23. Търсене в пространството от състояния. Генетични алгоритми.

Пространство на състоянията. Основни понятия. Формулировка на основните типове задачи за търсене в пространството на състоянията: търсене на път до определена цел, формиране на стратегия при игри за двама играчи, намиране на цел при спазване на ограничителни условия. Методи за информирано (евристично) търсене на път до определена цел: best-first search, beam search, hill climbing, A. Генетични алгоритми – основен алгоритъм, типове кръстосване и мутация.*

1. Основни понятия

Решаването на много задачи (но не на всички), традиционно смятани за интелектуални, може да бъде сведено до последователно преминаване от едно описание (формулировка) на задачата към друго, еквивалентно на първото или по-просто от него, докато се стигне до това, което се смята за решение на задачата. Такива задачи са например задачите от областта на игрите, аналитичните преобразования на математически изрази, решаването на алгебрични уравнения и др.

Състояние: едно описание (формулировка) на задачата в процеса на нейно то решаване.

Различават се няколко типа състояния: начално, междинни, крайни (целеви) състояния.

Оператор: начин (правило, алгоритъм, функция, връзка и т. н.), по който едно състояние се получава от друго.

Пространство на състоянията (ПС): съвкупността от всички възможни състояния, които могат да се получат от дадено начално състояние.

Най-естественият начин за представяне на ПС е чрез ориентиран граф (който може да бъде и дърво) с възли състоянията, и ориентирани дъги - операторите. Възможно е представяне на ПС и чрез низ, списък или на естествен език.

Основните действия свързани с пространството от състоянията са: генериране на състояния, оценяване на състоянията и дефиниране на цялото ПС.

Генериране на състояния представлява намиране на един или всичките наследници на дадено състояние. В процеса на генерирането се стига до два типа възли по отношение на генерирането на наследници: открит възел (съответства на състояние, от което могат да се генерират още наследници) или закрит възел (вече са генерирани всички негови наследници, или изобщо няма наследници или е целеви, или няма смисъл да се генерират негови наследници, тъй като е безперспективно).

Оценяването на състояние се извършва се с помощта на оценяваща функция V, която за дадено състояние дава оценката му съгласно някакви критерии (в частност, преценява дали дадено състояние е целево или не е такова, или установява степента му на близост до съответната цел). Последният въпрос е математически - въвеждане на метрика (разстояние между състояния).

В много случаи не се генерира цялото ПС, а само част от него. Например, ако се търси едно (произволно) целево състояние, възможно е да се стигне до цел преди да се генерира цялото ПС и тогава генерирането на останалата част от ПС не е необходимо.

Основните типове задачи, свързани с ПС, са

- генериране на ПС

- решаване на задачи върху генерирано ПС (търсене на цел, търсене на минимален път до цел, намиране на печеливша стратегия при игра и др.)
- комбинирана задача едновременно постепенно генериране на ПС и оценка на генерираните до даден момент състояния.

Проблемите при (методите за) решаване на тези задачи са сходни и затова често се говори само за търсене (а се подразбира и/или генериране).

2. Основните типове задачи за търсене в пространството на състоянията

2.1. Търсене на път до (определена) цел (path finding)

Първият основен тип задачи, които се решават с помощта на търсене в ПС е търсене на път до (определена) цел (path finding) - търси се път от някакво начално състояние до определено целево състояние (или до кое да е състояние от определено множество от целеви състояния). Пример за такава задача е търсенето на оптимален път, задачата за търговския пътни и др. За решаване на тази задача (по-точно, за генериране и обхождане на необходимата част от ПС) се използват два основни типа методи: пълно изчерпване (неинформирано, сляпо търсене) и евристично (информирано) търсене.

Пълното изчерпване (неинформирано търсене) се използва, ако нищо не се знае за задачата. То е крайно неефективно и често дори практически невъзможно (поради получаване на комбинаторен взрив и изчерпване на компютърните ресурси. Същността на този тип алгоритми се състои в последователното генериране на състоянията от ПС и сравняването им с целевото или списъка с целеви състояния. Основните стратегии за генерирането и обработването на състоянията при този вид търсена са обхождането в дълбочина (Depth-First Search), обхождането в ширина (Breadth-First Search) и комбинираното обхождане (в ширина и дълбочина).

