1.1 Przykłady programów opartych na szukaniu

Programy oparte na przeszukiwaniu przestrzeni stanów i tworzeniu drzewa możliwości należą do jednych z pierwszych prób rozwiązania zagadnień wymagających inteligencji. Wymienimy tutaj kilka wczesnych prób.

1.1.1. Teoretyk logiki

Logic Theorist (A. Newell, J.C. Shaw, H.A. Simon, 1956), był to jeden z pierwszych programów heurystycznych. Jego twórcy tak sformułowali swoje cele.

Nie jesteśmy zainteresowani metodami, które gwarantują rozwiązania lub wymagają wielu obliczeń. Chcielibyśmy raczej zrozumieć jak np. matematyk jest wstanie dowodzić twierdzeń, chociaż nie wie, kiedy zaczyna, w jaki sposób i czy w ogóle mu się to uda.

Heurystyki programu LT ograniczają przestrzeń poszukiwania rozwiązań przez odpowiednie dobranie sformułowania problemu; w ramach tej przestrzeni wykonuje się ślepe przeszukiwanie. Program dowodzi twierdzenia zawarte w *Principia Mathemathica* Whitheada i Russela dotyczące rachunku zdań. Opiera się na 5 aksjomatach:

1. $(p \vee p) \Rightarrow p$
2. $p \Rightarrow (q \vee p)$
3. $(p \vee q) \Rightarrow (q \vee p)$
4. $[p \vee (q \vee r)] \Rightarrow [q \vee (p \vee r)]\}$
5. $(p \Rightarrow q) \Rightarrow [(r \vee p)] \Rightarrow (r \vee q)]$

Typowe twierdzenia dowodzone przez LT:

2.01 $(p \Rightarrow \neg p) \Rightarrow \neg p$
2.31 $[p \vee (q \vee r)] \Rightarrow [(p \vee q) \vee r)]$
2.45 $\neg (p \vee q) \Rightarrow \neg p$

Dane, którymi można się w danym momencie posłużyć to aksjomaty i twierdzenia poprzednio udowodnione. Z 52 twierdzeń z *Principia Mathemathica* Teoretyk Logiki udowodnił 38. Program działa rozumując od tyłu, twierdzenia usiłuje redukować do aksjomatów, stosuje przy tym 3 operatory redukcji:

> Oderwanie: by pokazać X szukaj $A \Rightarrow X$ i dowiedź A.
> Łączenie w przód: by pokazać X postaci $A \Rightarrow C$ poszukaj coś (aksjomat, twierdzenie) postaci $A \Rightarrow B$ i dowiedz $B \Rightarrow C$.
> Łączenie w tył: by pokazać X postaci $A \Rightarrow C$ poszukaj coś w postaci $B \Rightarrow C$ i dowiedz $A \Rightarrow B$.

Problem: jak rozpoznać równoważność postaci przy różnych zmiennych? Jakie operatory można stosować?

> Podstawienie: Zmienną zastąpić można podstawieniem, np. $p \Rightarrow (q \vee p)$ podstawiamy $p \vee q$ zamiast p i mamy $(p \vee q) \Rightarrow [q \vee (p \vee q)]$.

> Wymiana: operator $\Rightarrow$ wymienić można zgodnie z jego definicją w rachunku zdań tj. $p \Rightarrow q$ na $\neg p \vee q$ (łatwo sprawdzić tabele wartości):

Ogólny algorytm: przeszukujemy przestrzeń stanów na ślepo, rozumując do tyłu. Test, czy dowód został już zakończony: przez porównanie z już udowodnionymi twierdzeniami i aksjomatami. Jeśli dowód nie jest jeszcze skończony do listy problemów dodajemy kolejny problem cząstkowy. Stosujemy oderwanie, łączenie wprzód i w tył aż do wyczerpania pamięci lub otwartych problemów.

Subtelności: by oszczędzić na czasie nie porównuje się po każdej transformacji ze wszystkimi twierdzeniami lub aksjomatami; wstępnie określa się podobieństwo, np. czy tyle samo argumentów.

Dodatki: powstające problemy porządkuje się w/g prostoty, odrzucając bardzo skomplikowane.

