1 Szukanie

1.1 Wstęp

Jedną z najważniejszych metod informatyki jest szukanie (searching). Trzeci tom fundamentalnego dzieła „The art of computer programming" Donalda Knutha poświęcony jest w całości algorytmom szukania. Konieczność szukania występuje w problemach dedukcji, rozumowania, wnioskowania, planowania, dowodzenia itp. Zastosowania metod opartych na szukaniu znaleźć można we wszystkich gałęziach AI. Niektórzy uczeni wszystko, co oparte jest na przeszukiwaniu drzewa możliwości wiążą z sztuczną inteligencją, np. techniki AI w fizyce czy chemii kwantowej sprowadzają się w większości do zastosowania algorytmów szukania. Chociaż jest to zbyt daleko idące uogólnienie trzeba stwierdzić, że algorytmy szukania rozwinęły się przede wszystkim w zastosowaniach związanych ze sztuczną inteligencją.

Szukajcie a (być może) znajdziecie! Bez szukania natomiast na pewno nie znajdziecie.

Zanim zaczniemy szukać trzeba wiedzieć gdzie prowadzić poszukiwania, tj. jak **reprezentować** problem, który chcemy rozwiązać i jak określić przestrzeń poszukiwań. Możemy tu wyróżnić 3 elementy. Po pierwsze, mamy jakąś **bazę danych** zawierających fakty czy możliwości, obszar dostępnych ruchów. Mogą w niej być tabele, listy słów, zbiory formuł, sieci semantyczne. Np. dowodząc twierdzenia matematyczne mamy bazę danych złożoną z aksjomatów, lematów i poprzednio udowodnionych twierdzeń. Korzystając z tego chcemy dojść do dowodzonego twierdzenia. Po drugie, mamy **przestrzeń możliwych operacji** prowadzących do zmian sytuacji w tej bazie danych. Sytuację w danym momencie określamy jako stan bazy danych. Operacje dokonywane są przez operatory powodujące zmianę tych stanów, manipulujące bazą danych. Przykładem są tu reguły wnioskowania w rachunku logicznym, reguły ruchów w grach, reguły heurystyczne dla upraszczania wyrażeń w rachunku symbolicznym. Trzeci element to **strategia kontrolna**, określająca start oraz sposób działania: do czego i który operator warto zastosować by dojść do pożądanej sytuacji końcowej (dowodu twierdzenia, przewagi na planszy w czasie gry). Osiąganie celu stosując odpowiednią sekwencję operatorów do sytuacji początkowej jest formą dowodzenia. Każde zastosowanie operatora zmienia sytuację, zmienia się baza danych i może to wpływać na strategię. Ta z kolei powinna dopuszczać różne sekwencje operatorów, które wydają się obiecujące. Ponieważ sekwencji takich może być bardzo dużo powstaje problem przeszukiwania.

Mamy 4 typy problemów:

1. Znany jest stan bazy i znane efekty działania.
2. Dany jest zbiór możliwych stanów początkowych, znane efekty działań.
3. Ograniczona znajomość stanów, działania zależne od warunków (np. ograniczona widoczność w trudnym terenie).
4. Problemy wymagające eksploracji: działania tworzą nowe warunki, potrzeba zdobywania nowej wiedzy (np. Pathfinder).

Wyróżnia się dwa rodzaje rozumowania. **Rozumowanie bezpośrednie** (do przodu) jest wtedy, gdy stosuje się operatory do faktów znanych by wytworzyć nową sytuację, bliższą sytuacji docelowej, np. mata w szachach lub uzyskanie przewagi na planszy. **Rozumowanie wsteczne** wychodzi od celu i stara się idąc do tyłu dotrzeć do czegoś znanego. Wyobrażając sobie drzewiastą strukturę schodzimy albo w dół albo do góry ale idąc do góry mamy większe szanse na trafienie na coś znanego jeśli nasza baza danych jest już spora. Możemy też zdefiniować podcele, bliskie celowi końcowemu, a do nich pod-podcele, a więc rozbić całe rozumowanie na etapy.

Rozumowanie do przodu określa się jako „data driven" lub z dołu do góry. Rozumowanie do tyłu jako „goal directed" lub z góry na dół. Kombinacje tych dwóch sposobów są również spotykane, w szczególności można porównywać różnice pomiędzy obecnym celem a sytuacją w zbiorze wygenerowanych stanów - jest to analiza środków i celów (means-ends analysis). Z różnicy wnioskuje się jaki operator najbardziej warto zastosować by ja zmniejszyć, a jeśli nie daje się zastosować operatora to wyznacza się nowy podcel.

Niezwykle ważna jest też **reprezentacja wiedzy** i sytuacji w bazie danych. „State-space representation" to

reprezentacja w której używa się bezpośredniego rozumowania a operatory tworzą w każdym kroku jeden nowy stan (obiekt) w bazie. Rozumowanie wstecz prowadzące do jednego stanu również posługuje się taką reprezentacją. Jeśli jednak problem dzieli się na zbiór podproblemów, np. całkowanie $2/(x^2-1)$ rozbija się na +, 1/(x-1), -1/(x+1) to mamy reprezentację redukowania problemów (problem-reduction representation). Najczęstszą reprezentacją sytuacji w grach, gdzie mamy element nieprzewidywalnej odpowiedzi przeciwnika, jest drzewo gry. Można też spotkać inne reprezentacje, zależnie od specyfik rozwiązywanego problemu.

