**46. Zastosowanie sztucznej inteligencji w systemach informatycznych**

Pytanie 23. + (ew. 42. ) + parę słów o systemach ekspertowych

|  |  |
| --- | --- |
| **Obszar zastosowań** | **Przykłady zastosowań decyzyjnych** |
| Transport | * rozwiązanie **problemu komiwojażera** w firmie transportowej / spedycyjnej / dostawczej / **cateringowej**, |
| Sprzedaż | • symulacja zapasów  **• problem transportowy w hurtowni**  • określenie pierwszego nakładu książki  **• rozmieszczenie punktów sprzedaży**  • samouczący się model konsumenta |
| Organizacja | • **harmonogramowanie pracy**  • sporządzanie tras pociągów towarowych  • sporządzanie tras pojazdów kołowych  • **kolejność lądowań samolotów**  • kolejność jazdy pociągów po linii jednotorowej  • **projektowanie sieci telekomunikacyjnych**  • **optymalizacja sieci komputerowej**  • układanie podziału godzin w szkole  • planowanie systemu dostaw wody |
| Finanse | • szacowanie ryzyka ubezpieczeniowego  • prognozowanie bankructwa  • **określanie zdolności kredytowej**  • przewidywanie opóźnień w płatnościach  • analiza cyklów czasowych  • symulacja rynku akcji  • **optymalizacja portfela akcji**  • model gracza na rynku akcji  • modelowanie rynków finansowych  • **wykrywanie oszustw finansowych**  • alokacja nakładów dla różnych programów  społecznych |
| Produkcja | • **równoważenie linii produkcyjnej**  • określanie wielkości partii  • sekwencyjność produkcji  • proces planowania  • magazynowanie  **• zarządzanie obciążeniami**  **• optymalizacja działania reaktorów jądrowych**  • programowanie symulatorów lotniczych |

# Zastosowanie sztucznej inteligencji w systemach ekonomicznych i w podejmowaniu decyzji

Sebastian Wyrwał

O sztucznej inteligencji mówi się dużo. Praktycznie każda osoba studiująca informatykę, czy kierunek pokrewny, słyszała o algorytmach genetycznych, czy systemach ekspertowych. Niektórzy jednak wciąż traktują zagadnienia związane ze sztuczną inteligencją z przysłowiowym przymrużeniem oka. Trzeba powiedzieć, że sztuczna inteligencja jest dziedziną obejmującą wiele zagadnień, również tych niepowiązanych bezpośrednio z programowaniem, czy projektowaniem systemów informatycznych. Poniżej czytelnik może znaleźć kilka zagadnień należących, bądź związanych ze sztuczną inteligencją:

Algorytmy Genetyczne,

Programowanie Genetyczne,

Systemy ekspertowe,

Automatyczne pozyskiwanie wiedzy,

Inżynieria wiedzy.

**Metody sztucznej inteligencji znajdują zastosowanie:**

* Wmedycynie,
* Do rozpoznawania obrazów,
* Jako metoda optymalizacji,
* Do wspomagania decyzji,
* Do pozyskiwania wiedzy z baz danych,
* W zarządzaniu,
* Do projektowania układów elektronicznych,
* Do obróbki obrazu,
* W ekonomii i ekonometrii,
* W grach logicznych,
* Do klasyfikacji.

Przydatność algorytmów genetycznych jako metody optymalizacji nie jest kwestionowana np. w przypadku funkcji, których minimum nie można wyznaczyć tradycyjnymi metodami matematycznymi.

