Тема №5 ЕВОЛЮЦІЙНІ ТЕХНОЛОГІЇ ТА ГЕНЕТИЧНІ АЛГОРИТМИ

План

5.1. Концептуальні засади еволюційної теорії.
5.2. Основні положення теорії генетичних алгоритмів.
5.3. Моделі генетичних алгоритмів. Канонічний ГА.
5.4. Програмне забезпечення та сфери застосування генетичних алгоритмів.

Література:

Основна

Черняк О. І., Захарченко П. В. Інтелектуальний аналіз даних: Підручник. Київ, 2014

Додаткова

1. Гладков Л. А., Курейчик В. В., Курейчик В. М. Генетические алгоритмы. Москва : Физматлит, 2006.
2. Дюк В. А., Самойленко А. П. Data Mining: учебный курс. Санкт-Петербург : Питер, 2001.
3. Емельянов В. В.Теория и практика эволюционного моделирования. Москва :Физматлит, 2003.
4. Каширина И. Л. Введение в эволюционное моделирование.–Воронеж : ВГТУ, 2007.
5. Панченко Т. В. Генетические алгоритмы. Астрахань: Астраханский университет, 2007.
6. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. Москва : Горячая Линия Телеком, 2007.
7. Чубукова И. А. Data Mining. Москва : БИНОМ, 2008.

5.1. Концептуальні засади еволюційної теорії

Еволюційна теорія, синонімом якої в зарубіжній літературі є термін *«Evolutionary computation»*, довела свою ефективність як при вирішенні складно формалізованих задач штучного інтелекту (розпізнавання образів, кластеризація, асоціативний пошук), так і при вирішенні трудомістких задач оптимізації, апроксимації, інтелектуальної обробки даних. До переваг еволюційної теорії відносяться адаптивність, здібність до навчання, паралелізм, можливість побудови гібридних інтелектуальних систем на основі комбінування з парадигмами штучних нейромереж і нечіткої логіки. Багатообіцяючою виглядає передумова створення єдиної концепції еволюційних обчислень, що включають генетичні алгоритми, генетичне програмування, еволюційні стратегії і еволюційне програмування. На думку багатьох дослідників, ці парадигми є аналогами процесів, що відбуваються в

живій природі, і на практиці довели свою непримітивність. Один з піонерів еволюційної теорії Л. Фогель взагалі бачить теорію еволюції і самоорганізації як базову концепцію для всіх інтелектуальних процесів і систем, що значно розширює сферу застосування традиційної парадигми інтелектуального аналізу даних. Навіть якщо це не так, і в природі відбувається революция, ніхто не може сказати, що алгоритми еволюційних обчислень невірні.

Основна теза підходу, названого еволюційним, – замінити процес моделювання складного об'єкту моделюванням його еволюції. Він спрямований на застосування механізмів природної еволюції при аналізі складних систем інтелектуальної обробки інформації. У своїй теорії походження видів Ч. Дарвін відкрив і обґрунтував основний закон розвитку органічного світу, охарактеризувавши його взаємодією трьох наступних чинників:

– спадкоємній мінливості;

– боротьби за існування;

– природного відбору.

Еволюційні обчислення – термін, зазвичай використовують для загального опису алгоритмів пошуку, оптимізації або навчання, заснованих на формалізованих принципах природного еволюційного процесу. Еволюційні методи призначені для пошуку переважних рішень і засновані на статистичному підході до дослідження ситуацій та ітераційному наближенні до шуканого стану систем. На відміну від точних методів дослідження операцій еволюційні методи дозволяють знаходити рішення, близькі до оптимальних, за прийнятний час, а у відмінності від відомих евристичних методів оптимізації характеризуються істотно меншою залежністю від особливостей додатку (тобто більш універсальні) і у багатьох випадках забезпечують кращий ступінь наближення до оптимального рішення. Основна перевага еволюційної теорії полягає в можливості вирішення багатомодальних (що мають декілька локальних екстремумів) задач із великою розмірністю за рахунок поєднання елементів випадковості і детермінованості точно так, як це відбувається в природному середовищі.

Спроба порівняльного аналізу конкуруючих евристичних алгоритмів з погляду їх результативності демонструє два важливі практичні наслідки:

1. Пошук еволюційного алгоритму, який перевершує всі алгоритми, що конкурують з ним, не має сенсу без точного опису конкретних задач і цільових функцій, для яких еволюційний алгоритм має перевагу перед іншими алгоритмами. Не можна розраховувати знайти один алгоритм, який буде результативніше останніх для будь-яких цільових функцій оптимізації.

2. Щоб знайти добре рішення для заданого класу задач, необхідно спочатку ідентифікувати характеристичні особливості класу задач і тоді на їх основі шукати відповідний алгоритм.

