ЛЕКЦІЯ 7
ОСОБЛИВОСТІ ФОРМУВАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ

1. Представлення даних у генах
2. Приклади кодування параметрів задачі в генетичному алгоритмі
3. Основна теорема про генетичні алгоритми
4. Будівельні блоки (Building blocks)
5. Еволюційні алгоритми
6. Еволюційні алгоритми в нейронних мережах

ВСТУП

Основний (класичний) генетичний алгоритм (також називаний елементарним або простим генетичним алгоритмом) складається з наступних кроків:

1) ініціалізація, або вибір вихідної популяції хромосом;
2) оцінка пристосованості хромосом у популяції;
3) перевірка умови зупинки алгоритму;
4) селекція хромосом;
5) застосування генетичних операторів;
6) формування нової популяції;
7) вибір «найкращої» хромосоми.

Блок-схема основного генетичного алгоритму зображена на рис. 12.

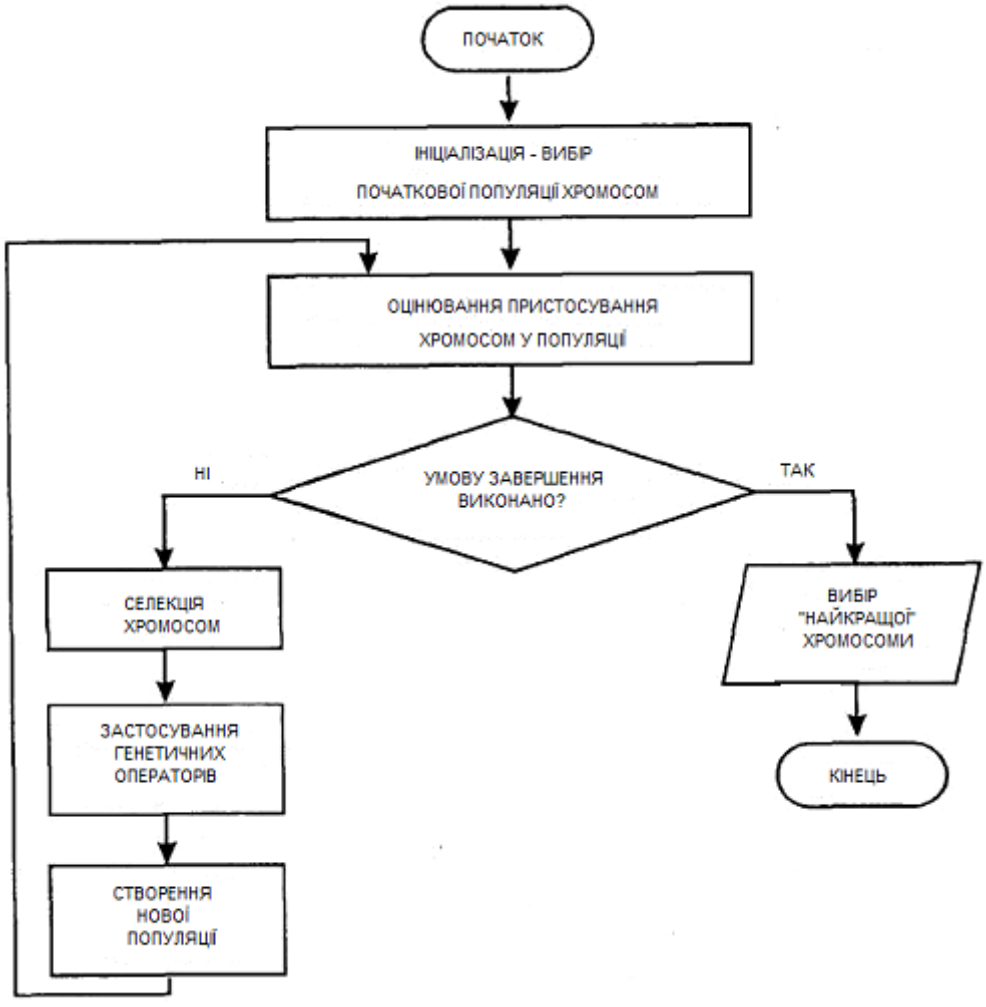

Рис. 12. Блок-схема генетичного алгоритму.

Розглянемо конкретні етапи цього алгоритму більш докладно з використанням додаткових подробиць, представлених на рис. 13.

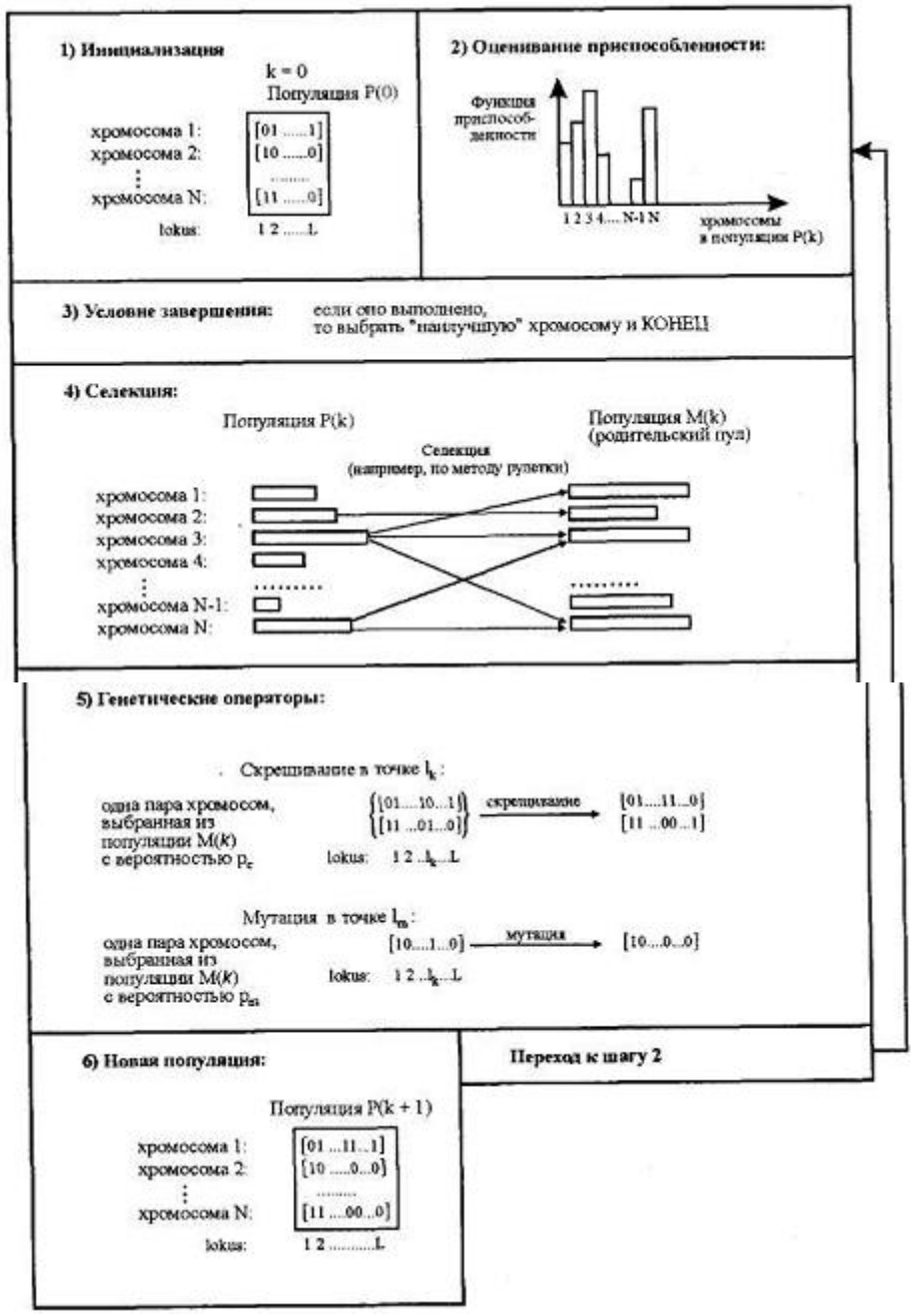

Рис. 13. Схема виконання генетичного алгоритму.

Генетичні оператори

Частково про генетичні оператори ми вже говорили. Зупинимося на їхньому описі більш докладно. Генетичні оператори необхідні, щоб застосувати принципи спадковості і мінливості до віртуальної популяції. Крім відмінних рис, про які буде розказано нижче, у них є така властивість як ймовірність. Тобто описувані оператори не обов'язково застосовуються до всіх особин, які схрещуються, що вносить додатковий елемент

невизначеності в процес пошуку рішення. У даному випадку, невизначеність не має на увазі негативний фактор, а є таким собі "степенем волі" роботи генетичного алгоритму. Тут описуються тільки два найпоширеніші і необхідних оператори. Існують і інші генетичні оператори (наприклад, інверсія), але вони застосовуються дуже рідко і тому ми про них говорити не будемо.

Оператор кросінговера

Оператор кросінговера (crossover operator), також називаний кросовером, є основним генетичним оператором, за рахунок якого виконується обмін генетичним матеріалом між особами. Оператор моделює процес схрещування особин. Нехай є дві батьківські особи з хромосомами X={xi , i=1,L} і Y={yi , i=1,L}. Випадковим чином визначається точка усередині хромосоми, у якій обидві хромосоми діляться на дві частини і обмінюються ними. Назвемо цю точку точкою розриву. Узагалі говорячи, в англомовній літературі вона називається точкою кросінговера (crossover point), просто, точка розриву більш образна назва і до того ж дозволяє в деяких випадках уникнути тавтології. Описаний процес зображено на рис.7.14.

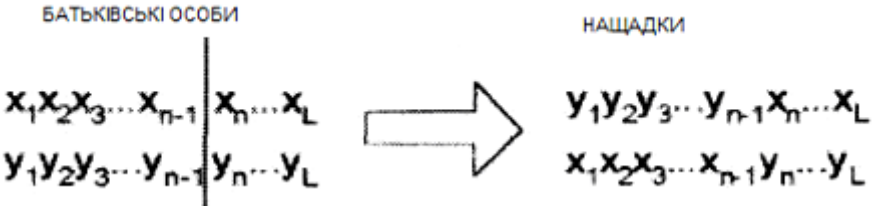

Рис.14. Кросінговер

Даний тип кросінговера називається одноточковим, так як при ньому батьківські хромосоми розрізаються тільки в одній випадковій точці. Також існують 2-х і n-точковий оператори кросінговера. У 2-х точковому кросінговері точок розриву 2, а n-точковий кросінговер є своєрідним узагальненням 1- і 2-точкового кросінговерів для n > 2.

Крім описаних типів кросінговерів є ще однорідний кросінговер. Його особливість полягає в тім, що значення кожного біта в хромосомі нащадка визначається випадковим чином з відповідних бітів батьків. Для цього вводиться деяка величина 0<p0<1, і якщо випадкове число більше p0, то на n-у позицію першого нащадка потрапляє n-й біт першого батька, а на n-у позицію другого - n-й біт другого батька. У протилежному випадку до першого нащадка потрапляє біт другого батька, а до другого - першого. Така операція проводиться для всіх бітів хромосоми.

Ймовірність кросінговера найвища серед генетичних операторів і дорівнює зазвичай 60% і більше.

Оператор мутації

Оператор мутації (mutation operator) необхідний для "вибивання" популяції з локального екстремуму і сприяє захистові від передчасної збіжності. Досягаються це за рахунок того, що інвертується випадково обраний біт у хромосомі, що і показано на рис. 15.

