*дата :*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| КНC-13 | Розрахункова робота | Фреймворк HeuristicLab |  |  |
| Шмігельський В. | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р. З. | |

1. **Опис фреймворку.**

HeuristicLab це програмне середовище для евристичних і еволюційних алгоритмів, розроблене співробітниками Heuristic and Evolutionary Algorithm Laboratory (HEAL). HeuristicLab приділяє велику увагу представленню графічного користувальницького інтерфейсу, так, що користувачі не повинні мати особливих навичок програмування, для налаштування і розширення алгоритмів для конкретної задачі.

Алгоритми HeuristicLab представлені у вигляді операторних графів, а зміна або перегрупування операторів може бути зроблена шляхом перетягування без фактичного написання коду. Таким чином, програмне забезпечення намагається змістити можливості розробки алгоритмів від інженера програмного забезпечення до користувача і практичного спеціаліста. Розробники можуть ще розширити функціональність на рівні коду і можуть використовувати механізм модуля HeuristicLab, що дозволяє їм інтегрувати користувальницькі алгоритми, представлення рішень або проблеми оптимізації.

Стефан Вагнер і Майкл Афенцелер розпочали розробку HeuristicLab в 2002 році. Основна мотивація для розробки HeuristicLab полягала в створенні незалежного від парадигми, гнучкого, розширюваного і комфортного середовища для евристичної оптимізації з використанням сучасних концепцій програмування. Оскільки інфраструктура Microsoft .NET, здавалося, відповідала цим вимогам, вона була обрана в якості середовища розробки, а C# в якості мови програмування.

Перша офіційно доступна версія HeuristicLab була версія 1.0, випущена в 2004 році, та з покращеною версією 1.1 в 2005-му. Розробка наступної версії HeuristicLab розпочалася в тому ж році. Серед багатьох речей, було вирішено, що HeuristicLab 2.0 повинен забезпечити абсолютно новий користувальницький досвід і зняти тягар програмування з користувача. Таким чином, HeuristicLab 2.0 була першою версією, що містила графічні інструменти для створення алгоритмів, однак через складність користувальницького інтерфейсу HeuristicLab 2.0 так і не був випущений для публічного доступу. Влітку 2007-го року було прийнято рішення про необхідність нової ітерації HeuristicLab, яка мала б поєднувати в собі зручність і простоту використання версії 1.1 з концепціями моделювання алгоритмів версії 2.0. Пакет HeuristicLab 3.0 був випущений в середині компанії на початку 2008-го року. Протягом наступних двох років HeuristicLab поступово поліпшувався, що призвело до випуску версії 3.3 влітку 2010-го року, у вигляді програмного забезпечення з відкритим вихідним кодом.

1. **Можливості та функції фреймворку.**

**Дизайнер алгоритмів**

Однією з особливостей, яка відрізняє HeuristicLab від багатьох інших метаевристичних фреймворків програмного забезпечення є розробник алгоритму. HeuristicLab дозволяє моделювати алгоритми в графічному вигляді без необхідності написання будь-якого вихідного коду (рис.1). Алгоритми в HeuristicLab є композицією операторів, які з'єднані один з одним. Ця послідовність операторів називається графом операторів і може бути переглянута і відредагована для будь-якого алгоритму в HeuristicLab.

HeuristicLab також пропонує, так званий, програмований оператор, який може включати в себе вихідний код, який може бути записаний всередині HeuristicLab. Дивлячись як працюють інші алгоритми можна спробувати нові ідеї, починаючи з існуючого алгоритму і поступово змінюючи його. Звичайно, нові алгоритми також можуть бути створені шляхом написання коду.

**Дизайнер експерименту**

В дизайнері експериментів HeuristicLab можуть бути складені, виконані і проаналізовані різні алгоритми з різними налаштуваннями параметрів і проблем. Це дуже корисно для задач налаштування параметрів, де необхідно виконати і порівняти різні параметри. HeuristicLab пропонує цілий ряд інструментів для графічного аналізу результатів.

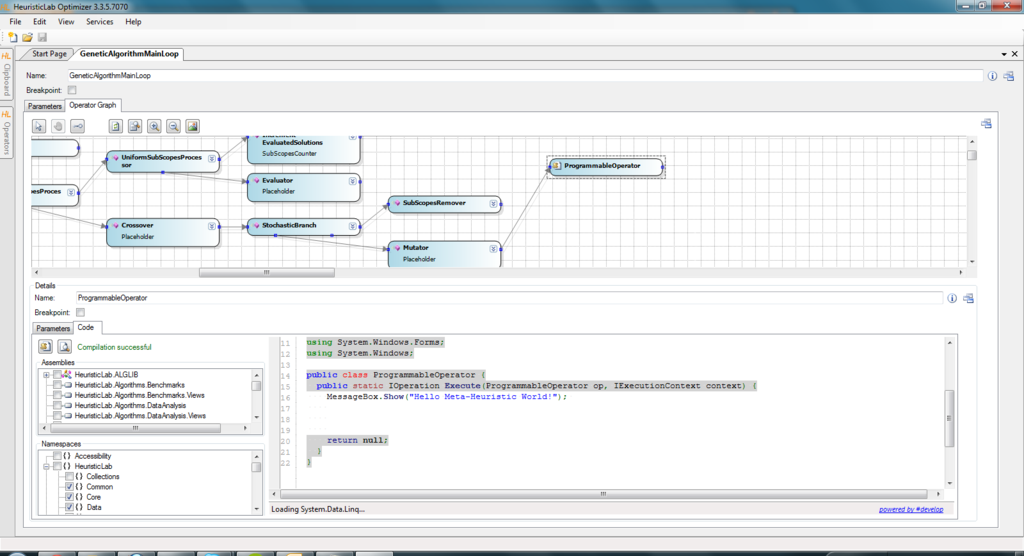


Рис.1. Дизайнер алгоритмів та програмований оператор.