## Systemy ekonomiczne, podejście tradycyjne i z zastosowaniem sztucznej inteligencji

Zastosowanie sztucznej inteligencji w systemach ekonomicznych i wspomagających decyzje kredytowe, bądź inwestycyjne jest dyskutowane, aczkolwiek nie negowane. Niespełna rok temu odbywała się prezentacja pewnego systemu służącego do oceny ryzyka przy udzielaniu kredytów. Pomimo tego, że system był bardzo rozbudowany, to do przewidywania wartości (lub mówiąc ściślej rozkładu) pewnych wskaźników ekonomicznych, na podstawie ich wartości historycznych używano prostych modeli matematycznych. Dlaczego? Na takie pytanie padła odpowiedź, że zastosowanie sztucznej inteligencji wymaga czasu i nabrania do niej większego zaufania ze strony nieufnych klientów jakimi są banki. Prawdopodobnie jedyną rzeczą z zakresu sztucznej inteligencji możliwą do zastosowania w tym systemie (choć do nieco innych celów) były sieci neuronowe.

Modele matematyczne polegają - w pewnym uproszczeniu - na przyjęciu pewnej zależności funkcyjnej. Ich zaletą jest to, że wynik otrzymujemy natychmiast, co jest istotne w systemach działających online. Wadą jest to, że należy przyjąć pewien model, czyli innymi słowy, przyjąć pewną zależność funkcyjną. Systemy ekonomiczne - czyli pełniące funkcję doradczą dla inwestorów giełdowych - można roboczo podzielić na dwie grupy:

* systemy wspomagające wybór akcji, jakie powinny znaleźć się w portfelu.
* Systemy do znajdowania zależności pomiędzy wskaźnikami giełdowymi i przewidywania ich wartości w przyszłości.

Trywializując można by powiedzieć, że giełda jest miejscem gdzie można - posiadając szczęście lub wiedzę - w łatwy sposób osiągnąć duże dochody. Trzeba tylko wiedzieć jakie będą ceny akcji lub przynajmniej znać tendencje zmian tych cen. Metod analizy jest wiele. Istnieją metody wykorzystujące modele matematyczne lub doświadczenie analityków. Niektórzy twierdzą, że ciągi notowań giełdowych są nieprzewidywalne... Przeszukując Internet można znaleźć wiele systemów wspomagających graczy giełdowych. Większość tych systemów jest dość droga, a ich cena, rzędu kilkuset dolarów, może odstraszać. W każdym wypadku trudne, o ile w ogóle możliwe, jest sprawdzenie jak taki system w rzeczywistości działa, tzn. jaką metodę stosuje.

Załóżmy, że ceny akcji pewnej hipotetycznej spółki Y były następujące:

Przyjmijmy, że mamy dzisiaj 11.03.2000, a my jesteśmy inwestorami posiadającymi kapitał 10000 zł. Stoimy przed dylematem jak zainwestować nasz kapitał. Notowania spółki wykazują tendencje zwyżkową. Jeżeli za posiadany kapitał kupimy 96 akcji spółki, a cena akcji w następnym notowaniu wyniesie 110 zł, to zarobimy nieco ponad 568 zł. (abstrahujemy od podatków itp.). Trywialne jest stwierdzenie, że zarobienie 6% kapitału w ciągu dwóch dni, praktycznie bez pracy, jest bardzo dobrym interesem. Oczywiście nie wiemy jakie notowania w praktyce będzie miała spółka Y 13 marca 2000.

Można założyć, że notowanie spółki Y, nazwijmy je, y zależy w pewien sposób od pewnego wektora X, czyli bardziej formalnie:

y (t) =f (X)

gdzie:

X=[x1,x2,...,xn] jest wektorem

pewnych parametrów

y (t) jest ceną akcji w notowaniu t

Celowo nie uzależniono wartości wektora X od czasu, pozostawiając to zagadnienie na razie otwarte.

