Технічні науки

УДК 004.021

Квасняк Катерина Миколаївна

студентка

Харківського національного університету радіоелектроніки

Квасняк Екатерина Николаевна

студентка

Харьковского национального университета радиоэлектроники

Kvasniak Kateryna

Student of the

Kharkiv National University of Radio Electronics

Науковий керівник:

Афанасьєва Ірина Віталіївна

доцент кафедри ПІ

Харківський національний університет радіоелектроніки

ГЕНЕТИЧНИЙ АЛГОРИТМ ДЛЯ ЗДІЙСНЕННЯ ПОШУКУ НАЙКРАЩОГО ВАРІАНТУ ПЛАНУВАННЯ ЧАСУ ШЛЯХОМ ВАРІАЦІЇ ШУКАНИХ ПАРАМЕТРІВ У ВЕБ-СЕРВІСІ ДЛЯ ПЛАНУВАННЯ ОСОБИСТОГО ЧАСУ

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДЛЯ ОСУЩЕСТВЛЕНИЯ ПОИСКА ЛУЧШЕГО ВАРИАНТА ПЛАНИРОВАНИЯ ВРЕМЕНИ С ПОМОЩЬЮ ВАРИАЦИИ ИСКОМЫХ ПАРАМЕТРОВ В ВЕБ-СЕРВИСЕ ДЛЯ ПЛАНИРОВАНИЯ ЛИЧНОГО ВРЕМЕНИ

GENETIC ALGORITHM FOR SEARCHING THE BEST OPTION OF TIME PLANNING BY VARIATION OF WANTED PARAMETERS IN WEB SERVICE FOR PLANNING PERSONAL TIME

Анотація. *Досліджено еволюційний алгоритм для пошуку найкращого варіанту планування особистого часу шляхом комбінування і варіації шуканих параметрів.*

Ключові слова: *функція пристосованості, генетичний алгоритм, хромосоми, популяція, генотип, пошук.*

Аннотация. *Исследован эволюционный алгоритм поиска наилучшего варианта планирования личного времени путем комбинирования и вариации искомых параметров.*

Ключевые слова: *функция приспособленности, генетический метод, хромосомы, популяция, генотип, поиск.*

Summary. *An evolutionary algorithm for finding the best option for personal time planning by combining and varying the required parameters is studied.*

Key words: *fitness function, genetic algorithm, chromosomes, population, genotype, search.*

Генетичний алгоритм – це метод, що відображає природну еволюцію методів вирішення проблем. Він використовується для процедури пошуку, що заснований на механізмах природного відбору і спадкоємства.

Класичний генетичний алгоритм має наступні етапи:

1) ініціалізація параметрів (вибір популяції хромосом);
2) знаходження оцінки пристосованості;
3) вибір параметрів для застосування генетичних операторів;
4) формування нової популяції;
5) вибір найкращих параметрів.

Під час дослідження генетичного алгоритму, написання програмного коду та оцінки правильності його виконання було створена популяція (кінцева множина завдань) з фенотипами. Групою випадкових фенотипів

для даного дослідження була група завдань/планів з обмеженнями, такими як «Завдання №1 має бути виконано лише о десятій годині дня» або «Завдання №3 має бути виконано лише після завдання №2». Тобто, фенотипи у даному випадку – це точка простору пошуку (search points), параметри задачі. Цей набір значень відповідає генотипу, тобто структурі.

Після формування множини особин з параметрами задач оцінюється група випадкових фенотипів та можливих варіацій планів. При цьому для кожної варіації рахується оцінка придатності (fitness score). Найкращі фенотипи/варіації за допомогою оператора кросовера створюють нові варіації, тобто графіки потомства. Цей оператор застосовується для рекомбінації генетичного алгоритму.

Деякі фенотипи отримують випадкові мутації. Формується нова популяція (нова вибірка). Далі цей процес повторюється до тих пір, поки не отримується найкращий зразок популяції. Це все необхідно для знаходження найкращого варіанту планування особистого часу.

На рис. 1 зображена програмна реалізація для рекомбінації генетичного алгоритму для знаходження найкращого варіанту планування часу:

```
32
33  def crossover(p1, p2, r_cross):
34      # діти - копії батьків за замовченням
35      c1, c2 = p1.copy(), p2.copy()
36      # перевірка на рекомбінацію
37      if rand() < r_cross:
38          # вибір точки перетину, яка не знаходиться на кінці рядка
39          pt = randint(1, len(p1)-2)
40          # виконати кросовер
41          c1 = p1[:pt] + p2[pt:]
42          c2 = p2[:pt] + p1[pt:]
43      return [c1, c2]
44
```

Рис. 1. Оператор кросовера

Функція crossover() реалізує перехрещення, використовуючи випадкове число в діапазоні [0,1], щоб визначити, чи виконується кросовер, а потім вибирає дійсну точку розділення, якщо має бути виконано кросовер.