1.2.1. General Problem Solver czyli ogólny rozwiązywacz problemów

GPS (General Problem Solver) to projekt rozpoczęty przez A. Newella, J.C. Shawa, H.A. Simona w 1957 roku. Autorzy postawili sobie dwa cele:

Rozwiązywanie problemów wymagających inteligencji.
Stworzenie teorii rozwiązywania problemów przez ludzi.

Program ten rozwijany był przez około 10 lat, ostatnie prace zrezygnowały z drugiego celu. GPS był pierwszym programem ogólnym, nie związanym z wiedzą specyficzną dla danej dziedziny. Składał się z dwóch odrębnych części:

abstrakcyjnego rozwiązywacza problemów
wiedzy o zadaniu, zawartej w strukturach danych, tworzących ”środowisko problemu”.

W strukturach danych zawarta jest informacja o obiektach i dozwolonych transformacjach tych obiektów. Zadaniem GPS było przekształcić stan początkowy w końcowy, posługując się mieszaniną różnych technik: analizą **środków-celów** (means-ends analysis). Różnice pomiędzy obiektem końcowym a aktualnym klasyfikowane są według typów, działanie operatorów też według typów różnic, jakie tworzą. Program szuka w głąb stosując po jednym operatorze, gdy ocenia, że zrobiło` się trudno, cofa się. Jeśli wybrany operator nie da się zastosować GPS potrafi dokonać transformacji obiektu tak, by operator był stosowalny. Program „ukierunkowany jest na cel”, zapisując historię w grafie typu AND/OR.

Cel w GPS to najważniejsza ze struktur danych: zawiera obecną sytuację, żądaną sytuację, historię przekształceń wykonanych na obecnej sytuacji by dojść do pożądanej. Cele zawierają 3 typy działań: przekształcenie A w B, redukcja różnicy A i B przez modyfikację A, oraz zastosowanie operatora do obiektu A. Początkowe zadanie przedstawiane jest GPS w postaci transformacji z A do B.

Wybór operatorów do transformacji wymaga, by spełnione były wstępne warunki ich zastosowania. Klasyfikuje się typ różnicy obiektów A, B, porządkuje według ocenianej trudności, wybiera się z listy operatorów odpowiedniej do redukcji różnic danego typu.

Program zawiera różne heurystyki by ograniczyć proces przeszukiwania, porzucając niektóre cele przynajmniej tymczasowo:

Każdy cel powinien być prostszy niż cel wyjściowy
Nie należy powtarzać takich samych celów
Nowy obiekt (cel) nie powinien być dużo większy niż cel początkowy

Przykłady

Początkowo zastosowano GPS do tych samych problemów co Teoretyka Logiki. Używano 12 operatorów reprezentujących reguły wnioskowania, np.:

1. $A \vee B \Leftrightarrow B \vee A$
2. $A \wedge B \Leftrightarrow B \wedge A$
3. $A \vee B \Leftrightarrow \neg(\neg A \wedge \neg B)$
4. $A \Rightarrow B \Leftrightarrow \neg A \vee B$

Zdefiniowano 6 możliwych różnic, podanych tu od trudnych do łatwych:

Występowanie zmiennej tylko w jednym z wyrażeń
Występowanie zmiennej różną liczbę razy
Różnice w znaku
Różnice w użyciu $\wedge$
Różnice w położeniu składowych w wyrażeniach

Przykładowy problem: $L_1 = R \wedge (\neg P \Rightarrow Q) \Leftrightarrow L_0 = (Q \vee P) \wedge R$.
Celem jest przekształcenie problemu L_1 w L_0

GPS miał być ogólny, stosując analizę celów i środków oraz heurystyczne reguły szukania niezależne od specyficznych obszarów wiedzy. Niestety, reprezentacja obiektów i operatorów nie dała się całkiem uniezależnić od rodzaju problemów. Początkowo GPS rozwiązał tylko dwa problemy pozalogiczne. Dopiero rozszerzenie przez Newella i Ernsta (1969) zwiększyło jego możliwości. Wprowadzono uzupełniający opis problemu przy pomocy list ograniczeń, więzów, ulepszoną reprezentację operatorów i kilka innych usprawnień.

