ЛЕКЦІЯ 7.
ОСНОВИ ЕВОЛЮЦІЙНОГО ПІДХІДУ У ШТУЧНОМУ ІНТЕЛЕКТІ

Питання лекції

Вступ
1. Природний відбір у природі
2. Основні поняття генетичних алгоритмів
3. Особливості генетичних алгоритмів
4. Класичний генетичний алгоритм
5. Мурашині алгоритми
6. Еволюційні моделі
Висновки

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

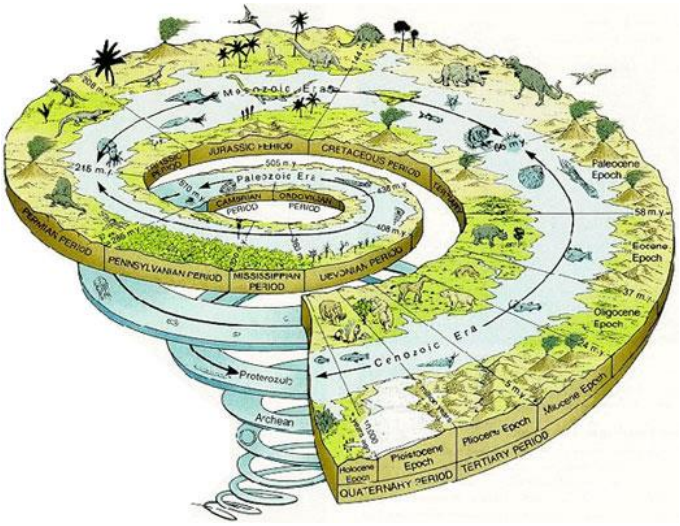
4. Д. Рутковская, М. Пилиньский, Л. Рутковский. Нейронные сети, генетические алгоритмы и нечеткие системы —1-е. — М.: Гарячая линия – Телеком, 2007. — С. 384. — ISBN 5-93517-103-1

ВСТУП

Для створення інтелектуальних систем часто застосовують **еволюційний підхід**, що умовно можна поділити на:

• **Еволюційні алгоритми** (моделювання загальних закономірностей еволюції). Це системи, які використовують ***еволюційні принципи розвитку популяції***. Вони успішно використовуються для завдань функціональної оптимізації і можуть легко бути описані математичною мовою.

• **Еволюційні моделі**. Це системи, які відтворюють біологічні популяції чи системи і поки що не є корисними в прикладному сенсі. Еволюційні моделі більше схожі на біологічні системи, мають складну поведінку, мало спрямовані на вирішення технічних завдань. До цих систем відносять так зване «штучне життя».



Еволюційні алгоритми

До них відносяться:
• Генетичні алгоритми
• Мурашині алгоритми
• Еволюційні стратегії
• Еволюційне програмування
• Генетичне програмування

Еволюційні алгоритми - термін, що часто використовується для загального опису алгоритмів пошуку, оптимізації або навчання, що засновані на формалізованих принципах природного еволюційного процесу.

Еволюційні методи призначені для пошуку бажаних рішень і засновані на статистичному підході до дослідження ситуацій та ітераційному наближенні системи до шуканого стану. ***На відміну від точних методів математичного програмування*** *еволюційні методи дозволяють знаходити рішення, близькі до оптимальних, за прийнятний час, а* ***на відміну від відомих евристичних методів оптимізації*** *характеризуються істотно меншою залежністю від особливостей додатку (більш універсальні) і в багатьох випадках забезпечують кращу ступінь наближення до оптимального рішення.*

Основною перевагою еволюційних методів оптимізації **є можливості вирішення багатомодальних (з кількома локальними екстремумами) завдань з великою розмірністю** за рахунок поєднання елементів випадковості і детермінованості точно так само, як це відбувається у природному середовищі. Детермінованість цих методів полягає в моделюванні природних процесів відбору, що відбуваються за строго визначеними правилами, основним з яких є закон еволюції: «перемагає найсильніший».

Іншим важливим чинником ефективності еволюційних алгоритмів є відтворення процесів розвитку. Розглянуті варіанти рішень можуть за певним правилом породжувати нові рішення, які будуть на - слідувати кращі риси своїх попередників. В якості випадкового елемента в методах еволюційних обчислень використовується моделювання процесу мутації. З її допомогою характеристики того чи іншого рішення можуть бути випадково змінені, що призведе до нового

напрямку в процесі еволюції рішень і може прискорити процес вироблення кращого рішення.

Переваги еволюційних алгоритмів

• Широка область застосування.

• Можливість проблемно-орієнтованого кодування рішень, підбору початкових умов, 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 вид цільової функції.

• Ясність схеми і базових принципів еволюційних обчислень.

• Інтегрованість еволюційних обчислень з іншими некласичними парадигмами штучного інтелекту, такими як штучні нейромережі та нечітка логіка.

Недоліки еволюційних алгоритмів

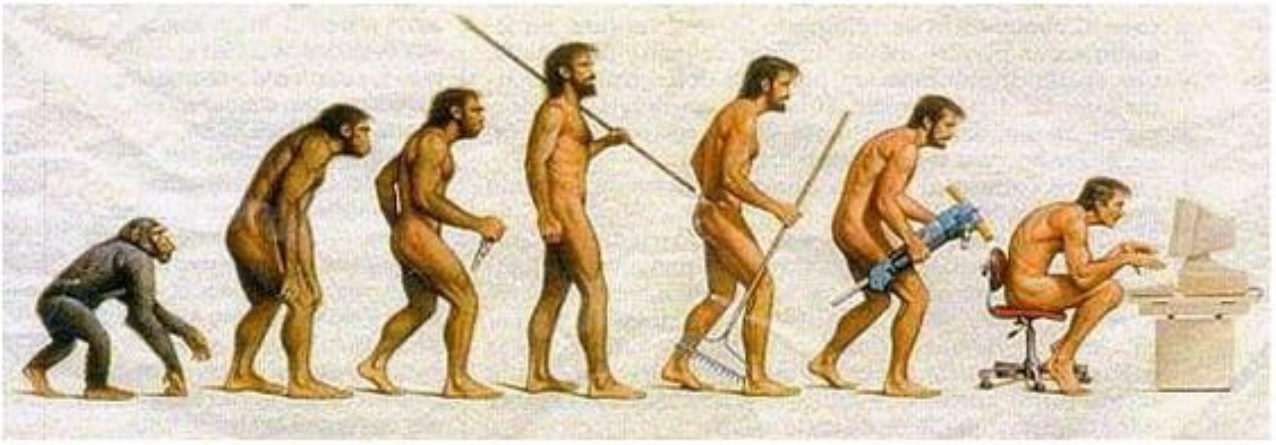
• Евристичний характер еволюційних обчислень не гарантує оптимальності отриманого рішення (правда, на практиці, найчастіше, важливо за заданий час отримати одне або кілька субоптимальних альтернативних рішень, тим більше, що вихідні дані в задачі можуть динамічно змінюватися, бути неточними або неповними).

• 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 заключних фазах моделювання еволюції (оператори пошуку в еволюційних алгоритмах не орієнтовані на швидке попадання в локальний оптимум).

• Невирішеність питань самоадаптації.

Еволюційна теорія стверджує, що кожен біологічний вид цілеспрямовано розвивається і змінюється для того, щоб щонайкраще пристосуватися до навколишнього середовища. У процесі еволюції багато видів комах і риб придбали захисне фарбування, їжак став невразливим завдяки голкам, людина стала власником дуже складної нервової системи. *Можна сказати, що еволюція - це процес оптимізації всіх живих організмів.* Розглянемо, якими ж засобами природа вирішує цю задачу оптимізації.

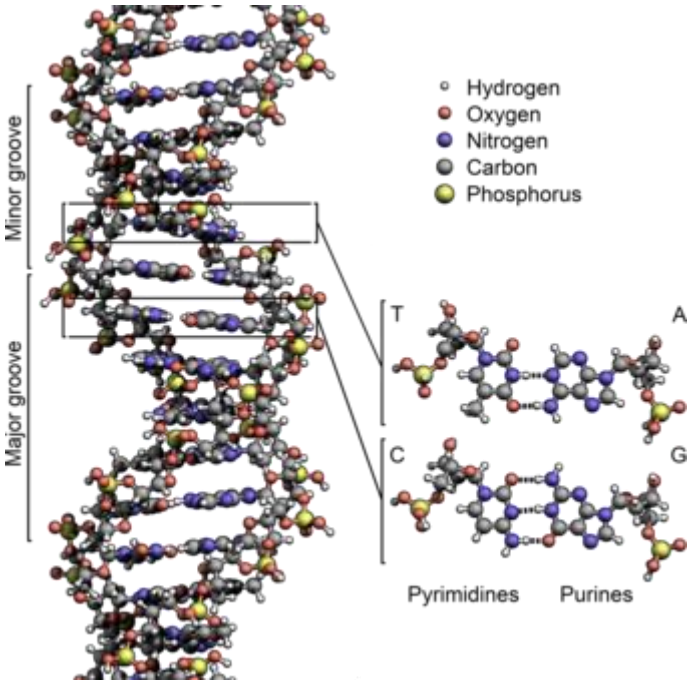


Основний механізм еволюції - це природний відбір. *Його суть полягає в тому, що більш пристосовані особи мають більше можливостей для виживання і розмноження і, отже, приносять більше нащадків, ніж погано пристосовані особи.* При цьому завдяки передачі генетичної інформації (генетичному спадкуванню) нащадки успадковують від батьків основні їхні якості. Таким чином, нащадки сильних індивідуумів також будуть відносно добре пристосованими, а їхня частка в загальній масі особин буде зростати. Після зміни декількох десятків або сотень поколінь середня пристосованість особин даного виду помітно зростає.

1. Природний відбір у природі

Щоб зробити зрозумілими принципи роботи генетичних алгоритмів, пояснимо також, як улаштовані механізми генетичного спадкування в природі. У кожній клітині будь-якої тварини утримується вся генетична інформація цієї особи. Ця інформація записана у виді набору дуже довгих молекул ДНК (Дезоксирибо Нуклеїнова Кислота). Кожна молекула ДНК - це ланцюжок, що складається з молекул нуклеотидів чотирьох типів, що позначаються A, T, C і G.

Власне, інформацію несе порядок проходження нуклеотидів у ДНК. Таким чином, генетичний код індивідума - це просто дуже довгий рядок символів, де використовуються всього 4 букви. У тваринній клітині кожна молекула ДНК оточена оболонкою - таке утворення називається *хромосомою*.



Кожна уроджена якість особи (колір ока, спадкоємні хвороби, тип волосся і т.д.) кодується визначеною частиною хромосоми, що називається *геном* цієї властивості. Наприклад, ген кольору ока містить інформацію, що кодує визначений колір очей. Різні значення гена називаються його *алелями*.

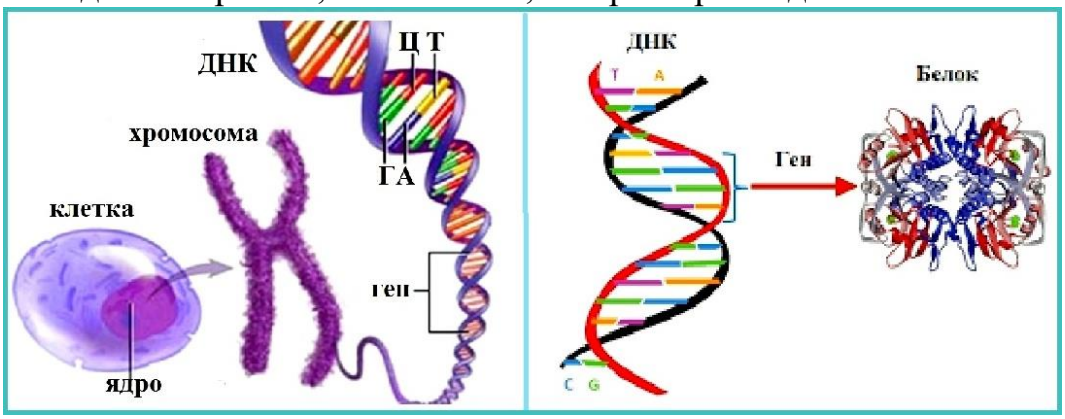
При розмноженні тварин відбувається злиття двох батьківських статевих клітин і їхні ДНК взаємодіють, утворюючи ДНК нащадка. Основний спосіб

взаємодії - **кроссовер** (cross-over, схрещування). При кроссовері ДНК предків розділяються на дві частини, а потім обмінюються своїми половинками.