Problem polega jednak na tym, że nie znamy ani parametrów wektora X ani zależności funkcyjnej f. Rozkład notowań w czasie to akcjogram. Dla naszej hipotetycznej spółki jest on pokazany na rysunku 1. Jeżeli postać funkcji f i wektora X była by znana np.

y=2.13\*WIG+6

gdzie:

y jest notowaniem spółki Y w notowaniu t

WIG jest to Warszawski Indeks Giełdowy w notowaniu t

Czyli bardziej formalnie:

f (x) =2.13\*x1+6

X=[WIG]

byłby to klasyczny model matematyczny. Funkcja f w tym wypadku jest dobrze znaną ze szkoły funkcją liniową,

y=ax+b

która jest najprostszym, ale stosowanym w praktyce w systemach, modelem matematycznym. Przy dzisiejszych możliwościach technologicznych system może przechowywać bardzo wielkie ilości różnych danych. Dane te mogą być wykorzystane do budowy i testowania modelu. Model musi być jednak zdefiniowany przez człowieka. System taki dobierze współczynniki tego modelu, w naszym wypadku będzie to a oraz b, ale nie wymyśli za nas postaci zależności funkcyjnej.

## Programowanie genetyczne (PG)

Sztuczna inteligencja przychodzi z pomocą, kiedy nie da się zaproponować żadnego modelu matematycznego. Model może być jednak zbudowany przy pomocy metod sztucznej inteligencji bez udziału człowieka. Posiadamy pewne dane, których część (podzbiór) może być składowymi naszego wektora X. Należy podkreślić, że dużo łatwiej jest w praktyce zaproponować takie dane (niezbędne i tak w modelu matematycznym) niż zaproponować zależność funkcyjną. Można ponadto zaproponować bardzo dużą ilość tych danych i pozwolić aby system sobie wybrał z jakich będzie korzystał, budując zależność funkcyjną. Metodą, która może być tu zastosowana nazywa się Programowanie Genetyczne (PG). Programowanie Genetyczne jest metodą ewolucyjną. Oznacza to, że w pamięci maszyny hoduje się populacje osobników, które odpowiadają funkcjom-modelom. W trakcie działania systemu powstają nowe populacje. Jeżeli przetwarzanie wykonywane jest poprawnie, to wysoce prawdopodobne jest to, że w następnych populacjach będzie występować grupa coraz to lepszych osobników-modeli.

Można to przyrównać do ewolucji opisanej przez Darwina, która zakłada, że w trakcie ewolucji wydadzą potomstwo osobniki najbardziej przystosowane do warunków zewnętrznych. W programowaniu genetycznym wybiera się grupę najlepszych osobników z populacji, które wydadzą potomstwo. Idea ta jest szeroko stosowana w algorytmach genetycznych. Tam jednak osobniki zazwyczaj reprezentują wartości funkcji, lub jej argumentów, a nie samą funkcję. W programowaniu genetycznym z punktu widzenia systemów ekonomicznych osobnik nie reprezentuje konkretnej wartości ale postać funkcji.

Drzewa oraz ich wartościowanie

Aby przybliżyć idee PG należy przypomnieć co to jest drzewo. Drzewo jest grafem nie zawierającym cykli, którego jeden wierzchołek jest wyróżniony i nazywany jest korzeniem. Na rysunku korzeń to czarna elipsa. Elipsy narysowane linią kropkowaną to liście.

Fragment P to prawe poddrzewo drzewa, fragment L to poddrzewo lewe. Poddrzewo jest również drzewem. P1 jest zatem prawym poddrzewem drzewa L. Drzewo można zatem określić jako korzeń, mający (zależną od drzewa) określoną liczbę poddrzew (być może zerową - pojedynczy węzeł to również drzewo). Drzewo jest zatem strukturą rekurencyjną. Drzewa są bardzo często stosowaną w informatyce strukturą danych. Poddrzewo P składa się z korzenia i dwóch liści, korzeń tego drzewa może reprezentować funkcję, a liście - argumenty tej funkcji, tak jak na rysunku 3. Mamy w tym wypadku wyrażenie a+b, nic nie szkodzi aby wyrażenie to potraktować jak funkcję dwuargumentową w postaci:

+ (a,b) =a+b

Dodawanie jest oczywiście funkcją dwuargumentową, a to czy używamy notacji a+b, czy + (a,b) nie ma w zasadzie znaczenia. Ważne jest to, że drzewo odpowiada pewnemu wyrażeniu, które z kolei odpowiada funkcji. Za pomocą drzew możemy reprezentować dowolne funkcje. Dany węzeł ma tyle dzieci ile argumentów ma funkcja, którą on reprezentuje. Liście drzewa (na rysunku 2 węzły rysowane linią kropkowaną) reprezentują argumenty funkcji (lub stałe) reprezentowanej przez całe drzewo.