**Інфраструктура плагінів**

Кожна функціональність HeuristicLab доступна в якості плагіна. Розробники можуть створювати і повторно використовувати плагіни для інтеграції нових функцій і розширення функціональних можливостей HeuristicLab.

**Інші особливості та функції**

* Моделі генетичного програмування можуть бути спрощені. Дерева генетичного програмування можуть бути експортовані в MATLAB, LaTeX, Excel або інші формати.
* Алгоритми, проблеми, досліди і результати можуть бути збережені. Алгоритми можуть виконуватися, призупинятися, зберігатися, відновлюватися і продовжуватися.
* Алгоритми і експерименти можуть виконуватися паралельно на багатоядерних і розподілених обчислювальних системах.
* Графіки можуть бути налаштовані і експортовані в різні графічні формати.
* Результати та інші дані можуть бути скопійовані з Microsoft Excel або інших додатків.
* HeuristicLab можна поєднувати із зовнішніми додатками, такими як імітаційні моделі, з метою оптимізації їх параметрів.
* Підтримка розподілених обчислень (HeuristicLab Hive) на основі моделі master-slave (ведучий-підлеглий).

1. **Інші особливості та функції**

HeuristicLab включає в себе велику кількість алгоритмів та проблем.

**Алгоритми**

1. Машинне навчання: перехресна перевірка; генетичне програмування; генетичний алгоритм відбору потомства; k-середніх; лінійний дискримінантний аналіз; лінійна регресія; нелінійна регресія; поліноміальна логітна класифікація; регресія і класифікація найближчих сусідів; регресія та класифікація нейронних мереж; регресія і класифікація випадкового лісу; векторна регресія та класифікація; регресія гаусівського процесу та класифікація найменших квадратів; дерева з градієнтним додаванням; градієнтна бустинг регресія; аналіз компонентів найближчих сусідів; еластичні мережі.
2. Популяційні алгоритми: ALPS; CMA-ES; еволюційна стратегія; генетичні алгоритми; острівний генетичний алгоритм; острівний генетичний алгоритм відбору потомства; еволюційна стратегія відбору потомства (OSES); генетичний алгоритм відбору потомства (OSGA); безпараметрична піраміда популяції (P3); оптимізація методом рою часток; RAPGA; SASEGASA; розсіюваний пошук.
3. Алгоритми, засновані на траєкторних обчисленнях: локальний пошук; табу-пошук; імітація відпалу; пошук змінних сусідів.
4. Багатоцільові алгоритми: NSGA-II.
5. Оптимальні алгоритми: угорський алгоритм; LM-BFGS.
6. Розширювані алгоритми: алгоритми, визначені користувачем.
7. Інші алгоритми: тести продуктивності.

**Проблеми(задачі)**

1. Генетичне програмування: штучна мураха; газонокосарка; перевірка на парність; мультиплексор; robocode; торгівля; символьна класифікація; символьна регресія; символьна регресія koza-style; прогнозування символьних часових рядів; граматична еволюція;
2. Аналіз даних: класифікація; кластеризація; регресія.
3. Комбінаторні задачі: оманлива пастка; HIFF; планування роботи магазину; ранець; упаковка в контейнери; лінійне присвоєння; one max; орієнтування; квадратичне присвоєння; тест-функції; задача комівояжера; імовірнісний комівояжер; задача маршрутизації.
4. Оптимізація на основі моделювання: проблема зовнішньої оцінки; проблема оптимізації параметрів Matlab; проблема оптимізації параметрів Scilab.
5. Розширювані проблеми: програмована проблема; проблема, визначена користувачем.
6. **Встановлення фреймворку.**

Остання стабільна версія HeuristicLab 3.3.14 була випущена 24 липня 2016.

HeuristicLab розроблена на основі Microsoft .NET Framework. Для того, щоб програма працювала належним чином, потрібно завантажити і встановити пакет програмного забезпечення Microsoft .NET Framework 4.5. HeuristicLab не містить налаштувань для встановлення, і її не потрібно інстальовувати. Після розпакування файлів з архіву, який потрібно скачати з офіційного сайту, потрібно всього лише знайти в папці файл HeuristicLab3.3.exe та запустити його.

1. **Початок роботи з фреймворком HeuristicLab.**

Після запуску програми з’являється заставка, доки завантажуються всі плагіни. Після цього відкривається стартове вікно програми (рис.2). Стандартна конфігурація, яку можна завантажити на цій сторінці, поставляється в комплекті з двома додатками:

* HeuristicLab Optimizer: оптимізатор є основним додатком HeuristicLab Framework, надаючи різноманітні метаевристичні алгоритми, стандартні реалізації проблем, зручний випробувальний стенд і модулі для розробки і візуалізації алгоритмів.
* HeuristicLab Plugin Manager: архітектура HeuristicLab заснована на під’єднуваних плагінах. Всі алгоритми і задачі оптимізації реалізуються як плагіни і взаємодіють один з одним за допомогою чітко визначених і розширюваних інтерфейсів. Менеджер плагінів дозволяє переглядати, встановлювати та оновлювати наявні плагіни або керувати своїми власними плагінами.

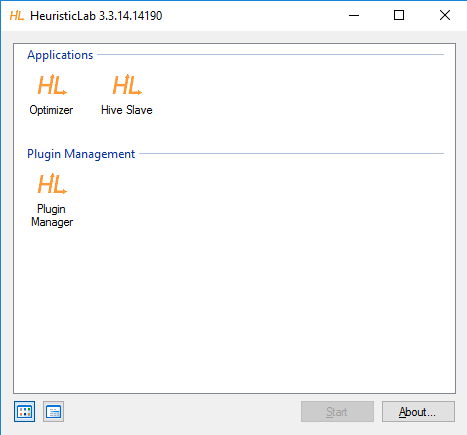


Рис.2. Стартове вікно програми.