При спадкуванні можливі **мутації** через радіоактивність або інші впливи, у результаті яких можуть змінитися деякі гени в статевих клітинах одного з батьків. Змінені гени передаються нащадкові і додають йому нові властивості. Якщо ці нові властивості корисні, вони, швидше за все, збережуться в даному виді - при цьому відбудеться стрибкоподібне підвищення пристосованості виду. Ключову роль в еволюційній теорії грає **природний відбір**. Його суть полягає в тому, що найбільш пристосовані особи краще виживають і приносять більше нащадків, чим менш пристосовані. Помітимо, що сам по собі природний відбір ще не забезпечує розвиток біологічного виду.

Тому дуже важливо зрозуміти, яким чином відбувається спадкування, тобто як властивості нащадка залежать від властивостей батьків. **Основний закон спадкування інтуїтивно зрозумілий кожному - він полягає в тому, що нащадки схожі на батьків.** Зокрема, нащадки більш пристосованих батьків будуть, швидше за все, одними з найбільш пристосованих у своєму поколінні. Щоб зрозуміти, на чому заснована ця подібність, потрібно трохи поглибитися в будову природної клітини - у світ генів і хромосом. Майже в кожній клітині будь-якої особи є набір хромосом, що несуть інформацію про цю особу. Основна частина хромосоми - нитка ДНК, що визначає, які хімічні реакції будуть відбуватися в даній клітині, як вона буде розвиватися і які функції виконувати.

Ген - це відрізок ланцюга ДНК, відповідальний за визначену властивість особи, наприклад за колір очей, тип волосся, колір шкіри і т.д.



Уся сукупність генетичних ознак людини кодується за допомогою приблизно 60 тис. генів, довжина яких складає більш 90 млн. нуклеотидів.

Розрізняють два види клітин: статеві (такі, як сперматозоїд і яйцеклітина) і соматичні. У кожній соматичній клітині людини утримується 46 хромосом. Ці 46 хромосом - насправді 23 пари, причому в кожній парі одна з хромосом отримана від батька, а друга - від матері. Парні хромосоми відповідають за однакові ознаки - наприклад, батьківська хромосома може містити ген чорного кольору ока, а парна їй материнська - ген блакитного кольору. **Існують визначені закони, що керують участю тих або інших генів у розвитку особи.** Зокрема, у нашому прикладі нащадок буде чорнооким, оскільки ген блакитних очей є "слабким" і придушується геном будь-якого іншого кольору.

У статевих клітинах хромосом тільки 23, і вони непарні. При заплідненні відбувається злиття чоловічої і жіночої статевих клітин і виходить клітина зародка,

що містить 46 хромосом. Які властивості нащадок одержить від батька, а які - від матері? Це залежить від того, які саме статеві клітини брали участь у заплідненні. Процес вироблення статевих клітин (так називаний мейоз) в організмі піддається ймовірностям, завдяки яким нащадки багато в чому відрізняються від своїх батьків. При мейозі, відбуваються наступне: парні хромосоми соматичної клітини зближаються впритул, потім їхні нитки ДНК розриваються в декількох випадкових місцях і хромосоми обмінюються своїми частинами (рис. 1).

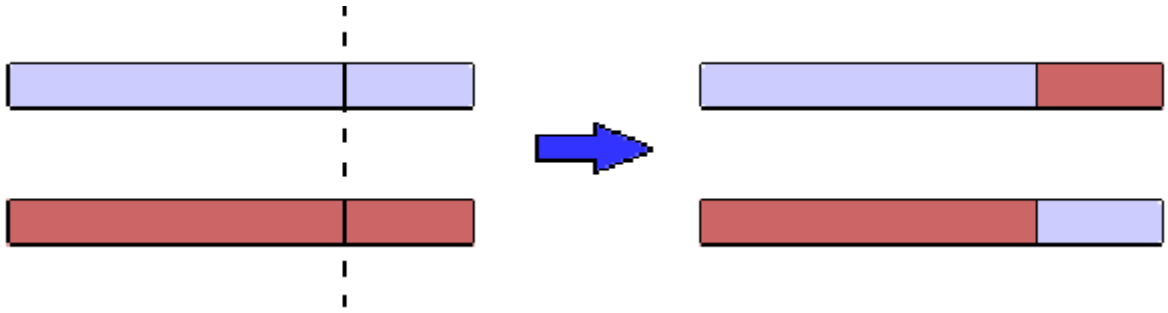


Рис. 1. Умовна схема кросінговера

Цей процес забезпечує появу нових варіантів хромосом і називається "кросінговер". Кожна із знову з'явившихся хромосом виявиться потім усередині однієї з статевих клітин, і її генетична інформація може реалізуватися в нащадках даної особи.

Другий важливий фактор, що впливає на спадковість, - це *мутації,* що виражаються в зміні деяких ділянок ДНК. Мутації також випадкові і можуть бути викликані різними зовнішніми факторами, такими, як радіоактивне опромінення. Якщо мутація відбулася в статевій клітині, то змінений ген може передатися нащадкові і проявитися у виді спадкоємної хвороби або в інших нових властивостях нащадка. Вважається, що саме мутації є причиною появи нових біологічних видів, а кросінговер визначає вже мінливість усередині виду (наприклад, генетичні розходження між людьми).

2. Основні поняття генетичних алгоритмів

Генетичний алгоритм (англ. Genetic algorithm) — це евристичний алгоритм пошуку, використовуваний для рішення задач оптимізації і моделювання шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію. Є різновидом еволюційних обчислень (англ. evolutionary computation). Відмінною рисою генетичного алгоритму є акцент на використання оператора «схрещування», що робить операцію рекомбінації рішень-кандидатів, роль якої аналогічна ролі схрещування в живій природі. «Батьком-засновником» генетичних алгоритмів, вважається Джон Холланд (англ. John Holland), книга якого «Адаптація в природних і штучних системах» (англ. Adaptationin Naturaland Artificial Systems) є основною працею в цій області досліджень.

При описі генетичних алгоритмів використовуються означення, запозичені з генетики. Наприклад, мова йде про популяції особин, а як базові поняття

застосовуються ген, хромосома, генотип, фенотип, алель. Також використовуються відповідним цим термінам означення з технічного лексикона, зокрема, ланцюг, двоїчна послідовність, структура.

Популяція - це кінцева множина особин.

Особи, що входять у популяцію, у генетичних алгоритмах представляються *хромосомами* з закодованим у них множинами параметрів задачі, тобто рішень, які інакше називаються точками в просторі пошуку (search points). У деяких роботах особи називаються організмами.

Хромосоми (інші назви - **ланцюжки або кодові послідовності**) - це упорядковані послідовності генів.

Ген (також називаний властивістю, знаком або детектором) - це атомарний елемент генотипу, зокрема, хромосоми.

Генотип або структура - це набір хромосом даної особи. Отже, особами популяції можуть бути генотипи або одиничні хромосоми (у досить розповсюдженому випадку, коли генотип складається з однієї хромосоми).

Фенотип - це набір значень, що відповідають даному генотипові, тобто декодована структура або множина параметрів задачі (рішення, точка простору пошуку).

Алель - це значення конкретного гена, також обумовлене як значення властивості або варіант властивості.

Локус або позиція вказує місце розміщення даного гена в хромосомі (ланцюжку). Множина позицій генів - це **локи**.

Дуже важливим поняттям у генетичних алгоритмах вважається **функція пристосованості (fitness function)**, інакше називана **функцією оцінки**. Вона представляє міру пристосованості даної особи в популяції. Ця функція відіграє найважливішу роль, оскільки дозволяє оцінити ступінь пристосованості конкретних особин у популяції і вибрати з них найбільш пристосовані (тобто найбільші значення функції, що мають пристосованості) відповідно до еволюційного принципу виживання «найсильніших» (найкраще пристосувалися). Функція пристосованості також одержала свою назву безпосередньо з генетики. Вона впливає на функціонування генетичних алгоритмів і повинна мати точне і коректне означення. У задачах оптимізації функція пристосованості, як правило, оптимізується (точніше кажучи, максимізується) і називається **цільовою функцією.** У задачах мінімізації цільова функція перетворюється, і проблема зводиться до максимізації. У теорії керування функція пристосованості може приймати вид *функції похибки*, а в теорії ігор - *вартісної функції*. На кожній ітерації генетичного алгоритму пристосованість кожної особи даної популяції оцінюється за допомогою *функції пристосованості*, і на цій основі створюється наступна популяція особин, що складають множину потенційних рішень проблеми, наприклад, задачі оптимізації.

Чергова популяція в генетичному алгоритмі називається **поколінням**, а до знову створюваної популяції особин застосовується термін «нове покоління» або «покоління нащадків».

Приклад

Нехай дана деяка складна функція (цільова функція), що залежить від декількох змінних, і потрібно знайти такі значення змінних, при яких значення функції максимальне. Задачі такого роду називаються задачами оптимізації і вони зустрічаються на практиці дуже часто.

Один з найбільш наочних прикладів - *задача розподілу інвестицій*. У цій задачі змінними є обсяги інвестицій у кожен проект (10 змінних), а функцією, яку потрібно максимізувати - сумарний дохід інвестора. Також приведені значення мінімального і максимального об'єму вкладення в кожний із проектів, що задають область зміни кожної із змінних.

Спробуємо розв'язати цю задачу, застосовуючи відомі нам природні способи оптимізації. *Будемо розглядати кожен варіант інвестування (набір значень змінних) як індивідуум, а прибутковість цього варіанта - як пристосованість цього індивідуума*. Тоді в процесі еволюції (якщо ми зуміємо його організувати) пристосованість індивідуумів буде зростати, а виходить, будуть з'являтися усе більш і більш дохідні варіанти інвестування. Зупинивши еволюцію в деякий момент і вибравши найкращого індивідуума, ми одержимо досить гарне рішення задачі.

Тут **генетичний алгоритм** - це проста модель еволюції в природі, яка реалізована у вигляді комп'ютерної програми, що здійснює оптимізацію.

У ньому використовуються як аналог механізму генетичного спадкування, так і аналог природного відбору. При цьому зберігається біологічна термінологія в спрощеному виді. От як моделюється генетичне спадкування

Хромосома - Вектор (послідовність) з нулів і одиниць.

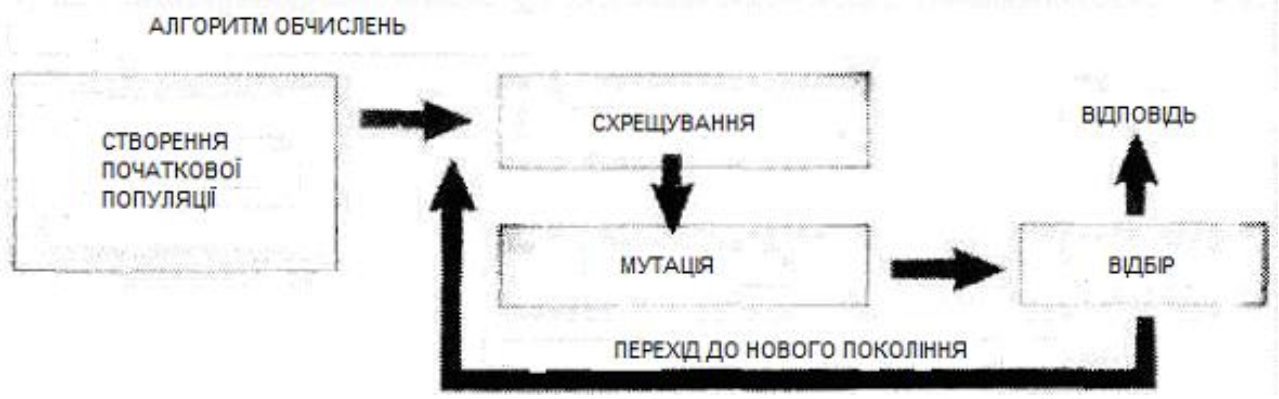
Кожна позиція (біт) називається **геном**.

Ідивідум= генетичний код. - Набір хромосом= варіант рішення задачі

Кросовер - Операція, при якій дві хромосоми обмінюються своїми частинами

Мутація - Випадкова зміна однієї або декількох позицій в хромосомі.

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



Життєвий цикл популяції - це кілька випадкових схрещувань (за допомогою кроссовера) і мутацій, у результаті яких до популяції додається якась кількість нових індивідуумів. **Відбір у генетичному алгоритмі** - це процес формування нової популяції зі старої, після чого стара популяція гине. Після відбору до нової

популяції знову застосовуються операції кросовера і мутації, потім знову відбувається відбір, і так далі.