З цієї точки зору еволюційні обчислення володіють наступними достоїнствами і недоліками.

Достоїнства еволюційних обчислень:

– широка сфера застосування;

– можливість проблемно-орієнтованого кодування рішень, підбору початкової популяції, комбінування еволюційних обчислень з не еволюційними алгоритмами, продовження процесу еволюції до тих пір, поки є необхідні ресурси;

– придатність для пошуку в складному просторі рішень великої розмірності;

– відсутність обмежень на вигляд цільової функції;

– ясність схеми і базових принципів еволюційних обчислень;

– інтегрованість еволюційних обчислень з іншими некласичними парадигмами штучного інтелекту, такими, як штучні нейромережі і нечітка логіка.

Недоліками еволюційних обчислень є:

– евристичний характер еволюційних обчислень не гарантує оптимальності отриманого рішення (правда, на практиці, часто, важливо за заданий час отримати одне або декілька субоптимальних альтернативних рішень, тим більше, що початкові дані в задачі можуть динамічно мінятися, бути неточними або неповними);

– відносно висока обчислювальна трудомісткість, яка проте долається за рахунок розпаралелювання на рівні організації еволюційних обчислень і на рівні їх безпосередньої реалізації в обчислювальній системі;

– відносно невисока ефективність на завершальних фазах моделювання еволюції (оператори пошуку в еволюційних алгоритмах не орієнтовані на швидке попадання в локальний оптимум);

– невирішеність питань самоадаптації.

5.2. Основні положення теорії генетичних алгоритмів

Генетичний алгоритм (ГА) – це проста модель еволюції в природі, реалізована у вигляді комп'ютерної програми. Він широко використовується, для проектування структури мостів, для пошуку максимального відношення міцність/вага, для визначення найменш марнотратного розміщення при нарізці форм з тканини. ГА можуть також використовуватися для інтерактивного управління процесом, наприклад на хімічному заводі, або балансуванні завантаження на багатопроцесорному комп'ютері.

У генетичному алгоритмі використовується як аналог механізму генетичного спадкоємства, так і аналог природного відбору. При цьому зберігається біологічна термінологія в спрощеному вигляді. От як моделюється генетичне спадкоємство:

Хромосома	Вектор (послідовність) з нулів і одиниць. Кожна позиція (біт) називається геном.
Індивідуум = генетичний код	Набор хромосом = варіант рішення задачі.
Кросовер	Оцерація при якій дві хромосоми обмінюються своїми частинами.
Мутація	Випадкова зміна однієї або кількох позицій в хромосомі
Пристосованість індивідуума	Значення цільової функції на цьому індивідуумі
Виживання найбільш пристосованих	Популяція наступного покоління формується відповідно до цільової функції. Чим пристосованіше індивідуум, тим більше вірогідність його участі в кросовері, тобто розмноженні.

Щоб змоделювати еволюційний процес, згенеруємо спочатку випадкову популяцію – декілька індивідуумів з випадковим набором хромосом (числових векторів). Генетичний алгоритм імітує еволюцію цієї популяції як циклічний процес схрещування індивідуумів і зміни поколінь (рис. 5.1).

Життєвий цикл популяції – це декілька випадкових схрещувань (за допомогою кросовера) і мутацій, в результаті яких до популяції додається якась кількість нових індивідуумів. Відбір в генетичному алгоритмі – це процес формування нової популяції із старої, після чого стара популяція гине. Після відбору до нової популяції знову застосовуються операції кросовера і мутації, потім знову відбувається відбір, і так далі.

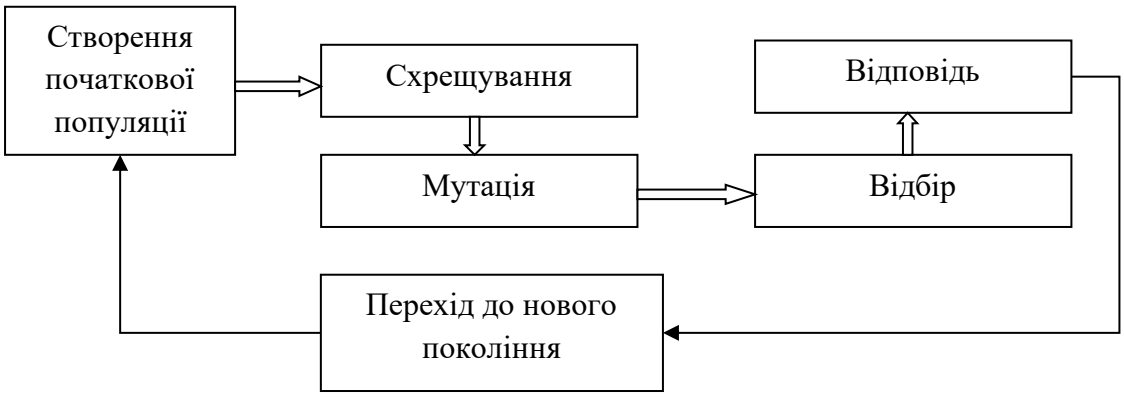

Рис. 5.1. Алгоритм генетичних обчислювань

Відбір в генетичному алгоритмі тісно пов'язаний з принципами природного відбору в природі таким чином:

Створення початкової популяції →
Схрещування →
Мутація →
Відбір →
Відповідь →

Перехід до нового покоління

Таким чином, модель відбору визначає, яким чином слід будувати популяцію наступного покоління. Як правило, вірогідність участі індивідуума в схрещуванні береться пропорційній його пристосованості. Часто використовується так звана стратегія елітизма, при якій декілька кращих індивідуумів переходять в наступне покоління без змін, не беручи участь в кросовері і відборі. У будь-якому випадку кожне наступне покоління буде в середньому краще попереднього. Коли пристосованість індивідуумів перестає помітно збільшуватися, процес зупиняють і як рішення задачі оптимізації беруть якнайкращого із знайдених індивідуумів.

Повертаючись до задачі оптимального розподілу інвестицій, пояснимо особливості реалізації генетичного алгоритму в цьому випадку.

- Індивідуум = варіант рішення задачі = набір з 10 хромосом X_j.
- Хромосома X_j = об'єм вкладення в проект = 16-розрядний запис цього числа.
- Оскільки обсяги вкладень обмежені, не всі значення хромосом є допустимими. Це враховується при генерації популяцій.
- Оскільки сумарний обсяг інвестицій фіксований, то реально варіюються лише 9 хромосом, а значення 10-ої визначається по ним однозначно.

Таким чином, ***перевагами генетичних алгоритмів*** є:

– не вимагання жодної інформації про поверхню відповіді;

– розриви, що існують на поверхні відповіді мають незначний ефект та не впливають на ефективність оптимізації;

– стійкість до попадання в локальний оптимум;

– добре працюють при вирішенні великомасштабних проблем оптимізації;

Можуть бути використані для широкого класу задач, зокрема; з середовищем, що змінюється.

Недоліками генетичних алгоритмів слід вважати:

– складність роботи у разі, коли необхідно знайти точний глобальний оптимум;

– час виконання функції оцінки великий;

– конфігурація є не простою (кодування рішення).

5.3. Моделі генетичних алгоритмів. Канонічний ГА (Canonical GA - J. Holland)

Ця модель алгоритму є класичною. Вона була запропонована Джоном Холландом в його знаменитій роботі «Адаптація в природних і штучних середовищах». Часто можна зустріти опис простого ГА (*Simple GA*), він відрізняється від канонічного тим, що використовує або рулеточний, або турнірний відбір. Модель канонічного ГА має наступні характеристики.

Фіксований розмір популяції.

Фіксована розрядність генів.

Пропорційний відбір.

Одноточечний кросовер і одноточечна мутація.

Наступне покоління формується з нащадків поточного покоління без «елітизму».

1. Ініціалізація початкової популяції. Покласти номер епохи $t = 0$. Ініціалізувати випадковим чином генотипів особин і сформувати з них випадкову популяцію. Обчислити пристосованість особин популяції $F(0) = (f_1(0), \dots, f_m(0))$, а потім – середню пристосованість по попляції $f_{\text{ср}} = \sum_{i=1}^{m} f_i(0)/m$.

2. Вибір батьків для схрещування. Збільшити номер епохи на одиницю: $t = t + 1$. Визначити випадковим чином номер першого батька $l \in \{1, \dots, m\}$, призначивши імовірність випадання будь-якого номера пропорційній величині $f_h(t)/f_{\text{ср}}(t)$. Повторним випробуванням визначити номер другого батька k.

3. Формування генотипу нащадків. Із заданою ймовірністю p_c провести над генотипами вибраних батьків одноточковий кросовер. Далі до кожного з отриманих нащадків з імовірністю p_M застосувати оператора мутації.

4. Оновлення популяції. Помістити нащадків в популяцію, заздалегідь видаливши з неї батьків. Обчислити пристосованості нащадків і відновити значення середньої пристосованості популяції $f_{\text{ср}}(t)$. Якщо формування популяції не завершене, перейти до кроку 2.

Генітор (Genitor).

У моделі генітор використовується специфічний спосіб відбору .