Для початку роботи із HeuristicLab Optimizer, потрібно слідувати наступним крокам:

1. Відкрити алгоритм: натиснути New Item в панелі інструментів та обрати потрібний алгоритм.
2. Відкрити проблему в алгоритмі: у вкладці Problem натиснути New Proplem.
3. Задати параметри: завантажити відповідний екземпляр проблеми; HeuristicLab включає в себе багато добре відомих еталонних прикладів, які можна легко завантажити. Задати параметри алгоритму вкладці Parameters.
4. Запуск алгоритму: Натиснути Start/Resume Algorithm щоб виконати алгоритм. Зачекати, доки алгоритм завершиться, або натиснути Pause Algoritm, щоб перервати його виконання, або натиснути Stop Algorithm щоб зупинити виконання.
5. Перевірка результатів: перевірити результати роботи алгоритму у вкладці Results. Натиснути Start/Resume щоб продовжити виконання алгоритму або Reset, щоб підготувати алгоритм до нового запуску.

Для зручності використання, тут наявний ряд попередньо сконфігурованих прикладів на стартовій сторінці оптимізатора оптимізатора. Двічі клацнувши їх можна відкривати і експериментувати з ними (рис.3).

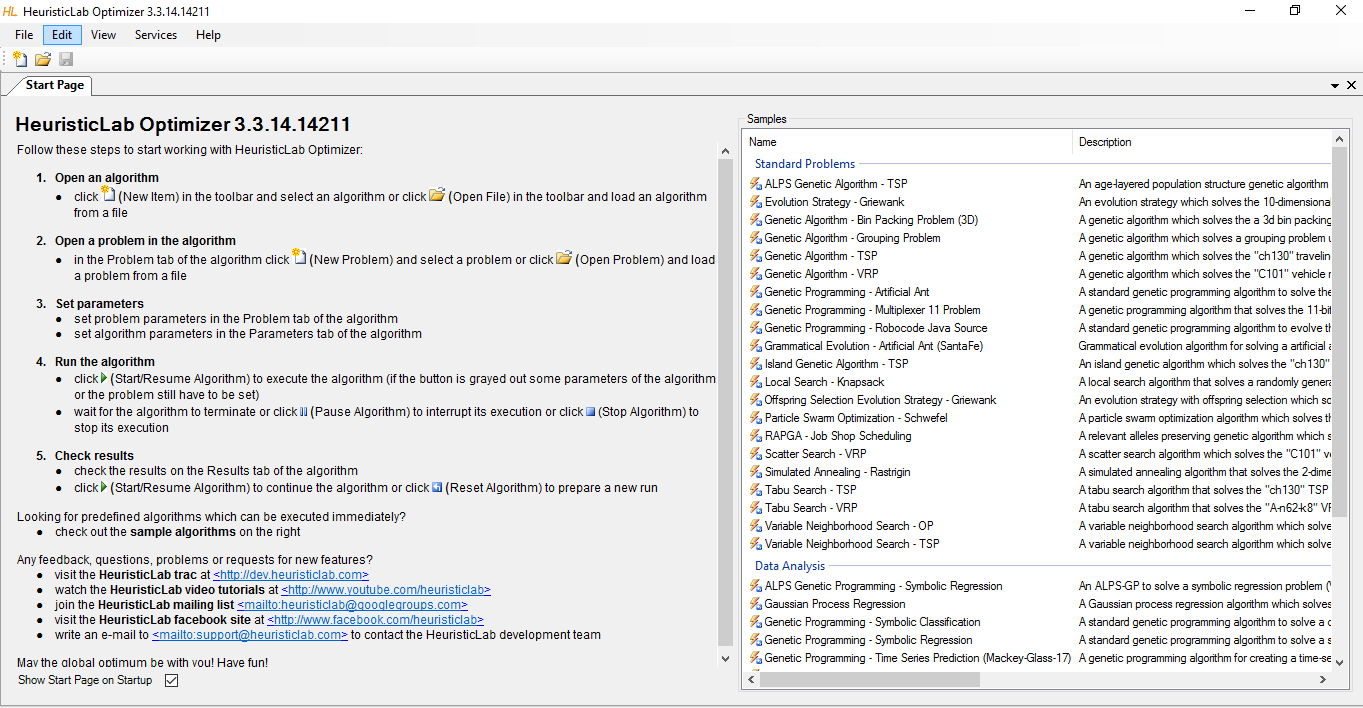


Рис.3. Головна сторінка оптимізатора.

1. **Приклади роботи з фреймворком.**

Приклад №1

Для демонстрації роботи програми, першим прикладом я обрав задачу комівояжера, з використанням генетичного алгоритму (Genetic Algorithm - TSP). У вкладці Algorithm задаємо необхідні параметри для виконання: тип схрещування, кількість поколінь, імовірність мутації, розмір популяції тощо (рис.5). Аналогічно задати параметри для задачі комівояжера (рис.4). У вкладці Results будуть відображатися результати виконання алгоритму. Також можна переглянути візуалізацію алгоритму під час його виконання та наявні графіки результатів роботи алгоритму (рис.6, рис.7). Результат виконання:

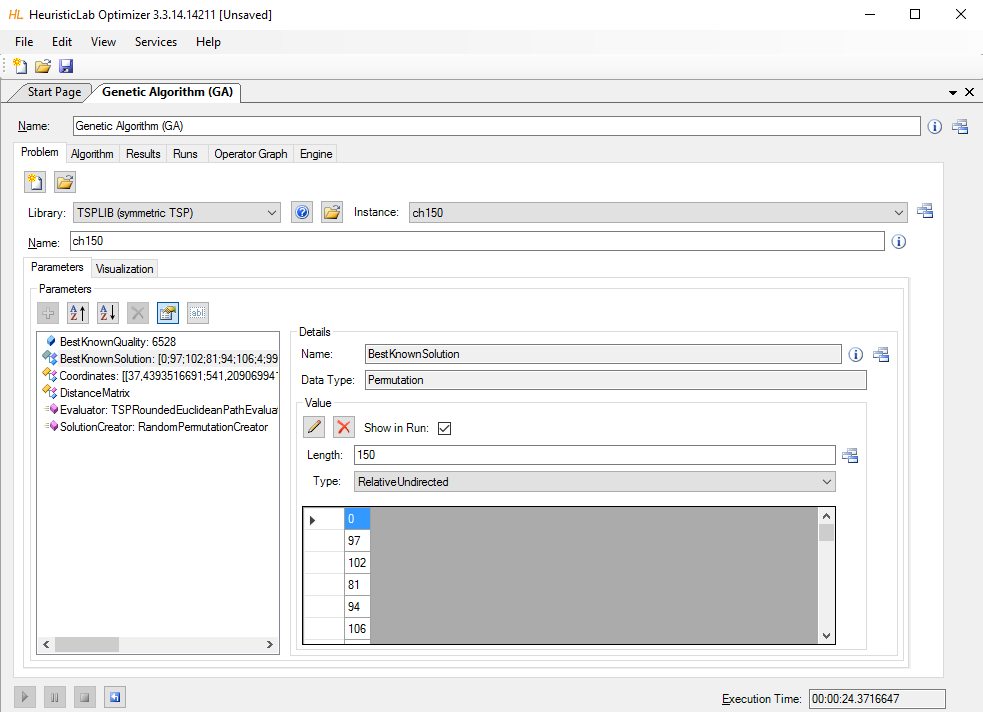


Рис.4. Задання параметрів для задачі комівояжера.

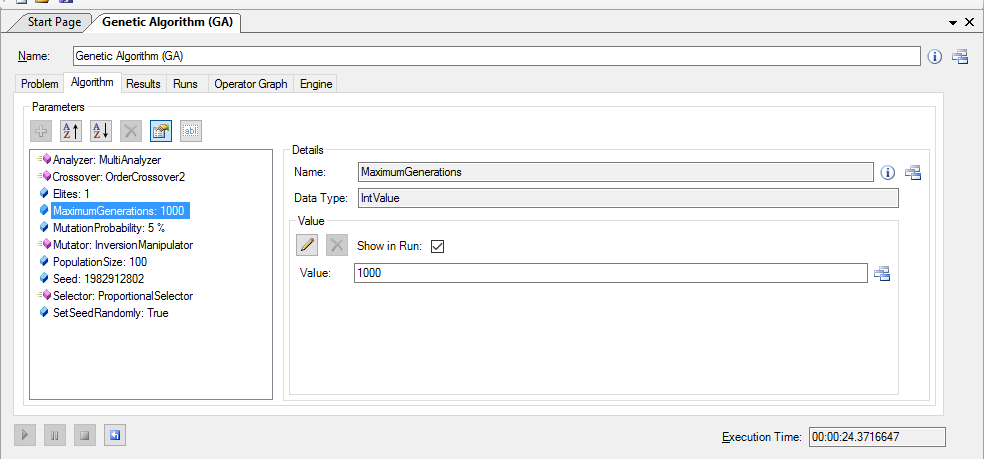


Рис.5. Задання параметрів для виконання алгоритму.

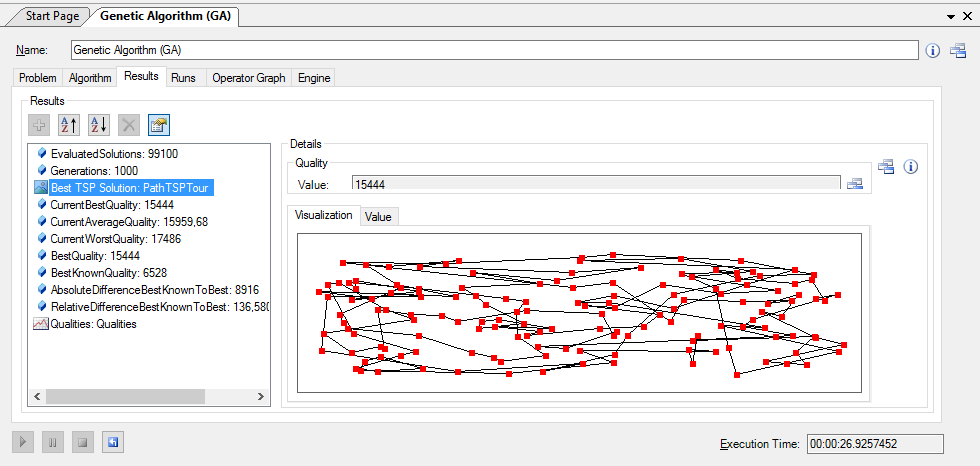


Рис.6. Візуалізація результату виконання алгоритму.

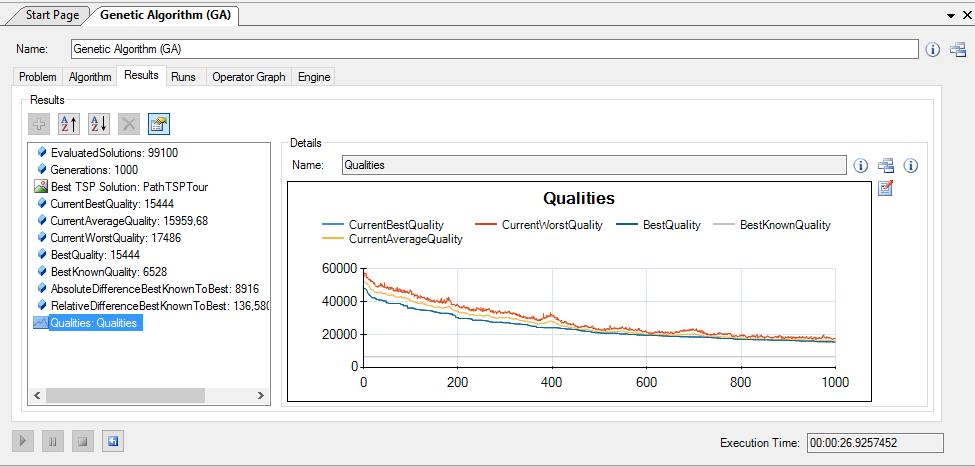


Рис.7. Графік результату роботи алгоритму.