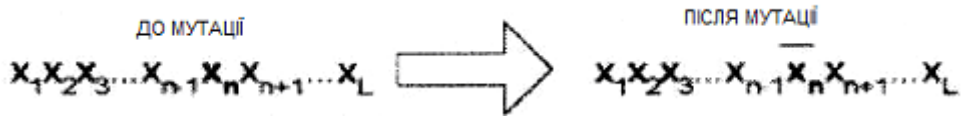

Рис. 15. Мутація

Так само як і кросінговер, мутація проводиться не тільки по одній випадковій точці. Можна вибирати деяку кількість точок у хромосомі для інверсії, причому їхнє число

також може бути випадковим. Також можна інвертувати відразу деяку групу точок, що йдуть підряд. Ймовірність мутації значно менша ймовірності кросінговера і рідко перевищує 1%. Серед рекомендацій з вибору ймовірності мутації нерідко можна зустріти варіанти 1/L або 1/N, де L - довжина хромосоми, N - розмір популяції. Необхідно також відзначити, що деякі автори вважають, що оператор мутації є основним пошуковим оператором і відомі алгоритми, що не використовують інших операторів (кросінговер, інверсія і т.д.) крім мутації.

Використовуються також кросінговери:

- Арифметичний кросінговер (Arithmetical crossover).
- Геометричний кросінговер (Geometrical crossover,).
- BLX-a кросінговер (BLX-a crossover).
- Імітований бінарний кросінговер (Simulated binary crossover).
- Нечітка рекомбінація (Fuzzy recombination).

Стратегії формування нового покоління

Після схрещування особин необхідно вирішити проблему про те, які з нових особин ввійдуть у наступне покоління, а які - ні, і що робити з їх предками. Є два способи:

1. Нові особи (нащадки) займають місця своїх батьків. Після чого настає наступний етап, у якому нащадки оцінюються, відбираються, дають потомство і поступаються місцем своїм "дітям".

2. Створюється проміжна популяція, що містить у собі як батьків, так і їхніх нащадків. Члени цієї популяції оцінюються, а потім з них вибираються N найкращих, котрі і ввійдуть у наступне покоління.

Ті хто знайомий з еволюційними стратегіями, то з цими способами ви вже зустрічалися. Який з цих двох варіантів краще – відповісти однозначно важко. Вочевидь другий варіант практичніший (так як не дозволяє заміняти пристосованих батьків на "невідомо кого"), але тут може бути більше проблем з передчасною збіжністю, чим у першому варіанті. До того ж він вимагає сортування масиву розміром (як мінімум) 2*N. Узагалі говорячи, можна комбінувати стратегії відбору і формування наступного покоління як завгодно - обмежень немає ніяких.

Принцип "елітизму"

Суть цього принципу полягає в тім, що в нове покоління включаються кращі батьківські особи. Їхнє число може бути від 1 і більше. Використання "елітизму" дозволяє не втратити гарне проміжне рішення, але, у той же час, через це алгоритм може "застрягти" у локальному екстремумі. Однак, досвід використання принципу "елітизму" , дозволяє зробити висновок, що в більшості випадків "елітизм" анітрохи не шкодить пошукові рішення, і головне - це надати алгоритмові можливість аналізувати різні рядки з простору пошуку.

У класичному описі генетичного алгоритму мається на увазі створення популяції нащадків і заміщення батьківських особин їх "дітьми". Такий підхід досить гарний, але не особливо ефективний, тому що нащадки, отримані в результаті генетичних перетворень, можуть бути гірші батьківських особин. У результаті з'явилося кілька підходів, "виправляючих" дане явище, які можна об'єднати, використовуючи, як ми казали, поняття "елітизм" (elitism) (іноді "стратегія елітизму" (elitist strategy). Нижче буде приведено короткий опис цих підходів.

Умовно елітизм можна розділити на два класи-підходи, назвемо їх конкурентним і неконкурентним. Коротко їхня суть у наступному:

1. Конкурентний підхід. Батьківські особи "змагаються" з нащадками і переможці (або переможець) переходять у наступне покоління.

2. Неконкурентний підхід. У цьому випадку частина батьківської підпопуляції (випадкова або визначена за заданим правилом) переходить у нове покоління без яких-небудь заперечень з боку електорату.

Іншими словами, у першому класі нащадки після створення мають, загалом, рівні права з батьками на те, щоб перейти в нове покоління і визначальну роль тут грає пристосованість особи, а не її положення на генеалогічному дереві. В другому класі старі індивіди мають визначений пріоритет і навіть якщо всі нащадки будуть краще будь-якого батька, якась частина батьківської підпопуляції неминуче буде присутня у новому поколінні.

1. Представлення даних у генах

Для рішення деякої задачі за допомогою ГА її дані необхідно *представити у виді генів особи*. Для цього насамперед необхідно визначити які параметри задачі необхідно настроїти. Наприклад, якщо ми намагаємося апроксимувати за допомогою ГА набір точок функцією виду $f(x)=A*exp(k*x)$, де A, k - константи, x - незалежна змінна, то як параметри будуть виступати A і k, тому що від їхнього значення залежить вид і поводження функції. Отже, маємо два параметри і, отже, два гени.

Наступний крок - це *вибір числа розрядів у кожнім гені*. Тут необхідно враховувати, що з одного боку, чим більше розрядів, тим краще, тому що вище точність і т.д., але з іншого боку - велике число розрядів тягне збільшення часу пошуку рішення (збіжності). Варто відзначити, що з ростом числа параметрів (у деяких задачах вони обчислюються сотнями) "зайві" розряди позначаються усе сильніше і сильніше. Тому необхідно знайти компроміс між точністю і швидкістю. Візьмемо для задачі апроксимації 16-розрядні гени, при цьому параметри будуть змінюватися від -32,768 до 32,767 із кроком у 0,001. Дані числа отримані, виходячи з того, що 16-розрядне число може приймати одне з 2^{16}= 65536 значень від 0 до 65535. Якщо припустити, що 0 буде в середині цього інтервалу, причому кожне значення розділиться потім на 1000, то одержимо інтервал зміни параметрів і крок зміни. Після того, як обрано параметри, їхнє число і розрядність, необхідно вирішити, як безпосередньо записувати дані. Можна використовувати звичайне кодування, коли $1011=1011_2=11_{10}$, або коди Грея, коли $1011=1110_2=14_{10}$.

Незважаючи на те, що коди Грея тягнуть неминуче кодування/декодування даних, вони дозволяють уникнути деяких проблем, що з'являються в результаті звичайного кодування. Можна лише сказати, що перевага коду Грея в тому, що якщо два числа розрізняються на 1, то і їхні двійкові коди розрізняються тільки на один розряд, а в двійкових кодах не все так просто.

Варто відзначити, що кодувати і декодувати у коди Грея досить зручно, описати це можна так: спочатку копіюється самий старший розряд, потім:

З двійкового коду в код Грея: G[i] = XOR(B[i+1], B[i])

З коду Грея в двоїчний код: B[i] = XOR(B[i+1], G[i])

Тут, G[i] - і-й розряд коду Грея, а B[i] - і-й розряд бінарного коду. Наприклад, послідовність чисел від 0 до 7 у двоїчном коді:

{000, 001, 010, 011, 100, 101, 110, 111},

а в кодах Грея:

{000, 001, 011, 010, 110, 111, 101, 100}.

Отже, тепер задано всі необхідні характеристики генів, залишається тільки представити все це на якій-небудь мові програмування. Наприклад, на C++ одна особа із хромосомою з двох 16-розрядних генів може "виглядати" так:

```
short int Individual [2];
1. class Individual {
2. short int Genes [2];
3. };
```

```
4. unsigned char Individual [32];
```

Останній приклад на перший погляд здається як мінімум дивним: навіщо для завдання одного біта використовувати цілий байт? Однак бувають випадки, коли використовуються небінарні алфавіти, тобто кожен розряд може приймати не 2 різні значення "0" або "1", а більше, наприклад, "1", "2", "4", "8". Саме для таких ситуацій останній приклад і зручний. Надалі мова буде йти тільки про бінарний алфавіт. Крім того, в останньому способі легше використовувати коди Грея. Для того щоб задати популяцію з 50 особин, можна поступити одним з наступних способів:

```
1. short int Population[50][2];
2. class Population {
3. Individual Inds[50];
4. };
5. unsigned char Population[50][32];
```

Слід також зазначити, що в задачі апроксимації, узятої як приклад, у процесі "еволюції" будуть приймати участь усі гени особи, тобто генетичні оператори будуть застосовуватися до кожного гена. Однак, це не завжди необхідно, наприклад, розглянемо задачу компонування, тобто коли на деякій відомій поверхні потрібно розмістити N елементів різної площі так, щоб вони не накладалися один на одний. Для простоти будемо вважати, що всі елементи мають прямокутну форму.

Параметрами, що настроюються (оптимізуються), будуть координати кожного елемента, тобто кожна хромосома буде містити два гени, що відповідають координатам лівого верхнього кута елемента. Але крім цього необхідно знати, який елемент представляє дана особу. Для цього необхідно додати в набір генів особи ще один ген, що містить номер елемента. При цьому нам потрібно зберегти значення цього гена незмінним на протязі всього процесу пошуку рішення. Таким чином, одержуємо особу із трьома генами, два з яких обробляються генетичними операторами, а один ні.

Крім розглянутих задач апроксимації і компонування можна привести *варіанти кодування для деяких інших задач*:

• Оптимізація функцій: гени - незалежні змінні;

• Настроювання ваг штучної нейронної мережі: гени - синаптичні ваги нейронів;

• Штучне життя (Artificial Life): гени - характеристики особи (сила, швидкість, і т.д.), також повинні бути незмінні гени, що позначають тип особи (рослина або тварина);

• Задача про найкоротший шлях: гени - пункти пересування. Уся хромосома цілком представляє із себе маршрут з початкової точки в кінцеву, причому не завжди існуючий.

І останнє, розрядність генів необов'язково повинна бути фіксованою, багато сучасних алгоритмів самостійно підбудовують число розрядів для кожного гена в залежності від проміжних результатів пошуку рішення.

2. Приклади кодування параметрів задачі в генетичному алгоритмі

Вибір вихідної популяції зв'язаний із представленням параметрів задачі у формі хромосом, тобто з так називаним хромосомним представленням. Це представлення визначається способом кодування. У класичному генетичному алгоритмі застосовується двійкове кодування, тобто алелі всіх генів у хромосомі рівні 0 або 1. Довжина хромосом залежить від умов задачі, точніше кажучи - від кількості точок у просторі пошуку.

У прикладі 3.6 оптимізується та ж функція, однак увага читача акцентується на іншому способі кодування хромосом для іншої області визначення змінної x.

Приклад 3.6 (на самостійне вивчення)

Розглянемо задачу, аналогічну задачі з приклада 3.5, тобто будемо шукати максимум функції, заданою формулою

$$f(x) = 2x^2+1 \quad (8.1)$$

, але для змінної x, що приймає дійсні значення з інтервалу [a, b], де a = 0, b = 3,1. Припустимо, що нас цікавить рішення з точністю до одного знака після коми. Пошук рішення зводиться до перегляду простору, що складається з 32 точок 0,0; 0,1; ...; 2,9; 3,0; 3,1. Ці точки (фенотипи) можна представити у виді хромосом (генотипів), якщо використовувати бінарні п'ятиланкові ланцюжки, оскільки за допомогою 5 бітів можна одержати 2^5=32 різних кодові комбінації. Отже, можна використовувати таку ж множину кодових послідовностей, як і в прикладі 3.5, причому хромосома [00000] буде відповідати числу 0,0, хромосома [00001] - числу 0,1 і т.д., аж до хромосоми [11111], що відповідає числу 3,1.

Таким чином, ми можемо відтворити послідовність етапів генетичного алгоритму (так само, як у прикладі 3.5), не забуваючи, що конкретним хромосомам (генотипам) у даному прикладі відповідають інші фенотипи. Ті кодові послідовності, що у прикладі 3.5 представляли фенотипи 0, 1, ..., 29, 30, 31, у розглянутій ситуації позначають значення x, рівні 0,0; 0,1;...;2,9; 3,0; 3,1. У зв'язку з тим, що генетичний алгоритм заснований на випадковому виборі вихідної популяції і хромосом для наступного перетворення методом колеса рулетки, а також батьківських пар для схрещування і точки схрещування, то генетичний алгоритм у поточному прикладі буде виконуватися аналогічно, але не ідентично попередньому прикладові.