При **евристичното търсене** се използват интуитивни или експертни знания за конкретната задача, които водят до повишаване на ефективността на търсенето при известен риск (пренебрегват се или изобщо не се генерират части от ПС). Компютърните алгоритми на евристичното програмиране са винаги точни, но с тях не винаги се достига до оптимално решение, а понякога въобще не се достига до решение. Някои от основните алгоритми за информирано търсене са best-first search, beam search, hill climbing, A* и други.

Best-first search

Методът на най-доброто спускане (Best-first search) е информиран (евристичен) метод, но все пак вариант на пълното изчерпване. Той е вариант на комбинация на търсене в дълбочина и широчина (следва се определен път, но когато друг път се окаже по-перспективен, се преминава към него). Тук евристиката е свързана с избора на посока, а не с отсичането. Методът се нарича на най-добро спускане при предположение, че оценяващата функция намалява към целево състояние. На всяка стъпка от изпълнението на съответния алгоритъм се избира най-добрият от възлите (възелът с най-добра оценка), генерирани до момента (а не измежду наследниците на текущото състояние). Това се прави с използване на подходяща оценяваща функция. След това се генерират наследниците на избрания възел. Ако някой от тях съвпада с търсената цел, процесът се прекратява. В противен случай всички генерирани наследници на избрания възел се добавят към множеството на възлите, генерирани до момента. Отново се избира най-добрият от генерираните възли и процесът продължава по описания начин, като се отбелязва всеки възел, чиито всички наследници са оценявани. Следователно обикновено се започва с търсене в дълбочина по най-перспективния клон, но ако там не се намери решение, започва изследване на изоставените клонове на по-горните равнища. Избира се следващ перспективен, но изоставен преди клон и се тръгва по него, като предишният клон отново не се забравя (към него можем да се върнем, когато останалите се окажат по-лоши) и т.н. Методът е ефективен, но не е оптимален. Времевата сложност на метода е $O(b^m)$, но използването на подходяща евристика може да доведе до съществено подобрение. Пространствената сложност на метода е $O(b^m)$, тъй като се съхраняват всички достигнати състояния.

Beam search

Методът за търсене в лъч (beam search) е оптимизация на Best-first-search, при която не се пазят всички частични решения, а само предварително определен брой от тях - k (ширината на лъча). Това намалява изискването за паметта от $O(b^m)$ до O(k). При k = 1 алгоритъмът се доближава до Hill Climbing.

Hill Climbing

При методът на изкачването (Hill Climbing) с подходяща евристична функция се оценяват състоянията, към които може да се премине от текущото състояние и се избира най-доброто, ако е с по-добра оценка от текущото. Това позволява да не се генерира цялото ПС, но е възможно решението да не бъде намерено, дори и да съществува. Търсенето се извършва еднопосочно, без възможност за връщане назад. Съществуват няколко ситуации, в които алгоритъмът не може да намери най-доброто решение:

- локален екстремум – състоянието е по-добро от съседните (наследниците си), но не е най-доброто в цялото ПС
- плато – съседните състояния (наследниците на текущото състояние) изглеждат също толкова добри, колкото и текущото
- хребет – никой от възможните оператори не води до по-добро състояние от текущото, макар че два или повече последователни оператори биха могли да доведат до такова състояние.

Ефективността на алгоритъма зависи много силно от евристичната функция, но често сложността на избирането на добра такава е съизмерима с решаването на самата задача.

A*

Търсенето с минимизиране на общата цена на пътя (A*) също е вариант на метода на най-доброто спускане. Евристичната функция, която се използва за избор на най-подходящо следващо състояние има следния вид: f(x) = g(x) + h(x). Тук функцията g връща като резултат цената на изминатия път от началния възел до текущия x, а евристичната функция h връща като резултат приблизителна стойност на цената на оставащата част от пътя от x до целта. Стратегията е пълна, ако разклоненията (наследниците) на всеки възел от ПС са краен брой и цената на преходите е положителна. Методът също така е и оптимален, ако евристиката е приемлива (оптимистична), т.е. никога не надценява стойността (цената) h* на оставащия път (ако h*(x) < h(x) за всеки възел x). Времевата и пространствената сложност на метода са експоненциални.