Приклад №2

Наступний приклад – алгоритм локального пошуку, задача про ранець (Local search - Knapsack). Вона полягає в тому, щоб покласти якомога більшу кількість цінних речей у ранець при умові, що місткість рюкзака обмежена. Аналогічно задаю параметри для алгоритму та для самої задачі.

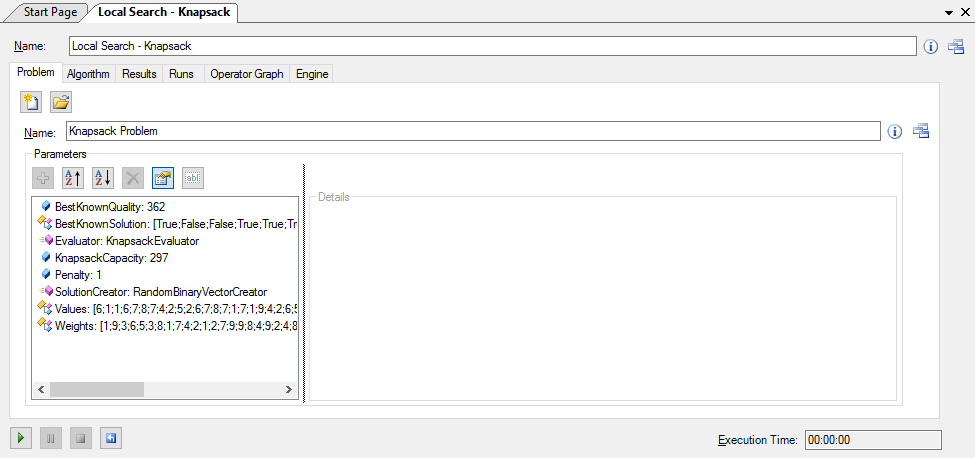
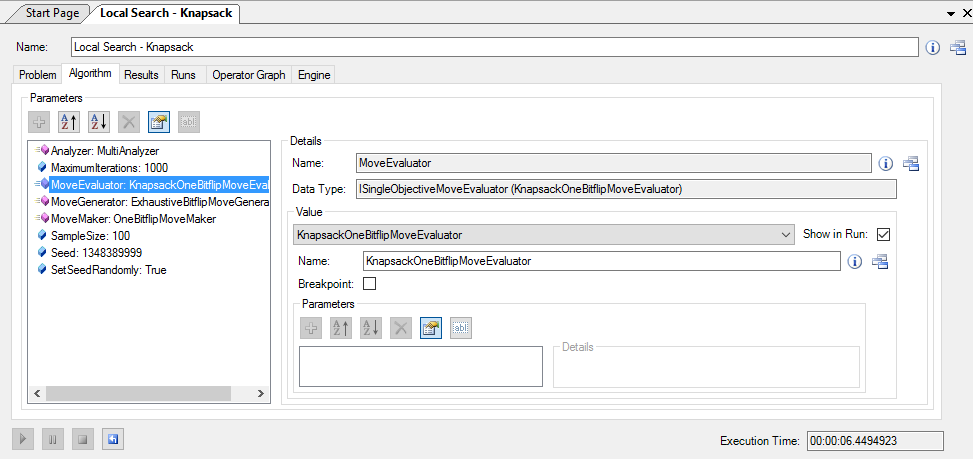
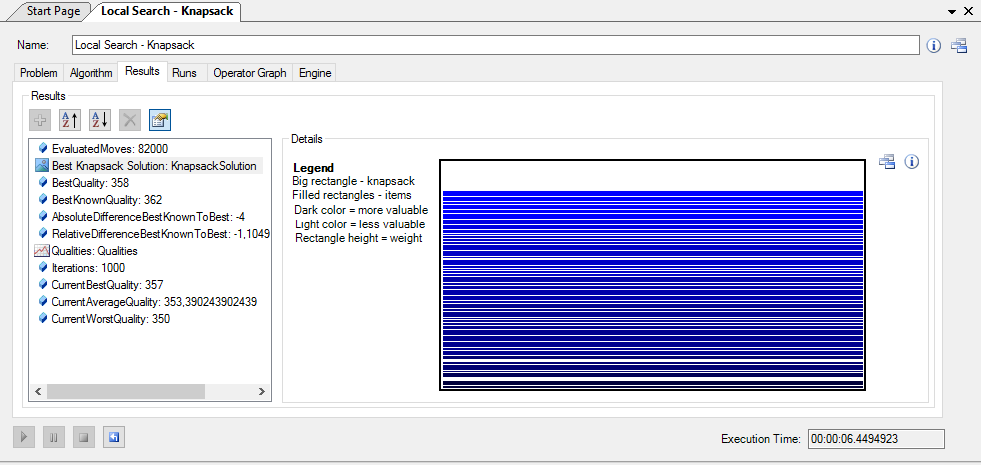
Результат виконання алгоритму відображається у вкладці Results. Тут можна переглянути результат виконання алгоритму у вигляді діаграми та графіка:

Рис.8. Задання параметрів для задачі.

 Рис.9. Задання параметрів для алгоритму.



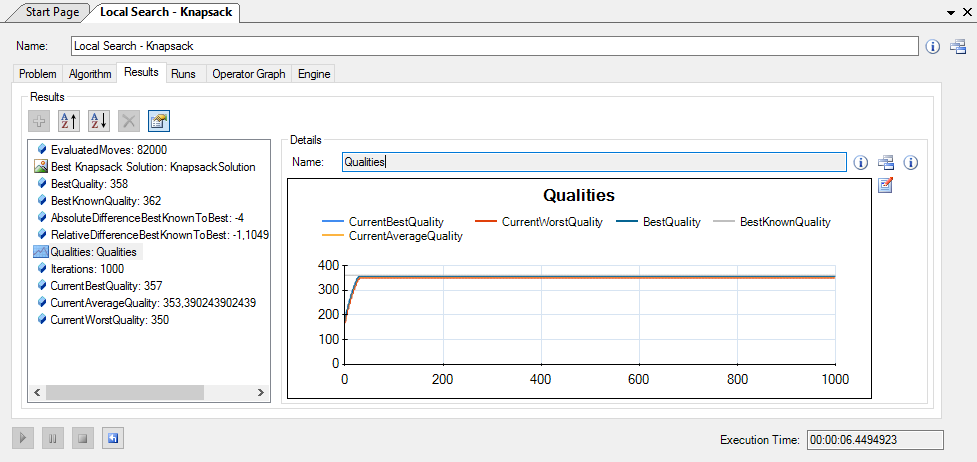
Рис.10. Візуалізація результатів у вигляді діаграми.