Grafy lub struktury drzewiaste stanowią naturalne struktury opisujące **strategie kontrolną** w procesie szukania. Początkowy węzeł reprezentuje wyjściową sytuację, kolejne zastosowanie operatorów tworzy nowe węzły. Drzewa to szczególne przypadki grafów w których dany węzeł ma tylko jednego poprzednika (ojca). Stosuje się też różne specjalne grafy np. grafy i/lub. Z każdym węzłem wiąże się pewien opis stanu bazy. Drzewo wszystkich możliwości wyznacza przestrzeń szukania. Zwykle dążymy do tego by drzewo tworzone w czasie szukania było małym podzbiorem całej przestrzeni szukania. Czasami ta przestrzeń jest nieskończona a czasem tylko ogromnie wielka. Np. oceniono liczbę gier w szachach na 10^{120} a w warcabach 10^{40}. Jak znaleźć drogę do rozwiązania w tak wielkiej przestrzeni tworząc najmniejszy graf szukania?

W prostych przypadkach dobrze działa metoda „wygeneruj i testuj". Generator nowych stanów czy możliwości produkuje hipotezy a tester je sprawdza. Dobre generatory powinny być w stanie wygenerować wszystkie możliwe hipotezy (zupełność), unikać powtarzania tych samych hipotez oraz używać wszystkich informacji pozwalających wstępnie ograniczyć możliwe hipotezy.

Oczywiście przy szukaniu należy korzystać z niepełnej wiedzy czyli **wiedzy heurystycznej**. Czasami zapobiega to kombinatorycznej eksplozji drzewa możliwości. Heurystyczny oznacza „służący odkryciu". W AI heurystyczny oznacza proces mogący - ale nie gwarantujący - doprowadzić do rozwiązania, strategię, trik, regułę kciuka. Heurystyczny jest więc przeciwstawieniem ślepego przeszukiwania.

1.2 Reprezentacja problemu w przestrzeni stanów

Reprezentacja to zbiór konwencji dotyczących opisu pewnej klasy rzeczy. Opis problemu wykorzystuje te konwencje w konkretnym przypadku. Stan systemu jest opisem pozwalającym na określenie przyszłości systemu. W przestrzeni stanów wprowadzić można reprezentację graficzną tak, by węzły reprezentowały stany systemu a łuki przejścia pomiędzy tymi stanami.

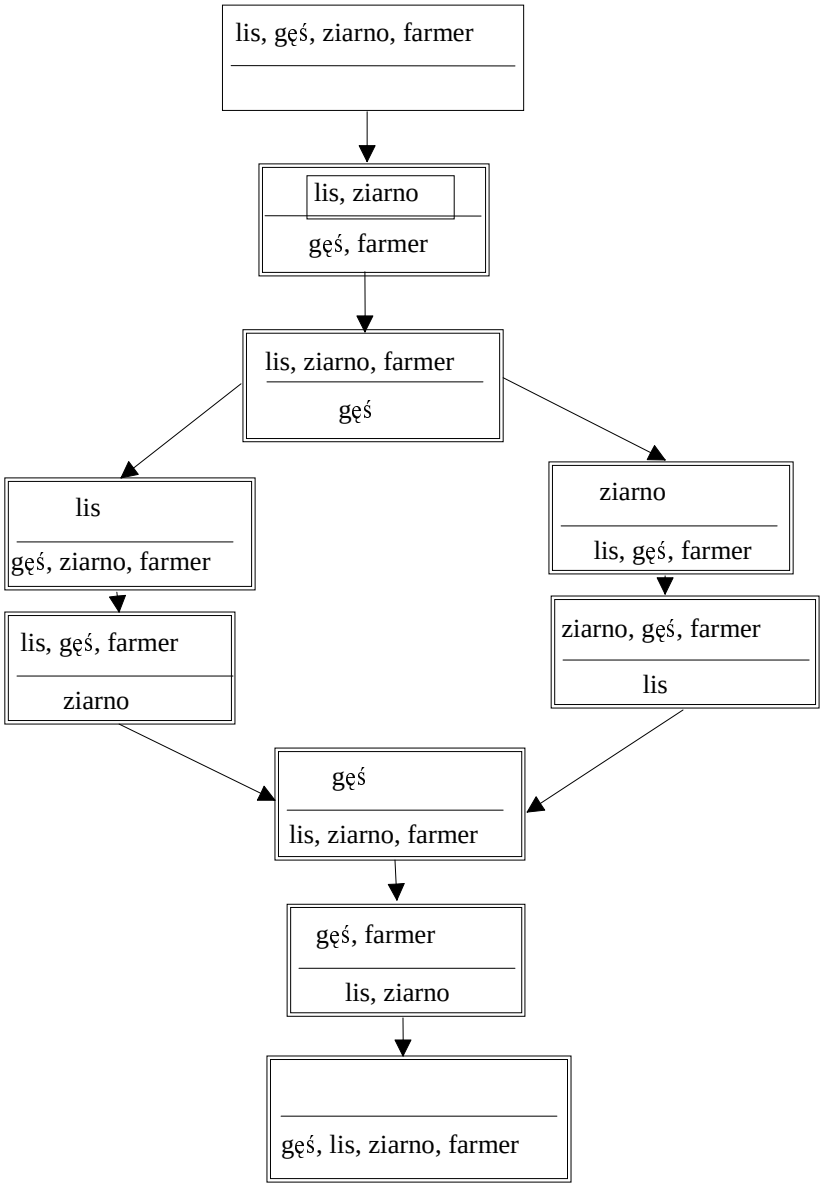
Prostym przykładem reprezentacji problemu w przestrzeni stanów może być gra w 8-kę.

2	1	7
3		8
5	4	6

Przestrzeń stanów liczy sobie 9!/2=181440 elementów. Stan opisywany jest w pełni przez macierz 3 na 3. Operacje na obiektach polegają na ich przesuwaniu, ale zamiast osobnych operacji na 1 .. 8 lepiej jest zdefiniować operacje na pustym polu jako ruchy w 4 kierunkach u, d, l, r. Te ruchy to nasz zbiór operatorów *O*. Potrzebujemy też zbioru stanów wyjściowych *S* i końcowych *G*. Problem zdefiniowany jest jako zbiór *(S,O,G).* Rozwiązanie problemu to skończony ciąg transformacji lub zastosowań operatorów zamieniający *S* w *G*.