Відбір у генетичному алгоритмі тісно зв'язаний із принципами природного відбору в природі в такий спосіб:

Пристосування індивідуума	Значення цільової функції на цьому індивідуумі
Виживання найбільш пристосованих	Популяція наступного покоління формується у відповідності з цільовою функцією. Чим пристосованіший індивідуум, тим більша ймовірність його участі у кросовері, тобто розмноженні

Таким чином, модель відбору визначає, яким чином варто будувати популяцію наступного покоління. *Як правило, ймовірність участі індивідуума в схрещуванні береться пропорційно його пристосуванню*. Часто використовується так звана стратегія елітизма, при якій кілька кращих індивідуумів переходять у наступне покоління без змін, не беручи участі у кросовері і доборі. У будь-якому випадку кожне наступне покоління буде в середньому краще попереднього. Коли пристосованість індивідуумів перестає помітно збільшуватися, процес зупиняють і як розв’язання задачі оптимізації беруть найкращого зі знайдених індивідуумів.

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

1. Індивідум = варіант рішення задачі = набір з 10 хромосом X_j

2. Хромосома X_j = обсяг вкладення в проект, j = 16-розрядний запис цього числа

3. Так як обсяги вкладень обмежені, не всі значення хромосом є припустимими. Це враховується при генерації популяцій.

4. Так як сумарний обсяг інвестицій фіксований, то реально варіюються тільки 9 хромосом, а значення 10-ої визначається по них однозначно.

Нижче приведені результати роботи генетичного алгоритму для трьох різних значень сумарного обсягу інвестицій *K* (рис. 2). Квадратами на графіках прибутків відзначено, який обсяг вкладення в даний проект рекомендовано генетичним алгоритмом. Видно, що при малому значенні *K* інвестуються тільки ті проекти, які прибуткові при мінімальних вкладеннях.

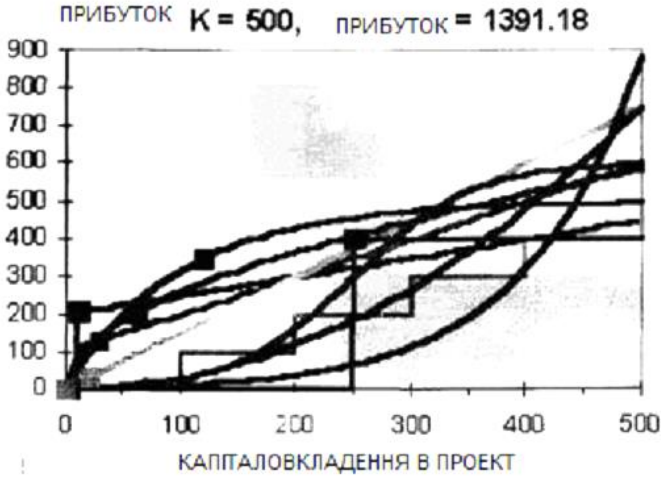


Рис. 2.

Якщо збільшити сумарний обсяг інвестицій, стає прибутковим вкладати гроші й у більш дорогі проекти (рис. 3).

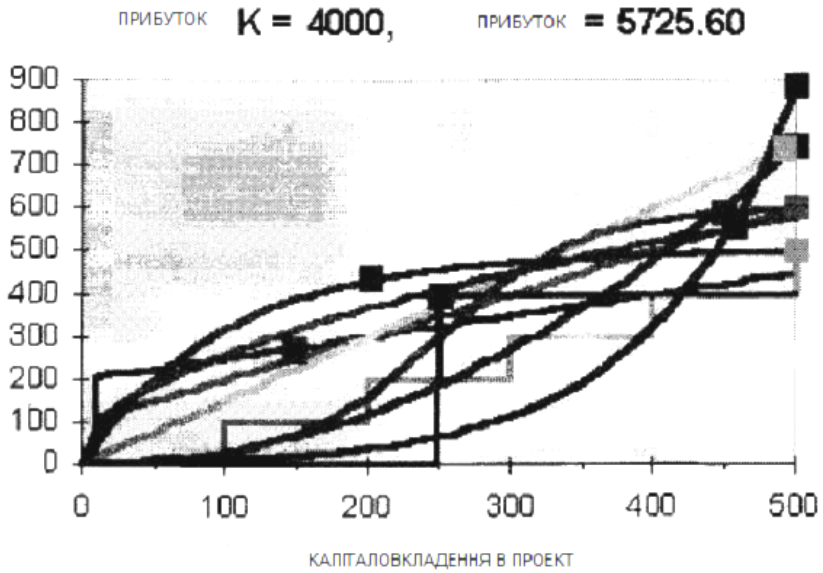


Рис. 3.

При подальшому збільшенні *K* досягається поріг максимального вкладення в прибуткові проекти, і інвестування в малоприбуткові проекти знову набуває сенсу.

3. Особливості генетичних алгоритмів

Генетичний алгоритм - новітній, але не єдино можливий спосіб розв’язання задач оптимізації. Віддавна відомі два основних шляхи розв’язання таких задач - **переборний** і **локально-градієнтний**. У цих методів свої достоїнства і недоліки, і в кожному конкретному випадку варто подумати, який з них вибрати. Розглянемо достоїнства і недоліки стандартних і генетичних методів на прикладі класичної *задачі комівояжера* (TSP - travelling salesman problem). Суть задачі полягає в тому, щоб знайти найкоротший замкнений шлях обходу декількох міст, заданих своїми координатами. Виявляється, що вже для 30 міст пошук оптимального шляху являє собою складну задачу, що спонукала розвиток різних нових методів (у тому числі нейромереж і генетичних алгоритмів).

Кожен варіант розв’язання (для 30 міст) - це числовий ряд, де на j-ому місці стоїть номер j-ого по порядку обходу міста. Таким чином, у цій задачі 30 параметрів, причому не всі комбінації значень припустимі. Природно, першою ідеєю є повний перебір усіх варіантів обходу.

Переборний метод найбільш простий по своїй суті і тривіальний у програмуванні. Для пошуку оптимального розв’язання (точки максимуму цільової функції) потрібно послідовно обчислити значення цільової функції у всіх можливих точках, запам'ятовуючи максимальне з них (рис.7.4).

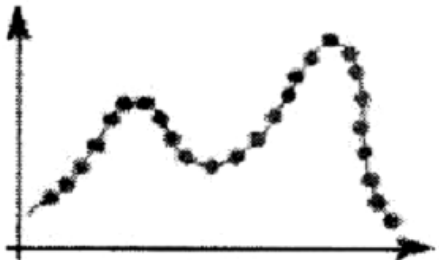


Рис. 4.

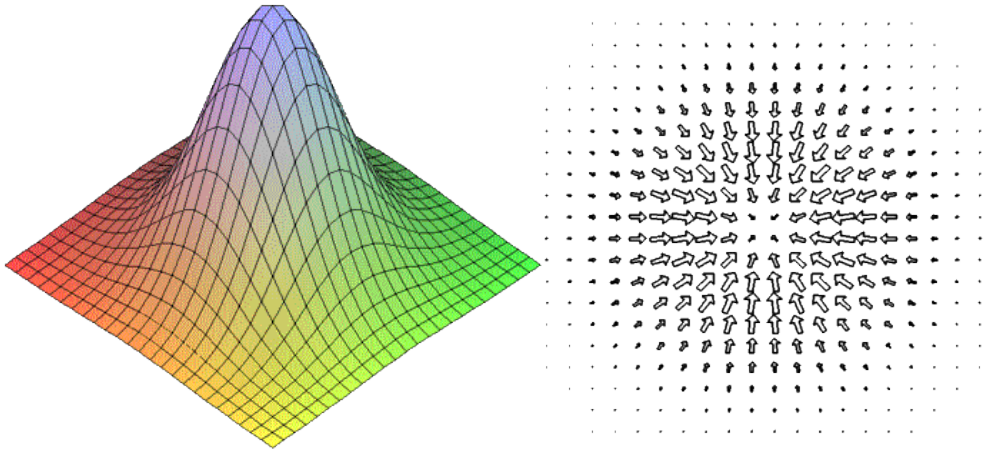
Недоліком цього методу є велика обчислювальна вартість (складність). Зокрема, у задачі комівояжера буде потрібно прорахувати довжини більш 10^{30} варіантів шляхів, що зовсім нереально.

Однак, якщо перебір усіх варіантів за розумний час можливий, то можна бути абсолютно упевненим у тому, що знайдене розв'язання дійсно оптимальне.

Другий популярний спосіб заснований на *методі градієнтного спуска*.

Градіє́нт — міра зростання або спадання в просторі якоїсь фізичної величини на одиницю довжини.

Для позначення градієнта використовується оператор Гамільтона ∇, або оператор {**grad**}



Операція градієнта перетворює пагорб (ліворуч), якщо дивитися на нього зверху, на поле векторів (праворуч). Видно, що вектори спрямовані «вгору», і чим крутіший нахил, тим вони довші.

При *методі градієнтного спуска* спочатку вибираються деякі випадкові значення параметрів, а потім ці значення поступово змінюють, домагаючись найбільшої швидкості росту цільової функції (рис.5).

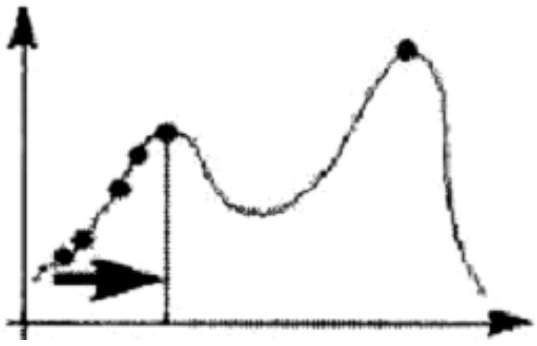


Рис. 5

Досягши локального максимуму, такий алгоритм зупиняється, тому для пошуку глобального оптимуму будуть потрібні додаткові зусилля.

Градієнтні методи працюють дуже швидко, але не гарантують оптимальності знайденого розв’язання. Вони ідеальні для застосування в так званих унімодальних задачах (рис. 6), де цільова функція має єдиний локальний максимум (він же - глобальний). Легко бачити, що задача комівояжера не є унімодальною.

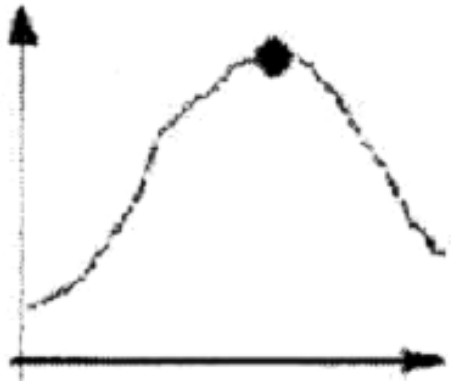
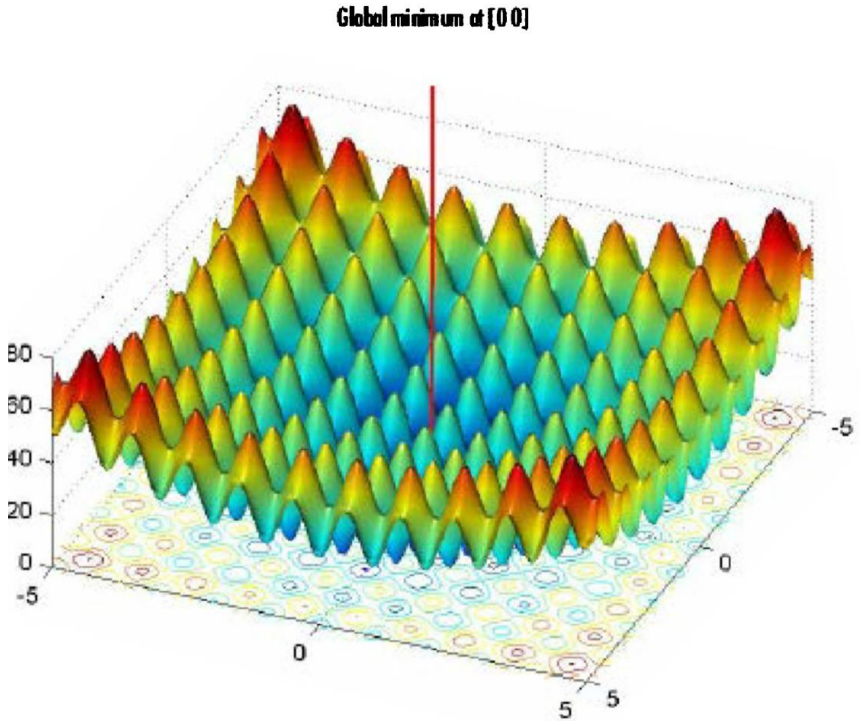


Рис. 6.