Рис.11.Візуалізація результатів у вигляді діграми.

Приклад №3

Даний приклад ілюструє стандартний алгоритм генетичного програмування для розв’язування задачі штучної мурахи (Genetic Programming – Artificial Ant). Для алгоритму, у вкладці Algorithm задаються параметри генетичного алгоритму: тип схрещування, максимальна кількість ітерацій, імовірність мутації, розмір популяції, тип відбору тощо. У вкладці Problem задаються налаштування для вирішення задачі: оператор, що використовується для оцінки рішень, кількість кроків часу, котрі штучна мураха має для збору всіх продуктів харчування, оператор для створення рішення, світ для штучної мурахи із розкиданими продуктами харчування тощо. Вкладка Result відображає результат виконання алгоритму.

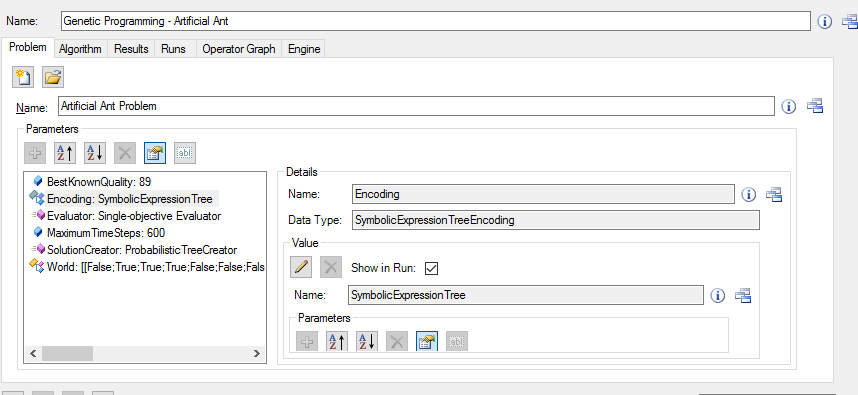


Рис.12. Задання налаштувань для вирішення задачі.

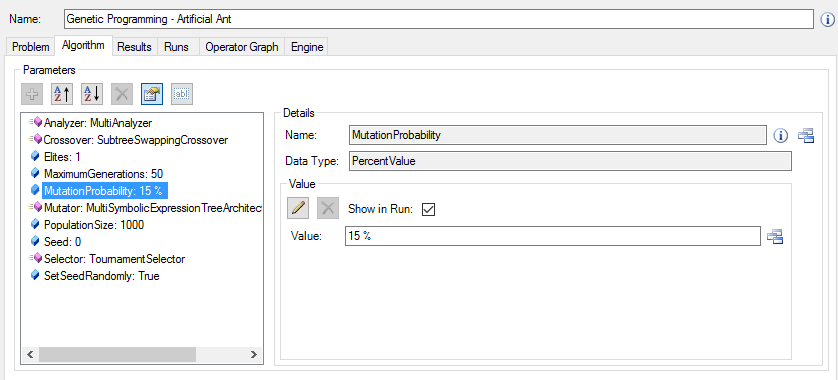


Рис.13. Задання налаштувань для виконання алгоритму.

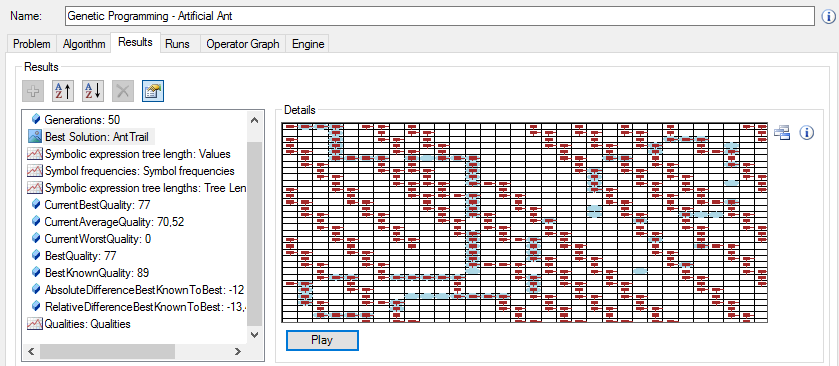


Рис.14. Відображення найкращого рішення у вигляді мурашиних слідів.

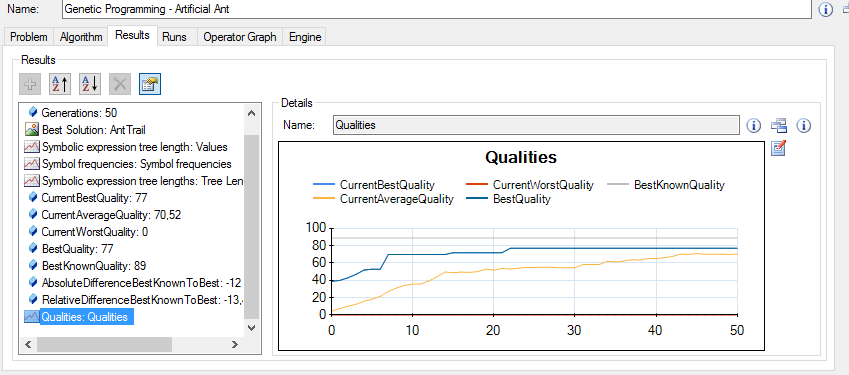


Рис.15. Графік з поточним найкращим та найгіршим результатом роботи.