У результаті виконання цього алгоритму буде обрано найкраще рішення, що представляється хромосомою [11111] зі значенням фенотипу 3,1. Функція пристосованості цієї хромосоми дорівнює 20,22; це максимально можливе значення. Помітимо, що якби в прикладі 3.6 нас цікавило рішення з точністю, що перевищує один знак після коми, то інтервал [0; 3,1] необхідно було б розбити на більшу кількість підінтервалів, і для кодування відповідно більшої кількості чисел потрібні були більш довгі хромосоми (з довжиною, що перевищує 5 бітів). Аналогічно, розширення області визначення змінної x також зажадає застосування більш довгих хромосом. З цих спостережень можна зробити *висновок, що довжина хромосом залежить від ширини області визначення x і від необхідної точності рішення*.

Представимо тепер задачу з приклада 3.6 у більш загальному виді. Припустимо, що шукається максимум функції $f(x_1,x_2.....x_n)>0$ для $x_i \in [a_i, b_i] \subset R;\ i = 1, 2, ..., n$ і потрібно знайти рішення з точністю до q знаків після коми для кожної змінної x_i .. У такій ситуації необхідно розбити інтервал $[a_i, b_i]$ на $(bi\text{-}ai)\cdot 10^q$ однакових підінтервалів. Це означає застосування дискретизації з кроком r= 10^{-q} . Найменше натуральне число m_i, що задовольняє нерівності

$$(bi\text{-}ai)\cdot 10^q \le 2^{m_i} - 1 \quad (3.4)$$

визначає необхідну і достатню довжину двійкової послідовності, необхідної для кодування числа з інтервалу [a_i, b_i] із кроком r.

Кожному елементу такої двійкової послідовності відповідає десяткове значення числа, що представляється даним кодом (з урахуванням правил перекладу десяткових чисел у двійкову форму). Нехай y_i - позначає десяткове значення двійкової послідовності, що кодує число x_i. Значення x_i можна представити виразом

$$x_i = a_i + y_i \frac{b_i - a_i}{2^{m_i} - 1}. \quad (3.5)$$

Таким способом задаються *фенотипи*, що відповідають кодовим послідовностям з довжиною m_i . Приклад 3.6 - це окремий випадок задачі в даній постановці за умови, що i= 1 і q=1. Вираз (3.5) - це наслідок із простого лінійного відображення інтервалу [a_i, b_i] на інтервал $[0, 2^{m_i} -1]$, де 2^{m_i} - десяткове число, закодоване двійковою послідовністю

довжиною m_i, і складене винятково з одиниць, а 0 - це, мабуть, десяткове значення двійкової послідовності довжиною m_i, складеної тільки з нулів.

Зверніть увагу, що якщо $a_i = -25$, $b_i = 25$ і застосовується крок r=0,05, то відповідно до формули (3.4) одержуємо m_i =10.

3. Основна теорема про генетичні алгоритми

Для того щоб краще зрозуміти функціонування генетичного алгоритму, будемо використовувати поняття схеми і сформулюємо основну теорему, що відноситься до генетичних алгоритмів і називається **теоремою про схеми**. Поняття схема було введено Холландом і використовується для аналізу роботи ГА. Зокрема розглядаються процеси конструювання і руйнування визначеної схеми протягом розвитку популяції (schema dynamics).

Схемою називається рядок виду

$$(a_1, a_2, ..., a_i, ..., a_l),\ a_i \in \{0, 1, *\}.$$

Символом "*" у деякому розряді позначається те, що там може бути як 1, так і 0. Наприклад, для двох бінарних рядків

"111000111000" і

"110011001100" схема буде виглядати в такий спосіб:

"11*0****1*00". Тобто за допомогою схем можна як би виділяти загальні ділянки двійкових рядків і маскувати розходження. Маючи в складі схем m символів "*" можна закодувати (узагальнити) 2^m двійкових рядків. Так, наприклад, схема "01*0*1" описує набір рядків

{"010001", "010011", "011001", "011011"}.

Визначальною довжиною схеми *(schema defining length)* називається відстань між двома крайніми символами "0" і/або "1". Для схеми "01*0*1" визначальна довжина дорівнює 5, а для схеми "**0**1*" визначальна довжина дорівнює 3.

Порядок схеми (schema order) - це ще одна характеристика схеми і дорівнює вона числу фіксованих позицій у рядку, тобто загальному числу "0" і "1". Для схем "01*0*1" і "**0**1*" порядки рівні 4 і 2 відповідно.

Тепер про те, яке відношення схема має до генетичних алгоритмів. Справа в тому, що хромосома, по суті, є двійковим рядком. У той же час особі, якій належить хромосома, що містить набір генів-параметрів задачі, поставлена у відповідність величина, що характеризує її пристосованість. Так як схема є узагальненням декількох бінарних рядків (хромосом), то можна говорити, що особи, які володіють хромосомами, що відповідають одній схемі більш пристосовані, а особи з хромосомами, які відповідають іншій схемі - менш пристосовані.

Можна сказати, що *зміст роботи ГА* полягає в пошуку двійкового рядка визначеного виду з усієї множині бінарних рядків. Простір пошуку складає 2^L рядків, а його мірність дорівнює L (L-мірний простір), де L - довжина хромосоми. Схема відповідає деякій гіперплощині в цьому просторі. Дане твердження можна проілюструвати в такий спосіб. Нехай розрядність хромосоми дорівнює 3, тоді усього можна закодувати 2^3 =8 рядків. Представимо куб у 3-мірному просторі. Позначимо вершини цього куба 3-розрядними бінарними рядками так, щоб мітки сусідніх вершин відрізнялися рівно на один розряд, причому вершина з міткою "000" знаходилася б на початку координат. Варіант позначення зображено на рис. 3.20.

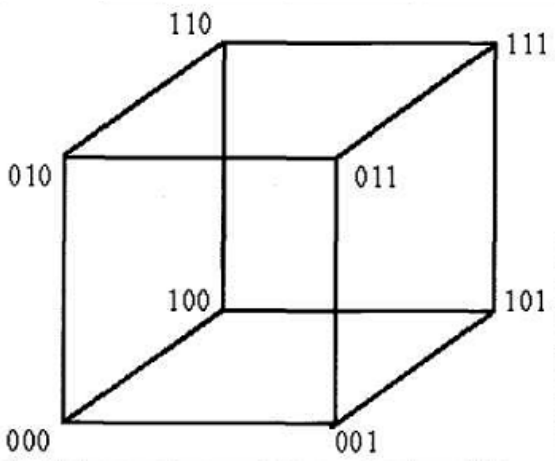

Рис. 3.20. 3-мірний куб

Якщо взяти схему виду "**0", то вона опише ліву грань куба, а схема "*10" - верхнє ребро цієї грані. Очевидно, що схема "***" відповідає всьому просторові. Якщо взяти двійкові рядки довжиною 4 розряди, то розбивка простору схемами можна зобразити на прикладі 4-мірного куба з пойменованими вершинами (рис. 3.21).

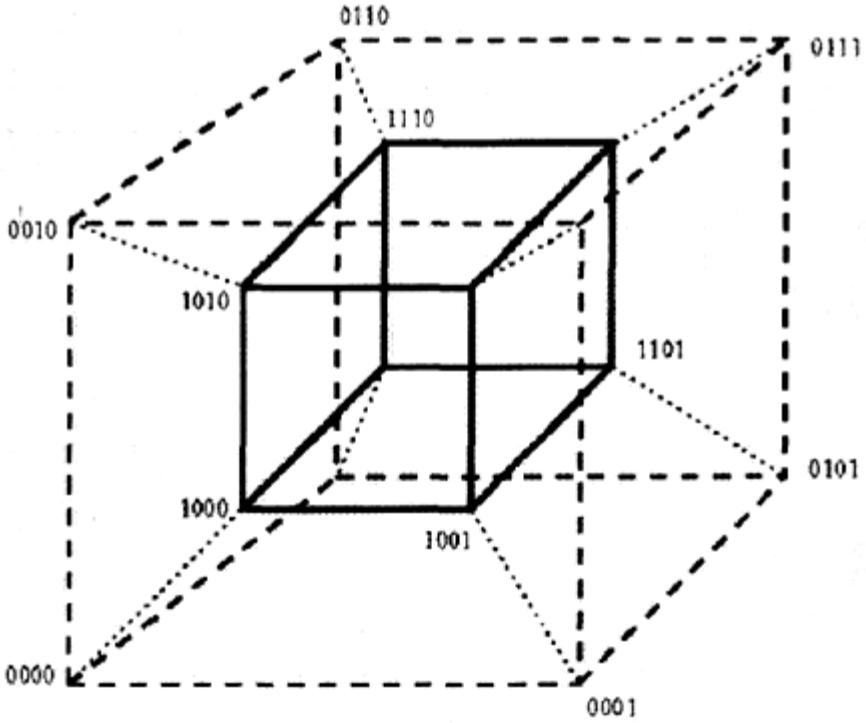

Рис. 3.21. 4-мірний куб

Тут схемі "*1**" відповідає гіперплощина, що включає задні грані зовнішнього і внутрішнього куба, а схемі "**10" - гіперплощина з верхніми ребрами лівих граней обох кубів.

Розбивку простору пошуку можна представити і по іншому. Представимо координатну площину, у якій по одній осі ми будемо відкладати значення двійкових рядків, а по іншій - значення цільової функції (рис. 3.22).

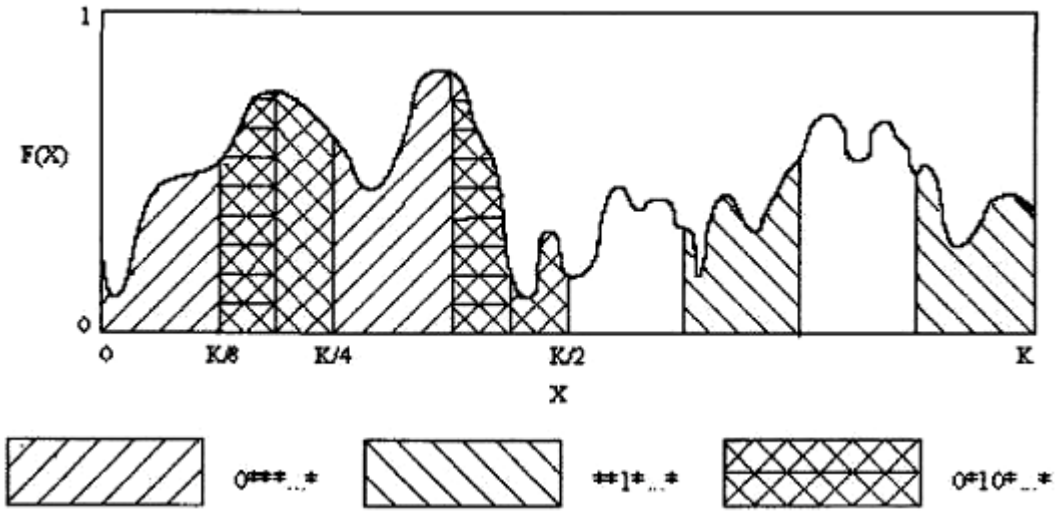

Рис. 3.22. Розбивка простору

Ділянки простору, які заштриховані різним стилем, відповідають різним схемам. Число К в правій частині горизонтальної осі відповідає максимальному значенню бінарного рядка - "111...111". З рисунка видно, що схема "0***...*" покриває всю ліву половину відрізка, схема "**1*...*" - 4 ділянки шириною в одну восьму частину, а схема "0*10*...*" - ліві половини ділянок, що знаходяться на перетині перших двох схем. У такий спосіб і відбувається розбивка простору в цьому випадку. Як ми вже говорили, поняття схема було введено для визначення множини хромосом, що володіють деякими загальними властивостями, тобто подібних одна одній. Якщо алелі приймають значення 0 або 1 (розглядаються хромосоми з двійковим алфавітом), то схема являє собою множину хромосом, що містять нулі й одиниці на деяких заздалегідь визначених позиціях. При розгляді схем зручно використовувати розширений алфавіт {0, 1, *}, у який крім 0 і 1 уведено додатковий символ *, що позначає будь-яке припустиме значення, тобто 0 або 1; символ * у конкретній позиції означає «усе рівно» (don't care). Наприклад,

$$10*1 = \{1001, 1011\}$$

$$*01*10 = \{001010, 001110, 101010, 101110\}$$

Вважається, що хромосома належить до даної схеми, якщо для кожної j-ї позиції (локусу), j = 1, 2, ..., L, де L - довжина хромосоми; символ, що займає j-у позицію хромосоми, відповідає символові, що займає j-у позицію схеми, причому символові * відповідають як 0, так і 1. Те ж саме означають твердження хромосома відповідає схемі і хромосома представляє схему. Відзначимо, що якщо в схемі присутні m символів *, то ця схема містить 2^m хромосом. Крім того, кожна хромосома (ланцюжок) довжиною L належить до 2^L схемам. У таблицях 3.2 і 3.3 представлені схеми, до яких належать ланцюжки довжиною 2 і 3 відповідно.