Kolejność zastosowań operatorów lub następstwo stanów najlepiej przedstawia się w postaci grafu, np. w problemie wędrującego komiwojażera - jak znaleźć najkrótszą drogę przez N miast odwiedzając każde z nich dokładnie jeden raz - startujemy od miasta A jako wierzchołku grafu, na drugim poziomie mamy węzły AB, AC, .., na trzecim podwęzły ABC, ABD, ... ACB, ACD ... i na N-tym mamy N! węzłów końcowych obrazujących wszystkie możliwe drogi. Rozmiary grafów w realnych problemach rosną kombinatorycznie, konieczne są więc „inteligentne" metody ich przeszukiwania.

Innym prostym przykładem obrazującym użyteczność reprezentacji stanów w postaci grafu jest dziecięca zagadka: jak przewieźć lisa, gęś i ziarno małą łódką na drugą stronę rzeki, jeśli zmieści się w niej nie więcej niż jedna rzecz (oczywiście oprócz nas). Należy tu rozważyć wszystkie możliwości - każda z 3 rzeczy plus przewożący je farmer mogą być po jednej lub drugiej stronie rzeki, więc mamy 2^4=16 możliwości. Wśród nich jest 6 niebezpiecznych i 10 akceptowalnych. Na rysunku przedstawiłem tylko możliwości akceptowalne



Rys. Graf rozwiązań dla prostego problemu logicznego

pomijając węzły ślepe, które nie prowadzą do rozwiązania. Zagadka ta znana jest również pod nazwą problemu misjonarzy i kanibali i dla większej liczby osób lub obiektów nie jest wcale trywialna, gdyż wymaga tworzenia pośrednich etapów, które „oddalają się" od pożądanego rozwiązania.

W tym przypadku reprezentacja w przestrzeni stanów jest dobrą reprezentacją prowadzącą do łatwego rozwiązania problemu. Ogólnie można zauważyć, że:

Dobór odpowiedniej reprezentacji to znaczna część rozwiązania problemu.

Jeśli liczba węzłów rośnie eksponencjalnie lub kombinatorycznie mówimy, że zagadnienie jest nieobliczalne. Od reprezentacji sytuacji zależy często bardzo wiele. Rozważmy dla przykładu następujące zadanie: czy 31 domin może pokryć wszystkie pola szachownicy z której usunięto 2 rogi? Mamy tu bardzo wiele możliwości ale wystarczy zauważyć, że domino pokrywa zawsze biało/czarne pole a usuwamy dwa pola o tym samym kolorze. Jak znaleźć w konkretnym przypadku odpowiednią reprezentację problemu tak, by dał się łatwo rozwiązać, albo przynajmniej lepszą reprezentację niż któraś z ogólnie stosowanych? Jest to często najtrudniejsza część pracy.

Dobra reprezentacja uwidacznia relacje pomiędzy istotnymi elementami lub stanami, pozwala na ujawnienie się wszystkich więzów ograniczających możliwe relacje, nieistotne szczegóły zostają usunięte, powinna być przy tym zrozumiała, kompletna (zawierająca wszystko, co potrzeba do rozwiązania), zwięzła i pozwalająca na

efektywne wykorzystanie w modelu komputerowym.

1.3 Redukcyjna reprezentacja problemu

Podstawowymi strukturami są tu nie stany, ale cele, czyli opisy problemu. Początkowy opis problemu poddaje się serii transformacji, aż dochodzimy do problemów elementarnych. Redukcyjna reprezentacja składa się więc z:

Opisu początkowego problemu
Zbioru operatorów transformujących dany problem na problemy cząstkowe
Zbioru problemów elementarnych

Przykład: Wieża z Hanoi

Mamy trzy wielkości krążków, A, B, C, i trzy kołki, *i, j, k*. W tym przypadku problem przesunięcia n>1 krążków z *i* na *k* rozbija się na podproblemy:

Przesuń stos n-1 klocków z *i* na *j*
Przesuń jeden klocek z *i* na *k*
Przesuń stos n-1 klocków z *j* na *k*

Problemem elementarnym jest oczywiście przesunięcie pojedynczego klocka. Opis problemu zawiera dane: ile jest klocków na stosie do przesunięcia, z którego kołka, na który kołek. Rozwiązanie możemy znowu przedstawić przy pomocy drzewa. Pewnym uogólnieniem prostych drzew są grafy AND/OR, na których zaznacza się, czy dany problem daje się rozwiązać, jeśli wszystkie podproblemy dają się rozwiązać (węzeł AND), czy też wystarczy rozwiązać tylko jeden z nich (węzeł OR).

Którą z tych dwóch reprezentacji - opisu problemów czy przestrzeni stanów - należy użyć jest kwestią wygody. Można tłumaczyć problemy z jednej reprezentacji do drugiej ale często w jednej z nich łatwiej się dają rozwiązać niż w drugiej. Jest wiele innych metod reprezentacji do których wrócimy w następnym rozdziale, gdyż metody reprezentacji wiedzy to centralne zagadnienie sztucznej inteligencji.