2.2. Формиране на стратегия при игра за двама играчи

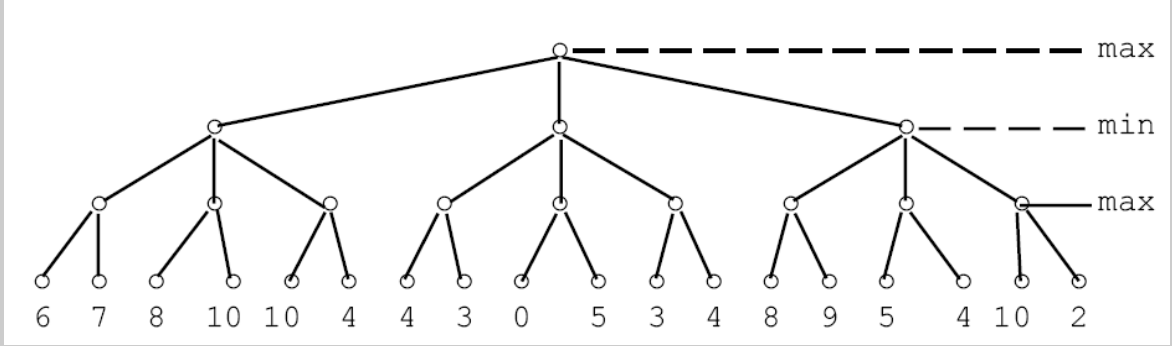
Вторият основен тип задачи за търсене в пространство от състоянията са задачите за формиране на стратегия при игра за двама играчи. Ще разглеждаме т. нар. интелектуални игри (с пълна информация), които се играят от двама играчи и върху хода на които не оказват влияние случайни фактори (като разпределение на картите при раздаване и др.). Двамата играчи играят последователно и всеки от тях има пълна информация за игровата ситуация, в частност всеки от тях знае какъв ход е избрал другият и от какво множество от възможни ходове е бил направен този избор. В определен момент от развитието на играта става ясно, че единият играч печели (и следователно другият губи) или играта завършва наравно. Примери за такива игри са шахмат, шашки, кръстчета и нулички, хо, ним и мн. др. Игрите на карти или други със случайни фактори се наричат игри с непълна информация например бридж.

Minimax

Основна стратегия при обхождането на дървото на състоянията на играта е **minimax** стратегията. Това е метод за намиране на най-добрия ход на първия играч при предположение, че другият играч също играе оптимално. Той изисква построяване на цялото дърво на състоянията (ДС) и намиране на оценките на листата (статични оценки). Предполага се, че двамата играчи имат противоположни интереси, изразени в това, че единият търси ход от стратегия, който води до получаване на позиция с максимална оценка, а другият търси ход (стратегия), който води до получаване на позиция с минимална оценка. Минимаксната процедура е метод за получаване на оценките (придобити или породени) на възлите от по-горните равнища на ДС, които позволяват на първия играч да избере най-добрия си ход. За определеност се приема, че първият ход се прави от максимизиращия играч, т. е. от този, който се стреми към получаване

на максимална оценка. Другият играч, който се стреми към получаване на минимална оценка, се нарича обикновено минимизиращ играч. Предполага се, че и двамата играчи играят оптимално.

Получаването на (придобитите или породени) оценки на възлите от по-горните равнища става по следния начин.



Ако сме в корена на дървото и на ход е максимизиращият играч, той ще избере един от трите възможни хода този, който води до позиция с максимална оценка. Ако започнем обхождането на дървото от най-левия клон, тук следващият ход е на минимизиращия играч. Той ще избере от трите възможни хода този, който води до позиция с най-малка оценка, т. е. ще избере хода с минимална оценка. При това оценките на тези три хода ще се получат в резултат на оценка на възможните ходове на максимизиращия играч в случая те зависят от оценките на листата на дървото. Така от долу на горе могат да се оценят всички възли и да се намери най-добрата оценка, която може да получи максимизиращият играч, а също и пътят от листата до корена на дървото, който води до тази оценка. Той включва най-добрия ход на максимизиращия играч.