Użycie drzew w PG pozwala na łatwą reprezentację wyrażeń - funkcji, przy równoczesnej łatwości ich wartościowania i wykonywania na osobnikach operacji, mających na celu wyprodukowanie osobników wchodzących w skład populacji potomnej.

Wartościowanie ma na celu przypisanie osobnikowi-drzewu liczby określającej jego stopień przystosowania. Wartościowanie odbywa się rekurencyjnie, tzn. zakładając, że węzeł, który właśnie wartościujemy reprezentuje funkcję +, to wartościowanie tego węzła polega na obliczeniu sumy wartościowań jego dwóch dzieci. A więc jeśli dany węzeł reprezentuje stałą, to wartościowaniem tego węzła jest po prostu wartość tej stałej. Jeżeli węzeł reprezentuje funkcję, to wartościowanie polega na wartościowaniu poddrzew reprezentujących argumenty tej funkcji, a następnie obliczeniu wartości funkcji dla tak obliczonych argumentów. Przyjmjmy że D jest całym drzewem, L jest lewym poddrzewem D, P jest prawym poddrzewem D, a f jest funkcją (dwuargumentową) zawartą w korzeniu D. Wartościowanie drzewa D wygląda następująco: Var (D) =f (Var (L), Var (P))

Wartościowanie poddrzew przeprowadzamy w ten sam sposób.

Przykład:

Dla prostego drzewa z rysunku 3 wartościowanie dla a=1 i b=2 przebiegać będzie następująco:

Var (D) =+ (Var (a), Var (b))

czyli:

Var (D) =Var (a) +Var (b) =a+b

dla powyższych wartości a oraz b

mamy:

Var (D) =a+b=1+2=3

Dla drzewa z rysunku 4 i następujących wartości a=1, b=2, c= P/2 mamy:

Var (D) =+ (Var (a), Var (P)) =a+Var (P)

Var (P) =\* (b, Var (P1)) =b\*Var (P1)

Var (P1) =sin (Var (c)) =sin (c)

Składając wyrażenia "od dołu" w jedno otrzymujemy:

Var (P) =b\*sin (c)

Var (D) =a+b\*sin (c)

Przyjmując wartości jak powyżej otrzymamy:

Var (D) =1+2\*sin ( P/2) =3

gdzie:

D całe drzewo

P prawe poddrzewo drzewa D

P1 prawe poddrzewo drzewa P

Wartościowanie drzewa odpowiada zatem obliczeniu wartości funkcji reprezentowanej przez całe drzewo, dla danych wartości argumentów tej funkcji.

Przykład:

Jeżeli drzewo reprezentuje funkcję

f (a,b,c) =a+b\*c, wtedy wartościowanie dla a=1, b=2, c=3 odpowiada obliczeniu wartości funkcji f (1,2,3) =7.

Wartościowanie drzew jest niezbędne do określenia przystosowania osobnika (reprezentowanego przez to drzewo). Na podstawie przystosowania poszczególnych osobników w populacji wybiera się osobniki, które będą miały potomstwo.