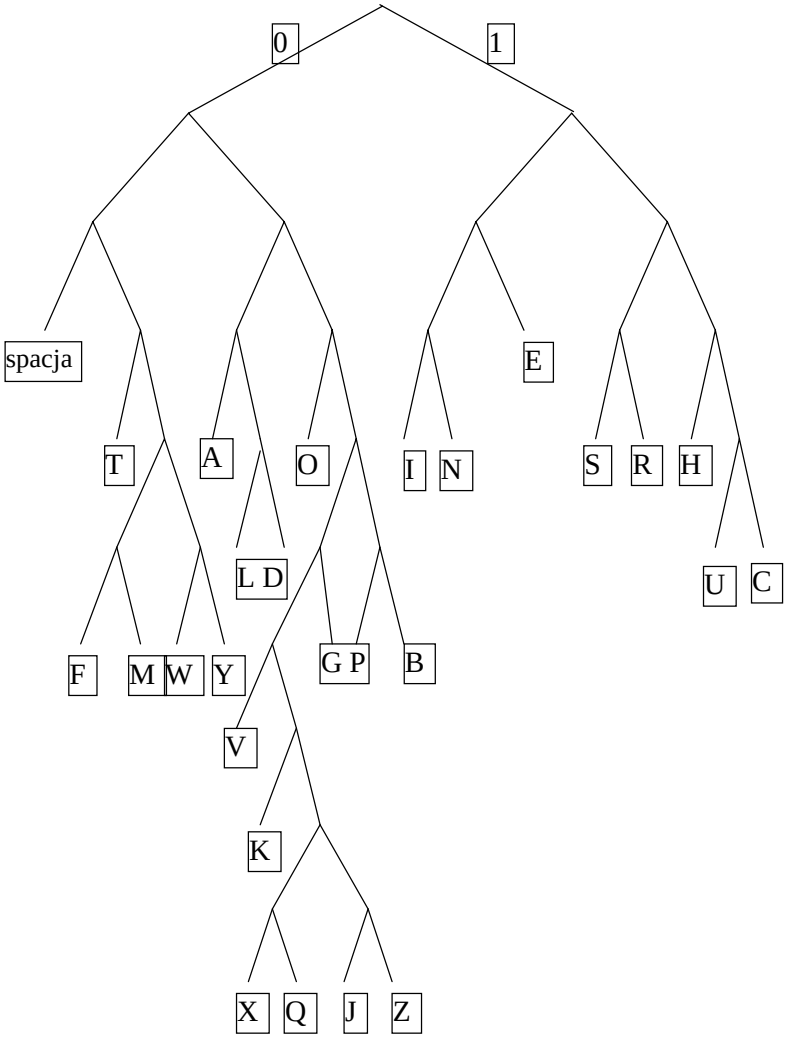
1.4 Metody szukania na ślepo

Reprezentacja stanów, reprezentacja redukcyjna i inne metody prowadzą w praktyce do tych samych problemów tj. do konieczności szukania rozwiązania w obszernej przestrzeni możliwych stanów lub możliwych zastosowań operatorów redukujących. Jest wiele metod prowadzenia takiego szukania (szczegółowe omówienie zawiera książka Bolca i Cytowskiego). Ograniczymy się tutaj tylko do podstawowych metod przeszukiwania „na ślepo". Szukanie drogi w grafie prowadzącej od początkowego węzła do węzła, który jest celem daje się przedstawić jako proces tworzenia drzewa poszukiwań. Drzewo jest szczególnym rodzajem grafu w którym każdy z węzłów ma tylko jednego rodzica. Dzięki temu węzły drzewa można utożsamić z drogami. Drzewo ma swój korzeń (root node) i liście, czyli węzły nie mające swoich następników lub dzieci. Określanie poddrzewa danego węzła nazywa się też rozwijaniem węzła. O węzłach, które nie są do końca rozwinięte mówi się, że są „otwarte" a o całkowicie rozwiniętych „zamknięte". W drzewie określić można średnią liczbę rozgałęzień (branching factor). Jeśli wynosi ona *k* to rozwinięcie węzła na głębokość *d* prowadzi średnio do k^d nowych węzłów, czyli do eksplozji kombinatorycznej. Jedynie dla prostych przypadków można wyliczyć wszystkie możliwe ścieżki drzewa przeszukiwań i wybrać z nich najlepsze rozwiązanie. Skrajnie różna wersja tej metody, określanej jako „generuj rozwiązanie i testuj czy jest właściwe" polega na przypadkowym wyborze jednego z rozwiązań i sprawdzeniu, czy jest ono właściwe. W literaturze amerykańskiej takie postępowanie określa się nazwą „**procedury Brytyjskiego muzeum**" - szanse na znalezienie jakiegoś obiektu w ogromnym Muzeum Brytyjskim przypadkowo błądząc po salach i magazynach są bardzo małe. Potrzebne są bardziej systematyczne podejścia.

1.1.4. Szukanie „w głąb”

Podstawowym rodzajem przeszukiwania grafów jest szukanie w głąb. Jeśli w danej sytuacji wszystkie posunięcia lub wszystkie kolejne stany są równie prawdopodobne to robimy następny krok tworząc, na grafie przedstawiony jako przesunięcie się na kolejny, niższy poziom, aż dochodzimy do poziomu, na którym nie ma już dalszych możliwych przekształceń. Jeśli poziom ten reprezentuje właściwe rozwiązanie zakończyliśmy proces szukania. Formalny algorytm wygląda następująco:

- Utwórz jednoelementową listę składającą się z ścieżki o zerowej długości wychodzącej z korzenia



Drzewo binarne (drzewo Huffmana) obrazujące minimalną liczbę bitów potrzebną do zakodowania litery w języku angielskim (uwzględniając częstości występowania liter). Algorytm przeszukiwani a w głąb pozwoli szybko odnaleźć litery zakodowane przy pomocy dużej liczby początkowych zer, algorytm szukania w szerz litery o najbardziej oszczędnym kodowania ale litery o najmniej oszczędnym kodowaniu jest stosunkowo trudno znaleźć (w tak małym drzewie nie ma z tym oczywiście problemu).

  - Dopóki pierwsza ścieżka na liście nie kończy się na węźle zawierającym cel, lub lista nie jest pusta:
    - Usuń pierwszą ścieżkę; utwórz wszystkie ścieżki wychodzące z końcowego węzła
      - Odrzuć ścieżki zawierające pętle
      - Jeśli któryś z nowo utworzonych węzłów jest celem zakończ pętlę
      - Dodaj na szczyt listy pozostałe ścieżki
- Jeśli znaleziono rozwiązanie to ogłoś sukces i podaj ścieżkę prowadzącą do celu, w przeciwnym przypadku ogłoś klęskę.