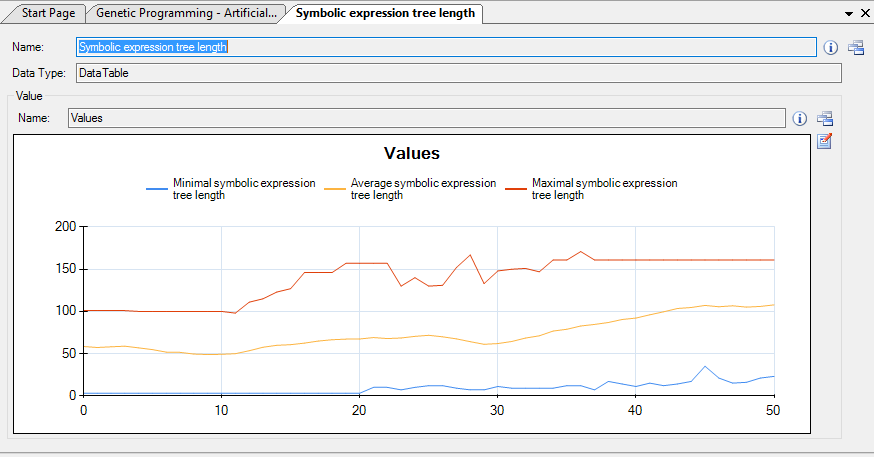


Рис.16. Символьне представлення довжини дерева алгоритму.

Приклад №4

Даний приклад демонструє еволюційну стратегію, яка вирішує десятивимірну тесову функцію Грівонка (Evolution Strategy - Griewank). У налаштуваннях алгоритму, вкладка Algorithm, вказується оператор, що використовується для аналізу; розмір популяції нащадків; максимальна кількість ітерацій; оператор мутації; кількість батьків на одного нащадка; розмір популяції; оператор схрещування; стратегічний параметр тощо. Вкладка Problem містить наступні налаштування: верхня та нижня границі в кожному вимірі; оператор для оцінки рішення; максимізація чи мінімізація функції; розмірність функції тощо. Результат виконання прикладу:

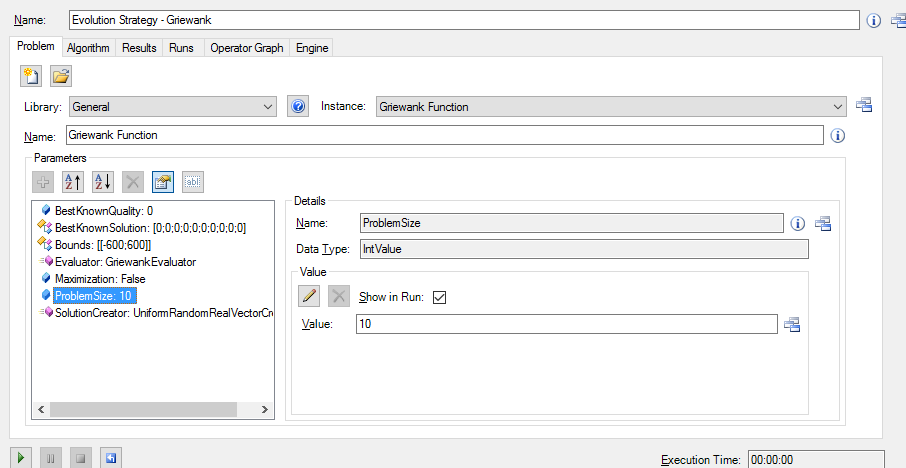


Рис.17. Налаштування параметрів для вирішення задачі.

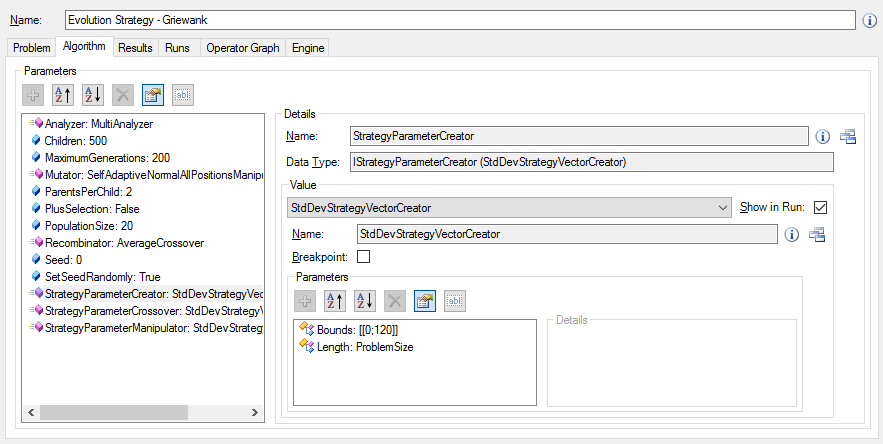


Рис.18. Налаштування параметрів алгоритму.

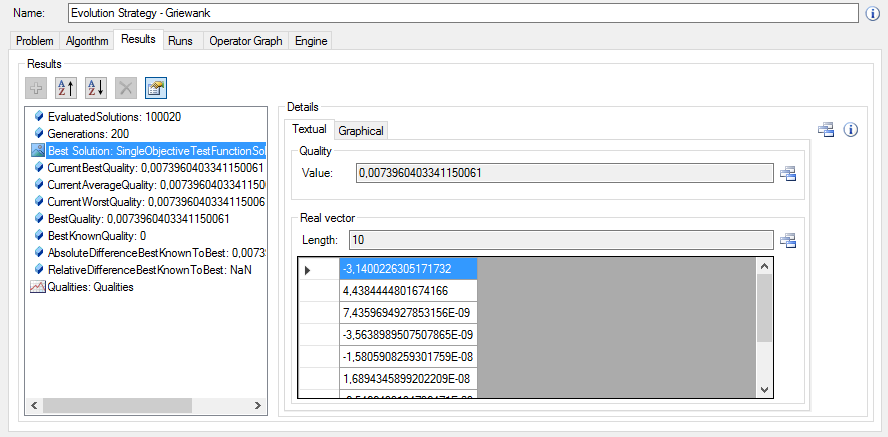


Рис.19. Найкраще рішення для тестової функції.