Таблиця 3.2. Схеми, до яких належать ланцюжки довжиною 2

ЛАНЦЮГИ	СХЕМИ			
	1	2	3	4
00	**	*0	0*	00
01	**	*1	0*	01
10	**	*0	1*	10
11	**	*1	1*	11

Таблиця 3.3. Схеми, до яких належать ланцюжки довжиною 3

ЛАНЦЮГИ	СХЕМИ							
	1	2	3	4	5	6	7	8
000	***	**0	*0*	0**	*00	0*0	00*	000
001	***	**1	*0*	0**	*01	0*1	00*	001
010	***	**0	*1*	0**	*10	0*0	01*	010
011	***	**1	*1*	0**	*11	0*1	01*	011
100	***	**0	*0*	1**	*00	1*0	10*	100
101	***	**1	*0*	1**	*01	1*1	10*	101
110	***	**0	*1*	1**	*10	1*0	11*	110
111	***	**1	*1*	1**	*11	1*1	11*	111

Ланцюжки довжиною 2 відповідають чотирьом різним схемам, а ланцюжки довжиною 3 - вісьмом схемам.

Генетичний алгоритм базується на принципі трансформації найбільш пристосованих особин (хромосом). Нехай *P(0)* означає вихідну популяцію особин, а *P(k)* - поточну популяцію (на *k*-й ітерації алгоритму). З кожної популяції *P(k)*, *k* = 0, 1, ... методом селекції вибираються хромосоми з найбільшою пристосованістю, які включаються в так званий батьківський пул (mating pool) *M(k)*. Далі, у результаті об'єднання особин з популяції *M(k)* у батьківські пари і виконання операції схрещування з ймовірністю p_c, а також операції мутації з ймовірністю рт формується нова популяція *P(k+1)*, у яку входять нащадки особин з популяції *M(k)*. Отже, для будь-якої схеми, що представляє гарне рішення, було б бажаним, щоб кількість хромосом, що відповідають цій схемі, зростало зі збільшенням кількості ітерацій *k*.

На відповідне перетворення схем у генетичному алгоритмі впливають 3 фактори: *селекція хромосом, схрещування і мутація*. Проаналізуємо вплив кожного з них з погляду збільшення очікуваної кількості представників окремо узятої схеми. Позначимо розглянуту схему символом S, а кількість хромосом популяції P(k), що відповідають цій схемі - *c(S, k)*.

Отже, *c(S, k)* можна вважати кількістю елементів (тобто потужністю) множини *P(k)*∩*S*.

Почнемо з *дослідження впливу селекції*. При виконанні цієї операції хромосоми з популяції P(k) копіюються в батьківський пул M(k) з ймовірністю, обумовленою виразом (3.3). Нехай F(S, k) позначає середнє значення функції пристосованості хромосом з популяції P(k), що відповідають схемі S. Якщо

$$P(k)S = \{ch_1 \ldots\ldots ch_{c(S,k)}\},$$

то

$$c(S,k) = \frac{\sum_{i=1}^{c(S,k)} F(ch_i)}{c(S,k)} \qquad (3.6)$$

Величина F(S, k) також називається пристосованістю схеми S на k-й ітерації.

Нехай $\Im^{(k)}$ $\Im$(k) позначає суму значень функцій пристосованості хромосом з популяції P(k) потужністю N, тобто

$$\Im(k) = \sum_{i=1}^{N} F(ch_i^{(k)}) \tag{3.7}$$

Позначимо через $\overline{F}(k)$ середнє значення функції пристосованості хромосом цієї популяції, тобто

$$\overline{F}(k) = \frac{1}{N}\Im(k). \tag{3.8}$$

Нехай $ch_i^{(k)}$ позначає елемент батьківського пула M(k). Для кожного $ch_r^{(k)} \in M(k)$ і для кожного $i = 1,\ldots, c(S, k)$ ймовірність того, що $ch_r^{(k)}=ch_i$ визначається відношенням $F(ch_i)/F(k)$. Тому очікувана кількість хромосом у популяції M(k), що рівні ch_i, складе

$$N\frac{F(ch_i)}{\Im(k)} = \frac{F(ch_i)}{\overline{F}(k)}$$

Таким чином, очікувана кількість хромосом з множини P(k)∩ S, відібраних для включення в батьківський пул M(k), буде дорівнювати

$$\sum_{i=1}^{c(S,k)} \frac{F(ch_i)}{\overline{F}(k)} = c(S,k)\frac{F(S,k)}{\overline{F}(k)},$$

що випливає з виразу (3.6).

Оскільки кожна хромосома з батьківського пулу M(k) одночасно належить популяції P(k), то хромосоми, що складають множину M(k)∩S - це ті ж самі особи, що були відібрані з множини P(k)∩S для включення в популяцію M(k). Якщо кількість хромосом батьківського пулу M(k), що відповідають схемі S (тобто кількість елементів множини M(k)∩S), позначити b(S, k), то з приведених міркувань можна зробити наступний висновок:

Висновок 3.1

Очікуване значення b(S, k), тобто очікуване значення кількості хромосом батьківського пулу M(к), що відповідають схемі S, визначається виразом

$$E[b(S,k)] = c(S,k)\frac{F(S,k)}{\overline{F}(k)}. \tag{3.9}$$

З цього випливає, що якщо схема S містить хромосоми зі значенням функції пристосованості, що перевищує середнє значення (тобто пристосованість схеми S на k-й ітерації виявляється більшою, ніж середнє значення функції пристосованості хромосом з популяції P(k), і тому $F(S, k)/F(k)>1$), то очікувана кількість хромосом з батьківського пулу M(k), що відповідають схемі S, буде більше кількості хромосом з популяції P(k), що відповідають схемі S. Тому можна стверджувати, що селекція викликає поширення схем із пристосованістю «краще середньої» і зникнення схем з «гіршою» пристосованістю.

Перш ніж приступити до аналізу впливу генетичних операторів схрещування і мутації на хромосоми з батьківського пулу, визначимо необхідні для подальших міркувань поняття *порядку й охоплення схеми*. Нехай L позначає довжину хромосом, що відповідають схемі S.

Означення 3.1

Порядок (order) схеми S, інакше називаний рахунковістю схеми і що позначається o(S) - це кількість сталих позицій у схемі, тобто нулів і одиниць у випадку алфавіту {0, 1, *}.

Наприклад,

o(10*1) = 3 o(*01*10) = 4 o(**0*Г) = 2 o(*101**) = 3

Порядок схеми o(S) дорівнює довжині L за винятком кількості символів *, що легко перевірити на представлених прикладах (для L = 4 з одним символом * і для L = 6 із двома, чотирма і трьома символами *). Легко помітити, що порядок схеми, що складається винятково із символів *, дорівнює нулеві, тобто o(****) = 0, а порядок схеми без єдиного символу * дорівнює L; наприклад, o(10011010) = 8. Порядок схеми o(S) - це завжди ціле число з інтервалу [0, L].

Означення 3.2

Охоплення (defining length) схеми S, яке називається також довжиною схеми (не плутати з довжиною L) і що позначається d(S) - це відстань між першим і останнім сталим символом (тобто різниця між правими і лівої крайніми позиціями, що містять сталі символи). Наприклад,

d(10*1) = 4-1 =3
d(**0*1*) = 5-3 = 2
d(*01*10) = 6-2 = 4
d(*101**) = 4-2 = 2

Охоплення схеми d(S) - це ціле число з інтервалу [0, L - 1]. Відзначимо, що охоплення схеми зі сталими символами на першій і останній позиції дорівнює L-1 (як у першому з приведених прикладів). Охоплення схеми з єдиною сталою позицією дорівнює нулеві, зокрема, d(**1*)=0. Охоплення характеризує змістовність інформації, яка міститься в схемі.

Перейдемо до міркувань про вплив операції схрещування на обробку схем у генетичному алгоритмі. Насамперед відзначимо, що одні схеми виявляються більш підданими знищенню в процесі схрещування, ніж інші. Наприклад, розглянемо схеми

S1 = 1****0* і S2 = **01***, а також хромосому ch=[1001101], що відповідає обом схемам. Видно, що схема S2 має більше шансів «пережити» операцію схрещування, ніж схема S1, що більше піддана «розщепленню» у точках схрещування 1, 2, 3, 4 і 5. Схему S2 можна розділити тільки при виборі точки схрещування, рівної 3. Звертаємо увагу на охоплення обох схем, що - це очевидно - виявляється істотним у процесі схрещування.

У ході аналізу впливу операції схрещування на батьківський пул M(k) розглянемо деяку хромосому з множини M(k)∩S, тобто хромосому з батьківського пулу, що відповідає схемі S.

Ймовірність того, що ця хромосома буде відібрана для схрещування, дорівнює p_c. Якщо жоден з нащадків цієї хромосоми не буде належати до схеми S, то це означає, що точка схрещування повинна знаходитися між першим і останнім сталим символом даної схеми. Ймовірність цього дорівнює d(S)/(L-1). З цього можна зробити наступні висновки:

Висновок 3.2 (вплив схрещування)

Для деякої хромосоми з M(k)∩S ймовірність того, що вона буде відібрана для схрещування і жоден з її нащадків не буде належати до схеми S, обмежена зверху величиною p_c

$$p_c \frac{d(S)}{L-1}$$

Ця величина називається *ймовірністю знищення схеми S*.

Висновок 3.3

Для деякої хромосоми з M(k)∩S ймовірність того, що вона не буде відібрана для схрещування або, що хоча б один з її нащадків після схрещування буде належати до схеми S, обмежена знизу величиною

$$1\text{-}p_c\frac{d(S)}{L-1}$$

Ця величина називається *ймовірністю виживання схеми S*.

Легко показати, що якщо дана хромосома належить до схеми S і відбирається для схрещування, а друга батьківська хромосома також належить до схеми S, то обидва їх нащадка теж будуть належати до схеми S. Висновки 3.2 і 3.3 підтверджують значимість показника охоплення схеми d(S) для оцінки ймовірності знищення або виживання схеми.

Розглянемо тепер *вплив мутації на батьківський пул* M(k). Оператор мутації з ймовірністю p_m випадковим чином змінює значення в конкретній позиції з 0 на 1 і зворотно. Очевидно, що схема переживе мутацію тільки в тому випадку, коли всі її сталі позиції залишаться після виконання цієї операції незмінними.