Program pamięta dane o kolejnych odwiedzanych węzłach i wybranych łukach, co pozwala na cofanie się po

osiągnięciu końcowego liścia o jeden poziom wyżej. Wymagania dotyczące pamięci wiążą się więc jedynie z głębokością szukania. Jeśli jednak systematycznie badamy rozwiązania na grafie, począwszy od strony lewej do prawej, a prawdziwe rozwiązanie leży gdzieś w prawej części końcowych liści drzewa rozwiązań procedura ta nie prowadzi do celu szybciej od procedury szukania na ślepo. Jeśli ścieżki są bardzo długie (graf bardzo głęboki) można wprowadzić ograniczenie głębokości przeszukiwania. Algorytm ten gwarantuje osiągnięcie celu w przypadku skończonej przestrzeni przeszukiwań - przeszukiwanie przebiega wówczas po całej przestrzeni. Jeśli przestrzeń jest zbyt duża to czas potrzebny na takie przeszukiwanie może być zbyt długi. Nie ma też gwarancji, że znalezione rozwiązanie jest optymalne (tzn. osiągnięte w najmniejszej liczbie kroków, w najprostszy sposób).

1.2.4. Szukanie „wszerz”

Drugim podstawowym sposobem szukania jest rozwijanie na każdym poziomie wszystkich węzłów następnej generacji, czyli szukanie wszerz. Sprawdzane są węzły kolejno na danym poziomie. Jeśli rozwiązanie położone jest niezbyt głęboko procedura szukania wszerz prowadzi szybciej do jego znalezienia. Wymagania pamięci są jednak w tym przypadku znacznie większe, gdyż trzeba pamiętać znaczną część drzewa, ze wszystkimi węzłami znajdującymi się na górnych poziomach. Jeśli rozwiązanie leży na którymś z głębszych poziomów procedura szukania wszerz może być niewykonalna ze względu na ograniczenia dotyczące pamięci. Z drugiej strony daje się ona zastosować nawet wówczas, gdy w drzewie poszukiwań istnieją nieskończenie długie drogi. By procedura ta była efektywna średni stopień rozgałęzienia nie powinien być zbyt duży.

1.5 Szukanie heurystyczne

Uniknięcie ślepego przeszukiwania jest głównym zadaniem metod heurystycznych. Procedurę szukania w głąb należy więc zmodyfikować, wprowadzając funkcje oceny przydatności poszczególnych ruchów i porządkując węzły na danym poziomie według oceny ich „atrakcyjności”. Takie procedury stosowane są często w grach planszowych, np. przy grze w warcaby czy szachy. Zadanie często sprowadza się do maksymalizacji funkcji oceny statycznej i można w tym przypadku stosować różne metody maksymalizacji. Najprostszym rozwiązaniem jest „wspinanie do góry” (hill climbing), czyli metoda gradientowa, wybierająca na każdym kroku maksymalnie korzystne rozwiązanie. Funkcja heurystyczna ocenia tu, jak bardzo dany stan oddalony jest od celu, czyli stanu pożądanego. Algorytm tej procedury wygląda następująco:

- Oceń stan początkowy; jeśli nie jest on poszukiwanym celem wykonaj:
  - Utwórz nowy stan
  - Jeśli nowy stan jest stanem celu zakończ.
  - Jeśli nowy stan jest bliżej stanu celu przyjmij go jako bieżący; w przeciwnym przypadku zignoruj go.
  - Jeśli nie można już utworzyć nowych stanów zatrzymaj się.

Jest to więc szukanie w głąb ukierunkowane przez funkcję celu. Ponieważ nie gwarantuje to znalezienia absolutnego maksimum modyfikuje się tą metodę tak, by na każdym kroku mieć listę kilku najbardziej obiecujących węzłów. Taką modyfikację określa się mianem „**najpierw najlepszy**”. Formalny algorytm szukania metodą „najpierw najlepszy” z heurystyczną funkcją *f* oceniającą węzły wygląda następująco:

- Utwórz jednoelementową listę składającą się korzenia
  - Dopóki lista nie jest pusta lub nie osiągnięto węzła celu:
  - Znajdź węzeł *n* dla którego funkcja *f(n)* przyjmuje minimum
    rozwiń *n* tworząc wszystkie węzły potomne
  - jeśli któryś z węzłów potomnych jest poszukiwanym celem zakończ
    - dla każdego węzła potomnego *n'*:
      - oceń węzeł funkcją *f(n')*
      - jeśli węzeł wcześniej się pojawił zostaw tylko ten z lepszą wartością *f(n')*
- Jeśli znaleziono rozwiązanie to ogłoś sukces i podaj ścieżkę prowadzącą do celu, w przeciwnym przypadku ogłoś klęskę.

Metoda nie gwarantuje znalezienia optymalnego rozwiązania a konieczność sprawdzania czy węzeł wcześniej się pojawił zwiększa wymagania dotyczące pamięci i złożoności obliczeń.

Wariantem tej strategii jest procedura **A***, oceniająca węzły przez dodanie do kosztów ich uzyskania (złożoności ścieżki do nich prowadzącej) ocenę odległości od stanu końcowego. Pomaga to - chociaż nie gwarantuje - w szukaniu optymalnego rozwiązania. Funkcja heurystyczna *f(n)* składa się z sumy kosztu dojścia do węzła *g(n)* i oceny kosztu dojścia do węzła celu *h(n).*

- Rozpocznij od początkowego węzła i utwórz nowe węzły *n*;
  - Posortuj nowe węzły *n* korzystając z funkcji *f (n)= g(n) + h(n);*
    - Wybierz najlepszy *n'*
    - Jeśli *n'* jest celem skończ;
    - Jeśli nie rozwijaj dalsze węzły

Ocena kosztu dana funkcją *h(n)* może być zawyżona; można pokazać, że jeśli prawdopodobieństwo przecenienia odległości o *d* jest niewielkie to algorytm A* rzadko znajdzie rozwiązanie, którego koszt jest większy niż *d* od optymalnego rozwiązania.