Типова практична задача, як правило, *мультимодальна і багатомірна*, тобто містить багато параметрів. Для таких задач не існує жодного універсального методу, що дозволяв би досить швидко знайти абсолютно точне розв’язання (рис. 7).

Приклад складної функції для двох незалежних змінних (функція Растригина)

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$



Однак, комбінуючи переборний і градієнтний методи, можна сподіватися отримати хоча б наближене розв'язання, точність якого буде зростати при збільшенні часу розрахунку (рис. 8).

Генетичний алгоритм являє собою саме такий комбінований метод. Механізми схрещування і мутації в якомусь змісті реалізують переборну частину методу, а добір кращих розв'язань - градієнтний спуск. На рисунку показано, що така комбінація дозволяє забезпечити стійко гарну ефективність генетичного пошуку для будь-яких типів задач (рис. 9).

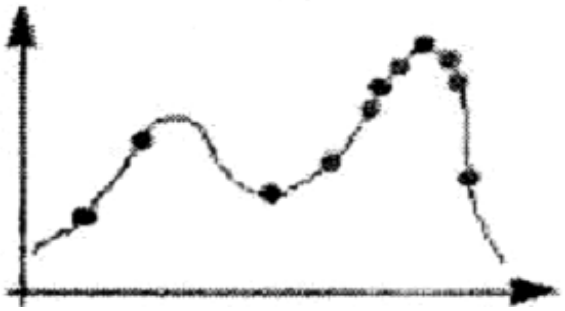


Рис. 8.

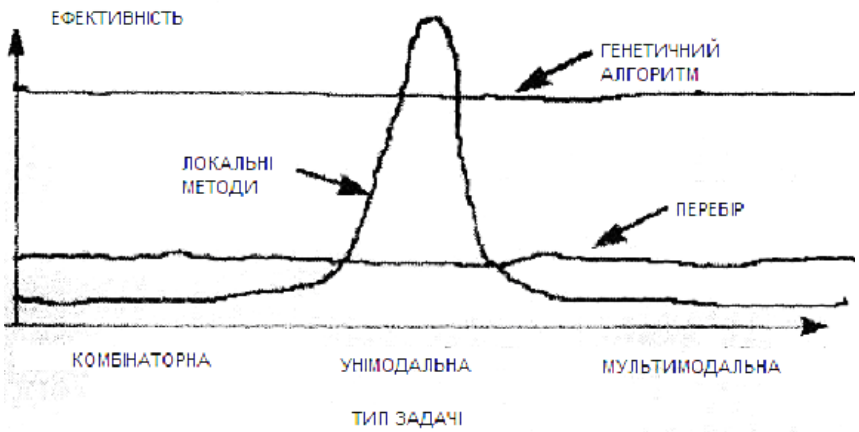


Рис. 9.

Отже, якщо на деякій множині задана складна функція від декількох змінних, то *генетичний алгоритм - це програма, що за розумний час знаходить точку, де значення функції досить близьке до максимально можливого*. Вибираючи прийнятний час розрахунку, ми одержимо одне з кращих розв'язань, що узагалі можливо одержати за цей час.

Генетичні алгоритми в основному застосовуються для розв'язання задач оптимізації, тобто задач, у яких є деяка функція декількох змінних $F(x_1, x_2, ... ,x_n)$ і необхідно знайти або її максимум, або її мінімум. Функція F називається *цільовою функцією*, а змінні - *параметрами функції*. Генетичні алгоритми "пришиваються" до даної задачі в такий спосіб.

Параметри задачі є генетичним матеріалом - *генами*. Сукупність генів складає *хромосому*. Кожна особа має свою хромосому, а, отже, свої набори параметрів. Підставивши параметри в цільову функцію, можна одержати якесь значення. Те, наскільки це значення задовольняє поставленим умовам, визначає характеристику особи, що називається **пристосованістю** (**fitness**). Функція, що визначає пристосованість повинна задовольняти наступній умові: *чим "краща" особа, тим вище пристосованість*. Генетичні алгоритми працюють з популяцією як правило фіксованого розміру, що складається з особин, заданих за допомогою способу, описаного вище. Особи "схрещуються" між собою за допомогою *генетичних операторів* (про те, як відбувається цей процес – буде описано окремо), і в такий спосіб виходять *нащадки*, причому частина нащадків замінюють представників більш старого покоління у відповідності *зі стратегією формування нового покоління*. Вибір особин для схрещування проводиться відповідно *до селективної стратегії (selection strategy)*. Заново сформована популяція знову оцінюється, потім вибираються найбільш гідні для схрещування особи, що схрещуються між собою, виходять «діти» і займають місце «старих» індивідумів і т.д. Усі це продовжується доти поки не знайдеться особа, гени якої представляють оптимальний набір параметрів, при яких значення цільової функції близьке до максимуму або мінімуму, або дорівнює йому. Зупинка роботи ГА може відбутися також у випадку, якщо популяція вироджується, тобто якщо практично немає розмаїтості в генах особин популяції, або якщо просто вийшов ліміт часу. Виродження популяції називають *передчасною збіжністю (premature convergence)*.

Може створитися враження, що ГА є просто перекрученим варіантом випадкового пошуку. Але пристосованість була введена зовсім не даремно. Справа в тім, що вона безпосередньо впливає на шанс особи взяти участь у схрещуванні з наступним «народженням дітей». Вибираючи щораз для схрещування найбільш пристосованих особин, можна з визначеним степенем упевненості стверджувати, що нащадки будуть або не набагато гірші, ніж батьки, або кращі їх.

Теоретичні аспекти, що випливають на ГА:

- Представлення даних в генах
- Генетичні оператори
- Моделі ГА
- Функції

 Стратегії відбору і формування нового покоління

Приклади задач де застосовуються ГА

• Екстремальні задачі (пошук точок мінімуму і мінімуму);
• Задачі про найкоротший шлях;
• Задачі компонування;
• Складання розкладів;
• Апроксимація функцій;
• Добір (фільтрація) вхідних даних;
• Настроювання штучної нейронної мережі;
• Моделювання штучного життя (Artificial life systems) ;
• Біоінформатика (згортання білків і РНК);
• Ігрові стратегії;
• Нелінійна фільтрація;
• Агенти, що розвиваються/машини (Evolvable agents/machines);
• Оптимізація запитів в базах даних
• Різноманітні задачі на графах (задача комівояжера, розфарбування, знаходження паросполученнь)
• Навчання штучної нейронної мережі
• Штучне життя

Деякі розділи можуть містити підпункти. Так, наприклад, екстремальні задачі включають у себе цілий клас задач лінійного і нелінійного програмування.

Усього застосувань дуже багато, тому приведений список не є вичерпним.

4. Класичний генетичний алгоритм

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.

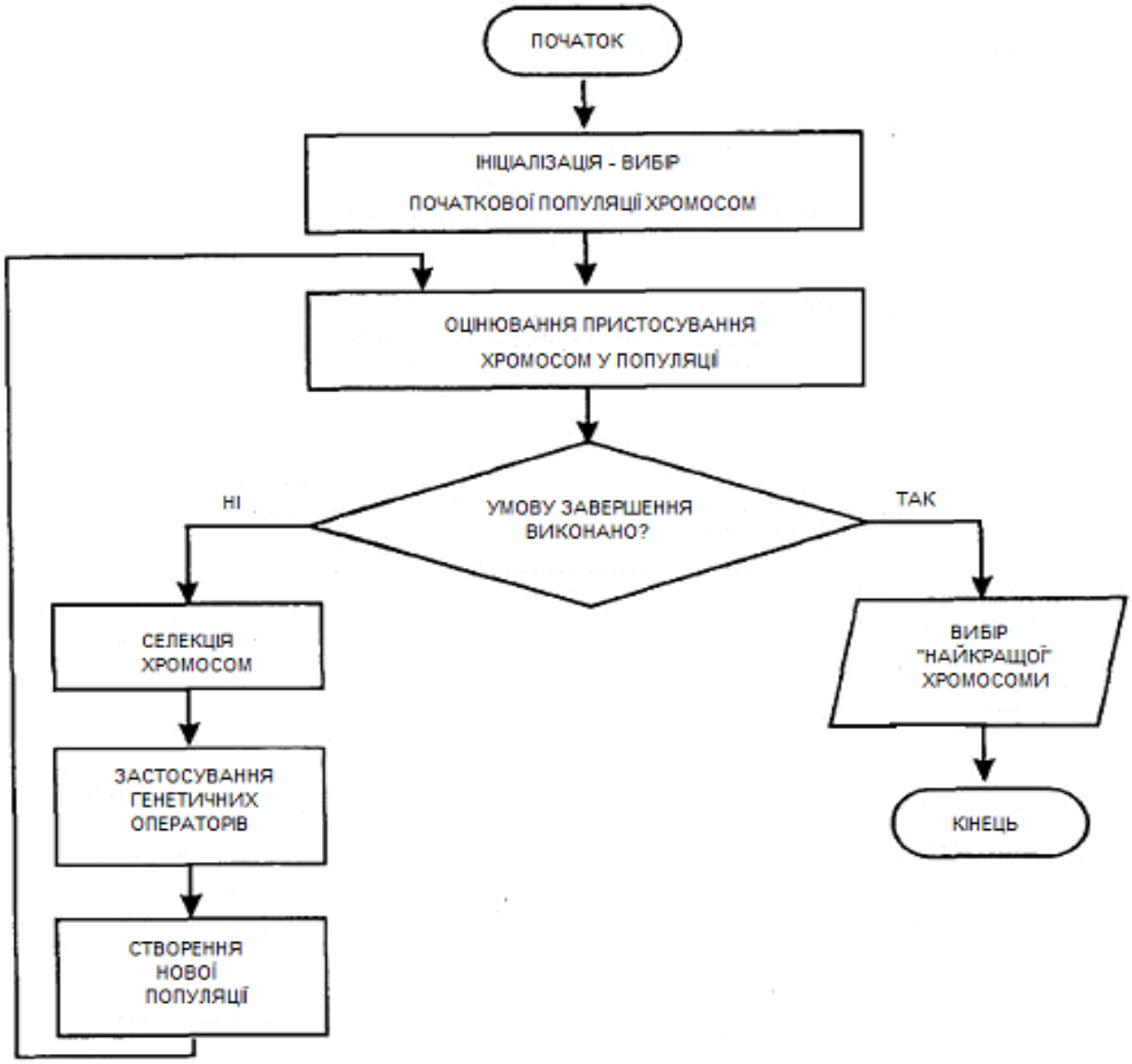


Рис. 12. Блок-схема генетичного алгоритму.

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

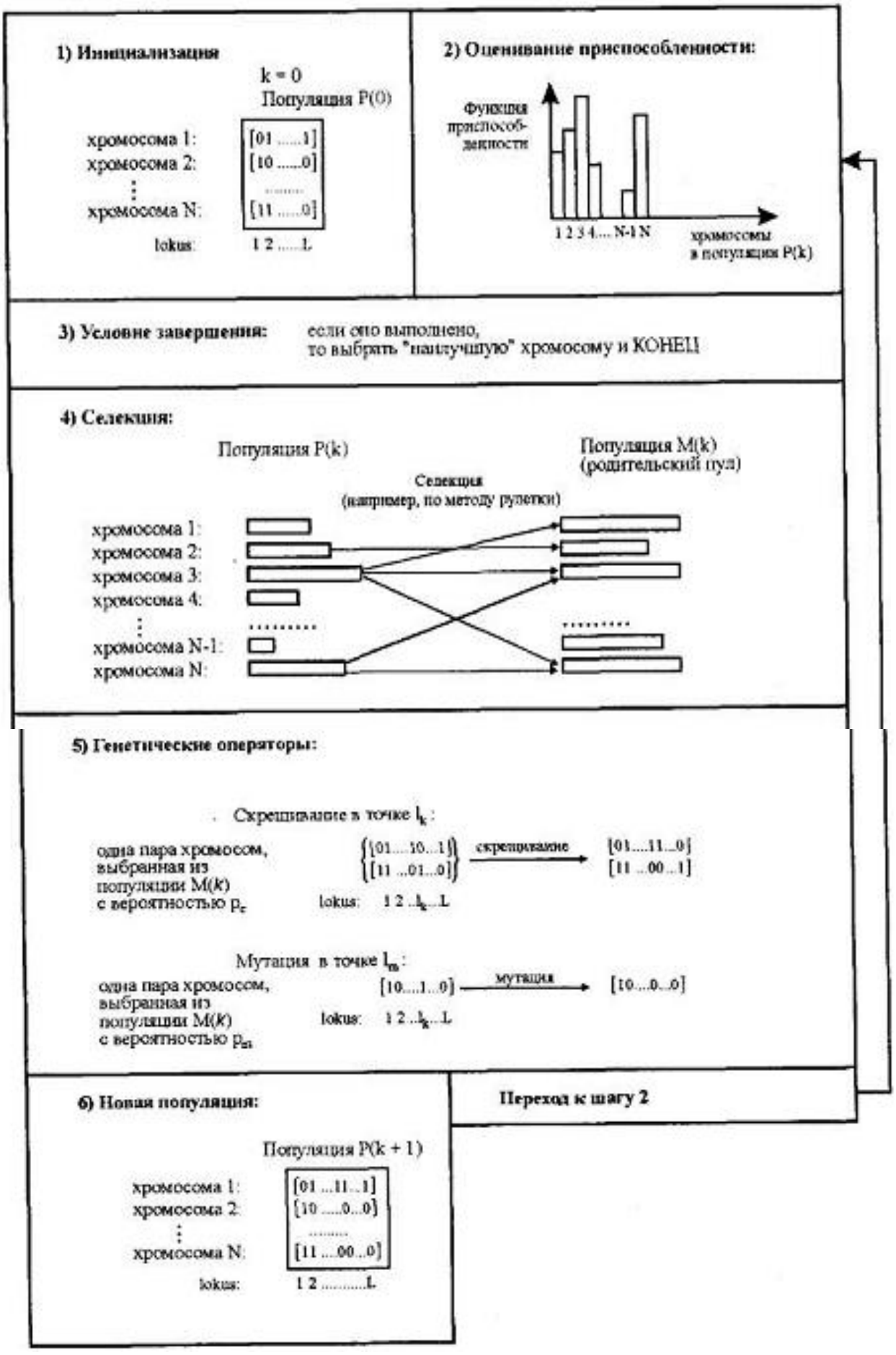


Рис. 13. Схема виконання генетичного алгоритму.