Хромосома з батьківського пулу, що належить до схеми S (тобто хромосома з множини $M(k)\cap S$) залишиться в цій схемі тоді і тільки тоді, коли жоден символ цієї хромосоми, що відповідає сталим символам схеми S, не зміниться в процесі мутації. Ймовірність такої події дорівнює $(1\text{-}p_m)^{o(S)}$.

Цей результат можна представити у формі наступного висновку:

Висновок 3.4 (вплив мутації)

Ймовірність того, що деяка хромосома з $M(k)\cap S$ буде належати до схеми S після операції мутації, визначається виразом

$$(1\text{-}p_m)^{o(S)}$$

Ця величина називається **ймовірністю виживання схеми S після мутації**.

Висновок 3.5

Якщо ймовірність мутації p_m мала ($p_m << 1$), то можна вважати, що ймовірність виживання схеми S після мутації, яка визначена у висновку 3.4, приблизно дорівнює

$$1 - p_m o(S).$$

Ефект спільного впливу селекції, схрещування і мутації (висновки 3.1 - 3.4) з урахуванням факту, що якщо хромосома з множини $M(k)\cap S$ дає нащадка, що відповідає схемі S, то він буде належати до $P(k+1)\cap S$, веде до побудови наступної схеми репродукції:

$$E[c(S,k+1)] \geq c(S,k)\,\frac{F(S,k)}{\overline{F}(k)}\left(1\text{-}p_c\frac{d(S)}{L-1}\right)(1\text{-}p_m)^{o(s)} \qquad (3\ 10)$$

Залежність (3.10) показує, як змінюється від популяції до популяції кількість хромосом, що відповідають даній схемі. Ця зміна викликається трьома факторами, представленими в правій частині виразу (3.10), зокрема: F(S,k)/F(k) відбиває роль середнього значення функції пристосованості, $1\text{-}p_c d(S)/(L\text{-}1)$ показує вплив схрещування і $(1\text{-}p_m)^{o(S)}$ - вплив мутації. Чим більше значення кожного з цих факторів, тим більшим виявляється очікувана кількість відповідностей схемі S у наступній популяції. Висновок 3.5 дозволяє представити залежність (3.10) у виді

$$E[c(S,k+1)] \geq c(S,k)\,\frac{F(S,k)}{\overline{F}(k)}\left(1\text{-}p_c\frac{d(S)}{L-1} - p_m o(s)\right) \qquad (3\ 11)$$

Для великих популяцій залежність (3.11) можна апроксимувати виразом

$$c(S,k+1) \geq c(S,k)\,\frac{F(S,k)}{\overline{F}(k)}\left(1\text{-}p_c\frac{d(S)}{L-1} - p_m o(s)\right) \qquad (3\ 12)$$

З формул (3.11) і (3.12) випливає, що очікувана кількість хромосом, що відповідають схемі S у наступному поколінні, можна вважати функцією від фактичної кількості хромосом, що належать цій схемі, відносній пристосованості схеми, а також порядку й

охопленн. схеми. Помітно, що схеми з пристосованістю вище середньої і з малим порядком і охопленням характеризуються зростанням кількості своїх представників у наступних популяціях. Подібний ріст має показниковий характер, що випливає з виразу (3.9). Для великих популяцій цю формулу можна замінити рекурентною залежністю виду

$$c(S,k+1) = c(S,k)\,\frac{F(S,k)}{\overline{F}(k)} \qquad (3.13)$$

Якщо допустити, що схема S має пристосованість на ε% вище середньої, тобто

$$F(S,k) = \overline{F}(k) + \varepsilon\,\overline{F}(k),$$

то при підстановці виразу (3.12) у нерівність (3.11) у припущенні,що ε не змінюється в часі, при старті від k = 0 одержуємо

$$c(S,k) = c(S,0)(1+\varepsilon)^k,$$
$$\varepsilon = (F(S,k) - \overline{F}(k))/\,\overline{F}(k), \qquad (3.15)$$

тобто ε > 0 для схеми з пристосованістю вище середньої і ε<0 – у противному випадку.

Рівність (3.15) описує геометричну прогресію. З цього випливає, що в процесі репродукції схеми, які виявилися кращими (гіршими) середніх, вибираються на чергових ітераціях генетичного алгоритму в показниково зростаючих (убуваючих) кількостях. Зверніть увагу, що залежності (3.9) - (3.13) засновані на припущенні, що функція пристосованості F приймає тільки додатні значення. При використанні генетичних алгоритмів для рішення оптимізаційних задач, у яких цільова функція може приймати і від'ємні значення, необхідні деякі додаткові співвідношення між оптимізуємою функцією і функцією пристосованості. Кінцевий результат, одержуваний з виразів (3.10) - (3.12), можна сформулювати у формі теореми. Це основна теорема генетичних алгоритмів, інакше називана теоремою про схеми .

Теорема 3.1

Схеми малого порядку, з малим охопленням і з пристосованістю вище середньої формують показниково зростаючу кількість своїх представників у наступних поколіннях генетичного алгоритму.

Відповідно до приведеної теореми важливим питанням стає кодування, що повинне забезпечувати побудову схем малого порядку, з малим охопленням і з пристосованістю вище середньої. Непрямим результатом теореми 3.1 (про схеми) можна вважати наступну гіпотезу, названу *гіпотезою про цеглинки (або про будівельні блоки)*.

Гіпотеза 3.1

Генетичний алгоритм прагне досягти близького до оптимального результату за рахунок комбінування гарних схем (із пристосованістю вище середньої) малого порядку і малого охоплення. Такі схеми називаються *цеглинками (або будівельними блоками)*.

Гіпотеза про будівельні блоки висунута на підставі теореми про схеми з урахуванням того, що генетичні алгоритми досліджують простір пошуку за допомогою схем малого порядку і малого охоплення, які згодом беруть участь в обміні інформацією при схрещуванні. Незважаючи на те, що для доведення цієї гіпотези виконувалися значні дослідження, однак у більшості нетривіальних доданків приходиться спиратися на емпіричні результати. Протягом останніх років опубліковано численні роботи, присвячені застосуванням генетичних алгоритмів, що підтверджують цю гіпотезу. Якщо вона вважається правильною, то проблема кодування здобуває критичне значення для генетичного алгоритму; кодування повинне реалізувати концепцію малих будівельних блоків. Якість, яка забезпечує генетичним алгоритмам явну перевагу перед іншими традиційними методами, безсумнівно полягає в обробці великої кількості різних схем.

Проаналізуємо обробку схем генетичним алгоритмом на прикладі.

Приклад 3.7

Розглянемо приклад, що полягає в знаходженні хромосоми з максимальною кількістю одиниць. Припустимо, що хромосоми складаються з 12 генів, а популяція нараховує 8 хромосом. (Зрозуміло, що найкращою буде хромосома, що складається з 12 одиниць).

В умовах цього приклада розглянемо схему S_0 = **********11 і покажемо, як змінюється кількість представників цієї схеми і пристосованість у процесі виконання генетичного алгоритму. Довжина L=12, а охоплення і порядок схеми S0 складають відповідно $d(S_0)=1$ і $o(S_0)=2$. У вихідній популяції з приклада схемі S0 відповідають дві наступні хромосоми:

ch_3 = [011101110011]

ch_7 = [101011011011]

З формули (3.10) випливає, що після селекції і схрещування кількість хромосом, що відповідають схемі S_0, повинна бути більше або дорівнює 2,5. Нагадаємо, що ймовірності схрещування і мутації вважаються рівними відповідно $p_c = 1$ і $p_m = 0$. Пристосованість схеми S_0 у вихідній популяції, що позначається $F(S_0, 0)$, дорівнює 8 і перевищує середню пристосованість усіх хромосом цієї популяції F=5,75, що легко розрахувати по формулах (3.6) - (3.8).

У прикладі 3.4 після селекції і схрещування в новій популяції отримані чотири хромосоми, що відповідають схемі S0:

Ch_1 = [001111011011]

Ch_3 = [111011011011]

Ch_7 = [011101011011]

Ch_8 = [101011110011]

Пристосованість схеми S_0 у новій популяції, тобто $F(S_0, 1)$, складе 8,25, тоді як середня пристосованість хромосом цієї популяції F(1)=7, що також випливає з формул (3.6) - (3.8). Нова популяція характеризується великим середнім значенням функції пристосованості особин у порівнянні з попередньою (вихідною) популяцією, що уже відзначалося в прикладі 3.4. Крім того, у новій популяції пристосованість схеми S_0 виявляється кращою, а кількість представників цієї схеми - великим у порівнянні з попередньою популяцією.

4. Будівельні блоки (Building blocks)

Будівельний блок (ББ) (*building block (BB*)), як ми вже відзначали, - це одне з ключових понять у теорії генетичних алгоритмів, поряд зі схемою і теоремою схем. До будівельних блоків (як правило) прийнято відносити схеми з малою визначальною довжиною, малим порядком і високою пристосованістю. Пристосованість ББ найчастіше визначається як середня пристосованість особин, хромосоми яких містять даний будівельний блок. У гіпотезі про будівельні блоки (building blocks hypothesis) вважається, що в процесі роботи генетичного алгоритму, у міру наближення до глобального оптимуму, порядок і пристосованість будівельного блоку збільшуються. Тобто на початку еволюції відносно високою пристосованістю володіють будівельні блоки малої визначальної довжини, потім, у результаті добору і рекомбінації, з'являються більш пристосовані і "довгі" будівельні блоки. Таким чином, говорять про обробку будівельних блоків (building blocks processing) у результаті роботи генетичного алгоритму.

Виділяють наступні проблеми в обробці будівельних блоків:

1. Ідентифікація.
2. Змішування.

Під ***ідентифікацією*** розуміють проблему знаходження (локалізації) будівельного блоку. Іншими словами, яку саме ділянку хромосоми можна вважати будівельним блоком.

Очевидно, що мінімальний по розмірах ББ представляє один розряд, а максимальний - усю хромосому. Але це в найпростішому варіанті, а у випадку "не граничних умов" кількість "претендентів" на звання будівельного блоку дуже велика. Зокрема, для хромосоми довжиною n може існувати C(n, 2) різних позицій для будівельних блоків 2-го порядку, C(n, 3) різних позицій для будівельних блоків третього порядку і т.д., де C(n, m) - кількість сполучень з n по m (обчислюється як n!/(m!(n-m)!), де "!" - факторіал, тобто n! = 1*2*...*(n-1)*n, де "*" - позначає операцію множення).

Проблема ***змішування*** полягає в тому, що, навіть якщо будівельні блоки знайдені, важко визначити, які з них варто змішувати (грубо говорячи, поміщати в одну хромосому), а які – ні, так як висока пристосованість різних будівельних блоків не гарантує високої пристосованості хромосом, отриманих у результаті їхньої комбінації.

У силу позначених проблем приділяється досить багато уваги розробці операторів і підходів, що дозволили б забезпечити ефективну (BB-wise) обробку будівельних блоків. Також існує думка, що ефективна ідентифікація і змішування є критичними умовами для забезпечення успішної роботи генетичного алгоритму.

Крім ідентифікації і змішування існує проблема *невизначеності пристосованості будівельного блоку (building block fitness noise)*. Оскільки той самий будівельний блок може входити як у гарні (з погляду розв'язуваної задачі) хромосоми, так і в не дуже, то пристосованість цього будівельного блоку практично завжди визначається з деяким "шумом". Це ускладнює задачу вибору між двома будівельними блоками, тому що навіть якщо пристосованість одного з них вища, ніж пристосованість іншого, завжди є ймовірність зробити неправильний вибір. Говорячи по-іншому, у результаті селекції може бути обрана особа з менш пристосованим будівельним блоком. Графічно це можна представити в такий спосіб (рис.3.24)

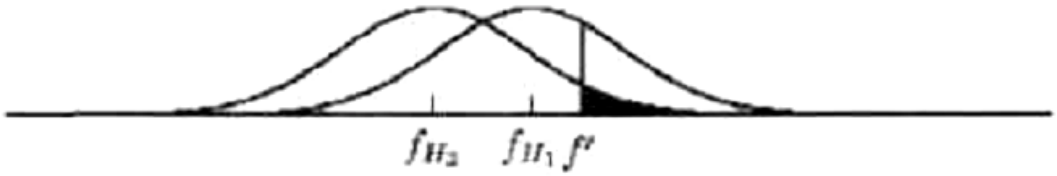

Рис. 3.24.