W nowej wersji program rozwiązał zadania z 11 różnych dziedzin, np. gier, całkowania symbolicznego, dowodzenia twierdzeń, ale pogorszyły się jego możliwości logiczne i przestał działać dobrze w rachunku zdań.

Główny sukces GPS nie leżał w jego użyteczności do rozwiązywania problemów, ale w lepszym zrozumieniu sposobu i trudności z rozwiązywaniem problemów. Jego twórcy porównywali zapis rozumowania, sporządzony przez uczestników eksperymentu w czasie rozwiązywania danego problemu z wydrukami otrzymanymi z GPS i doszli do wniosku, że jest to niezłe pierwsze przybliżenie do specyficznego rodzaju rozumowania wykonywanego przez ludzi. Heurystyki uznano za najważniejsze w procesie rozwiązywania problemów. Niestety, złożone problemy wymagają czegoś więcej.

1.3.1. Geometra (Herbert Gelernter, 1959, IBM New York)

Zadanie: udowadnianie twierdzeń geometrycznych na poziomie szkoły średniej. Program napisany w Fortranie. Był to pierwszy program, który radził sobie z koniunkcją podcelów.

Aksjomaty przyjęto za operatory redukujące problem, np: by pokazać przystawanie Δ pokazać, że jedna strona i 2 kąty lub 2 boki i jeden kąt są jednakowe.

Program rozumuje wstecz, tworząc drzewo podproblemów AND/OR. Wejściem jest opis problemu i zbiór współrzędnych punktów, dowodem zbiór stwierdzeń redukujących cel do trywialnego lub aksjomatu. Redukcję szukania osiągnięto posługując się reprezentacją geometryczną problemu, sprawdza się z nią poprawność celów i odrzuca szybko błędne.

1.4.1. SAINT=Symbolic Automatic INTegrator (J. Slagle, 1961, MIT)

Program SAINT napisany został w LISPie i miał na celu wykonywanie całkowania symbolicznego na poziomie pierwszego roku studiów. Problemem postawionym w pracy doktorskiej Slagla było: czy komputer może rozpoznawać struktury, występujące w wyrażeniach symbolicznych w inteligentny sposób? Było to jedno z pierwszych zastosowań reprezentacji redukcji problemóow.

Inna wczesna próba programu do algebry symbolicznej na MIT to program SIN, czyli Symbolic Integration, który w 1967 napisał Joel Moses. SAINT potrafił już rozwiązać 84 z 86 problemów z egzaminów na MIT a SIN rozwiązywał najtrudniejsze całki z tablic. Idee tych programów użyte zostały w pierwszych programach do obliczeń symbolicznych, takich jak MACSYMA.

1.5.1. Inne programy oparte na prostych algorytmach szukania

STRIPS (R. Fikes, N. Nilsson, 1971, SRI International)

Jest to program planujący ruchy robota w pokoju ze skrzynkami i pudłami. rozwiązaniem jest plan ruchów, model świata zawierał pokoje połączone drzwiami i ściany jako obiekty statyczne oraz pudła jako ruchome. Stosowano opis przy pomocy rachunku predykatów, operatorami były akcje, jakie wykonywać może robot. Operatory mają wstępne warunki stosowalności, np. przesunięcie wymaga zbliżenia się. Wynikiem działania jest zmiana modelu świata.

ABSTRIPS (E. Sacerdoti, 1974) to modyfikacja programu STRIPs przy pomocy techniki hierarchicznego planowania, dokonana by uniknąć eksplozji kombinatorycznej. Najpierw robi się szkic rozwiązania a potem planuje szczegóły.

1.2 Szachy

Pierwszy program szachowy napisał już w 1958 roku Alex Bernstein. Nie był to dobry program.