Ініціалізація, тобто формування вихідної популяції, полягає у випадковому виборі заданої кількості хромосом (особин), що представляються двоїчними послідовностями фіксованої довжини.

Оцінювання пристосованості хромосом у популяції складається в розрахунку функції пристосованості для кожної хромосоми цієї популяції. Чим більше значення цієї функції, тим вище «якість» хромосоми. Форма функції пристосованості залежить від характеру розв'язуваної задачі. Передбачається, що функція пристосованості завжди приймає ненегативні значення і, крім того, що для рішення оптимізаційної задачі потрібно максимізувати цю функцію. Якщо вихідна форма функції пристосованості не задовольняє цим умовам, то виконується відповідне перетворення (наприклад, задачу мінімізації функції можна легко звести до задачі максимізації).

Перевірка умови зупинки алгоритму. Визначення умови зупинки генетичного алгоритму залежить від його конкретного застосування. В оптимізаційних задачах, якщо відомо максимальне (або мінімальне) значення функції пристосованості, то *зупинка алгоритму може відбутися після досягнення очікуваного оптимального значення, можливо - із заданою точністю. Зупинка алгоритму також може відбутися у випадку, коли його виконання не приводить до поліпшення вже досягнутого значення. Алгоритм може бути зупинений після закінчення визначеного часу виконання або після виконання заданої кількості ітерацій.* Якщо умова зупинки виконана, то виконується перехід до завершального етапу вибору «найкращої» хромосоми. У противному випадку на наступному кроці виконується селекція.

Селекція хромосом полягає у відборі (по розрахованим на другому етапі значенням функції пристосованості) тих хромосом, що будуть брати участь у створенні нащадків для наступної популяції, тобто для чергового покоління. Такий відбір виконується відповідно до принципу природного відбору, по якому найбільші шанси на участь у створенні нових особин мають хромосоми з найбільшими значеннями функції пристосованості. Існують різні методи селекції. Найбільш популярним вважається так називаний метод рулетки (roulette wheel selection), що свою назву одержав за аналогією з відомою азартною грою. Кожній хромосомі може бути зіставлений сектор колеса рулетки, величина якого встановлюється пропорційно значенню функції пристосованості даної хромосоми. Тому чим більше значення функції пристосованості, тим більше сектор на колесі рулетки. Усе колесо рулетки відповідає сумі значень функції пристосованості всіх хромосом розглянутої популяції.

Кожній хромосомі, що позначається ch_i для i=1,2,..., N (де N позначає чисельність популяції) відповідає сектор колеса v(ch_i), виражений у відсотках відповідно до формули

$$v(ch_i) = p_s(ch_i)100\% \ , \qquad (7.2)$$

де

$$p_s(ch_i)=\frac{F(ch_i)}{\sum_{i=1}^{N} F(ch_i)} \quad (7.3)$$

причому $F(ch_i)$ - значення функції пристосованості хромосоми ch_i , а $p_s(ch_i)$ - ймовірність селекції хромосоми ch_i . Селекція хромосоми може бути представлена як результат повороту колеса рулетки, оскільки «вигравша» (тобто обрана) хромосома відноситься до того сектора колеса, що випав. Очевидно, що чим більше сектор, тим більше ймовірність «перемоги» відповідної хромосоми. Тому ймовірність вибору даної хромосоми виявляється пропорційною значенню її функції пристосованості.

Якщо все коло колеса рулетки представити у виді цифрового інтервалу [0, 100], то вибір хромосоми можна ототожнити з вибором числа з інтервалу [a, b], де a і b позначають відповідно початок і закінчення фрагмента кола, що відповідає цьому секторові колеса; очевидно, що $0 \le a < b \le 100$. У цьому випадку вибір за допомогою колеса рулетки зводиться до вибору числа з інтервалу [0, 100], що відповідає конкретній точці на колі колеса. Інші методи селекції будуть розглядатися далі.

У результаті процесу селекції створюється батьківська популяція, яка також називана батьківським пулом (mating pool) з чисельністю N, рівною чисельності поточної популяції.

Застосування генетичних операторів до хромосом, відібраним за допомогою селекції, приводить до формування нової популяції нащадків від створеної на попередньому кроці батьківської популяції.

У класичному генетичному алгоритмі застосовуються два основних генетичних оператори: оператор схрещування (crossover) і оператор мутації (mutation). Однак слід зазначити, що оператор мутації грає явно другорядну роль у порівнянні з оператором схрещування. Це означає, що схрещування в класичному генетичному алгоритмі виконується практично завжди, тоді як мутація - досить рідко. Ймовірність схрещування, як правило, досить велика (зазвичай $0{,}5 \le p_c \le 1$), тоді як ймовірність мутації встановлюється досить малою (найчастіше $0 \le p_т \le 0{,}1$). Це випливає з аналогії зі світом живих організмів, де мутації відбуваються надзвичайно рідко. У генетичному алгоритмі мутація хромосом може виконуватися на популяції батьків перед схрещуванням або на популяції нащадків, утворених у результаті схрещування.

Оператор схрещування. На першому етапі схрещування вибираються пари хромосом з батьківської популяції (батьківського пула). Це тимчасова популяція, що складається з хромосом, відібраних у результаті селекції і призначених для подальших перетворень операторами схрещування і мутації з метою формування нової популяції нащадків. На даному етапі хромосоми з батьківської популяції поєднуються в пари. Це виконується випадковим способом відповідно до ймовірності схрещування pc. Далі для кожної пари відібраних у такий спосіб батьків розігрується позиція гена (локус) у хромосомі, що визначає так називану точку схрещування. Якщо хромосома кожного з батьків складається з L генів, то очевидно, що точка схрещування k являє собою натуральне число, менше L. Тому фіксація точки схрещування зводиться до випадкового вибору числа з інтервалу [1, -1]. У

результаті схрещування пари батьківських хромосом виходить наступна пара нащадків:

1) нащадок, хромосома якого на позиціях від 1 до k складається з генів першого батька, а на позиціях від k+1 до L - з генів другого батька;

2) нащадок, хромосома якого на позиціях від 1 до k складається з генів другого батька, а на позиціях від k+1 до L - з генів першого батька.

Оператор мутації з ймовірністю $p_т$ змінює значення гена в хромосомі на протилежне (тобто з 0 на 1 або назад). Наприклад, якщо в хромосомі [100110101010] мутації піддається ген на позиції 7, то його значення, рівне 1, змінюється на 0, що призводить до утворення хромосоми [100110001010]. Як уже згадувалося вище, ймовірність мутації зазвичай дуже мала, і саме від неї залежить, буде даний ген мутировати чи ні. Ймовірність $p_т$ мутації може емулюватися, наприклад, випадковим вибором числа з інтервалу [0, 1] для кожного гена і добором для виконання цієї операції тих генів, для яких розігране число виявляється меншим або рівним значенню $p_т$.

Формування нової популяції. Хромосоми, отримані в результаті застосування генетичних операторів до хромосом тимчасової батьківської популяції, включаються до складу нової популяції. Вона стає так називаною поточною популяцією для даної ітерації генетичного алгоритму. На кожній черговій ітерації розраховуються значення функції пристосованості для всіх хромосом цієї популяції, після чого перевіряється умова зупинки алгоритму й або фіксується результат у виді хромосоми з найбільшим значенням функції пристосованості, або здійснюється перехід до наступного кроку генетичного алгоритму, тобто до селекції. У класичному генетичному алгоритмі вся попередня популяція хромосом заміщується новою популяцією нащадків, що має ту ж чисельність.

Вибір «найкращої» хромосоми. Якщо умова зупинки алгоритму виконана, то варто вивести результат роботи, тобто представити шукане рішення задачі. Кращим рішенням вважається хромосома з найбільшим значенням функції пристосованості.

На завершення варто визнати, що генетичні алгоритми успадкували властивості природного еволюційного процесу, що складаються в генетичних змінах популяцій організмів з часом. Головний фактор еволюції - це природний відбір (тобто природна селекція), що приводить до того, що серед генетично розрізняючихся особин однієї і тієї ж популяції виживають і залишають у спадщину тільки найбільш пристосовані до навколишнього середовища. У генетичних алгоритмах також виділяється етап селекції, на якому з поточної популяції відбираються і включаються в батьківську популяцію особи, що мають найбільші значення функції пристосованості. На наступному етапі, що іноді називається еволюцією, застосовуються генетичні оператори схрещування і мутації, що виконують рекомбінацію генів у хромосомах.

Операція схрещування полягає в обміні фрагментами

ланцюжків між двома батьківськими хромосомами. Пари батьків для схрещування вибираються з батьківського пула випадковим чином так, щоб ймовірність вибору конкретної хромосоми для схрещування дорівнювала б ймовірності pc. Наприклад, якщо в якості батьків випадковим чином вибираються дві хромосоми з батьківської популяції чисельністю N, то $pc = 2N$. Аналогічно, якщо з батьківської популяції чисельністю N вибирається 2z хромосом ($z \leq N/2$), які

утворять z пар батьків, то pc=2z/N. Звертаємо увагу, що якщо всі хромосоми поточної популяції об'єднані в пари до схрещування, то pc=1. Після операції схрещування батьки в батьківській популяції заміщаються їхніми нащадками.

Операція мутації змінює значення генів у хромосомах із заданою ймовірністю pт способом, представленим при описі відповідного оператора. Це приводить до інвертування значень відібраних генів з 0 на 1 і зворотно. Значення $p_т$, як правило, дуже мале, тому мутації піддається лише невелика кількість генів. Схрещування - це ключовий оператор генетичних алгоритмів, що визначає їхню можливості й ефективність.

Мутація грає більш обмежену роль. Вона вводить у популяцію деяку розмаїтість і попереджає втрати, що могли б відбутися унаслідок виключення якого-небудь значного гена в результаті схрещування.

Основний (класичний) генетичний алгоритм, як ми вже відмічали, відомий у літературі як інструмент, у якому виділяються три види операцій: репродукції, схрещування і мутації. Терміни селекція і репродукція в даному контексті використовуються як синоніми. При цьому репродукція в даному випадку зв'язується скоріше зі створенням копій хромосом батьківського пула, тоді як більш розповсюджений зміст цього поняття означає процес формування нових особин, що відбуваються від конкретних батьків . Якщо ми приймаємо таке тлумачення, то оператори схрещування і мутації можуть вважатися операторами репродукції, а селекція - добором особин (хромосом) для репродукції.

Приклад

Розглянемо простий приклад - задачу знаходження максимуму функції, заданої виразом (1) для цілочислової змінної x, що приймає значення від 0 до 31.

$$f(x) = 2x^2+1 \qquad (1)$$

Для застосування генетичного алгоритму необхідно насамперед закодувати значення змінної **x** у виді двійкових послідовностей. Очевидно, що цілі числа з інтервалу [0, 31] можна представити послідовностями нулів і одиниць, використовуючи їхнє представлення в двійковій системі числення.