Bardzo przydatną metodą poszukiwania globalnego maksimum jest **„simulated annealing”**, czyli metoda stopniowego studzenia, odkryta w 1983 roku. Idea tej metody oparta jest na analogii z procesem studzenia się materiałów. Przyroda bardzo efektywnie rozwiązuje problem poszukiwania globalnego minimum funkcji energii. Materiały ogrzane do wysokich temperatur i stopniowo oziębiane krystalizują w stanach o najniższych energiach. Zbyt szybkie chłodzenie spowodować może powstanie defektów, zamrożonych lokalnych fluktuacji lub niedoskonałości kryształów, o energiach wyższych niż energie konfiguracji bardziej symetrycznych. Istnienie fluktuacji energii powoduje, że prawdopodobieństwo wystąpienia konfiguracji o energii wyższej o ΔE od absolutnego minimum dane jest przez rozkład Boltzmana:

$$p(\Delta E) = \exp(-\Delta E / kT)$$

gdzie *T* jest temperaturą. Prawdopodobieństwo pojawienia się stanów o wyraźnie wyższej energii jest niewielkie przy niskiej temperaturze i stosunkowo duże przy temperaturze wyższej. Twierdzenie o stopniowym ostudzaniu mówi, że jeśli temperatura zmierzać będzie do zera nieskończenie wolno to układ zmierzać będzie do konfiguracji odpowiadającej absolutnemu minimum czyli ΔE=0. W praktyce dobór odpowiedniego schematu studzenia ma duży wpływ na wyniki. W procedurach heurystycznego szukania metoda stopniowego chłodzenia stosowana jest do generowania nowych węzłów i oceny ich przydatności. Przechowywane są dane węzła dającego najlepsze wyniki i wybierany nowy węzeł - jeśli jest on lepszy niż przechowywany to przyjmuje się go za nowy węzeł, jeśli jest gorszy to z prawdopodobieństwem p(ΔE) staje się on węzłem bieżącym z którego prowadzimy dalsze poszukiwania generując nowe węzły. Metodę stopniowego studzenia można uważać za ulepszoną metodę gradientową, usiłującą zbadać cały krajobraz i uniknąć lokalnych minimów. Formalny algorytm wygląda następująco:

- Oceń stan początkowy i przyjmij go za bieżący; jeśli jest on stanem celu to zakończ
- Nadaj zmiennej Min_stan wartość równą ocenie stanu bieżącego
- Ustal temperaturę *T* zgodnie z założonym schematem studzenia
  - Generuj nowy stan
  - Jeśli nowy stan jest stanem celu zakończ;
  - Oblicz różnicę wartości ocen ΔE stanu bieżącego i nowego
  - Jeśli nowy stan jest lepszy niż Min_stan nadaj tej zmiennej nową wartość i przyjmij go za bieżący
  - Jeśli jest gorszy to oblicz $p(\Delta E)$ i przyjmij z takim prawdopodobieństwem za bieżący
- Zmień *T* zgodnie z założonym schematem
- Wykonuj dopóki znalezione zostanie rozwiązanie lub nie będzie więcej stanów.

Przyjmowanie nowego stanu z danym prawdopodobieństwem oznacza wywołanie liczby losowej z przedziału (0,1) i jeśli będzie ona większa od tego prawdopodobieństwa to przyjmujemy ten stan, a jeśli mniejsza to nie. W praktyce metoda stopniowego chłodzenia daje często dobre rezultaty chociaż z punktu widzenia ilości obliczeń jest to metoda bardzo kosztowna. „Adaptive simulated annealing” to wersja tej metody pozwalająca na stosowanie niezależnych schematów obniżania temperatury do różnych parametrów. Eksponencjalne chłodzenia polega na zmniejszaniu temperatury o ustalony procent poprzedniej wartości, czyli $T_{i+1} = \alpha T_i$. Stosuje się też wersję z szybkim chłodzeniem, zwaną „simulated quenching”, w której temperaturę zmniejsza się o ustaloną wartość $T_{i+1} = T_i - \Delta T$. Wersja ta może łatwiej zgubić globalne minimum.

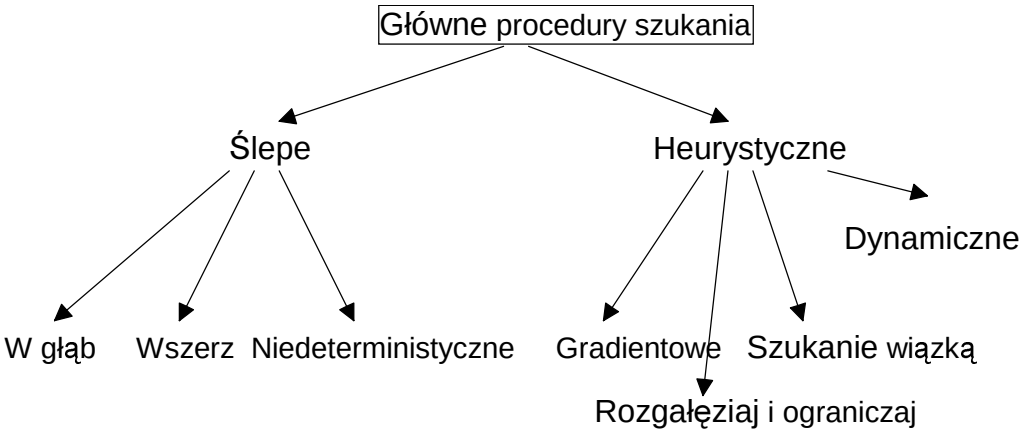
Poszukiwanie w głąb jest dobrym rozwiązaniem jeśli jesteśmy przekonani, że drogi kończą się ślepo lub prowadzą do rozwiązania po niezbyt wielkiej liczbie kroków. Jeśli podejrzewamy, że w procesie szukania mogą powstać bardzo długie drogi nie prowadzące do nikąd lub nieskończenie długie drogi bezpieczniej jest zastosować inne metody. Innym istotnym czynnikiem przy ocenie przydatności algorytmu szukania w głąb jest rodzaj oceny heurystycznej, którą potrafimy dokonać. Jeśli mamy absolutną miarę dobroci rozwiązania problemu to szukanie w głąb można zakończyć, gdy jej ocena jakiegoś węzła wypada dostatecznie dobrze. Jeśli jednak mamy tylko względną miarę, to nie wiemy czy kolejna grupa węzłów nie okaże się znacznie lepsza. Tak dzieje się np. dla problemu wędrującego sprzedawcy przy ocenie długości drogi prowadzącej przez N miast.