DEEP THOUGHT to program rozwijany od 1985 roku przez 4 amerykańskich studentów (T. Hsu, T. Anantharaman, M. Campbell, A. Nowatzyk) to program i sprzęt komputerowy do gry w szachy. W dziedzinie gier planszowych nie prowadzi się zbyt wielu badań ze względu na brak funduszy. Wyjątkiem są do pewnego stopnia szachy gdyż symulatory szachowe dają się sprzedać a program, który jest w stanie zwyciężyć z mistrzem świata, nadaje się dobrze do reklamy firmy i komputerów. Zarówno Intel jak i IBM zaangażował w konstrukcję komputerów szachowych i w badaniach nad rozwojem algorytmów opartych na szukaniu spore środki. Szkocki międzynarodowy mistrz szachowy ufundował nagrodę dla programu szachowego, który ogra go chociaż raz na cztery partie, ale przegrał już w 1985 roku i to wszystkie partie. Program Chess Genius w 1994 roku w kilku partiach zwyciężył mistrza świata, Gary Kasparova. Czas grania ograniczony był do 25 minut na zawodnika, ale program wykonywał się na zwykłym komputerze osobistym z procesorem Pentium. Tymczasem IBM pracował nad wyrosłym z wcześniejszego projektu ***Deep Thought*** systemem wieloprocesorowym ***Deep Blue***, który działając na 32-procesorowej stacji roboczej IBM RS6000/SP2, do której dołączono 256 wyspecjalizowanych obwodów scalonych, analizował 200-1000 milionów pozycji w ciągu sekundy! Dodatkowo program korzysta z biblioteki otwarć i końcówek, dającej mu wiedzę szachową porównywalną z wiedzą najlepszych szachistów.

Konstrukcja programów szachowych jest pewnym kompromisem pomiędzy wielkością pamięci i złożonością strategii gry a prostymi metodami szukania. *Deep Thought* używa specjalnego sprzętu (wyspecjalizowane mikroprocesory) i wykorzystuje procedury alfa-beta osiągając w skomplikowanych sytuacjach poziom przewidywania około 10 ruchów a w prostszych przypadkach nawet większy. Statyczna ocena sytuacji na planszy uwzględnia liczbę figur i ich wzajemne położenie. Istnieje prosta zależność pomiędzy liczbą ruchów, które można uwzględnić, a ilością punktów oceniających „siłę" programu. Mistrz świata ma obecnie ponad 2800 punktów. Przy ocenie do 5 ruchów w głąb program ma około 1500 punktów. Empirycznie stwierdzono, że przy uwzględnieniu dodatkowego poziomu od 5 do 10 poziomów w głąb przybywa około 200 punktów na poziom, czyli 2500 punktów przy przewidywaniu 10 ruchów. Ponieważ efektywna liczba ruchów, które należy rozważyć na każdym poziomie wynosi około 6-ściu (tj. po uwzględnieniu różnych strategii heurystycznych) przewiduje się, że kolejna generacja *Deep Blue* będzie robić plany na 14 ruchów w głąb i powinna zgromadzić znacznie więcej punktów niż mistrz świata. Jej siłę szacuje się na 3000 punktów.

Do decydującego meczu doszło na początku 1996 roku. Kasparow wygrał w nim dwie partie, dwie zremisował a jedną przegrał. W maju 1997 roku doszło do rewanżu, w którym sytuacja się odwróciła: Kasparow wygrał pierwszą partię, przegrał drugą, zremisował trzy kolejne i przegrał ostatnią. Pomimo wielu zarzutów, jakie środowisko szachowe usiłowało postawić firmie IBM - niektórzy szachiści nie mogli wprost uwierzyć, że komputer może wygrać z człowiekiem, chociaż już od dawna zostało bardzo niewielu zawodników, którzy zmierzyć się mogą z dobrymi programami szachowymi, nietrudno więc było przewidzieć, że kiedyś ten moment musiał nadejść - jest rzeczą jasną, że *Deep Blue* osiągnął poziom mistrzowski i w ciągu kilku lat mogą pojawić się programy dla komputerów osobistych, z którymi żaden człowiek nie wygra. Po raz pierwszy w tym meczu grając z maszyną Kasaprow kilka razy miał wrażenie, że stoi za nią jakaś ludzka inteligencja. W odróżnieniu od innych programów *Deep Blue* miała najwyraźniej długofalową strategię.