Спочатку, як і завжди, популяція ініціалізується, і її особини оцінюються.

Потім вибираються випадковим чином дві особини, схрещуються, причому виходить лише один нащадок, який оцінюється і займає місце менш пристосованої особини в популяції (а не одного з батьків). Після цього знову випадковим чином вибираються дві особини, і їх нащадок займає місце батьківської особини з найнижчою пристосованістю. Таким чином, на кожному кроці в популяції оновлюється лише одна особина. Процес продовжується до тих пір, поки придатності хромосом не стануть однаковими. У даний алгоритм можна додати мутацію нащадка після його створення. Критерій закінчення процесу, як і вигляд кросинговера і мутації, можна вибирати різними способами. Підводячи підсумки, можна виділити наступні характерні особливості. Фіксований розмір популяції. Фіксована розрядність генів. Особини для схрещування вибираються випадковим чином. Обмежень на тип кросовера і мутації немає. В результаті схрещування особин виходить один нащадок, який займає місце найменш пристосованої особини.

Метод переривистої рівноваги.

Даний метод заснований на палеонтологічній теорії переривистої рівноваги, яка описує швидку еволюцію за рахунок вулканічних і інших змін

земної кори. Для вживання даного методу в задачах інтелектуального аналізу даних пропонується після кожної генерації проміжного покоління випадковим чином перемішувати особині в популяції, а потім застосовувати основний ГА. У даній моделі для відбору батьківських пар використовується панміксія. Нащадки, що вийшли в результаті кросинговера, і найбільш придатні батьки випадковим чином змішуються. Із загальної маси в нове покоління попадуть лише ті особини, придатність яких вище середньою. Тим самим досягається управління розміром популяції залежно від наявності кращих особин. Така модифікація методу переривистої рівноваги може дозволити скоротити неперспективні популяції і розширити популяції, в яких знаходяться кращі індивідуальності.

Гібридний алгоритм (Hybrid Algorithm - L. «Dave» Davis).

Ідея гібридних алгоритмів полягає в поєднанні генетичного алгоритму з деяким іншим класичним методом пошуку, відповідним до даної задачі. У кожному поколінні всі згенеровані нащадки оптимізуються вибраним методом і потім заносяться в нову популяцію. Тим самим виходить, що кожна особина в популяції досягає локального оптимуму, поблизу якого вона знаходиться. Далі виробляються звичайні для ГА дії: відбір батьківських пар, кросинговер і мутації. На практиці гібридні алгоритми виявляються дуже вдалими. Це пов'язано з тим, що вірогідність попадання однієї з особин в область глобального максимуму досить велика. Після оптимізації така особина буде рішенням задачі. Відомо, що генетичний алгоритм здатний швидко знайти у всій зоні пошуку хороші рішення, але він може зазнавати труднощі в здобутті з них найкращих. Звичайний оптимізаційний метод може швидко досягти локального максимуму, але не може знайти глобальний. Поєднання двох алгоритмів дозволяє використовувати переваги обох.

CHC (Eshelman).

CHC розшифровується як *Cross-population selection, Heterogeneous recombination and Cataclysmic mutation*. Даний алгоритм досить швидко сходиться через те, що в ньому немає мутацій, наступних за оператором кросинговера, використовуються популяції невеликого розміру, і відбір особин в наступне покоління ведеться і між батьківськими особинами, і між їх нащадками. У даному методі для кросинговера вибирається випадкова пара, але не допускається, аби між батьками була маленька хеммінгова відстань або мала відстань між крайніми бітами. Для схрещування використовується різновид однорідного кросовера *HUX (Half Uniform Crossover)*, при якому нащадкові переходить рівно половина бітів кожного батька. Для нового покоління вибираються *N* кращих різних особин серед батьків і дітей. При цьому дублювання рядків не допускається. У моделі *CHC* розмір популяції відносно малий – близько 50 особин. Це виправдовує використання однорідного кросинговера і дозволяє алгоритму зійтися до рішення. Для здобуття більш менш однакових рядків CHC застосовує cataclysmic mutation: всі рядки, окрім самого пристосованого, піддаються

сильній мутації (змінюється біля третини бітів). Таким чином, алгоритм перезапускається і далі продовжує роботу, застосовуючи лише кросинговер.

ГА з нефіксованим розміром популяції (Genetic Algorithm with Varying Population Size - GAVaPS).