1.6 Inne metody szukania

Modyfikacją procedury szukania wszerz jest „**przeszukiwanie wiązką**" (beam search), czyli rozwijanie na każdym poziomie tylko ograniczonej liczby węzłów, które oceniane są jako najbardziej obiecujące. Niektóre węzły nie są więc rozwijane zupełnie. Decyzja o rozwijaniu ścieżek z danego węzła oparta jest na sumarycznej ocenie przydatności wszystkich węzłów pochodnych.

Innym wariantem metod szukania jest procedura mieszana, złożona z przeszukiwania wiązką oraz szukania w głąb (**branch-and-bound**, czyli „rozgałęziaj i ograniczaj"). W metodzie tej dokonuje się ograniczonego przeszukiwania wiązką i rozwija w pierwszym rzędzie te węzły, które są najbardziej obiecujące, wędrując w ten sposób w głąb aż do węzłów końcowych. Na każdym poziomie mamy więc rozwinięcie tylko kilku węzłów. Przeszukiwanie tą metodą ulepsza się na różne sposoby. Jednym z nich jest zastosowanie zasady **dynamicznego programowania**, polegająca w tym przypadku na obcinaniu tych wszystkich dróg, które prowadzą do tego samego węzła pośredniego, a które są dłuższe lub mniej optymalne od innych dróg jednocześnie rozwiniętych. Jeśli dwie lub więcej ścieżek prowadzi do tej samej sytuacji to wystarczy rozważyć tylko tą z nich, której koszt jest najmniejszy.

Odmianą programowania dynamicznego jest również usuwanie dróg oparte na ocenie odległości od pożądanego rozwiązania końcowego. Jeśli nie mamy dobrej miary odległości należy się postarać przynajmniej o ocenę jej dolnego kresu. Dodawanie dolnego kresu do już przebytej odległości pozwala na pozostawienie tylko tych węzłów z listy węzłów do rozwinięcia, dla których ocena z uwzględnieniem dolnego kresu pozostałej odległości jest najmniejsza.



Dobre rezultaty można czasami osiągnąć przy pomocy **niedeterministycznego szukania**. Proces ten podobny jest do wykorzystania algorytmów niedeterministycznych, np. metody Monte Carlo, do zagadnień optymalizacji. Rozwija się w nim przypadkowo wybrany węzeł. Jeśli mamy dobre funkcje heurystycznej oceny jak blisko lub jak podobny jest dany węzeł do węzła końcowego celu można użyć wielu różnych algorytmów dotyczących szukania globalnego minimum funkcji, np. algorytmów gradientowych, stopniowego studzenia czy algorytmów genetycznych. Jednakże metody minimalizacji również należą do zagadnień NP-trudnych i mogą prowadzić do nienajlepszych rozwiązań (lokalnych minimów) lub mieć trudności z określeniem kierunku poszukiwań jeśli wiele węzłów daje takie same wartości funkcji oceny. Prostą metoda globalnej minimalizacji jest metoda sympleksów. Istnieje mało znana wersja tej metody budująca jednocześnie wiele wielowymiarowych sympleksów. Algorytm P^* opiera się na poszukiwaniu minimum wzdłuż przypadkowo

wybranej prostej, po czym stosowana jest metoda gradientowa w celu znalezienia minimum leżącego w pobliżu tej prostej.

Szczególnym przykładem algorytmów Monte Carlo połączonych z wiedzą heurystyczną są **algorytmy genetyczne**. Wektor parametrów nazywa się tu chromosomem i poddaje różnym „operacjom genetycznym". Po wygenerowaniu pewnej populacji chromosomów ocenia się ich „funkcję przystosowania" f_i (fitness function, wartość minimalizowanej funkcji kosztu), oraz prawdopodobieństwo przeżycia, równe tej wartości podzielonej przez całkowitą sumę wszystkich f_i dla całej populacji. operacje genetyczne, takie jak jednopunktowe lub wielopunktowe krzyżowanie chromosomów, mutacje i bardziej wyspecjalizowane operacje, mają na celu zapewnienie lepszego przystosowania populacji. Niestety nie ma uniwersalnego sposobu na dobranie najlepszych operatorów, metody genetycznej optymalizacji są zwykle „ręcznie" dostrajane do problemu.