Нехай H1 і H2 два будівельних блоки, пристосованості яких розподілені по нормальному закону, причому f_{H_1} і f_{H_2} відповідно середні пристосованості блоків 1 і 2. Нехай також пристосованість H_1 більша, ніж пристосованість H_2. Якщо є невелике число хромосом, що містять розглянуті блоки, то їх пристосованості визначені з досить великою похибкою, іншими словами, дисперсія розподілів велика. Таким чином, площа взаємного "накладання" розподілів також значна, і, отже, значна ймовірність вибрати в результаті селекції менш пристосований будівельний блок. У випадку, представленому на рис. 3.24, ймовірність вибрати хромосому з пристосованістю більше f' і яка утримує схему H_2, дорівнює площі зафарбованої частини розподілу пристосованості 2-ї схеми. Якщо збільшити кількість хромосом які утримують розглянуті будівельні блоки (наприклад, збільшивши розмір популяції), то тоді розподіли пристосувань обох блоків "стискуються", так як значення пристосувань визначаються більш точно (рис.3.25). Тим самим зменшується ймовірність вибору менш пристосованого будівельного блоку.

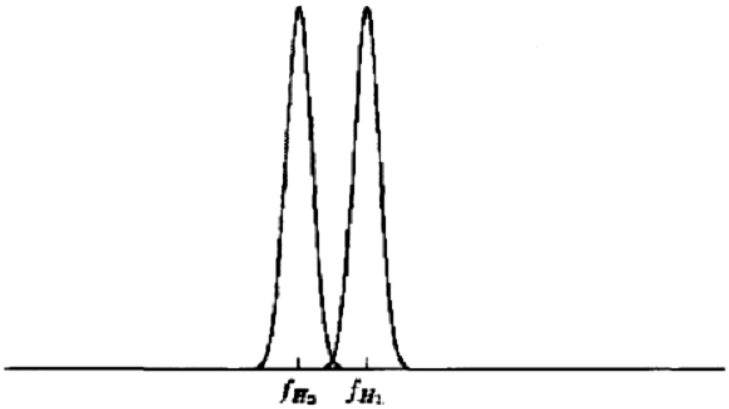

Рис. 3.25.

Незважаючи на те, що споконвічно будівельні блоки визначалися як схеми з визначеними властивостями, тобто стосовно до ГА з бінарним кодуванням, аналоги схем, а отже і будівельних блоків, можна знайти й в інших видах еволюційних обчислень (не тільки в генетичних алгоритмах). Вочевидь, концепція будівельних блоків давно вже живе "своїм життям", окремо від теореми схем, так як за ББ можна вважати будь-яку ділянку хромосоми визначеного виду поза залежністю від того, чи можливо описати хромосоми, що містять цей блок, однією схемою чи ні.

5. Еволюційні алгоритми

Генетичні алгоритми (genetic algorithms) разом з еволюційною стратегією й еволюційним програмуванням представляють три головних напрямки розвитку так називаного еволюційного моделювання (simulated evolution).

Незважаючи на те, що кожен з цих методів виник незалежно від інших, вони характеризуються рядом важливих загальних властивостей. Для кожного з них формується вихідна популяція особин, що у наступному піддається селекції і впливові різних генетичних операторів (найчастіше схрещуванню і/або мутації), що дозволяє знаходити більш гарні рішення.

Еволюційні стратегії - це алгоритми, створені як методи рішення оптимізаційних задач і засновані на принципах природної еволюції.

Еволюційне програмування являє собою підхід, запропонований американськими вченими спочатку в рамках теорії кінцевих автоматів і узагальнений пізніше для додатків до проблем оптимізації. Обидва напрямки виникли в шістдесятих роках XX століття.

Сконцентруємо увагу на найважливіших подібностях і розходженнях між еволюційними стратегіями і генетичними алгоритмами. Очевидно, що **головна подібність** полягає в тім, що обидва методи використовують популяції потенційних рішень і реалізують принцип селекції і перетворення найбільш пристосованих особин. Однак обговорювані підходи сильно відрізняються один від одного.

Перше розходження полягає в способі представлення особин. ***Еволюційні стратегії оперують векторами дійсних чисел, тоді як генетичні алгоритми - двійковими векторами***.

Друге розходження між еволюційними стратегіями і генетичними алгоритмами криється *в організації процесу селекції*. При реалізації еволюційної стратегії формується проміжна популяція, що складається з усіх батьків і деякої кількості нащадків, створених у результаті застосування генетичних операторів. За допомогою селекції розмір цієї проміжної популяції зменшується до величини батьківської популяції за рахунок виключення найменш пристосованих особин. Сформована в такий спосіб популяція утворює чергове покоління. Навпроти, у генетичних алгоритмах передбачається, що в результаті селекції з популяції батьків відбирається кількість особин, рівна розмірності

вихідної популяції, при цьому деякі (найбільш пристосовані) особи можуть відбиратися багаторазово. У той же час, менш пристосовані особи також мають можливість з'явитися в новій популяції. Однак шанси їхнього відбору пропорційні величині пристосованості особин. Незалежно від застосовуваного в генетичному алгоритмі методу селекції (наприклад, рулетки або рангового) більш пристосовані особи можуть відбиратися багаторазово. *При реалізації еволюційних стратегій особи відбираються без повторень*. В еволюційних стратегіях застосовується детермінована процедура селекції, тоді як у генетичних алгоритмах вона має випадковий характер.

Третє розходження між еволюційними стратегіями і генетичними алгоритмами стосується послідовності виконання процедур селекції і рекомбінації (тобто зміни генів у результаті застосування генетичних операторів). При реалізації еволюційних стратегій *спочатку виконується рекомбінація, а потім селекція*. У випадку виконання генетичних алгоритмів ця послідовність інвертується. При здійсненні застосування еволюційних стратегій нащадок утворюється в результаті схрещування двох батьків і мутації. Формована в такий спосіб проміжна популяція, що складається з усіх батьків і отриманих від них нащадків, надалі піддається селекції, що зменшує розмір цієї популяції до розміру вихідної популяції. При виконанні генетичних алгоритмів спочатку виконується селекція, що приводить до утворення перехідної популяції, після чого генетичні оператори схрещування і мутації застосовуються до особин (обираних з ймовірністю схрещування) і до генів (обираних з ймовірністю мутації).

Наступне розходження між еволюційними стратегіями і генетичними алгоритмами полягає в тім, що параметри генетичних алгоритмів (такі, як ймовірності схрещування і мутації) залишаються сталими протягом усього процесу еволюції, тоді як при реалізації еволюційних стратегій ці параметри піддаються безперервним змінам (так називана самоадаптація параметрів).

Ще одне розходження між еволюційними стратегіями і генетичними алгоритмами стосується трактування обмежень, що накладаються на розв'язувані оптимізаційні задачі. Якщо при реалізації еволюційних стратегій на деякій ітерації нащадок не задовольняє всім обмеженням, то він відкидається і включається в нову популяцію. Якщо таких нащадків виявляється багато, то еволюційна стратегія запускає процес адаптації параметрів, наприклад, шляхом збільшення ймовірності схрещування. У генетичних алгоритмах такі параметри не змінюються. У них може застосовуватися штрафна функція для тих особин, що не задовольняють накладеним обмеженням, однак ця технологія має багато недоліків.

В міру розвитку еволюційних стратегій і генетичних алгоритмів протягом останніх років істотні розходження між ними поступово зменшуються. Наприклад, при реалізації генетичних алгоритмів для рішення оптимізаційних задач усе частіше застосовується представлення хромосом дійсними числами і різні модифікації «генетичних» операторів, що має на меті підвищити ефективність цих алгоритмів. Подібні методи, що значно відрізняються від класичного генетичного алгоритму, за традицією зберігають колишню назву, хоча більш коректно було б називати їх «еволюційними алгоритмами».

Еволюційне програмування, також як і еволюційні стратегії, робить основний упор на адаптацію і розмаїтість способів передачі властивостей від батька до нащадків у наступних поколіннях. Познайомимося зі стандартним алгоритмом еволюційного програмування.

Вихідна популяція рішень вибирається випадковим чином. У задачах оптимізації значень дійсних чисел (прикладом яких може служити навчання нейронних мереж) особа (хромосома) представляється ланцюгом значень дійсних чисел. Ця популяція оцінюється щодо заданої функції (функції пристосованості). Нащадки утворюються від вхідних у цю популяцію батьків у результаті випадкової мутації. Селекція заснована на ймовірнісному виборі (турнірний метод), при якому кожне рішення змагається з хромосомами, випадковим чином обираними з популяції. Рішення-переможці (які є найкращими) стають

батьками для наступного покоління. Описана процедура повторюється так довго, поки не буде знайдено шукане рішення або не буде вичерпаний ліміт машинного часу.

Еволюційне програмування застосовується для оптимізації функціонування нейронних мереж. Так, як і інші еволюційні методи, воно не вимагає градієнтної інформації і тому може використовуватися для рішення задач, у яких ця інформація недоступна, або для її одержання потрібні значні обсяги обчислень. Одними з перших додатків еволюційного програмування вважаються задачі теорії штучного інтелекту, а самі ранні роботи стосувалися теорії кінцевих автоматів.

Спостерігається велика подібність між еволюційними стратегіями й еволюційним програмуванням у їхніх додатках до задач оптимізації безперервних функцій з дійсними значеннями. Деякі дослідники стверджують, що ці процедури, по суті, однакові, хоча вони і розвивалися незалежно одна від одної. Дійсно, обидва методи схожі на генетичні алгоритми. ***Принципове розходження між ними полягає в тому, що еволюційне програмування не зв'язане з конкретною формою представлення особин, оскільки оператор мутації не вимагає застосування якого-небудь спеціального способу кодування***.

Усі три представлених методи, тобто *генетичні алгоритми, еволюційні стратегії й еволюційне програмування поєднуються під загальною назвою еволюційні алгоритми (evolutionary algorithms)*. Також застосовується термін *еволюційні методи (evolutionary methods)*.

Еволюційними алгоритмами називаються й інші методи, що реалізують еволюційний підхід, зокрема, генетичне програмування (genetic programming), що представляє собою модифікацію генетичного алгоритму з урахуванням можливостей комп'ютерних програм. При використанні цього методу популяція складається з закодованих відповідним чином програм, що піддаються впливові генетичних операторів схрещування і мутації для знаходження оптимального рішення, яким вважається програма, що щонайкраще вирішує поставлену задачу. Програми оцінюються щодо визначеної спеціальним чином функції пристосованості. Цікавою представляється модифікація еволюційних алгоритмів для рішення оптимізаційних задач методом так називаної м'якої селекції, що запропонована Р. Галаром.

Для позначення різноманітних алгоритмів, заснованих на еволюційному підході, також застосовується *поняття еволюційних програм* (evolution programs). Цей термін поєднує як генетичні алгоритми, еволюційні стратегії й еволюційне програмування, так і генетичне програмування, а також інші аналогічні методи.

Еволюційні програми можна вважати узагальненням генетичних алгоритмів. Класичний генетичний алгоритм виконується при фіксованій довжині двійкових послідовностей і в ньому застосовуються оператори схрещування і мутації. Еволюційні програми обробляють більш складні структури (не тільки двійкові коди) і можуть виконувати інші «генетичні» операції. Наприклад, еволюційні стратегії можуть трактуватися як еволюційні програми, у яких хромосоми представляються дійсними (не двійковими) числами, а мутація використовується як єдина генетична операція.