Число 0 при цьому записується як 00000, а число 31 - як 11111. У даному випадку хромосоми здобувають вид двійкових послідовностей, що складаються з 5 бітів, тобто ланцюжками довжиною 5.

Також очевидно, що в ролі функції пристосованості буде виступати цільова функція *f(x)*, задана виразом (1). Тоді пристосованість хромосоми ch_i, i = 1,2.....N, буде визначатися значенням функції *f(x)* для **x**, рівного фенотипові, що відповідає генотипові ch_i . Позначимо ці фенотипи ch_i*. У такому випадку значення функції пристосованості хромосоми ch_i (тобто $F(ch_i)$) буде дорівнювати $f(ch_i*)$. Виберемо випадковим чином вихідну популяцію, що складається з 6 кодових послідовностей (наприклад, можна 30 разів підкинути монету); при цьому N=6.

Припустимо, що обрано хромосоми

ch_1 = [10011]

ch_2 = [00011]

ch_3 = [00111]

ch_4 = [10101]

ch_5 = [01000]
ch_6 = [11101]

Відповідні ним фенотипи - це представлені нижче числа з інтервалу від 0 до 31:
ch_1* = 19
ch_2* =3
ch_3*= 7
ch_4* = 21
ch_5* = 8
ch_6* = 29

По формулі $f(x) = 2x^2+1$ (1) розраховуємо значення функції пристосованості для кожної хромосоми в популяції й одержуємо

$F(ch_1)$ = 723
$F(ch_2)$= 19
$F(ch_3)$ = 99
$F(ch_4)$ = 883
$F(ch_5)$ = 129
$F(ch_6)$=1683

Селекція хромосом. *Методом рулетки*, вибираємо 6 хромосом для репродукції. Колесо рулетки представлено на рис. 10.

$$v(ch_i) = p_s(ch_i)100\% , \quad (2)$$

де

$$p_s(ch_i)=\frac{F(ch_i)}{\sum_{i=1}^{N} F(ch_i)} \quad (3)$$

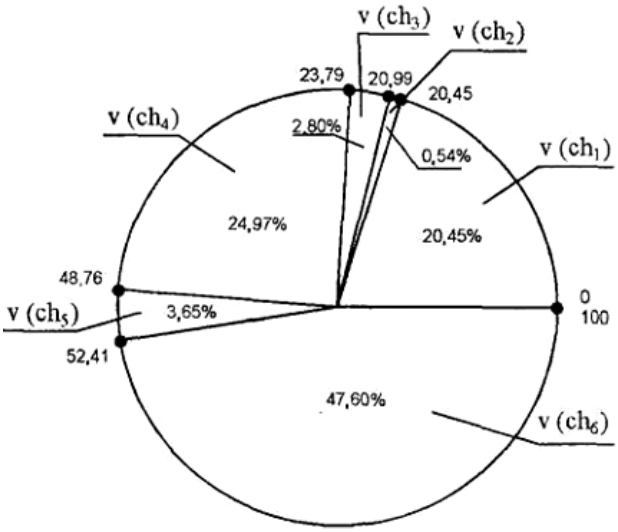


Рис. 10. Колесо рулетки для селекції в прикладі 2.

Припустимо, що обрано числа

97 26 54 13 31 88.

Це означає вибір хромосом

$ch_6\ ch_4\ ch_6\ ch_1\ ch_4\ ch_6$

Нехай схрещування виконується з ймовірністю p_c=1. Припустимо, що для схрещування сформовані пари

ch_1 і ch_4 , ch_4 і ch_6 , ch_6 і ch_6 .

Крім того, припустимо, що випадковим чином обрана точка схрещування, рівна 3 для хромосом ch_1, і ch_4, а також точка схрещування, рівна 2 для хромосом ch_4 і ch_6 (рис.11).

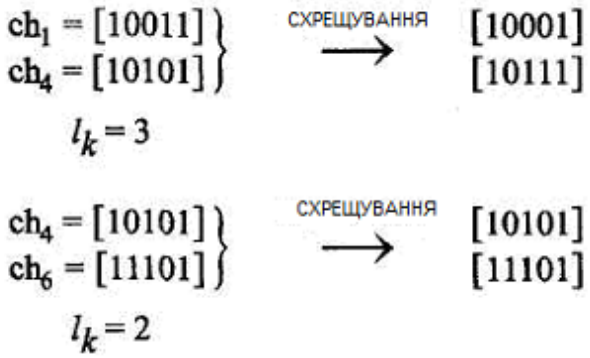


Рис. 11. Процес схрещування хромосом у прикладі 2.

За умови, що ймовірність мутації p_m=0, у нову популяцію включаються хромосоми

Ch_1 = [10001]
Ch_2 = [10111]
Ch_3 = [10101]
Ch_4 = [11101]
Ch_5 = [11101]
Ch_6 = [11101]

Для розрахунку значень функції пристосованості цих хромосом необхідно декодувати їхні двійкові послідовності, що представляють, і одержати відповідні їм фенотипи. Позначимо їх Ch_i*. У результаті декодування одержуємо числа (з інтервалу від 0 до 31)

Ch_1*= 17
Ch_2* = 23
Ch_3* = 21
Ch_4* = 29
Ch_5* = 29
Ch_6* = 29

Відповідно, значення функції пристосованості хромосом нової популяції, розраховані по формулі (1), складуть

$F(Ch_1)$= 579

$F(Ch_2)=1059$
$F(Ch_3)= 883$
$F(Ch_4)=1683$
$F(Ch_5) = 1683$
$F(Ch_6)=1683$

Легко помітити, що в цьому випадку середнє значення пристосованості зросло з 589 до 1262.

Звертаємо увагу, що якщо на наступній ітерації будуть сформовані для схрещування пари хромосом, наприклад, Ch_4 і Ch_2, Ch_5 і Ch_2 або Ch_6 і Ch_2 із точкою схрещування 2 або 3, то серед інших буде отримана хромосома [11111] з фенотипом, рівним числу 31, при якому оптимізуєма функція досягає свого максимуму. Значення функції пристосованості для цієї хромосоми виявляється найбільшим і складає 1923. Якщо таке сполучення пар у даній ітерації не відбудеться, то можна буде очікувати утворення хромосоми з найбільшим значенням функції пристосованості на наступних ітераціях. Хромосома [11111] могла бути отримана і на поточній ітерації у випадку формування для схрещування пари Ch_1 і Ch_6 із точкою схрещування 3.

Відзначимо, що при довжині хромосом, рівній 5 бітам, простір пошуку дуже малий і нараховує всього $2^5=32$ точки.

Представлений приклад має винятково демонстраційний характер. Застосування генетичного алгоритму для такого простого прикладу практично недоцільне, оскільки його оптимальне рішення може бути отримане миттєво. Однак цей приклад придатний для вивчення функціонування генетичного алгоритму.

Також варто згадати, що в малих популяціях часто зустрічаються ситуації, коли на початковому етапі кілька особин мають значно більші значення функції приналежності, чим інші особи даної популяції. Застосування методу селекції на основі «колеса рулетки» дозволяє в цьому випадку дуже швидко вибрати «найкращі» особи, іноді - протягом «життя» одного покоління. Однак такий розвиток подій вважається небажаним, оскільки він стає головною причиною передчасної збіжності алгоритму, що називається збіжністю до неоптимального рішення. З цієї причини використовуються й інші методи селекції, що відрізняються від колеса рулетки, або застосовується масштабування функції пристосованості.

5. Мурашині алгоритми

http://habrahabr.ru/post/105302/

Мураху не можна назвати кмітливою. Окрема мураха не в змозі прийняти жодного рішення. Вона влаштована вкрай примітивно: всі її дії зводяться до елементарних реакцій на навколишнє оточення і своїх побратимів. Мураха не здатна аналізувати, робити висновки і шукати рішення.

Ці факти, однак, ніяк не узгоджуються з успішністю мурах як виду. Вони існують на планеті понад 100 мільйонів років, будують величезні споруди,

забезпечують їх всім необхідним і навіть ведуть справжні війни. У порівнянні з повною безпорадністю окремих особин, досягнення мурах здаються немислимими.

Домогтися таких успіхів мурахи здатні завдяки своїй соціальності. Вони живуть тільки в колективах - колоніях. Всі мурахи колонії формують так званий ройовий інтелект. Особини, що складають колонію, не повинні бути розумними: вони повинні лише взаємодіяти за певними простими правилами і тоді колонія цілком буде ефективною.

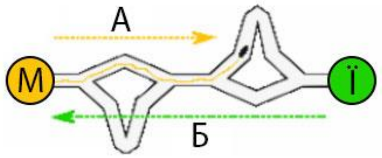
В колонії немає домінуючих особин, немає начальників і підлеглих, немає лідерів, що роздають вказівки і координують дії. Колонія є повністю самоорганізованою. Кожна з мурах володіє інформацією лише про локальну обстановку, жодна з них не має уявлення про всю ситуації в цілому - тільки про те, що дізналася сама або від своїх родичів, явно чи неявно. На неявних взаємодіях мурах засновано механізми пошуку найкоротшого шляху від мурашника до джерела їжі.

Кожен раз проходячи від мурашника до їжі і назад, мурахи залишають за собою доріжку феромонів. Інші мурахи, відчувши такі сліди на землі, будуть інстинктивно спрямовуватися до нього. Оскільки ці мурахи теж залишають за собою доріжки феромонів, то чим більше мурах проходить певним шляхом, тим більш привабливим він стає для їх родичів. При цьому, чим коротше шлях до джерела їжі, тим менше часу потрібно мурашкам на нього - а отже, тим швидше залишені на ньому сліди стають помітними.

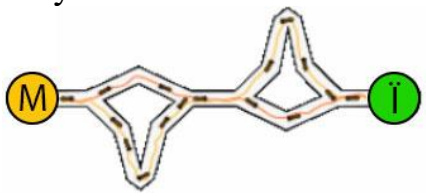
У 1992 році у своїй дисертації Марко Дориго запропонував запозичити описаний природний механізм для вирішення завдань оптимізації. Імітуючи поведінку колонії мурах в природі, мурашині алгоритми використовують багатоагентні системи, агенти яких функціонують по вкрай простим правилам. Мурашині алгоритми є ефективними при вирішенні складних комбінаторних завдань та задач оптимізації.

Ідея алгоритму

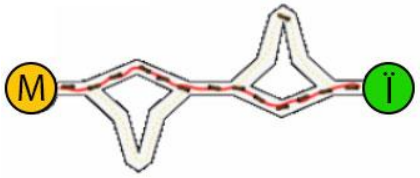
Мураха прямує від мурашника (М) у випадковому напрямку. Якщо вона знаходить їжу (Ї), то повертається до мурашника і залишає по собі слід з **феромону**.



Ці феромони приваблюють інших мурах, що знаходяться поруч, і скоріш за все вони рушать цим маршрутом. Повертаючись до мурашника, вони також залишають свій феромон і укріплюють доріжку.



Якщо існує два маршрути, то коротким за цей час встигне пройти більше мурах ніж по довгому і короткий маршрут стане більш вигідним. Довгші доріжки повільно зникають внаслідок випаровування феромонів.



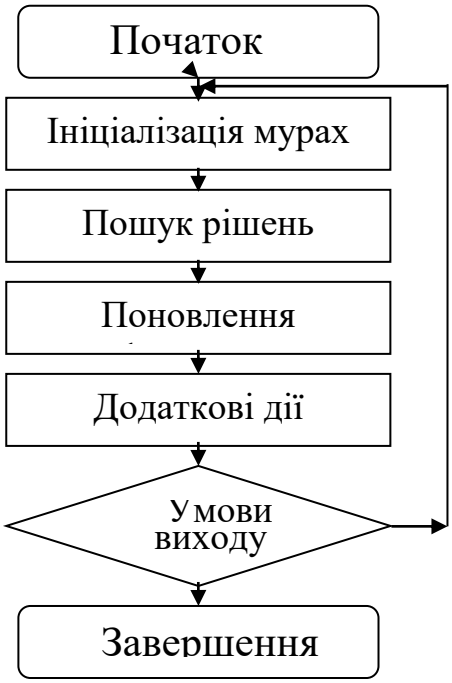
Базова ідея алгоритму мурахи полягає в оптимізації шляхом непрямого зв'язку між автономними агентами.

Мурашиний алгоритм моделює багатоагентну систему. Її агентів надалі будемо називати мурахами. Як і справжні мурахи, вони досить просто влаштовані: для виконання своїх обов'язків вони вимагають невелику кількість пам'яті, а на кожному кроці роботи виконують нескладні обчислення.