Ciekawa była reakcja prasy na mecz pomiędzy *Deep Blue* i Kaparowem: „Pointa jest niestety żałosna: nowy król szachów nawet nie wie, że wygrał". Inteligencja maszynowa coraz bardziej postrzegana jest jako zagrożenie dla mitu o tajemniczości i wyjątkowości procesów, leżących u podstaw „intuicji" i rozumienia człowieka, stąd próba jak największego zdeprecjonowania maszyny. Czy podobnie pisano o pierwszych samochodach, które potrafiły przegonić człowieka? One również nie wiedziały, że wygrały, i wcale się z tego nie cieszyły. Programy grające w szachy nie maja na celu zaprezentowania superumysłu ludzkiego typu lecz jedynie wykazanie, że procesy obliczeniowe pozwalają na rozwiązywanie zagadnień wymagających bardzo wysokiej i wyspecjalizowanej inteligencji.

1.3 Inne gry

Większość programów do gry w szachy nie zawiera żadnych elementów uczenia maszynowego. Artur Samuel napisał już przy końcu lat 50. samouczący się program do gry w **warcaby**. Program ten wygrał z mistrzem stanu Connecticut. Najwyższy poziom osiągnął program komputerowy Chinook napisany przez Jonathana Schaeffera

z Uniwersytetu Alberty. Po raz pierwszy mistrzostwa świata w kategorii „człowiek przeciwko maszynom" zorganizowano w Londynie w 1992. Mistrz świata w warcabach, Dr. Marion Tinsley, wygrał z Chinookiem 4-2 (przy 33 remisach). Program działał na 8-procesorowej stacji Silicon Graphics 4D/480 wyposażonej w 256 MB RAM, korzystając z bazy danych wszystkich końcówek z 1-7 figurami i prawie połową wszystkich partii z 8 figurami. Partie te zostały wcześniej przeanalizowane tak, by nie tracić czasu w trakcie rozgrywek Po raz pierwszy maszyna zdolna była zagrozić człowiekowi w mistrzowskich zawodach. Mecz zainteresował prasę i wywołał falę spekulacji na temat możliwości technologicznego rozwoju. Marion Tinsley powiedział o sobie: „Chinook zaprogramowany został przez Jonathana, ale ja zostałem zaprogramowany przez Boga". Wielu dziennikarzy wyrażało podobne przekonanie o wyższości ducha ludzkiego nad komputerami.

W roku 1994 mistrzostwa zorganizowano w Bostonie, ale Marion Tinsley wycofał się po 6 remisach z powodu choroby. Wicemistrz Don Lafferty zremisował z Chinookiem uzyskując wyniki 1-1 i remisując 18 razy. Rok później w Petal, Mississipi, przegrał 1-0 przy 31 remisach. Chinook dysponował pamięcią 512 MB. Po raz pierwszy maszyna zwyciężyła najlepszego gracza wśród ludzi.

Elementy uczenia maszynowego są bardzo ważne w programach grających w triktraka (Othello), szczególnie popularnej w Japonii. Już w latach 80. powstały bardzo dobre programy komputerowe osiągające mistrzowski poziom. Program Logistello, jeden z najlepszych w tej dziedzinie, w drugiej połowie lat 90. zwyciężył z mistrzem świata Takeshi Murakami 6 do 0. Informacje o Othello znaleźć można pod adresem:
http://www.armory.com/~iioa/.

Chińska gra *go* jest znacznie bardziej złożona niż szachy czy triktrak, pozwalając na każdym etapie na o wiele więcej możliwych ruchów niż szachy. Programy komputerowe do gry w *go* nie osiągają na razie wysokiego poziomu. Jedną z przyczyn jest brak poważniejszych prac nad rozwojem oprogramowania tego typu. Ważniejszą przyczyną może być mała przydatność klasycznych algorytmów opartych na szukaniu w sytuacji, w której mamy średnio 260 posunięć na każdym kroku. Konieczne jest wówczas najpierw rozpoznanie sytuacji na planszy technikami rozpoznawania wzorców, ocenienie tej sytuacji i dopiero potem szukanie odpowiedniego ruchu. Jest to zagadnienie znacznie trudniejsze niż samo szukanie. Nie ma jednak wątpliwości, że i w tej dziedzinie programy przewyższą ludzkie możliwości.