Розглянемо *узагальнений приклад еволюційної програми*. Припустимо, що шукається граф, що задовольняє певним обмеженням (наприклад, виконується пошук топології комунікаційної мережі, оптимальної за конкретними критеріями, наприклад, по вартості передачі і т.п.). Кожна особа в еволюційній програмі представляє одне з потенційних рішень, тобто в даному випадку деякий граф. Вихідна популяція графів P(0), сформована випадковим чином або створювана при реалізації якого-небудь евристичного процесу, вважається відправною точкою (k=0) еволюційної програми. Функція пристосованості, що зазвичай задається, зв'язана із системою обмежень розв'язуваної задачі. Ця функція визначає «пристосованість» кожного графа шляхом виявлення «кращих» і «гірших» особин. Можна запропонувати кілька різних операторів мутації, призначених для

трансформації окремих графів, і трохи операторів схрещування, що будуть створювати новий граф у результаті рекомбінації структур двох або більш графів. Дуже часто такі оператори обумовлюються характером розв'язуваної задачі. Наприклад, якщо шукається граф типу «дерево», то можна запропонувати оператор мутації, що видаляє галузь з одного графа і додає нову галузь, що поєднує два окремих підграфи. Інші можливості полягають у проектуванні мутації незалежно від семантики задачі, але з включенням у функцію пристосованості додаткових обмежень - «штрафів» для тих графів, що не є деревами.

Принципову різницю між класичним генетичним алгоритмом і еволюційною програмою, тобто еволюційним алгоритмом у широкому змісті, ілюструють рис. 3.82 і 3.83.

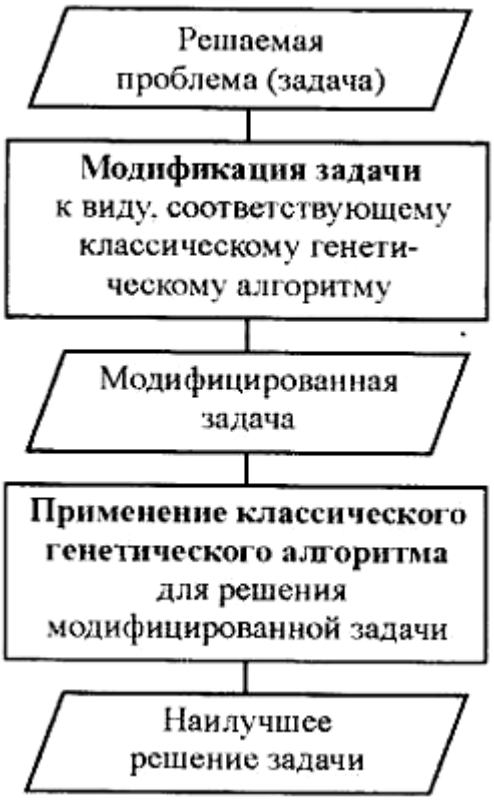

Рис. 4.82. Рішення задачі за допомогою класичного генетичного алгоритму

Класичний генетичний алгоритм, що оперує двійковими послідовностями, вимагає представити розв'язувану задачу в строго визначеному виді (відповідність між потенційними рішеннями і двійковими кодами, декодування і т.п.). Зробити це не завжди просто.

Рис. 3.83. Рішення задачі за допомогою еволюційного алгоритму (еволюційної програми)

Еволюційні програми можуть залишити постановку задачі в незмінному виді за рахунок модифікації хромосом, що представляють потенційні рішення (з використанням «природних» структур даних), і застосування відповідних «генетичних» операторів. Іншими словами, для рішення нетривіальної задачі можна або перетворити її до виду, необхідному для використання генетичного алгоритму (рис. 3.82), або модифікувати генетичний алгоритм так, щоб він задовольняв задачі (рис. 3.83).

При реалізації першого підходу застосовується класичний генетичний алгоритм, а при реалізації другого підходу - еволюційна програма Таким чином, модифіковані генетичні алгоритми можна в загальному випадку називати еволюційними програмами. Однак найчастіше зустрічається термін еволюційні алгоритми. Еволюційні програми також можуть розглядатися як еволюційні алгоритми, підготовлені програмістом для виконання на комп'ютері. Основна задача програміста полягає при цьому у виборі відповідних структур даних і «генетичних» операторів. Саме таке трактування поняття еволюційна програма представляється найбільш обґрунтованою.

Усі поняття, застосовувані в цьому розділі і стосовні головним чином до методів, заснованим на еволюційному підході, можна зіставити головному напрямкові досліджень - комп'ютерному моделюванню еволюційних процесів. Ця область інформатики називається Evolutionary Computation, що можна перевести як еволюційні обчислення.

До еволюційних алгоритмів також застосовується поняття технологія еволюційних обчислень. Можна додати, що назва генетичні алгоритми використовується як у вузькому змісті, тобто для позначення класичних генетичних алгоритмів і їхніх несуттєвих модифікацій, так і в широкому змісті - припускаючи будь-які еволюційні алгоритми, що значно відрізняються від «класики».

Останнім часом з'явилося багато нових методів, заснованих на еволюційному моделюванні, але базові технології, що використовуються - головним чином класичний генетичний алгоритм, еволюційні стратегії й еволюційне програмування.

6. Еволюційні алгоритми в нейронних мережах

Об'єднання генетичних алгоритмів і нейронних мереж відомо в літературі під абревіатурою COGANN (Combinations of Genetic Algorithms and Neural Networks). Це об'єднання може бути допоміжним (supportive) або рівноправним (collaborative). Допоміжне об'єднання двох методів означає, що вони застосовуються послідовно один за одним, причому один з них служить для підготовки даних, використовуваних при реалізації другого методу. При рівноправному об'єднанні обидва методи застосовуються одночасно. Класифікація цих типів об'єднань генетичних алгоритмів і нейронних мереж представлена в табл. 3.6.

Необхідно відзначити, що відповідно до зауважень, приведених в п. 7, термін «генетичні алгоритми» застосовується тут у більш широкому змісті, чим класичний генетичний алгоритм.

Таблиця 3.6. Об'єднання генетичних алгоритмів і нейронних мереж

Вид объединения	Характеристика объединения	Примеры использования
	Генетические алгоритмы и нейронные сети независимо применяются для решения одной и той же задачи	Однонаправленные нейронные сети, сети Кохонена с самоорганизацией и генетические алгоритмы в задачах классификации
Вспомогательное	Нейронные сети для обеспечения генетических алгоритмов	Формирование исходной популяции для генетического алгоритма
	Генетические алгоритмы для обеспечения нейронных сетей	Анализ нейронных сетей
		Подбор параметров либо преобразование пространства параметров
		Подбор параметров либо правила обучения (эволюция правил обучения)

Равно-правное	Генетические алгоритмы для обучения нейронных сетей	Эволюционное обучение сети (эволюция весов связей)
	Генетические алгоритмы для выбора топологии нейронной сети	Эволюционный подбор топологии сети (эволюция сетевой архитектуры)
	Системы, объединяющие адаптивные стратегии генетических алгоритмов и нейронных сетей	Нейронные сети для решения оптимизационных задач с применением генетического алгоритма для подбора весов сети
		Реализация генетического алгоритма с помощью нейронной сети
		Применение нейронной сети для реализации оператора скрещивания в генетическом алгоритме

Незалежне застосування генетичних алгоритмів і нейронних мереж

Генетичні алгоритми і нейронні мережі можуть незалежно застосовуватися для рішення однієї і тієї ж задачі. Цей підхід ілюструється на рис. 3.132.

Рис. 3.132. Генетичний алгоритм і нейронна мережа незалежно застосовуються для рішення однієї і тієї ж задачі.

Наприклад, описані незалежні застосування нейронних мереж, генетичних алгоритмів і алгоритму KNN «найближчий сусід» (K - means nearest neighbour) для рішення задач класифікації. У літературі приводяться порівняння тришарової односпрямованої нейронної мережі з навчанням по методу зворотного поширення похибки (навчання з учителем), мережі Кохонена із самоорганізацією (навчання без учителя), системи класифікації, заснованої на генетичному алгоритмі, а також алгоритму KNN «найближчий сусід». Автори ряду робіт вважають незалежне застосування цих методів для рішення задачі автоматичної класифікації результатів ЭМГ (электроміографія - реєстрація електричної активності м'язів) допоміжним об'єднанням. Відомі й інші роботи, у яких порівнюються можливості застосування різних методів (зокрема, генетичних алгоритмів і нейронних мереж) для рішення тих самих задач. Прикладом

задачі, яку можна вирішити за допомогою як нейронної мережі, так і генетичного алгоритму, може служити задача про комівояжера.

Нейронні мережі для підтримки генетичних алгоритмів

Більшість дослідників вивчали можливості застосування генетичних алгоритмів для забезпечення роботи нейронних мереж. До нечисленних зворотних випадків відноситься гібридна система, призначена для рішення задачі трасування, що класифікується як приклад допоміжного об'єднання нейронних мереж і генетичних алгоритмів. У цій системі генетичний алгоритм використовується в якості оптимізаційної процедури, призначеної для знаходження найкоротшого шляху. Нейронна мережа застосовується при формуванні вихідної популяції для генетичного алгоритму. Цей підхід схематично ілюструється на рис. 3.133.

Рис. 3.133. Допоміжне об'єднання нейронної мережі з генетичним алгоритмом.

Генетичні алгоритми для підтримки нейронних мереж

Підхід, заснований на використанні генетичного алгоритму для забезпечення роботи нейронної мережі, схематично представлено на рис. 3.134.

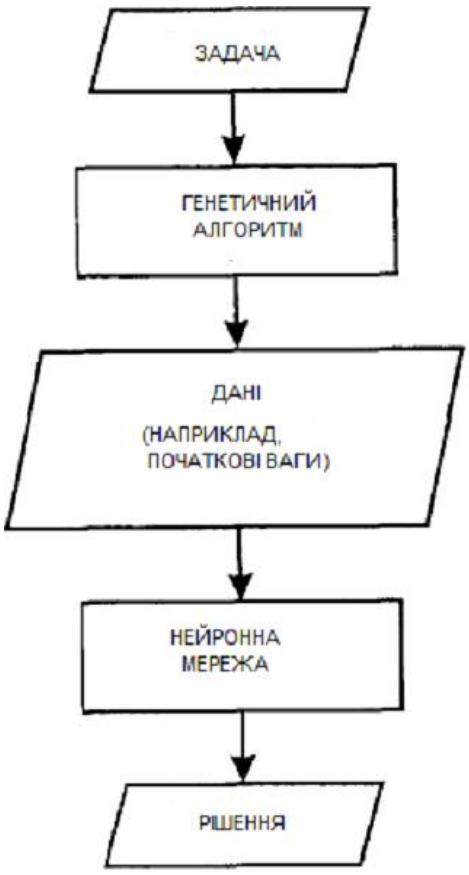

Рис. 3.134. Допоміжне об'єднання генетичного алгоритму з нейронною мережею.

Відомо багато робіт, присвячених подібному об'єднанню методів. *Можна виділити три області проблем:*

- застосування генетичного алгоритму для підбору параметрів або перетворення простору параметрів, використовуваних нейронною мережею для класифікації;
- застосування генетичного алгоритму для підбору правила навчання або параметрів, керуючих навчанням нейронною мережею;
- застосування генетичного алгоритму для аналізу нейронної мережі.

Дві перші області додатка генетичних алгоритмів у нейронних мережах, узагалі говорячи, дозволяють поліпшувати функціонування мереж (тобто вирішують проблему синтезу), тоді як третя служить для аналізу їхнього функціонування. Почнемо обговорення з останньої позиції.