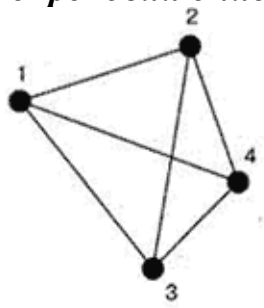
Кожна мураха зберігає в пам'яті список пройдених нею вузлів. Цей список називають **списком заборон (tabu list)** або просто пам'яттю мурашки. Вибираючи вузол для наступного кроку, мураха «пам'ятає» про вже пройдені вузли і не розглядає їх як можливих для переходу. На кожному кроці список заборон поповнюється новим вузлом, а перед новою ітерацією алгоритму - тобто перед тим, як мурашка знову проходить шлях - він очищується.

Окрім списку заборон, при виборі вузла для переходу мураха керується «**привабливістю**» ребер, які вона може пройти. *Привабливість залежить, по-перше, від відстані між вузлами (тобто від ваги ребра), а по-друге, від слідів феромонів, залишених на ребрі мурахами, що пройшли по ньому раніше.* Природно, що на відміну від ваг ребер, які є константними, сліди феромонів оновлюються на кожній ітерації алгоритму: як і в природі, з часом сліди випаровуються, а мурахи, що проходять, навпаки, посилюють їх.

Узагальнено, базовий мурашиний алгоритм, незалежно від модифікацій, можна представити у вигляді схеми



Покроковий опис загальної схеми



Припустимо, що навколишнє середовище для мурах представляє повнозв’язний неорієнтований граф. Кожне ребро має вагу, яка позначається як відстань між двома вершинами, що ним з’єднується. Граф є двохскерованим, тому мураха може подорожувати по грані в будь-якому напрямку.

Ймовірність включення ребра в маршрут окремої мурахи пропорційна до кількості феромонів на цьому ребрі, а кількість відкладеного феромону пропорційна до довжини маршруту. Чим коротший маршрут, тим більше феромону буде відкладено на його ребрах, отже, більша кількість мурах буде включати його в синтез власних маршрутів. *Моделювання такого підходу, що використовує тільки додатній зворотний зв'язок, призводить до передчасної збіжності - більшість мурашок рухається по локально-оптимальному маршруту*.

Уникнути цього можна моделюючи від’ємний зворотний зв'язок у вигляді випаровування феромону. Причому, якщо феромон випаровується швидко, то це призводить до втрати пам'яті колонії і забування хороших рішень, з іншого боку, збільшення часу випаровування може призвести до отримання стійкого локального оптимального рішення.

Мураха

Мураха - це програмний агент, який є членом великої колонії і використовується для вирішення певної проблеми. Мураха забезпечується набором простих правил, які дозволяють їй вибирати шлях у графі. Вона підтримує список вузлів, які вже відвідала і може пройти через кожен вузол лише один раз. Шлях між двома вузлами графа, за яким мураха відвідала кожен вузол, називається шляхом Гамільтона.

Вузли в списку "поточної подорожі" розташовуються в тому порядку, в якому їх відвідала мураха. Пізніше список використовується для визначення протяжності шляху між вузлами. Справжня мураха під час переміщення по шляху буде залишати за собою феромони. В мурашиному алгоритмі агент залишає феромони на ребрах графа після завершення подорожі.

Стартова точка, куди поміщається мураха, залежить від обмежень, накладених умовами завдання, оскільки для кожного завдання спосіб розміщення мурашок *є визначальним*. Або всі вони поміщаються в одну точку, або в різні з повтореннями, або без повторень.

На цьому ж етапі задається початковий рівень феромону. Він ініціалізується невеликим додатнім числом для того, щоб на початковому кроці ймовірності переходу в наступну вершину були нульовими.

Рух мурашки

Рух мурашки ґрунтується на простому ймовірнісному рівнянні. Якщо мураха ще не закінчила шлях, тобто не відвідала усі вузли мережі, для визначення наступного ребра шляху використовується рівняння

$$P = \frac{\tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}{\sum_k \tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}. \quad (1)$$

Тут $\tau(r,u)$ - інтенсивність ферменту на ребрі між вузлами *r* і *u*, $\eta(r,u)$ - функція, яка представляє вимір зворотньої відстані для грані, α - вага ферменту, а β - коефіцієнт евристики. Параметри α і β визначають відносну значимість двох параметрів, а також їх вплив на рівняння. Оскільки мураха подорожує тільки по вузлах, які ще не були відвідані (як зазначено списком табу), ймовірність обчислюється лише для ребер, які ведуть до ще не відвіданих вузлів. Ці ребра представляє змінна k.

Подорож мурахи

Пройдений мурахою шлях відображається, коли мураха відвідає всі вузли графа. Цикли заборонено, оскільки в алгоритм включено список табу. Після завершення довжина шляху може бути підрахована - вона дорівнює сумі довжин всіх ребер, якими подорожувала мураха. Рівняння (2) показує кількість феромону, який був залишений на кожному ребрі шляху для мурашки *k*. Змінна *Q* є константою.

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L^{k}(t)}. \quad (2)$$

Результат рівняння *є засобом вимірювання шляху,* - короткий шлях характеризується високою концентрацією феромонів, а більш довгий шлях - більш низькою. Далі, отриманий результат використовується в рівнянні (3), щоб збільшити кількість феромону вздовж кожного ребра пройденого мурахою шляху.

$$\tau_{ij}(t) = \Delta\tau_{ij}(t) + (\tau_{ij}^{k}(t) \times \rho). \quad (3)$$

Важливо, що дане рівняння застосовується до всього шляху, при цьому кожне ребро позначається феромоном пропорційно до довжини шляху. Тому слід дочекатися, поки мураха закінчить подорож і лише потім оновити рівні феромону, в іншому випадку справжня довжина шляху залишиться невідомою. Константа *p* - значення між 0 і 1.

Випаровування феромонів

На початку шляху у кожного ребра є шанс бути обраним. Щоб поступово видалити ребра, які входять в гірші шляхи графа, до всіх ребер застосовується процедура випаровування феромону. Використовуючи константу *p* з рівняння (3), отримуємо рівняння (4):

$$\tau_{ij}(t) = \tau_{ij}(t) \times (1 - \rho). \quad (4)$$

Для випаровування феромону використовується зворотний коефіцієнт оновлення шляху.

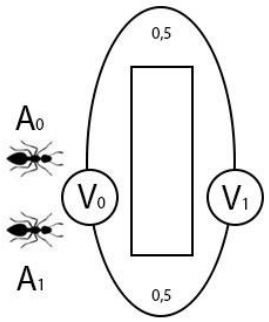
Повторний запуск

Після того, як шлях мурашки завершено, ребра оновлено відповідно до довжини шляху і сталося випаровування феромону на всіх ребрах, алгоритм запускається повторно. Список табу очищується, і довжина шляху обнулюється. Мурахам дозволяється переміщатися по графу, засновуючи вибір ребра на рівнянні (1). Цей процес може виконуватися для постійної кількості шляхів або до моменту, коли протягом кількох запусків не було відзначено повторних змін. Потім визначається кращий шлях, який і є рішенням.

Демонстраційний приклад

Щоб побачити, як працюють рівняння, розберемо функціонування розглянутого вище алгоритму на простому прикладі. Візьмемо простий сценарій з двома мурашками з прикладу яке розглянуто вище.

На рисунку показано приклад з двома ребрами між двома вузлами (V_0 і V_1). Кожне ребро ініціалізується і має однакові шанси на те, щоб бути обраним.

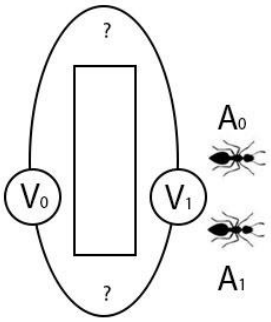


Два мурахи, що знаходяться у вузлі V_0 позначаються як A_0 і A_1. Оскільки ймовірність вибору будь-якого шляху однакова, в цьому циклі проігноруємо рівняння вибору шляху.

Дані для завдання:

- Число пройдених кроків: для A_0 - 20, для A_1 - 10
- Рівень феромону (Q / пройдена відстань): для A_0 - 0.5, A_1 - 1.0
- $\rho = 0.6$
- $\alpha = 0.3$
- $\beta = 1.0$

На наступному рисунку показано, що кожна мураха вибирає свій шлях (мураха A_0 йде по верхньому шляху, а мураха A_1 - по нижньому). Мураха A_0 зробила 20 кроків, а мураха A_1, - тільки 10. За рівнянням $\Delta\tau_{ij}^k(t) = \frac{Q}{L^k(t)}$. (2) розраховуємо кількість феромонів, яке має бути "нанесено".



Примітка: Роботу алгоритму можна змінити, якщо перевизначити його параметри (наприклад, α, β або ρ), наприклад надати більшу вагу феромонам або відстані між вузлами.

Далі за рівнянням $\tau_{ij}(t) = \Delta\tau_{ij}(t) + (\tau_{ij}^k(t) \times \rho).$ (3) обчислюється кількість феромону, яка буде застосовано.

Для мурахи A_0 результат становить: 0,1+(0,5*0,6)=0,4

Для мурахи A_1 результат становить: 0,1+(0,5*0,6)=0,4

Далі за допомогою рівняння $\tau_{ij}(t) = \tau_{ij}(t) \times (1 - \rho).$ (4) визначається, яка частина феромонів випарується і, відповідно, скільки залишиться. Результати (для кожного шляху відповідно) становлять:

0,4*(1,0-0,6)=0,16

0,7*(1,0-0,6)=0,28

Ці значення представляють нову кількість феромонів для кожного шляху (верхнього і нижнього, відповідно). Тепер перемістимо мурах назад у вузол V_0 і скористаємося імовірнісним рівнянням вибору шляху $P = \frac{\tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}{\sum_k \tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}.$ (1), щоб визначити, який шлях повинні вибрати мурахи. Ймовірність того, що мураха вибере верхній шлях (представлений кількістю феромону 0,16), становить:

$$\frac{(0,16)^{3,0} \times (0,5)^{1,0}}{(0,16)^{3,0} \times (0,5)^{1,0}} + (0,28)^{3,0} \times (1,0)^{1,0} = \frac{0,002048}{0,024} = P(0,085).$$

Ймовірність того, що мураха вибере нижній шлях (представлений кількістю феромону 0,28) становить:

$$\frac{(0,28)^{3,0} \times (1,0)^{1,0}}{(0,16)^{3,0} \times (0,5)^{1,0}} + (0,28)^{3,0} \times (1,0)^{1,0} = \frac{0,021952}{0,024} = P(0,915).$$

При зіставленні двох ймовірностей обидві мурахи виберуть нижній шлях, який є найбільш оптимальним.

Характерні особливості

Для алгоритму мурашиної колонії необхідно вказати:

- Закон виділення феромону.
- Закон випаровування феромону.
- Кількість агентів.
- Місця розміщення.

Ці характеристики вибираються з врахуванням особливостей завдання на основі експериментальних досліджень (евристики).

Алгоритм:

- Реалізує пошук наближених рішень.
- Має поліноміальну складність.
- Є одним з видів імовірнісних алгоритмів (закони виділення випаровування - імовірнісні закони).

Області застосування

Алгоритм оптимізації мурашиної колонії може бути успішно застосований для вирішення складних комплексних завдань оптимізації. Мета вирішення складних комплексних завдань оптимізації - пошук і визначення найбільш відповідного рішення для оптимізації (знаходження мінімуму або максимуму) цільової функції (ціни, точності, часу, відстані тощо) з дискретної множини можливих рішень.

Типовими прикладами вирішення такого завдання є *задача календарного планування, завдання маршрутизації транспорту, різних мереж (GPRS, телефонні, комп'ютерні тощо), розподіл ресурсів та робіт.* Ці задачі виникають у бізнесі, інженерії, виробництві та багатьох інших областях. Дослідження показали, що

метод мурашиних колоній може давати результати, навіть кращі ніж при використанні генетичних алгоритмів і нейронних мереж.

Модифікації класичного алгоритму

Результати перших експериментів із застосуванням мурашиного алгоритму для вирішення завдання комівояжера були багатообіцяючими, проте далеко не кращими порівняно з вже існуючими методами. Однак, простота класичного мурашиного алгоритму («мурашиної системи») залишала можливості для доробок - і саме алгоритмічні вдосконалення стали предметом подальших досліджень фахівців у галузі комбінаторної оптимізації. В основному, ці вдосконалення пов'язано з великим використанням історії пошуку та більш ретельним дослідженням областей навколо вже знайдених вдалих рішень.