Приклад №5

Наступний приклад – алгоритм розсіяного пошуку (Scatter search - VRP), для вирішення проблеми маршрутизації транспортних засобів. У вкладці Problems задаємо необхідні параметри для вирішення задачі маршрутизації: обираємо оператор, який сформує рішення; обираємо екземпляр проблеми VRP; вказуємо задачу максимізації або мінімізації. Вкладка Algorithm відповідає за налаштування роботи алгоритму, задаємо: аналізатор; тип схрещування; оператор для покращення рішення; к-сть ітерацій; розмір популяції, оператор для обчислення подібності між рішеннями та інші налаштування. Нижче наведений результат виконання алгоритму.

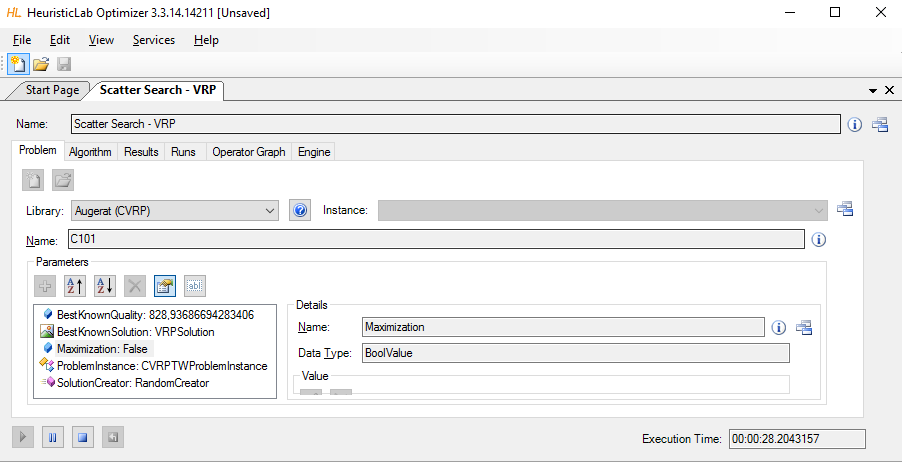


Рис.20. Налаштування задачі маршрутизації.

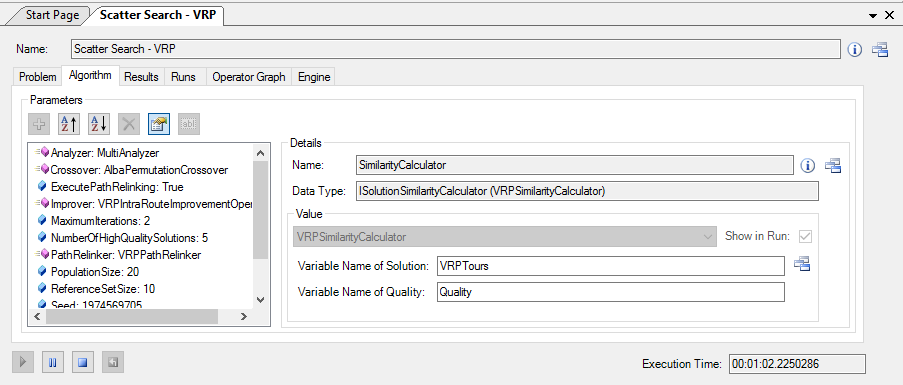


Рис.21. Налаштування алгоритму розсіяного пошуку.

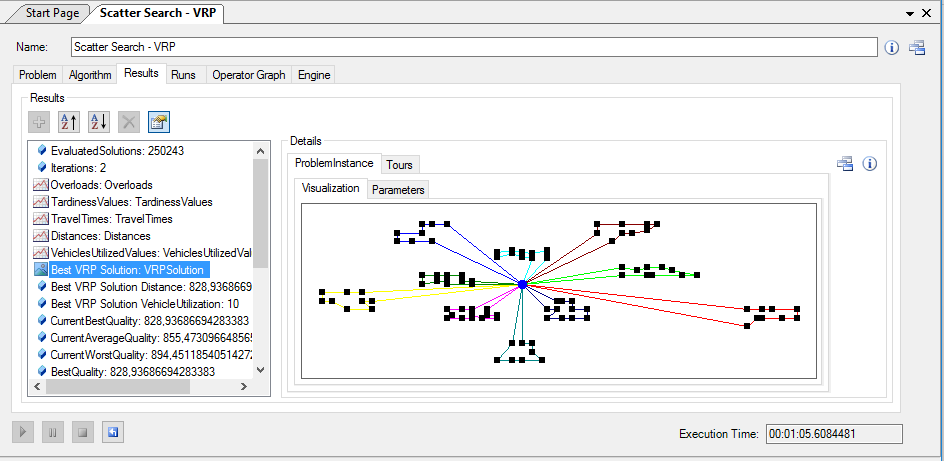


Рис.22. Візуалізація результатів виконання алгоритму.

**Висновок**

В дані розрахунковій роботі здійснено огляд фреймворку для евристичних і еволюційних алгоритмів HeuristicLab. Проведено огляд програмного середовища, описано його основні функції та можливості. Проведений огляд наявних алгоритмів та проблем, які можна вирішувати з допомогою фреймворку. Дана програма є легкою у використанні і не потребує навичок програмування, її не потрібно встановлювати, достатньо лише розпакувати скачаний архів та запустити відповідний файл на виконання. Фреймворк містить велику кількість різноманітних алгоритмів та задач, також наявні приклади до них. Для демонстрації роботи програми здійснив запуск кількох прикладів.