Dlaczego wartościujemy - czyli o uczeniu

W wielu metodach sztucznej inteligencji, w tym w PG, potrzebne jest uczenie. Załóżmy, że chce się przewidywać wartości akcji spółki Y, uzależniając cenę akcji wyłącznie od jednej zmiennej generowanej przez system. Niech zmienna ta będzie umownym numerem notowania. Dysponujemy ciągiem 10 poprzednich notowań, tak jak pokazano w tabeli 1. Weźmy przykładową populację drzew w systemie (tutaj zakładamy populację 5-ciu drzew, ale w praktyce przetwarza się populacje złożone z dziesiątek tysięcy drzew), które odpowiadają funkcjom przedstawionym w tabeli. Aby określić przystosowanie poszczególnych osobników reprezentowanych przez drzewa należy dokonać wartościowania tych drzew dla wszystkich punktów uczących.

Tabela 2 Drzewa w przykładowej populacji i odpowiadające im funkcje

DRZEWO

Funkcja

D1 f(x)=101

D2 f(x)=100+0,1\*x

D3 f(x)=1

D4 f(x)=100+(x-4,5)2

D5

f(x) =100+x2

Dla wszystkich drzew z populacji dla każdego punktu uczącego (notowania) należy obliczyć różnicę pomiędzy wartościowaniem w tym punkcie a rzeczywistą wartością notowania akcji w tym punkcie tj. błąd, który jest określony następująco.

Bp=|Wp-Varp (D)|

gdzie:

Bp błąd w punkcie uczącym p,

Wp rzeczywista wartość ceny akcji w punkcie p,

Varp (D) wartościowanie drzewa

D w punkcie p,

p punkt uczący, czyli notowanie.

Na rysunku 5 przedstawiono błędy dla punktów uczących (kolejnych notowań) dla drzew z populacji; widać, że najmniejsze błędy są dla drzew D1 i D2, nieco większe błędy są dla drzewa D4. Biorąc pod uwagę sumy błędów dla wszystkich punktów uczących możemy uszeregować drzewa wg rosnących błędów:

D1, D2, D4, D5, D3 Można ogólnie powiedzieć, że jest bardzo mało prawdopodobne aby drzewo D3 miało potomka, jak również jest wysoce prawdopodobne, aby drzewa D1 i D2 miały potomków, aczkolwiek istnieją różne metody wyboru drzew do reprodukcji. Niezbędne jest zawsze jednak wartościowanie drzew. Przystosowanie drzew formalnie określa funkcja przystosowania zwana funkcją fitness. Często jest to suma wartości błędów dla wszystkich punktów uczących.

Więcej o PG, czyli jak to działa

Budując lub konfigurując system określamy jak, a raczej z czego będą budowane drzewa. W programowaniu genetycznym określa się zbiór funkcji F oraz zbiór symboli terminalnych T. Funkcje ze zbioru F będą węzłami w budowanych drzewach, a symbole terminalne ze zbioru T, odpowiadające argumentom funkcji reprezentowanej przez całe drzewo lub stałym, będą liśćmi drzew.

Przykład:

Jeżeli w systemie zbory F i T będą określone następująco:

F={+, \*, sin}

T={a, b, c}

możliwe jest wygenerowanie drzewa jak na rysunku 4. Nic oczywiście nie stoi na przeszkodzie aby zbiory te zawierały więcej funkcji i symboli terminalnych tzn.:

F={+, -, \*, sin, cos, log}

T={a, b, c, d, e, f, g}.

gdzie:

+ oznacza dwuargumentową funkcję, która zwraca sumę swoich argumentów

- oznacza dwuargumentową funkcję, która zwraca różnicę swoich argumentów

\* oznacza dwuargumentową funkcję, która zwraca iloczyn swoich argumentów

sin oznacza jednoargumentową funkcję sinus

cos oznacza jednoargumentową funkcję cosinus

log oznacza jednoargumentową funkcję logarytm dziesiętny

a,b,c są argumentami funkcji reprezentowanej przez całe drzewo

Siła programowania genetycznego polega na tym, że nawet z niezbyt licznych zbiorów F oraz T wygenerować można bardzo dużo różnych drzew. Liczba możliwych do wygenerowania drzew wzrasta bardzo szybko wraz ze wzrostem liczności zbiorów F i T. Przy definiowaniu funkcji należy zachować ostrożność. Jak wiadomo, w matematyce, argumentem funkcji logarytm może być tylko liczba dodatnia. Wartość logarytmu dla liczby zero lub mniejszej od zera jest nieokreślona. W PG należy nieco "ulepszyć" definicję funkcji logarytm w stosunku do tej znanej z matematyki.