У генетичному алгоритмі з нефіксованим розміром популяції кожній особини приписується максимальний вік, тобто число поколінь, після яких особина гине. Впровадження в алгоритм нового параметра – віку – дозволяє виключити оператор відбору в нову популяцію. Вік кожної особини індивідуальний і залежить від її пристосованості. У кожному поколінні t на етапі відтворення звичайним способом створюється додаткова популяція з нащадків. Розмір додаткової популяції (*AuxPopsize(t)*) пропорційний розміру основної популяції (*Popsize(t)*) і рівний $AuxPopsize(t) = [Popsize(t)\ p_c\ ]$, p_c - вірогідність відтворення. Для відтворення особині вибираються з основної популяції з однаковою імовірністю незалежно від їх пристосованості. Після вживання мутації і кросинговера нащадкам приписується вік згідно значенню їх пристосованості. Вік є константою впродовж всієї еволюції особини (від народження до загибелі). Потім з основної популяції видаляються ті особини, термін життя яких витік, і додаються нащадки з проміжної популяції. Таким чином, розмір після однієї ітерації алгоритму обчислюється за формулою $Popsize(t+1) = Popsize(t)+AuxPopsize(t)-D(t)$, де $D(t)$ – число особин, які вмирають в поколінні t .

Простий паралельний ГА (Parallel implementations).

Генетичні алгоритми застосовуються і при паралельних обчисленнях. При цьому формується декілька популяцій, що живуть окремо. На етапі формування нового покоління за деяким правилом відбираються особини з різних популяцій. Згенерована таким чином популяція, в деяких випадках замінює всі інші. Таким чином, так або інакше, відбувається міграція особин однієї популяції в інші популяції. Тому паралельні ГА називають *міграціями*. Спершу розглянемо створення простого паралельного ГА з класичної моделі Холланда. Для цього використовуватимемо турнірний відбір. Заведемо $N / 2$ процесів (тут і далі процес розглядається як деяка машина, процесор, якої може працювати незалежно). Кожен з них вибиратиме випадково з популяції 4 особини, проводити 2 турніри, і переможців схрещувати. Отримані нащадки записуватимуться в нове покоління. Таким чином, за один цикл роботи одного процесу змінюється ціле покоління.

Міграція (Migration).

Модель міграції представляє популяцію як множину підпопуляцій. Кожна підпопуляція обробляється окремим процесором. Ці під популяції розвиваються незалежно одна від одної протягом однакової кількості поколінь T (час ізоляції). Після закінчення часу ізоляції відбувається обмін особинами між підпопуляціями (міграція). Кількість особин, що піддалися обміну (вірогідність міграції), метод відбору особин для міграції і схема міграції визначає частоту виникнення генетичного різноманіття в

підпопуляціях і обмін інформацією між популяціями. Відбір особин для міграції може відбуватися таким чином:

– випадкова одноманітна вибірка з числа особин;

– пропорційний відбір: для міграції беруться найбільш придатні особини.

Окремі підпопуляції в паралельних ГА можна умовно прийняти за вершини деякого графа. У зв'язку з цим можна розглядати топологію графа міграційного ГА. Найбільш поширеною топологією міграції є повний граф (рис. 5.2), при якій особині з будь-якої підпопуляцій можуть мігрувати в будь-яку іншу підпопуляцію. Для кожної підпопуляції повна кількість потенційних іммігрантів будується на основі всіх підпопуляцій. Мігруюча особина випадковим чином вибирається з цього загального числа.

При використанні в необмеженій міграції пропорційного відбору спочатку формується масив з найбільш придатних особин, відібраних по всіх підпопуляціях. Випадковим чином з цього масиву вибирається особина, і нею замінюють найменш придатну особину в підпопуляції 1. Аналогічні дії проробляємо з останніми підпопуляціями. Можливо, що якась популяція отримає дублікат своєї «хорошої» особини.

Інша основна міграційна схема – це топологія кільця (рис. 5.3). Тут особини передаються між сусідніми (по напряму обходу) підпопуляціями.

Таким чином, особини з однієї підпопуляції можуть мігрувати лише в одну – сусідню підпопуляцію.