Тази процедура изисква построяване на цялото ДС, което може да се окаже твърде голямо в общия случай на реална игра. Следователно това е точен, но крайно неефективен и често нереализуем метод поради “вечния” недостиг на компютърни ресурси.

Alpha-betha отсичане

В минимаксната процедура процесът на генерирането на ДС е напълно отделен от процеса на оценка на позициите. Оценката на позициите при този метод може да започне едва след завършването на генерирането на ДС. Оказва се, че това разделение води до силно неефективна стратегия. Обратно, ако листата се оценяват веднага след генерирането им и при първа възможност се пресмятат и съответните придобити (породени) оценки на възлите от по-горните равнища или поне се намерят подходящи горни граници на оценките на възлите от минимизиращите равнища и/или долни граници на оценките на възлите от максимизиращите равнища, то броят на операциите за намирането на същия резултат може да се намали значително. По същество α-β процедурата изисква генериране на ДС в дълбочина, при което се преценява безполезността от генериране на някои клонове, неоказващи влияние върху резултата. α-стойност на даден максимизиращ връх (от тип max) се нарича текущо установената долна граница на породената му оценка. β-стойност на даден минимизиращ връх (връх от тип min) се нарича текущо установената горна граница на породената му оценка. Могат да бъдат формулирани следните правила за прекратяване на генерирането и търсенето:

- α-отсичане. Не е необходимо да се извършва генериране и търсене върху всяко поддърво, което произлиза от min-връх, β-стойността на който е по-малка или равна на α-стойността на съответния max-родител.
- β-отсичане. Не е необходимо да се извършва генериране и търсене върху всяко поддърво, което произлиза от max-връх, α-стойността на който е по-голяма или равна на β-стойността на съответния min-родител.

При това ще се получи същият резултат, както и при пълната минимаксна процедура.

2.3 Търсене на цел при ограничителни условия

Третият тип задачи за търсене в ПС са свързани с търсене на цел при ограничителни условия (constraint satisfaction). Целта при този тип задачи е да се построи описание на търсеното целево състояние, което удовлетворява дадено множество от ограничителни условия.

Дадени са:

- множество от променливи v_1, v_2, … v_n (със съответни области на допустимите стойности Dv_i дискретни (крайни или изброими безкрайни) или непрекъснати)
- множество от ограничения (допустими/недопустими комбинации от стойности на променливите)

Търси се целево състояние (състояния), което е множество от свързвания със стойности на променливите $\{v_1=c_1, v_2=c_2, \ldots, v_n=c_n\}$, които удовлетворяват всички ограничения.

Примери за такива задачи са: задачата за осемте царици, решаване на криптограми, съставяне на разписания, различни проектантски задачи (с различни ограничения откъм време, пространство, цена и пр.), задача за зебрата, задача за триъгълниците и много други.

Общият алгоритъм за намиране на решение на задачата е следният: първоначалното множество от ограничителни условия се разширява, като в него се включват и ограниченията, които са логически следствия от входните. Този процес се нарича разпространяване на ограничителните условия. След това, ако не е намерено решение, се прави предположение (хипотеза) за някой от неуточнените параметри. По такъв начин ограниченията (ограничителните условия) се засилват, след което ново получените ограничителни условия отново се разпространяват и т. н. Целта е да се достигне до противоречие (тогава се осъществява връщане назад и се прави ново предположение за съответния параметър, ако това е възможно, и т. н.) или до намиране на решение (целево състояние е всяко състояние, което е описано с помощта на достатъчно силни за условията на конкретната задача ограничителни условия).

Описаният алгоритъм е максимално общ. За прилагането му в дадена конкретна област е необходимо да се използват (конкретизират) два типа допълнителни правила: правила, по които могат да се разпространяват ограниченията и правила, по които могат да се правят хипотези. При избора на обект, за който ще се направят допълнителни предположения (хипотези), се използват различни евристични правила например: избира се обект, който участва в повече на брой ограничителни условия (така по-бързо се разбира дали направеното предположение е правилно или не е); ако някой обект има повече възможни стойности от друг, то вторият е за предпочитане.