Nasz logarytm musi "akceptować" także wartość 0 i wartości mniejsze od zera. Można to zapewnić definiując funkcję logarytm tak aby dla argumentu równego 0 zwracała 0, a dla innych logarytm wartości bezwzględnej argumentu. Postępowanie takie ma na celu zapewnienie domkniętości zbioru funkcji F. Domkniętość polega na tym, że każda funkcja z F akceptuje dowolną wartość uzyskaną w wyniku wartościowania dowolnego drzewa zbudowanego z węzłów określonych w zbiorach F i T. Ponadto zbiór F musi posiadać własność dostateczności, która zapewnia, że rozwiązanie może być wyrażone za pomocą funkcji z F. Teoretyczna własność domkniętości niestety nie wystarcza. W praktyce rozwiązanie musi być łatwe do wyrażenia przy pomocy funkcji ze zbioru F.

Przykład:

Jeżeli F={\*, /} oraz T={a,b} to nie jest możliwe wygenerowanie drzewa odpowiadającego wyrażeniu a+b.

Jeżeli F={+,\*} oraz T={a,b} to wygenerowanie drzewa odpowiadającego wyrażeniu (a+b)2 jest możliwe ale mniej prawdopodobne niż w przypadku kiedy F={+,\*,pow}, jeżeli pow oznacza jednoargumentową funkcję zwracającą wartość argumentu. Można to sobie wyobrazić rozrysowując wyrażenie (a+b)2 w postaci drzewa (zapisane oczywiście jako (a+b) \* (a+b)) dla F={+,\*}, a następnie dla F={+, \*, pow}.

Teoretycznie można powiedzieć, że im bardziej "bogaty" jest zbiór funkcji tym lepiej. W trakcie przeprowadzonych eksperymentów okazało się jednak, że obecność niektórych funkcji w zbiorze F wyraźnie pogarsza wyniki.

PG posiada bardzo wiele różnych parametrów konfiguracyjnych mających wpływ na uzyskiwane wyniki. Poza określeniem zbiorów F oraz T, bardzo istotny jest rozmiar populacji, oraz określenie jak długo będzie wykonywane przetwarzanie, czyli mówiąc bardziej poprawnie, jakie jest kryterium zakończenia przetwarzania.

Początek przetwarzania, czyli wygenerowanie generacji początkowej

Przetwarzanie w PG rozpoczyna się wygenerowaniem generacji początkowej. Drzewa generacji początkowej generowane są automatycznie przez system. Zasadniczo istnieją trzy metody generowania drzew populacji początkowej:

full - metoda ta polega na generowaniu drzew o jednakowej, założonej z góry wysokości. Dopóki wysokość drzewa jest mniejsza od założonej bierze się symbole wyłącznie za zbioru F, jeżeli wysokość drzewa jest równa założonej bierze się wyłącznie symbole ze zbioru T.

grow - metoda ta polega na generowaniu drzew o wysokości nie większej niż założona. Dopóki wysokość drzewa jest mniejsza niż założona bierze się symbole zarówno ze zbioru F oraz T. Jeżeli wysokość drzewa jest równa założonej bierze się symbole ze zbioru T.

ramped half and half - polega na zastosowaniu obu powyższych metod, każdej z prawdopodobieństwem 0.5.

Proces generowania drzewa rozpoczyna się od wybrania losowej funkcji ze zbioru F. Liczba poddrzew drzewa, którego korzeń reprezentuje tę funkcję jest równa liczbie argumentów tej funkcji. Następnie proces jest powtarzany rekurencyjnie (czyli dla każdego poddrzewa itd.) zgodnie z jedną z powyższych metod.