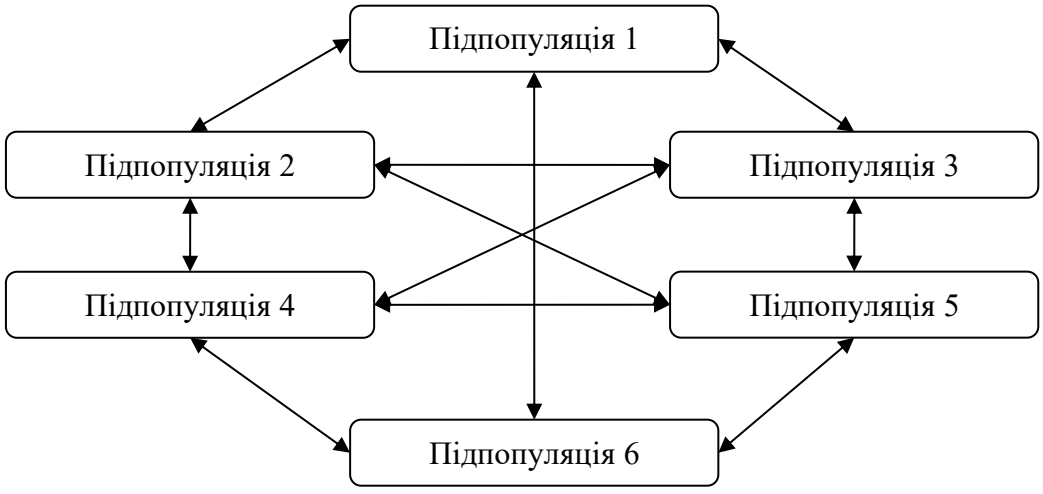

Рис. 5.2. Міграція з топологією повної мережі

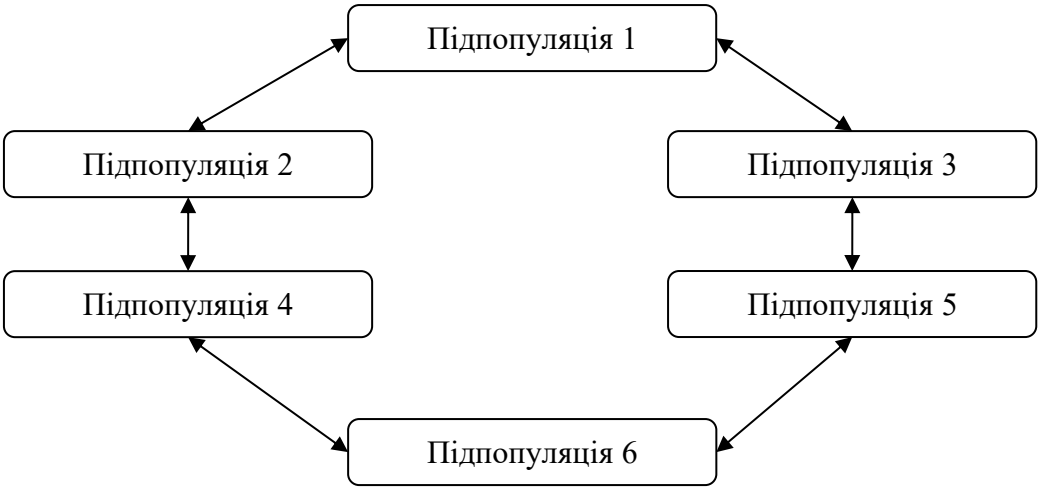

Рис. 5.3. Міграція з топологією кільця

5.4. Програмне забезпечення та сфери застосування генетичних алгоритмів.

Скільки завгодно красива і цікава теорія опановує маси лише тоді, коли на ринок виходять прості і зручні інструментальні засоби, що роблять експерименти з новими концепціями загальнодоступними. На щастя, для *A-life* такий інструмент вже існує. Основним компонентом систем *A-life* є пакети, що реалізовують генетичні алгоритми. На ринку програмних засобів представлено декілька таких пакетів:

Arlequin – аналіз генетичних для популяції даних;

Genepop – генетичний для популяції аналіз;

Genetix – генетичний для популяції аналіз (програма доступна лише французькою мовою);

MacClade – комерційна програма для інтерактивного еволюційного аналізу даних;

MEGA – молекулярно-еволюційний генетичний аналіз;

Populations - генетичний для популяції аналіз.

Найбільш поширеними вважаються два програмних пакета: пакет *Evolver* (перший з масових пакетів *GA*), а також пізніший і потужніший пакет *GeneHunter фірми Ward Systems Group*. Останній особливо популярний, оскільки входить до складу нейромережевого пакету *Ward*, активно вживаного банкірами для портфельної гри і валютного ділінга.

GeneHunter. Потужний програмний комплекс під назвою *GeneHunter* призначений для рішення оптимізаційних задач будь-якої складності за допомогою генетичних алгоритмів і складається з наступних трьох частин: надбудови для MS Excel, динамічної бібліотеки функцій *GALIB* і прикладів задач, вирішених на пакеті.