Jest kilka metod związanych bezpośrednio z algorytmami szukania choć nieco poza nie wykraczających. Jedną z nich jest **procedura minimaksu**, użyteczna w grach, w których usiłujemy maksymalnie wzmocnić swoja pozycję i maksymalnie osłabić pozycję przeciwnika (opisał ją Shannon w 1950 roku). Konieczna jest do tego procedura oceniająca daną sytuację z punktu widzenia każdej ze stron. Dla redukcji drzewa szukania i liczby statycznych ocen sytuacji reprezentowanych przez węzły stosuje się **zasadę alfa-beta**: jeśli któreś rozwiązanie jest na pewno marne nie trzeba tracić czasu by rozwijać jego gałąź. Parametry alfa i beta pozwalają określić w różnych gałęziach najlepsze wyniki obu stron, czyli minmizera i maksymizera. Schodząc o kilka poziomów w głąb ocenia się końcowe sytuacje z punktu widzenia maksymizera i cofa do poprzedniego poziomu bo ocenić optymalny ruch minimizera. Jeśli w innych gałęziach po zejściu w głąb zobaczymy, że sytuacja z punktu widzenia minimizera jest mniej korzystna to nie warto dalej rozważać tej gałęzi. Procedura alfa-beta zmniejsza szybkość eksplozji kombinatorycznej ale jej nie zapobiega. Jeśli czas przeznaczony na szukanie rozwiązania jest ograniczony stosuje się procedurę stopniowego pogłębiania, poszukując optymalnych rozwiązań na kolejno coraz głębszych poziomach. Nietrudno zauważyć, że praca konieczna do oceny sytuacji na nowym poziomie zawsze dominuje w całkowitych kosztach poszukiwania rozwiązania.

Agendy to listy zadań, które system może w danej chwili wykonać, połączone z uzasadnieniami dlaczego takie proponowane są takie zadania oraz z probabilistyczną oceną poszczególnych zadań. Rozumowania w oparciu o agendy sprowadza się do poszukiwania stanów o największym końcowym prawdopodobieństwie. W tego typu rozumowaniach stosuje się również metody spełniania ograniczeń, w których cel jest takim stanem systemu, w którym spełniona jest maksymalna liczba ograniczeń.

Analiza celów i środków (means-ends analysis) wykorzystuje różnice pomiędzy stanem aktualnym a docelowym usiłując znaleźć takie przejście w grafie które w maksymalny sposób redukuję tą różnicę. W odróżnieniu od poprzednich strategii szukania w metodzie tej usiłuje się niejako jednocześnie stosować rozumowania w przód i wstecz (dedukcyjne i redukcyjne), próbując połączyć ze sobą ich wyniki. Szkic algorytmu wygląda następująco:

- Dopóki cel nie zostanie osiągnięty lub nie będzie więcej nowych procedur wykonuj:
    - Opisz stan bieżący, cel i różnice miedzy nimi
    - Poszukaj procedury, która minimalizuje różnicę
    - Użyj tej procedury działając na stan bieżący by utworzyć nowy stan
- Jeśli cel zostanie osiągnięty ogłoś sukces; jeśli nie klęskę.

Wiele problemów najłatwiej jest opisać podając różne ograniczenia, jakim podlegają. Trzeba wówczas znaleźć rozwiązanie zgodne ze zbiorem ograniczeń (constraint satisfaction). Heurystyki używane są wtedy do określenia, który z węzłów opisujących bieżącą sytuację warto rozwijać. Algorytm wygląda następująco:

- Propaguj zadane ograniczenia:
    - Utwórz opis stanu zawierający wszystkie obiekty podlegające ograniczeniom
    - Powtarzaj aż wszystkie ograniczenia zostaną spełnione:
    - wybierz obiekt i spróbuj poddać go transformacji dającej maksymalnie dobre spełnianie warunków;
    - jeśli ograniczenia ulegną zmianie po transformacji obiektu otwórz wszystkie obiekty które podlegają tym samym ograniczeniom i poddaj je transformacjom.
- Zakończ

Jest jeszcze wiele innych metod heurystycznego przeszukiwania. W nielicznych przypadkach udowodnić można optymalność (w sensie generowania statystycznie najmniejszej liczby węzłów w procesie szukania) pewnych

metod, zależy to jednak bardzo od założeń dotyczących problemu. Ogólnie można stwierdzić, że zagadnienie szukania należy do zagadnień matematycznie trudnych i nie ma tu idealnych rozwiązań dobrych dla każdego przypadku.

Porównanie kilku metod szukania w zastosowaniu do dwóch problemów - szukania rozwiązań dla 8-mki przy utworzeniu stanu początkowego ze stanu końcowego za pomocą 20 lub 100 przypadkowych ruchów, oraz szukania najkrótszej drogi pociągiem pomiędzy miastami w USA, zilustrowane jest w tabeli. Widać z niej, że najlepsze rezultaty przy stosunkowo najmniejszych wymaganiach obliczeniowych daje procedura A*.

Rozwiązywanie 8-emki dla 20 i 100 ruchów

Algorytm	Pamięć 20 ruchów	Czas 20 ruchów	L. ruchów 20 ruchów	Pamięć 100 ruchów	Czas 100 ruchów	L. ruchów 100 ruchów
W głąb	-	-	-	-	-	-
W głąb do 5 poziomów	11.00	144.00	4.00	-	-	-
Wszerz	104.00	60.00	4.00	> 4500	> 2000	> 8
Wspinanie do góry	1.00	5.00	4.00	-	-	-
Najpierw najlepszy	6.00	5.00	4.00	148.00	148.00	26.00
A*	6.00	5.00	4.00	86.00	86.00	18.00

Szukanie minimalnej drogi pociągiem dla bliskich i dalekich miast

Algorytm	Pamięć, blisko	Czas, blisko	Długość trasy, blisko	Pamięć, daleko	Czas, daleko	Długość trasy, daleko
W głąb	-	-	-	-	-	-
W głąb do 5 poziomów	14.00	23.00	2990.00	-	-	-
Wszerz	103.00	31.00	1860.00	> 9999	> 3000	> 6000
Wspinanie do góry	1.00	4.00	2023.00	-	-	-
Najpierw najlepszy	7.00	4.00	2023.00	29.00	12.00	3592.00
A*	7.00	5.00	1860.00	15.00	23.00	2859.00

Zastosowanie wspomnianych tu metod szukania do poszukiwania dróg syntezy cząsteczek chemicznych znaleźć można w książce o metodach sztucznej inteligencji w chemii (Hippe 1993).

Przejdźmy teraz do zastosowania metod opartych na szukaniu w rozwiązywaniu problemów.