Właściwe przetwarzanie, czyli ewolucja

Po wygenerowaniu przez system populacji początkowej rozpoczyna się właściwe przetwarzanie. Trzeba pamiętać, że w PG przekształceniu podlega w większym stopniu cała populacja niż pojedynczy osobnik. Operacja reprodukcji polega na wybraniu z generacji osobnika, który zostanie następnie skopiowany bez żadnych zmian do generacji potomnej i poddany działaniu operatorów genetycznych. Do wybierania osobników do operacji reprodukcji używa się różnych metod selekcji. Metody selekcji opierają się na funkcji przystosowania osobnika. Wyróżniamy następujące metody selekcji:

proporcjonalna - prawdopodobieństwo reprodukcji danego osobnika jest równe stosunkowi przystosowania tego osobnika do sumy przystosowań wszystkich osobników w populacji.

rankingowa - polega na posortowaniu osobników wg wartości funkcji przystosowania, a następnie wyborze najlepszego.

turniejowa - polega na losowym wybraniu pewnej, określonej liczby osobników, a następnie wybraniu spośród nich najlepszego.

Najważniejszym operatorem genetycznym w PG jest krzyżowanie. Aby wyprodukować potomstwo potrzeba dwojga rodziców. Obydwoje rodzice są (najczęściej) wybierani wg metod selekcji omówionych powyżej. Produkcja potomstwa polega na wzięciu części z jednego rodzica i części z drugiego. U każdego z rodziców wybiera się losowo punkt i wymienia się poddrzewa wychodzące z tych punktów. W ten sposób powstaje dwóch potomków. Operator mutacji polegający na wprowadzeniu losowych zmian w strukturze drzewa ma w PG małe zastosowanie. Przetwarzanie jest oczywiście tak prowadzone, aby wielkość populacji w poszczególnych generacjach nie ulegała zmianie. Można przetwarzać przez zadaną z góry liczbę generacji (np. 20, 40, lub 100), albo dopóki nie osiągnięty zostanie zadowalający wynik. Wynikiem przetwarzania w sensie znajdowania zależności funkcyjnej jest na ogół najbardziej przystosowany osobnik, który kiedykolwiek się pojawił. Można zapytać czy zastosowanie PG daje lepsze wyniki niż ślepe przeszukiwanie. Trzeba zdać sobie sprawę, że przestrzeń przeszukiwania składająca się ze wszystkich możliwych do wygenerowania drzew z założonych zbiorów F oraz T jest bardzo duża i znacznie większa niż ilość wszystkich szacowań funkcji fitness podczas całego przetwarzana (równa, co oczywiste iloczynowi liczby generacji i ilości osobników w populacji). Zauważmy, że:

Rozwiązanie zostaje znalezione przy pomocy PG w n-tej, a nie w początkowej populacji.

W trakcie przetwarzania wzrasta średnie przystosowanie populacji w kolejnych generacjach.

W literaturze przedmiotu porównywano wyniki uzyskane przy użyciu PG oraz metod losowych. Dla tej samej (sumarycznej) liczby osobników przy pomocy ślepego przeszukiwania nie znaleziono nawet przybliżonego rozwiązania, podczas gdy przy użyciu PG problem udało się rozwiązać.

Zależność funkcyjna (czyli model) została znaleziona - co dalej?

Po znalezieniu modelu, czyli zależności funkcyjnej należy go przetestować. Testowanie polega na porównaniu wyników (np. przewidywania) uzyskanych przy pomocy modelu ze znanymi rzeczywistymi danymi. Zazwyczaj do uczenia wykorzystuje się podciąg uczący o długości 2/3 ciągu danych a do testowania ciąg testowy o długości 1/3 ciągu z danymi. Przykładowo jeśli do uczenia wykorzystano ciąg taki jak w tabeli 1 o długości 10, to do testowania modelu należy użyć następnych (nie ujętych w tej tabeli) pięciu notowań.