Elitist Ant System

Введення в алгоритм так званих «елітних мурах». Досвід показує, що проходячи ребра, що входять в короткі шляхи, мурахи з більшою ймовірністю будуть знаходити коротші шляхи. Ефективною стратегією є штучне збільшення рівня феромонів на найвдаліших маршрутах. Для цього на кожній ітерації алгоритму кожна з елітних мурах проходить шлях, який є найкоротшим із знайдених на даний момент.

Експерименти показують, що, до певного рівня, збільшення числа елітних мурах є досить ефективним, дозволяючи значно скоротити число ітерацій алгоритму. Однак, якщо число елітних мурах занадто велике, то алгоритм досить швидко знаходить субоптимальне рішення і застряє в ньому. Як і інші змінні параметри, оптимальне число елітних мурах слід визначати дослідним шляхом.

Ant-Q

Мурашиний алгоритм, який отримав свою назву за аналогією з методом машинного навчання Q-learning. В основі алгоритму лежить ідея про те, що мурашину систему можна інтерпретувати як систему навчання з підкріпленням. Ant-Q підсилює цю аналогію, запозичуючи багато ідей з Q-навчання.

Алгоритм зберігає Q-таблицю, що співставляє кожному з ребер величину, яка визначає «корисність» переходу по цьому ребру. Q-таблиця змінюється в процесі роботи алгоритму - відбувається навчання системи. Значення корисності переходу по ребру обчислюється виходячи із значень корисності переходу за наступними ребрам в результаті попереднього визначення можливих наступних станів. Після кожної ітерації корисності оновлюються виходячи з довжин шляхів, до складу яких було включено відповідні ребра.

Ant Colony System

Для підвищення ефективності в порівнянні з класичним алгоритмом введено три основних зміни.

По-перше, рівень феромонів на ребрах оновлюється не лише в кінці чергової ітерації, але і при кожному переході мурах з вузла у вузол. По-друге, наприкінці ітерації рівень феромонів підвищується тільки на найкоротшому із знайдених шляхів. По-третє, алгоритм використовує змінене правило переходу: або, з певною часткою ймовірності, мураха безумовно вибирає краще ребро у відповідності до довжини і рівня феромонів, або робить вибір так само, як і в класичному алгоритмі.

Max-min Ant System

Мурашиний алгоритм, в якому підвищення концентрації феромонів відбувається тільки на кращих шляхах з пройдених мурахами. Така велика увага до локальних оптимумів компенсується введенням обмежень на максимальну і мінімальну концентрацію феромонів на ребрах, які вкрай ефективно захищають алгоритм від передчасної збіжності до субоптимальних рішень.

На етапі ініціалізації, концентрація феромонів на всіх ребрах встановлюється рівною максимальній. Після кожної ітерації алгоритму тільки одна мураха залишає за собою слід - або найбільш успішна на даній ітерації, або, аналогічно до алгоритму з елітизмом, елітна. Цим досягається, з одного боку, більш ретельне дослідження області пошуку, з іншого - його прискорення.

ASrank

Модифікація класичного мурашиного алгоритму, в якому в кінці кожної ітерації мурахи ранжуються у відповідно до довжин пройдених ними шляхів. Кількість феромонів, що залишається мурахою на ребрах, таким чином, призначається пропорційно до її позиції. Для ретельного дослідження околу вже знайдених вдалих рішень, алгоритм використовує елітних мурах.

Висновок

Ефективність мурашиних алгоритмів порівнянна з ефективністю загальних метаевристичних методів, а в ряді випадків - і з проблемно-орієнтованими методами. Найкращі результати мурашині алгоритми показують для задач з великою розмірністю областей пошуку і можуть бути успішно застосовані для вирішення складних комбінаторних задач оптимізації. Мурашині алгоритми добре підходять для застосування разом з процедурами локального пошуку, дозволяючи швидко знаходити початкові точки для них.

Найбільш перспективними напрямками подальших досліджень у даному напрямку слід вважати аналіз способу вибору параметрів, що налаштовуються в алгоритмах. В останні роки пропонуються різні способи адаптації параметрів алгоритмів «на льоту». Оскільки від вибору параметрів сильно залежить поведінка мурашиних алгоритмів, саме до цієї проблеми звернено найбільшу увагу дослідників на даний момент.

6. Еволюційні моделі

Оскільки еволюція є основним механізмом обробки інформації в природних системах, **дослідники прагнуть побудувати теоретичні та комп'ютерні моделі, що реально пояснюють принципи роботи цього механізму**. Для досліджень цього напрямку характерне розуміння, що моделі повинні відтворювати структуру популяції, народження і зникнення особин, а також події між ними.

Найчастіше залучаються наступні концепції:

Ройовий інтелект. Описує колективну поведінку децентралізованої самоорганізованої системи. Розглядається в теорії штучного інтелекту як метод оптимізації. Системи ройового інтелекту, як правило, складаються з множини агентів (багатоагентна система), що локально взаємодіють між собою і з навколишнім середовищем. Самі агенти, зазвичай, є достатньо простими, але всі

разом, локально взаємодіючи, створюють так званий ройовий інтелект. Прикладом в природі може служити колонії мурашок, рій бджіл, зграя птахів, риб.

Соціологічний напрямок. Людське суспільство представляє реальний інструмент обробки інформації, який добре піддається спостереженню і задокументований (на відміну від людського мозку). Якщо ройовий інтелект орієнтовано на отримання складної поведінки в системі з простих елементів, цей підхід, навпаки, досліджує побудову простих і спеціальних структур на базі складних і універсальних об'єктів: «спільнота є менш розумна, ніж більшість її членів».

Для цього напрямку характерне прагнення дати соціологічним поняттям визначення з області інформатики. Наприклад, еліта визначається як носій певної приватної моделі реального світу, а базис (тобто народ) грає роль арбітра між елітами. Еволюційний процес полягає в народженні і загибелі еліт. Базис не в змозі розібратися в суті ідей і моделей, що представляються елітами, і не ставить перед собою такого завдання. Однак, в силу своєї незалученості зберігає здатність до ясної емоційної оцінки, що дозволяє йому легко відрізняти харизматичні еліти від загниваючих, що намагаються зберегти свої привілеї, розуміючи, що їхня ідея або модель не підтвердилася.

Колективний інтелект. Термін з'явився в середині 1980-х років в соціології при вивченні процесу колективного прийняття рішень. Дослідники визначили колективний інтелект як здатність групи знаходити вирішення завдань більш ефективним, ніж найкраще індивідуальне рішення в цій групі.

Досягнення людства народжені колективним розумом. *Люди - нейрони цивілізації.* Роблячи кожен свою справу, удосконалюючись в ньому і обмінюючись результатами своєї праці, ми навчилися створювати речі, які навіть не розуміємо. У своєму есе «Я, Олівець» економіст Леонард Рід зауважує, що жодна людина на світі не знає, як виготовити олівець, - це знання розподілено між тисячами шахтарів, дроворубів, дизайнерів і фабричних робітників.

Великі проекти не робляться поодинці. З часів будівництва єгипетських пірамід результат досягався умінням організувати спільну роботу великої кількості людей. *У війні та на виробництві добре зарекомендували себе ієрархічні схеми, а інформаційні технології за останню чверть століття довели їх практично до досконалості. Однак у сфері інновацій все залежить від особистого досвіду і творчого інтелекту окремої людини.* Звідси і ризики, що пов'язані з астрономічними гонорарами зіркових топ-менеджерів, які своєю інтуїцією повинні компенсувати відсутність технологій для роботи з прихованим знанням персоналу.

Діяльність колективного інтелекту, організована за аналогією з нейронною мережею, обіцяє досягнення якісно нових результатів.

Але як ефективно задіяти інтелект великого числа людей? Ради та комітети не бувають розумнішими і оригінальніше за своїх креативних і компетентних членів. Великі експертні групи обережні, повільні і неоригінальні. Виходить, інтелект, який розуміють як здатність до пошуку нестандартних рішень, неадитивний?

Так вважалося до недавнього часу.

Кожен з нас - багаторівнева структура з безлічі біологічних елементів, у тому числі відповідальних за інформаційні потоки. Кожна клітина, що бере участь в

інформаційному потоці, може мати багато інформаційних контактів. Кількість інформаційних зв'язків обумовлюють інтегровану функцію розуму порівнюють з кількістю піщинок в Сахарі, а кількість можливих комбінацій інформаційних зв'язків всередині цього людського колективного розуму - з кількістю атомів у Всесвіті.

Чи можливе щось подібне, в сенсі колективного розуму, для спільноти особин? З чисто теоретичної точки зору це можливо. Потрібно лише поширити міжклітинні інформаційні зв'язки на весь колектив особин. А от чи є в природі передавач інформації, здатний виконати подібну функцію, - це вже інше питання. Телепатію, якщо вона є, якось сумнівно навіть гіпотетично прив'язати до контактів між окремими клітинами, щоб забезпечити функціонування деякого єдиного колективного суперінтелекту з єдиною функцією свідомості.

Надію на прорив дає аналогія з мозком. Якщо взаємозв'язок нейронів породжує індивідуальний людський інтелект, то, ймовірно, і людей можна організувати в колективний інтелект, який буде сильніше за кожний індивідуальний. Треба тільки знайти правильну архітектуру, що дозволяє задіяти для розвитку бізнесу приховане знання, яке не можна перенести з голів людей в корпоративні бази даних. Розробка таких соціальних бізнес-додатків визначатиме розвиток інформаційних технологій на найближче десятиліття.

У пошуках архітектури колективного інтелекту сформувалися два напрямки, які можна умовно назвати **соціальним і семантичним**.

Соціальний напрямок. Увага концентрується на людях, об'єднаних в соціальну мережу, яка в ході комунікації виявляє ідейних лідерів і допомагає вдосконалювати їх пропозиції. На цьому шляху з'явилися методи мозкового штурму, форсайта, організаційних ігор, проте вони нестабільні за продуктивністю і не масштабуються за кількістю учасників.

Семантичний напрямок. Ставить в центр повідомлення, що несе ідею. По зв'язках між повідомленнями вибудовується семантична мережа, аналіз якої виявляє ключові ідеї. Так будується система наукових публікацій, індекси пошукових систем, рейтинги повідомлень в блогах. Але семантичні мережі схильні до різних типів нестійкості - накрутки, флейм, флешмоб.

Стабілізація і масштабування досягаються за рахунок об'єднання соціальних та семантичних мереж. Вже за наявності найпростіших технічних засобів їх інтеграції (скажімо, «френди + коменти») виникають колективні протоінтелектуальні явища, такі як медіавіруси і «розумні натовпи». В їх основі лежить поширення інформації на хвилі сильних емоцій, і це вже знайшло застосування в комерції та політиці.

Дещо ближче до колективного інтелекту знаходяться приклади **краудсорсингу**, коли компанії залучають допомогу тисяч людей в обмін на призи або участь в доходах. Ще складніше колективна інтелектуальна діяльність по створенню вільного ПЗ та інструментів агрегації інформації, наприклад, сайтів або ринків передбачень. Тут скромні вклади учасників обмінюються на повагу спільноти чи інший виграш.

Краудсорсінг в даний час активно розвивається в якості моделі для вирішення будь-якого виду проблем і завдань, що стоять як перед бізнесом, так і перед державою і суспільством в цілому. В рамках парадигми краудсорсінгу рішення

задачі передається розподіленій і дуже численній групи людей, за рахунок чого вартість і час досягнення результату радикально знижуються.

Приклади краудсорсінгових проектів:

Вікіпедія - електронна енциклопедія, що створюється переважно силами волонтерів.

InnoCentive - компанія, що запрошує вчених за конкурсну винагороду від $ 10 тис. до $ 100 тис. вирішувати завдання, які ставлять такі компанії, як Procter & Gamble, DuPont і BASF.

Threadless - компанія з виробництва футболок з Чикаго, процес розробки дизайну складається виключно з проведення онлайн-конкурсу, переможці щотижневого конкурсу отримують $ 2 тис. І їх робота запускається у виробництво.

Muji - японська меблева компанія, через свій корпоративний сайт збирає ідеї для своїх виробів і приймає рішення про запуск у виробництво за результатами конкурсу.

eBird - проект, який використовує ресурси любителів для спостереження за птахами.

NASA Clickworkers - проект NASA, створений з метою аналізу масиву знімків марсіанської поверхні силами астрономів-аматорів.

Peer-to-Patent - американський проект, заснований на принципі спільної роботи: державне патентне бюро на постійній основі працює з відкритою Інтернет-спільнотою, в розгляді заявок на патенти бере участь мережа волонтерів (вчені, технічні фахівці, люди, чия кваліфікація дозволяє брати участь у процесі патентування).