Inne zagadki, rozwiązywalne za pomocą programów do automatycznego dowodzenia twierdzeń:

Zagadka dotycząca zawodów

Mamy cztery osoby: Ramonę, Teresę, Stefana i Piotra.
Mają w sumie osiem miejsc pracy, każde z nich pracuje w dokładnie dwóch miejscach.
Wykonywane zawody to: kucharz, strażnik, pielęgniarka, urzędnik, policjant, nauczyciel, aktor i bokser.
Zawód pielęgniarki wykonuje mężczyzna.
Mężem kucharki jest urzędnik.
Ramona nie jest bokserem.
Piotr ukończył nie więcej niż 8 klas.
Ramona, kucharka i policjant poszli razem na golfa.

Pytanie: kto ma jaki zawód?
Rozwiązanie nie jest trudne i potrafią je znaleźć dzieci w gimnazjum. Z punktu widzenia programu rozumującego jest to jednak interesujące zagadnienie, bo wymaga uwzględnienia informacji podanej zarówno w jawny, jak i ukryty sposób. Jest to jeden z przykładów rozwiązywanych przez program rozumujący OTTER.
Rozwiązanie: kucharz i bokser – Teresa; strażnik i nauczyciel – Ramona, pielęgniarka i policjant – Stefan, urzędnik i aktor – Piotr.

Wracając do problemu domin i szachownicy – chociaż jest to problem trudny, współczesne programy rozumujące potrafią sobie z nim poradzić. Problem ten postawił w połowie lat 60. John McCarthy jako przykład zagadnienia, z którym komputer sobie nie poradzi. Człowiek radzi sobie z tym problemem ponieważ potrafi sobie wyobrazić szachownicę. Rozwiązanie polega na zauważeniu, że jeśli liczba czarnych pól nie jest równa liczbie pól białych to nie da się pokryć dominami całej szachownicy. Jeśli jednak usunąć pola tego samego koloru nie da się już tego argumentu użyć, jednakże programy rozumujące nadal sobie z tym problemem radzą.

Prawdomówni i kłamcy.

Na pewnej wyspie połowa mieszkańców zawsze mówi prawdę a połowa zawsze kłamie. Lądujesz na tej wyspie

i spotykasz trzech mieszkańców, A, B i C. Pytasz A: czy mówisz zawsze prawdę czy kłamiesz? Mruczy coś pod nosem i nie możesz nic zrozumieć. Pytasz się B co powiedział A i słyszysz „ A powiedział, że jest kłamcą". Wtrąca się C: „nie wierz B, on zawsze kłamie". Co możesz powiedzieć o A, B i C?

O A nic nie można powiedzieć, B jest kłamcą a C mówi prawdę.

1.4 Szukanie a ludzkie myślenie

W jaki sposób człowiek gra w szachy lub inne gry planszowe? Jak przebiega proces szukania? Często ludzie wyobrażają sobie, że jest to proces „intuicyjny", nie oparty na mechanizmach szukania. tymczasem grafy szukania tworzone na podstawie relacji ludzi komentujących swoje zachowania w czasie gry pokazują, że wykonują oni ograniczone poszukiwanie w głąb oparte na heurystykach związanych z doświadczeniem i zdolnością oceny przewagi danej sytuacji. Taki sposób szukania nazwano „zmodyfikowanym progresywnym pogłębianiem". Ogólna jakościowa teoria zachowania się człowieka w takich przypadkach wygląda następująco. Przestrzeń problemów złożona jest z sytuacji początkowej i sytuacji, do których potencjalnie chcemy doprowadzić (sytuacji przewagi). Operatorami są dozwolone ruchy prowadzące do nowych sytuacji. Uwaga skupiona jest na sytuacjach trudnych lub uznawanych za interesujące. Ze względu na ograniczenia pamięci nie da się zastosować wielu strategii przeszukiwania, jedynie w przypadkach lokalnych celów stosowane są bardziej systematyczne strategie przeszukiwania przestrzeni możliwych rozwiązań, w pozostałych przypadkach stosowania jest strategia progresywnie pogłębianego przeszukiwania. Teoria rozwiązywania problemów przez człowieka musi uwzględniać ograniczenia umysłu. Sztuczna inteligencja korzysta tu z badań psychologii poznawczej.