Аналіз нейронних мереж. Деякі дослідники застосовували генетичні алгоритми як допоміжний інструмент для з'ясування закономірностей функціонування нейронних мереж або аналізу ефективності їхньої роботи. Генетичний алгоритм використовувався для побудови «інструментальної системи», що полегшує розуміння функціонування мережі - попросту говорячи, для з'ясування, що і чому робить мережа. Таке розуміння необхідне для того, щоб нейромережевий класифікатор не сприймався в якості «чорної шухляди», що формує відповідь якимось таємничим чином, і щоб рішення по класифікації об'єктів були поясненими. Подібний «інструментарій» (explanation facilities) використовується в більшості експертних систем. Побудова цих інструментів для їхнього застосування в нейронних мережах вважається більш масштабною проблемою, що відноситься до аналізу мереж. Генетичний алгоритм застосовувався для побудови так званих кодових векторів (codebook vectors), що представляють собою вхідні сигнали, при яких функція активації конкретного вихідного нейрона мережі приймає максимальне або близьке до нього значення. Вхідні вектори представлялися в хромосомах множиною

дійсних чисел від 0,0 до 1,0. Аналізувалася нейронна мережа, яка призначена для рішення задачі класифікації. Аналогічний підхід застосовувався для мережі ART1 (часткового випадку ART із двійковими вхідними сигналами). За допомогою генетичного алгоритму також проводився аналіз нейронної мережі, використовуваної як модель асоціативного запам'ятовуючого пристрою. Приведені приклади характеризують допоміжне об'єднання генетичних алгоритмів і нейронних мереж, хоча і не можуть вважатися типовими стосовно схеми, представленої на рис. 3.134.

Підбір параметрів або перетворення простору параметрів.

Генетичний алгоритм використовується при підготовці даних для нейронної мережі, яка грає роль класифікатора. Ця підготовка може виконуватися шляхом перетворення простору параметрів або виділенням деякого підпростору, що містить необхідні параметри.

Перший з цих методів, так називане перетворення простору параметрів, застосовується найчастіше в алгоритмах типу «найближчий сусід», хоча відомі також його доданки в нейромережних класифікаторах. Другий підхід полягає у виділенні підмножини параметрів, що враховуються. Виявляється, що обмеження множини параметрів часто поліпшує функціонування нейронної мережі як класифікатора і, до того ж, скорочує обсяги обчислень. Подібне обмеження множини що враховуються нейронною мережею параметрів застосовувалося, зокрема, для контролю сценаріїв подій на ядерних об'єктах, а також для розпізнавання китайських ієрогліфів. Відомі й інші приклади підготовки даних для нейронних мереж за допомогою генетичних алгоритмів.

Підбір параметрів і правил навчання. Генетичний алгоритм також застосовується для підбора параметрів навчання - найчастіше швидкості навчання (learning rate) і так називаного моменту для алгоритму зворотного поширення похибки. Таке адаптивне уточнення параметрів алгоритму зворотного поширення (вони кодуються в хромосомах) у результаті еволюції може розглядатися як перша спроба еволюційної модифікації правил навчання. Замість безпосереднього застосування генетичного алгоритму для підбору параметрів навчання розвивається еволюційний підхід, спрямований на побудову оптимального правила (алгоритму) навчання.

Помітимо, що еволюційна концепція вже може розглядатися як перехід від допоміжного до рівноправного об'єднання генетичного алгоритму і нейронних мереж.

Застосування генетичних алгоритмів для навчання нейронних мереж

Думка про те, що нейронні мережі можуть навчатися за допомогою генетичного алгоритму, висловлювалася різними дослідниками. Перші роботи на цю тему стосувалися застосування генетичного алгоритму як метод навчання невеликих односпрямованих нейронних мереж, але в наступному було реалізовано застосування цього алгоритму для мереж з більшою розмірністю.

Як правило, задача полягає в оптимізації ваг нейронної мережі, що має апріорі задану топологію. Ваги кодуються у виді двійкових послідовностей (хромосом). Кожна особа популяції характеризується повною множиною ваг нейронної мережі. Оцінка пристосованості особин визначається функцією пристосованості, що задається у виді суми квадратів похибок, тобто різностей між очікуваними (еталонними) і фактично одержуваними значеннями на виході мережі для різних вхідних даних.

Приведемо два найважливіших аргументи на користь застосування генетичних алгоритмів для оптимізації ваг нейронної мережі. Насамперед, генетичні алгоритми забезпечують глобальний перегляд простору ваг і дозволяють уникати локальні мінімуми. Крім того, вони можуть використовуватися в задачах, для яких інформацію про градієнти одержати дуже складно або вона виявляється занадто дорогою.

Генетичні алгоритми для вибору топології нейронних мереж

Як найбільш очевидний спосіб об'єднання генетичного алгоритму з нейронною мережею інтуїтивно сприймається спроба закодувати в генотипі топологію об'єкта (у розглянутому випадку - нейронної мережі) із указівкою кількості нейронів і зв'язків між ними при наступному визначенні ваг мережі за допомогою будь-якого відомого методу.

Проектування оптимальної топології нейронної мережі може бути представлене у виді пошуку такої архітектури, яка забезпечує найкраще (щодо обраного критерію) рішення конкретної задачі. Такий підхід припускає перебір простору архітектури, складеного з усіх можливих варіантів, і вибір точки цього простору, найкращої щодо заданого критерію оптимальності.

З урахуванням достоїнств еволюційного проектування архітектури в останні роки була виконана велика кількість досліджень, у яких основна увага приділялася еволюції з'єднань нейронної мережі, тобто кількості нейронів і топології зв'язків між ними. Лише в деяких роботах розглядалася еволюція функцій переходів, хоча ці функції вважаються важливим елементом архітектури і впливають на функціонування нейронної мережі.

Також, як і у випадку еволюційного навчання, перший крок еволюційного проектування архітектури полягає у формуванні вихідної множини розглянутих варіантів.

Адаптивні взаємодіючі системи

До рівноправного об'єднання генетичних алгоритмів і нейронних мереж варто віднести комбінацію адаптивних стратегій обох методів, що складає єдину адаптивну систему. Можна привести три приклади систем такого типу.

Перший з них - це нейронна мережа для оптимізаційної задачі з генетичним алгоритмом для визначення ваг мережі.

Другий приклад відноситься до реалізації генетичного алгоритму за допомогою нейронної мережі. У цьому випадку нейронні підсистеми застосовуються для виконання генетичних операцій репродукції і схрещування.

У третьому прикладі, трохи схожому на попередній, нейронна мережа також застосовується як оператор схрещування в генетичному алгоритмі, призначеному для рішення оптимізаційних задач.

Представлені приклади стосуються такого рівноправного об'єднання генетичних алгоритмів і нейронних мереж, що у результаті дозволяє одержати більш ефективний алгоритм, який поєднує кращі якості обох методів.

Типовий цикл еволюції

Як тільки визначений вид еволюції вводиться в штучну нейронну мережу, відразу виникає потреба у відповідній йому схемі хромосомного представлення даних, тобто повинний бути створений спосіб генетичного кодування особин популяції. Розробка способу кодування вважається першим етапом такого еволюційного підходу, поряд з яким типовий процес еволюції включає наступні кроки:

- декодування;
- навчання;
- оцінювання пристосованості;
- репродукція;
- формування нового покоління.

Приведена на рис. 3.12 блок-схема зберігає свою актуальність, оскільки (як уже згадувалося в п. 3.18) вона відображає і класичний генетичний алгоритм, і так називані еволюційні програми, що засновані на генетичному підході й узагальнюють його принципи. Отже, цій універсальній блок-схемі відповідають різні еволюційні алгоритми, і

в кожнім з них у першу чергу повинна бути згенерована вихідна популяція хромосом. За аналогією з класичним генетичним алгоритмом ініціалізація (тобто формування цієї вихідної популяції) полягає у випадковому виборі необхідної кількості хромосом, які включаються в неї, що припускає відповідне генетичне кодування кожної особи. У класичному генетичному алгоритмі хромосоми представляються тільки двійковими послідовностями. При еволюційному підході вибір способу кодування являє собою важливу й актуальну задачу.

Далі відповідно до типового циклу еволюції випливає необхідність декодування кожної особи (хромосому) вихідної або поточної популяції для того, щоб одержати множину рішень (фенотипів) даної задачі. У випадку еволюції ваг, архітектур і/або правил навчання фенотипи представляють відповідно множини ваг, архітектур і правил навчання.

Згодом відповідно до генетичного алгоритму розраховуються значення функції пристосованості особин вихідної (або поточної) популяції. При нейромережному підході після декодування хромосом виходить множина нейронних мереж, для яких степінь пристосованості визначається за результатами навчання цих мереж.

При реалізації типового циклу еволюції необхідно сконструювати множину відповідних нейронних мереж (фенотипів):

- мережі з фіксованою архітектурою і множиною закодованих хромосомами ваг - у випадку еволюції ваг;
- мережі з закодованою хромосомами архітектурою - у випадку еволюції архітектури;
- мережі з випадково згенерованими архитектурами і початковими вагами - у випадку еволюції правил навчання.

Після навчання оцінюється пристосованість кожної особи, що входить у поточну популяцію. Помітимо, що також як і в прикладі максимізації функції (приклад 3.5), для оцінювання пристосованості хромосом необхідно їх спочатку декодувати і лише потім розрахувати значення функції пристосованості особин по їхніх фенотипах.

Наступний крок генетичного алгоритму - це селекція хромосом. Вибираються хромосоми, що підлягають репродукції, тобто формується батьківський пул, особи якого в результаті застосування генетичних операторів сформують популяцію нащадків. Селекція може бути заснована на методі рулетки або будь-якому іншому, наприклад, по алгоритму Уітлі (Whitley). Згідно з цими методами селекція виконується з ймовірністю, пропорційною пристосованості хромосом, або відповідно до їх рангу (при використанні рангового методу). Під репродукцією в даному випадку розуміється процес відбору (селекції) і копіювання (розмноження) хромосом для формування з них перехідної популяції (батьківського пула), особи якої будуть піддаватися впливові генетичних операторів схрещування, мутації і, можливо, інверсії.

Застосування генетичних операторів за обраним методом селекції хромосом відбувається аналогічно класичному генетичному алгоритмові, причому ці оператори можуть відрізнятися від схрещування і мутації базового алгоритму. Як відзначалося в п. 3.18, для конкретної задачі генетичні оператори можуть визначатися в індивідуальному порядку.

Також як і в класичному генетичному алгоритмі, у результаті застосування генетичних операторів з обраним методом селекції хромосом формується нова популяція особин (нащадків). Наступні кроки алгоритму повторюються для чергової популяції аж до виконання умови завершення генетичного алгоритму. На кожній ітерації формується нове покоління нащадків.

Найкраща особа з останнього покоління вважається шуканим рішенням даної задачі. У такий спосіб виходить найкраща множина ваг, найкраща архітектура або найкраще правило навчання.

ЛІТЕРАТУРА

1. Кононюк А.Ю. Нейроні мережі і генетичні алгоритми – К.:«Корнійчук», 2008. – 446 с. SBN 978-966-7599-50

2. Ясницкий Л.Н. Введення в штучний інтелект. — 1-е. — Издательский центр "Академия", 2005. — С. 176. — ISBN 5-7695-1958-4

3. Г. К. Вороновский, К. В. Махотило, С. Н. Петрашев, С. А. Сергєєв. Генетичні алгоритми, штучні нейроні мережі і проблеми віртуальної реальності. — Замовне. — Х.: ОСНОВА, 1997. — С. 112. — ISBN 5-7768-0293-8

4. Д. Рутковская, М. Пилиньский, Л. Рутковский. Нейронные сети, генетические алгоритмы и нечеткие системы —1-е. — М.: Гарячая линия – Телеком, 2007. — С. 384. — ISBN 5-93517-103-1