***Auto2Fit 3*.** Програмний комплекс є потужним і в той же час легким у використанні інструментом, призначеним для вирішення оптимізаційних задач і проведення складних розрахунків. Програма *Auto2Fit* має в своєму

розпорядженні вісім оптимізаційних алгоритмів, включаючи генетичні, і їх багаточисельні модифікації. Якщо говорити про ГА, то пакет дозволяє кодувати рішення для більшої ефективності, має шість типів операторів схрещування (кросинговера) і сім можливих варіантів відбору. Програма може працювати в одному з наступних режимів: швидкому і програмному. З додаткових можливостей *Auto2Fit* можна відзначити зручну навігацію по файлах, що реалізовується за допомогою дерева каталогів, робочу область, представлену у вигляді таблиць *Excel*, побудова 3D-графиков і інше. У комплект також включені багаточисельні приклади, у тому числі і класичні приклади задач оптимізації.

GeneBase. Бібліотека компонент для *Delphi*, що реалізовує генетичні алгоритми – потужний засіб рішення задач багатовимірної непараметричної оптимізації. У наборі є компонент, що реалізовує генетичний алгоритм, який може бути використаний для рішення задач багатовимірної непараметричної оптимізації. Зокрема він дозволяє знаходити субоптимальне рішення NP-повних задач.

Genetic Algorithm and Direct Search Toolbox. *Genetic Algorithm and Direct Search Toolbox* призначений для розширення функціональних можливостей пакету *MATLAB* і, зокрема, *Optimization Toolbox* новим виглядом алгоритмів оптимізації. У даному інструментарії містяться нові можливості по вживанню відомих алгоритмів оптимізації для такого класу задач, який представляє певні труднощі при вирішенні звичайними методами оптимізації. Подібні методи і алгоритми найчастіше використовуються у разі, коли шукана цільова функція є переривистою, істотно нелінійною, стохастичною і не має похідних або ці похідні є недостатньо визначеними. Цей модуль може розглядатися і як деяке доповнення до інших методів оптимізації як деякий засіб для пошуку прийнятної початкової точки розрахунку по основному алгоритму оптимізації. Алгоритм може служити і як засіб для подальшого уточнення раніше використаних основних алгоритмів.

5.5. Мурашині алгоритми та генетичне програмування

У останні два десятиліття при оптимізації складних систем дослідники все частіше застосовують природні механізми пошуку найкращих рішень. Ці механізми забезпечують ефективну адаптацію флори і фауни до довкілля впродовж мільйонів років. Останніми роками інтенсивно розробляється науковий напрям ***Natural Computing*** – «Природні обчислення», який об'єднує математичні методи з природними механізмами прийняття рішень, а саме:

Genetic Algorithms – генетичні алгоритми;
Evolution Programming – еволюційне програмування;
Neural Network Computing – еволюційне нейромережеві обчислення;
DNA Computing – ДНК обчислення;
Cellular Automata – клітинні автомати;

Ant Colony Algorithms – мурашині алгоритми.

Імітація самоорганізації мурашиної колонії складає основу мурашиних алгоритмів оптимізації – нового перспективного методу природних обчислень. Колонія мурах може розглядатися як багатоагентна система, в якій кожен агент (мурашка) функціонує автономно по дуже простих правилах. На противагу майже примітивній поведінці агентів, поведінка всієї системи виходить на подив розумною. Мурашині алгоритми серйозно досліджуються європейськими вченими з середини 90-х років. Вперше підхід був запропонований бельгійським дослідником Марко Доріго (*Marco Dorigo*). На сьогодні вже отримані важливі результати мурашиної оптимізації таких складних комбінаторних задач, як: задача комівояжера, задача оптимізації маршрутів вантажівок, задача розфарбовування графа, квадратичної задачі про призначення, оптимізації мережевих графіків, задача календарного планерування і інших. Особливо ефективні мурашині алгоритми при оптимізації процесів в розподілених нестаціонарних системах, наприклад трафіків в телекомунікаційних мережах.

Мурахи використовують два способи передачі інформації: прямий – обмін їжею, мандибулярний, візуальний і хімічний контакти, і непрямий – стігмержи (stigmergy). *Стігмержи* – це рознесений в часі тип взаємодії, коли один суб'єкт взаємодії змінює деяку частину довкілля, а останні використовують інформацію про її стан пізніше, коли знаходяться в її околиці. Біологічно стігмержи здійснюється через *феромон* (pheromone) – спеціальний секрет, що відкладається як слід при переміщенні мурав'їв. *Феромон* – досить стійка речовина, він може сприйматися мурашками декілька діб. Чим вище концентрація феромону на стежці, тим більше мурах по ній рухатиметься. З часом феромон випаровується, що дозволяє мурашкам адаптувати свою поведінку під зміни зовнішнього середовища. Розподіл феромону по простору пересування мурах є свого роду динамічно змінною глобальною пам'яттю мурашника. Будь-яка мурашка у фіксований момент часу може сприймати і змінювати лише один локальний елемент цієї глобальної пам'яті.