Chase i Simon (Newell 1990) przeprowadzili w 1973 roku badania nad szachistami. Jeśli pokazać fragment środkowego etapu rozgrywanej partii szachistom na różnym poziomie to okazuje się, iż istnieje wyraźna korelacja pomiędzy poziomem gry szachisty a jego zdolnością do odtworzenia położenia figur po 5-sekundowej ekspozycji szachownicy. Mistrz szachowy ułoży średnio 23 figury z 25 na pokazanych pozycjach a nowicjusz tylko 3 lub 4 figury. Jest to związane z pojemnością pamięci roboczej człowieka. Taki test spostrzeżeniowy jest najszybszym testem dla oceny umiejętności szachowych. Jeśli natomiast figury ułożone są przypadkowo to dobrzy szachiści robią podobną liczbę błędów jak początkujący, skarżąc się przy tym na „chaotyczność" położeń figur. Świadczy to o używaniu przez ich mózgi mechanizmu „porcjowania", czyli rozbicia całej struktury na znaczące fragmenty i pamiętania tych fragmentów. Przypomina to nieco proces kompilacji przyrostowej. Gra w szachy oparta jest na mechanizmach pamięci złożonych struktur pozwalających na stosowanie skomplikowanych funkcji oceny przewagi na szachownicy. We wszystkich dziedzinach wiedzy wydają się one przebiegać podobnie, eksperci pamiętają większe całości, co pozwala im lepiej analizować struktury. Simon i Gilmartin ocenili liczbę pamiętanych „prototypowych" struktur na około 50.000. Nie tylko mistrz szachowy potrafi szybko rozpoznać takie struktury, ale pamięta również plany działania odpowiednie w danej sytuacji, np. jeśli jakiś obszar przeciwnika uznany zostanie za słaby od razu skupia się nad rozwijaniem strategii ataku w tym kierunku.

Ekspert pamięta rozwiązania, nowicjusz musi je wymyślać od początku ucząc się na błędach. W obu przypadkach liczba rozważanych ruchów jest podobna. Eksperci nie tylko lepiej rozpoznają, ale też lepiej pamiętają pokazywane im sytuacje w eksperymentach, w których 30-sekundowa przerwa pomiędzy pokazaniem szachownicy a pytaniem wypełniona jest rozwiązywaniem innych problemów. W testach pamięci dotyczących słów czy symboli nie widać jednak różnicy. Lepsza pamięć eksperta ogranicza się więc do domeny jego ekspertyzy. Badano też sposób, w jaki szachiści kopiują położenia figur z jednej szachownicy na drugą. Eksperci układają sensowne fragmenty, tworząc bloczki (zgrupowania) po jednym rzucie oka. Ich bloczki są większe i potrafią zapamiętać ich więcej niż nowicjusze. System ekspertowy SOAR, zaproponowany przez Newella i Simona, wykorzystuje takie mechanizm porcjowania. Mechanizm porcjowania wykorzystywane jest w mnemotechnice, czyli metodach trenowania pamięci pozwalających na zapamiętanie długich list liczb czy wyrazów. Człowiek może zapamiętać 7-9 cyfr, ale po odpowiednim treningu może to być nawet 100 cyfr. Specjalizacja w przypominaniu sobie ciągów cyfr nie ulega jednak automatycznej generalizacji na ciągi liter czy symboli. Zdolności kognitywne mogą być ściśle związane z sytuacją, np. (Anderson 1990) Brazylijskie dzieci sprzedające owoce na ulicy potrafiły poprawnie obliczyć cenę 5 owoców po 35 cruzeiros w 98% przypadków lecz jedynie 37% z nich podało poprawnie wynik mnożenia 5*35 w szkolnym teście. Wiara w zbawczy wpływ nauczania łaciny w szkole (ma nauczać logicznego myślenia) nie znajduje więc potwierdzenia w badaniach kognitywistów.