Наступним кроком є створення функції для виконання мутації. Ця процедура перевертає біти з низькою ймовірністю, що контролюється гіперпараметром «r_mut»:

```
66  # оператор мутації
67  def mutation(bitstring, r_mut):
68      for i in range(len(bitstring)):
69          # перевірка мутації
70          if rand() < r_mut:
71              # зміна біту
72              bitstring[i] = 1 - bitstring[i]
```

Рис. 2. Оператор мутації

Слід зауважити, що генетичний алгоритм – це метод оптимізації на основі випадкових принципів. Він намагається покращити поточні рішення, застосовуючи до них деякі випадкові зміни. Оскільки такі зміни є випадковими, вони не завжди можуть давати кращі рішення. З цієї причини бажано зберегти попередні найкращі рішення (батьків) у новій популяції:

```python
def genetic_algorithm(objective, n_bits, n_iter, n_pop, r_cross, r_mut):
    # початкова сукупність бітових рядків
    pop = [randint(0, 2, n_bits).tolist() for _ in range(n_pop)]
    # масиви для збереження найкращого рішення
    best, best_eval = 0, objective(pop[0])
    # перераховування поколінь
    for gen in range(n_iter):
        # оцінка всіх варіацій у популяції
        scores = [objective(c) for c in pop]
        # перевірка нового найкращого рішення
        for i in range(n_pop):
            if scores[i] < best_eval:
                best, best_eval = pop[i], scores[i]
                print(">%d, new best f(%s) = %.3f" % (gen,  pop[i], scores[i]))
        # вибір батьків
        selected = [selection(pop, scores) for _ in range(n_pop)]
        # створення нового покоління
        children = list()
        for i in range(0, n_pop, 2):
            p1, p2 = selected[i], selected[i+1]
            # кросовер та мутація
            for c in crossover(p1, p2, r_cross):
                mutation(c, r_mut)
                # збереження наступного покоління
                children.append(c)
        pop = children
    return [best, best_eval]
```

Рис. 3. Генетичний алгоритм

З рис. 3 слід зазначити, що оцінка всіх варіацій у популяції, тобто оцінювання пристосованості, полягає в розрахунку функції пристосованості для кожної хромосоми (для кожного варіанту) 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 невід'ємні значення і, крім того, що для вирішення оптимізаційної задачі потрібно максимізувати цю функцію. Якщо вихідна форма функції пристосованості не задовольняє цим умовам, то виконується відповідне перетворення (наприклад, завдання мінімізації функції можна легко звести до задачі максимізації).

Результати виконання генетичного алгоритму можуть відрізнятися з огляду на стохастичний характер алгоритму чи процедури оцінки або

відмінності в чисельній точності. З графіка, що зображено на рис. 4, можна побачити, що еволюція досить швидко позбавляється від найгіршої генетичної інформації (найгірших варіацій планування особистого часу). Вже через 10 поколінь (сукупностей вибірок) можна побачити, що якість згенерованих поколінь однозначно покращилася.

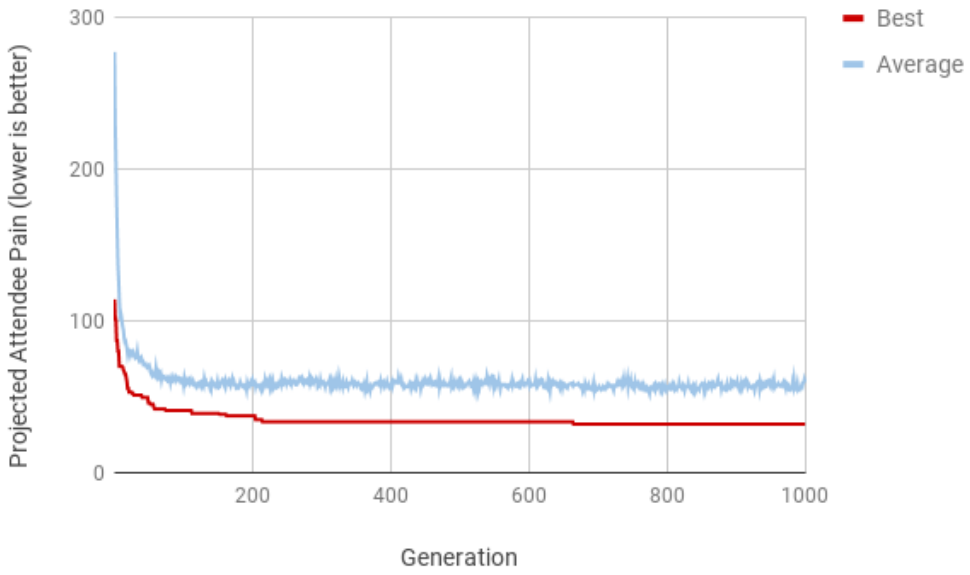

Рис. 4. Значення якості формування поколінь

З графіка помітно, що популяція нащадків характеризується набагато більш високим середнім значенням функції пристосованості, ніж популяція батьків. Слід звернути увагу, що в результаті виконання оператору кросовер завжди отримується хромосома з найбільшим значенням функції пристосованості, яким не володіють ні одна хромосома з батьківського популяції.

Отже, для знаходження найкращого варіанту планування особистого часу було досліджено генетичний алгоритм, що базується на пошуку та вибору «найкращої хромосоми» (найкращої варіації планування особистого часу). Під найкращим варіантом розуміється хромосома з найбільшим значенням функції пристосованості. Так як генетичний алгоритм успадкував властивості природного еволюційного процесу, селекція

(генерація нової вибірки/покоління) призводить до того, що з кожною наступною вибіркою значення функції пристосованості зростає. А це призводить до пошуку найкращого варіанту значення вибірки.

Література

1. Python Genetic Algorithm // basecs: [Веб-сайт]. URL: https://pygad.readthedocs.io/en/latest/ (дата звернення: 21.04.2022).
2. Janikow, C. Z., Michalewicz, Z., An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms. 1991.
3. Генетичний алгоритм // Wikipedia: [Веб-сайт] https://uk.wikipedia.org/wiki/Генетичний_алгоритм (дата звернення: 22.04.2022).