Ідея мурашиного алгоритму полягає в наступному: дві мурашки з мурашника повинні дістатися до їжі, яка знаходиться за перешкодою. Під час переміщення кожна мурашка виділяє трохи феромону, використовуючи його як маркер. При інших рівних умовах кожна мурашка вибере свою дорогу. Перша мурашка вибирає верхню дорогу, а друга – нижню. Оскільки нижня дорога в два рази коротше верхньої, друга мурашка досягне мети за час . Перша мурашка у цей момент пройде лише половину дороги. Коли одна мурашка досягає їжі, вона бере один з об'єктів і повертається до мурашника по тій же дорозі. За час друга мурашка повернеться в мурашник з їжею, а перша мурашка досягне їжі. При переміщенні кожної мурашки на дорозі залишається трохи феромону. Для першої мурашки за час дорога була покрита феромонами лише один раз. У той же самий час друга мурашка покрила дорогу феромонами двічі. За час перша мурашка повернулася в

мурашник, а друга мурашка вже встигла ще раз сходити до їжі і повернутися. При цьому концентрація феромону на нижній дорозі буде в два рази вище, ніж на верхній. Тому перша мурашка наступного разу вибере нижню дорогу, оскільки там концентрація феромону вища. У цьому і полягає базова ідея мурашиного алгоритму – оптимізація шляхом непрямого зв'язку між автономними агентами.

Достоїнствами мурашиних алгоритмів є:

– в порівняння з *GAs (Genetic Algorithms)* мурав'їнні алгоритми спираються на пам'ять всієї колонії замість пам'яті лише про попереднє покоління і менше схильні до неоптимальних початкових рішень (із-за випадкового вибору шляху);

– можуть використовуватися в динамічних додатках (швидко адаптуються до змін);

– мають важливу практику застосовування до множини різних задач.

Недоліки таких алгоритмів вважають:

– ускладнений теоретичний аналіз в результаті послідовності випадкових (незалежних) рішень і зміни розподілу вірогідності при ітераціях;

– збіжність алгоритму гарантується, але час збіжності не визначений;

– зазвичай необхідне вживання додаткових методів таких, як локальний пошук;

– сильно залежать від параметрів настройки, які підбираються лише виходячи з експериментів.

Генетичне програмування (*genetic programming, GP*) – одна з найзручніших і універсальних методик вирішення задач, що встають перед аналітиками. Воно застосовується для вирішення широкого круга проблем: символьної регресії (*symbolic regression*), аналізу даних (*data mining*), оптимізації і дослідження поведінки потомства (*emergent behavior*), що з'являється, в біологічних співтовариствах.

Ідею генетичного програмування (ГП) вперше запропонував Дж. Коза в 1992 році, спираючись на концепцію генетичних алгоритмів. **Генетичне програмування** – це розширення генетичної моделі навчання в область програмного забезпечення. Його об'єктом на відміну від генетичних алгоритмів є не символьні рядки фіксованої довжини, що кодують можливі рішення проблеми, а власне програми, виконуючи які і отримують різні варіанти рішення задачі. У генетичному програмуванні програми представляються у вигляді дерева граматичного розбору, а не у вигляді рядків коду, тому в ГП всі операції виконуються не над рядками, а над *деревами*. При цьому використовуються такі ж оператори, як і в генетичному алгоритмі: селекція, схрещування і мутація.

Генетичні програми не пишуться програмістами, а створюються в результаті наступного ітераційного покрокового процесу.

1. Генерується початкова популяція програм, що є випадковими композиціями функцій (арифметичних, логічних та ін. операцій) і терміналів (змінних і констант), узятих з множини функціональних і термінальних

елементів, що відносяться до вирішуваної проблеми. Якщо ми збираємося вивести програму, яка управляє транспортною логістикою, то до набору терміналів увійдуть наступні змінні – відстань до об’єкта, швидкість і вантажопідйомність, тип транспортних засобів і так далі. Набор функцій в цьому випадку включатиме різні математичні операції, як прості (множення, ділення, віднімання, складання), так і складніші.

2. Кожна програма виконується і їй привласнюється значення пристосованості, відповідне тому, наскільки добре вона вирішує задачу.

3. Створюється нова популяція комп'ютерних програм за рахунок:

– копіювання в неї найкращих програм старої популяції;

– створення нових програм за допомогою мутацій (випадкової зміни функцій і терміналів або навіть цілих піддерев);

– створення нових програм за допомогою схрещування тих, що існують. Операція схрещування реалізується за рахунок обміну піддерев двох хромосом, вибраних випадково (відповідно до їх пристосованості).

4. Всі програми знову виконуються і цикл повторюється до тих пір, поки не буде отриманий необхідний результат.