3. Генетични алгоритми

Генетичните алгоритми са вариант на стохастично търсене в лъч, при което новите състояния се генерират чрез комбиниране на двойки родителски състояния вместо чрез модифициране на текущото състояние. Състоянията се представят като низове над дадена крайна азбука (често като низове от нули и единици). Оценяваща функция (fitenss function) оценява пригодността (близостта до целта) на съответното състояние. Има по-големи стойности за по-добрите състояния. Алгоритъмът започва работа с множество (популация) от k случайно генерирани състояния (поколение 0). Използват се няколко принципа за генериране на следващите състояния от текущата популация: селекция, кръстосване, мутация.

При селекцията в генерирането на състоянията от следващото поколение участват някои от най-добрите представители на текущото поколение (съгласно оценяващата функция), избрани на случаен принцип.

При кръстосването се избират двойка "родителски" състояния и се определя т. нар. точка на кръстосването им (позиция в двата низа). Състоянието наследник се получава чрез конкатенация на началната част на първия и крайната част на втория родител. Възможно е да се получи и друг наследник, в конструирането на който участват неизползваните части на двамата родители.

При мутацията се извършват случайни промени в случайно избрана малка част от новата популация с цел да се осигури възможност за достигане на всяка точка от пространството на състоянията и да се избегне опасността от попадане в локален екстремум.

Кръстосването може да бъде извършвано по няколко начина, защото за различни задачи са подходящи различни видове кръстосване. При **кръстосването в единична точка** низът на новото състояние от началото си до точката на кръстосване е копие на началната част на единия родител, останалата му част е копие на съответната част на втория родител.

Родител 1 Родител 2 Резултат

8691247536 + 1234567892 = 8691567892

При **кръстосването в две точки** се избират две точки на кръстосване. Низът-резултат от началото си до първата точка на кръстосване е копие на съответната част от първия родител, частта на резултата от първата точка на кръстосване до втората точка на кръстосване е копие на съответната част на втория родител и останалото е копие на оставащата след втората точка на кръстосване част на първия родител.

Родител 1 Родител 2 Резултат

8691247536 + 1234567892 = 8691567536

При **аритметичното кръстосване** се извършва определена операция (аритметична, логическа и т.н.) между двамата родители и в резултат се получава новото потомство.

Родител 1 Родител 2 Резултат

1101011010 + 1101111101 = 1101011000

двоично кодиране, операция AND между съответните битове

Характерните приложения на генетичните алгоритми са в решаване на оптимизационни задачи (задача за търговския пътник, задача за раницата и др.), решаване на задачи за удовлетворяване на ограничения, задачи за избор на стратегия при игри за двама (по-общо, N) играчи, самообучение на невронни мрежи (уточняване на теглата на връзките между елементите в невронна мрежа с определена архитектура), генетично програмиране и други.

Общ алгоритъм:

1. Генериране на нулевата популация по случаен начин.
2. Оценяване на всяко състояние (спрямо fitness функцията)
3. Ако е изпълнено условието за край – КРАЙ
4. Избиране на две случайни състояния от най-добрите от текущата популация
5. Генериране на дете от двамата избрани родителя
    1. Получаване на низа на наследника чрез кръстосване на родителите
    2. Случайно с някаква вероятност се определя дали да се извърши мутация
6. Добавяне на ново полученото състояние към новата популация
7. Ако новата популация не е пълна преминаване към т. 3
8. Преминаване към т. 2

В зависимост от конкретната задача могат да бъдат използвани различни условия за прекратяване на изпълнението на един генетичен алгоритъм. Такива критерии, които често се използват на практика са: намиране на търсеното състояние, изтичане на определен брой итерации, съвпадение/близост на оценките на най-добрите състояния от последните няколко популации, наличие на висок процент състояния от текущата популация, които са идентични или подобни едно на друго и други.