Badania w zakresie kognitywistyki (nauk o poznaniu) przedstawione powyżej nie wpływają na razie na konstrukcję programów osiągających mistrzowski poziom w grze w szachy. Nie byłoby to wygodne gdyż nie mają one możliwości zdobycia tak wielkiego doświadczenia jak prawdziwy szachista, który w ciągu swojego życia rozgrywa ogromną liczbę partii. Brakuje jeszcze dostatecznie dużych bibliotek gier pozwalających na trenowa-

nie systemu. Drugi argument ma bardziej fundamentalne znaczenie. Człowiek nie potrafi utrzymać w swojej krótkotrwałej pamięci więcej niż kilka danych, na planszy ocenia tylko jedną sytuację. Komputer nie ma takich ograniczeń, każdy z procesorów może rozważać sytuację odmienną.

Badania nad „intuicją" pokazują, że ulegamy często złudzeniom kognitywnym, oceniając zupełnie błędnie sytuację. Dotyczy to zwłaszcza ocen prawdopodobieństw, jednak nawet zdolności do wyciągania prostych wniosków logicznych opierają się na pamiętanych schematach. Na przykład typowy sylogizm:

Każdy człowiek jest ssakiem.
Sokrates jest człowiekiem.
Wniosek: Sokrates jest ssakiem.

Nie sprawia nam trudności, ale nieco trudniej jest wnioskować gdy pojawia się zaprzeczenie:

Żaden rolnik nie jest żeglarzem.
Wszyscy Rurytanie to rolnicy.
Wniosek: Żaden Rurytanin nie jest żeglarzem.

Prawie nikt nie wyciąga poprawnego wniosku z następujących przesłanek:

Wszyscy członkowie gabinetu to złodzieje.
Żaden muzyk nie jest członkiem gabinetu.

Nawet najbardziej inteligentni ludzie mają trudności z wyciągnięciem wniosków w tym przypadku, często twierdząc, że nic się z tego nie da wywnioskować. Tymczasem jest tu jeden logiczny wniosek: są złodzieje nie będący muzykami, lub: nie każdy złodziej jest muzykiem. Nie mamy doświadczenia z rozpoznawaniem tego typu logicznych wzorców i dlatego ich odkrycie jest prawie niemożliwe. Myślenie opiera się więc na procesach szukania wykorzystujących utarte schematy wnioskowania i ogromną pamięć, pozwalająca na ocenę sytuacji

W procesach myślenia odkryto przykłady złudzeń kognitywnych, w pewnym sensie analogicznych do złudzeń optycznych. Jednym z najsłynniejszych jest Monty Hall Paradox, przedstawiany w postaci gry.

Mamy 3 kubki. Wychodzisz z pokoju, ja pod jednym z kubków ukrywam złotą monetę. Wracasz i wybierasz jeden z kubków. Ja – wiedząc, pod którym jest moneta – odkrywam jeden z pustych kubków. Masz teraz szansę zmienić swoją decyzję i pozostać przy już wybranym kubku lub wybrać pozostały.

Zabawę powtarzamy wielokrotnie. Czy najlepszą strategią jest:

1. zawsze trzymanie się pierwotnego wyboru,
2. zawsze zmiana,
3. czy przypadkowy wybór?

Publikacja tego problemu w *Scientific American* a potem w pismach dotyczących statystyki wywołała lawinę listów. Bardzo trudno jest zrozumieć rozwiązanie, nawet jeśli się pozna argumenty, świadczące na korzyść jednej ze strategii. Można oczywiście zrobić samemu eksperyment, ale wydaje się nam niesamowite, jak wiedza o braku może wpłynąć na zmianę prawdopodobieństwa. Wydaje się, że jest tu jakaś głębsza myśl, być może przydatna w zrozumieniu paradoksów mechaniki kwantowej?

Oczywiście najlepszą strategią jest ...

Literatura

Bolc L., Cytowski J., *Metody przeszukiwania heurystycznego, T.2*, PWN Warszawa 1991

Hippe Z, *Zastosowanie metod sztucznej inteligencji w chemii*, Wydawnictwo Naukowe PWN, Warszawa 1993

Newell A, *The unified theories of cognition*, Harvard University Press 1990