ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА ЕЛЕКТРОННА ІНЖЕНЕРІЯ
INFORMATION AND COMMUNICATION TECHNOLOGIES, ELECTRONIC ENGINEERING

№ 4 (1), 2024

ОЦІНКА ОБЧИСЛЮВАЛЬНОЇ СКЛАДНОСТІ ГЕНЕТИЧНОГО АЛГОРИТМУ

Я. Пиріг [ORCID: 0009-0001-2104-8439]

Національний університет «Львівська політехніка», вул. С. Бандери, 12, 79013, Львів, Україна

Відповідальний за рукопис: Ярослав Пиріг (e-mail: yaroslavpyrih@gmail.com)

(Подано 17 січня 2024)

Стаття присвячена оцінці обчислювальної складності генетичного алгоритму як одного із ключових засобів для розв’язання оптимізаційних задач. Розглянуто теоретичні аспекти обчислювальної складності алгоритмів та взаємозв'язок елементів генетичного алгоритму. Описано основні види обчислювальної складності алгоритмів: часову, просторову та асимптотичну. Наведено п’ять основних правил для розрахунку асимптотичної складності. Представлено математичний апарат для оцінки асимптотичної складності генетичного алгоритму, який враховує витрати на формування початкової популяції та на виконання еволюції. Еволюція відбувається через ітерації, під час яких покоління індивідів піддаються певним операціям з метою знаходження оптимального рішення (схрещування, мутації, декодування хромосоми тощо). ГА, в якості алгоритму глобального пошуку, розглядається для знаходження оптимального шляху без застрягання в локальних мінімумах. Для оцінки обчислювальної складності ГА розглянуто розв’язання задачі комівояжера (TSP) для 28 міст України з використанням модифікованої бібліотеки TSPLIB та платформи DEAP, створеної на мові програмування Python. Представлено блок-схему ГА, основними елементами якого є турнірний оператор відбору, оператор впорядкованого схрещування та інверсійний оператор мутації. Здійснено дослідження впливу розміру популяції та кількості поколінь на асимптотичну складність генетичного алгоритму при розв’язанні задачі TSP. Під час проведення дослідження розглядалась зміна розміру популяції ГА від 50 до 500 з кроком 50, при цьому для кожного такого значення моделювалось чотири набори кількості поколінь: від 50 до 200 із кроком 50. На основі отриманих результатів представлено лінійну залежність часу виконання ГА від розміру розглянутих вхідних даних. Показано, що найменша часова складність представленого ГА для наведеної задачі TSP становить 0.33848 секунди при розмірі популяції 50 та аналогічній кількості поколінь, тоді як найбільше значення – 3.752734 секунди при розмірі популяції 500 та кількості поколінь 200. Отримані результати можуть бути використані для оптимізації роботи ГА, зокрема в задачі TSP.

Ключові слова: *обчислювальна складність, генетичний алгоритм, задача комівояжера, популяція, покоління.*
УДК: 519.876

1. Вступ

Оптимізаційні задачі здебільшого потребують високоефективних та гнучких методів для пошуку оптимальних рішень в складних просторах параметрів. Генетичні алгоритми (ГА) стали невід'ємною частиною інструментарію оптимізації, завдяки їхній здатності ефективно долати проблеми глобального пошуку та оптимізації великих та складних просторів рішень [1-3].

Універсальність цього типу алгоритмів полягає в можливості їх використання для розв'язання різних оптимізаційних задач.

Зростання потужності обчислювальної техніки дає можливість розв'язувати складні задачі, впроваджуючи складніші алгоритми або істотно підвищуючи точність наявних. Розглядаючи загальний час роботи будь-якого алгоритму як сукупність двох складових - числа операцій і часу виконання однієї операції, можна передбачити середній час розв'язання задачі на ЕОМ заданого типу.

Однак слід зазначити, що, незважаючи на те, що час виконання кожної операції скорочується відповідно до закону Мура, різкого прискорення розв'язання алгоритмічних задач, як правило, не відбувається. Це пов'язано з ускладненням алгоритмічної складової через збільшення точності розрахунків, застосування більш якісних способів обробки тощо.

З позиції класичної теорії складності більшість завдань, що мають важливе практичне значення, є NP-важкі. Класична теорія складності аналізує і класифікує задачі та алгоритми за обсягом ресурсу (переважно часу), необхідного для досягнення необхідного результату, і тому оперує функціями обчислювальної складності [4].

Оцінка обчислювальної складності ГА є ключовим елементом для досягнення балансу між точністю вирішення задачі та ефективністю використання обчислювальних ресурсів, що в свою чергу дозволяє забезпечити оптимальні результати. У випадку задачі комівояжера, де велика кількість можливих комбінацій шляхів може призвести до великої обчислювальної складності, важливо розуміти, як ГА буде реагувати на збільшення розміру задачі та як ефективно використовувати обчислювальні ресурси для знаходження оптимального рішення [5].

Отже, функції обчислювальної складності в ГА допомагають визначити їхню продуктивність, оптимізувати параметри та визначати, в яких умовах вони можуть забезпечити найкращі результати. Ці функції стають ключовим елементом для поглибленого розуміння і вдосконалення ефективності ГА для широкого спектру застосувань. Таким чином, оцінка обчислювальної складності ГА є актуальним завданням.

2. Види обчислювальної складності алгоритмів

Для оцінки складності алгоритмів часто використовують поняття "клас", який визначають в залежності від часових ресурсів, необхідних для розв'язання задачі. При цьому найчастіше застосовують такі процедури: оцінка часових витрат для роботи даного алгоритму; порівняння двох або більше алгоритмів і віднесення їх або до одного, або до різних класів складності.

Класи складності алгоритмів можна розглядати як сукупність алгоритмів, що розв'язують співрозмірні обчислювальні задачі.

Такий клас зазвичай визначається трьома факторами: тип обчислювальної задачі, модель обчислення та обмежений обчислювальний ресурс. Наприклад, клас складності P (від англ. Polynomial) визначається як множина задач, розв'язання яких може бути отримано детермінованою машиною Тюрінга за поліноміальний час.

Поліноміальний час - це спосіб вимірювання часової складності алгоритмів. Алгоритм вважається розв'язаним за поліноміальний час, якщо час його виконання обмежується поліномом від розміру вхідних даних. Формально, алгоритм називається розв'язаним за поліноміальний час, якщо існує поліном $P(n)$, де n – такий розмір вхідних даних (ВД), при якому час виконання алгоритму не перевищує $P(n)$ для будь-якого значення n.

В теорії алгоритмічної складності класи складності визначаються вимогами до ресурсів алгоритму, а не вимогами до фізичних ресурсів. Основна обчислювальна модель - це машина Тюрінга, хоча можуть використовуватись і інші моделі. У такій машині замість використання стандартних інформаційних одиниць часу використовуються інформаційні одиниці – число певних кроків, потрібних для розв'язання задачі.

Основні види обчислювальної складності алгоритмів наведено на рисунку 1 [6]. Менш поширеною є усереднена складність, яка являє собою середній час, що витрачається на вхідні дані певного обсягу.

Часова складність зазвичай виражається як функція розміру ВД. Оскільки цю функцію, як правило, важко обчислити точно, а час виконання ВД малої розмірності зазвичай не має жодного значення, то зосереджуються на поведінці складності при збільшенні розміру ВД, тобто на асимптотичній поведінці.

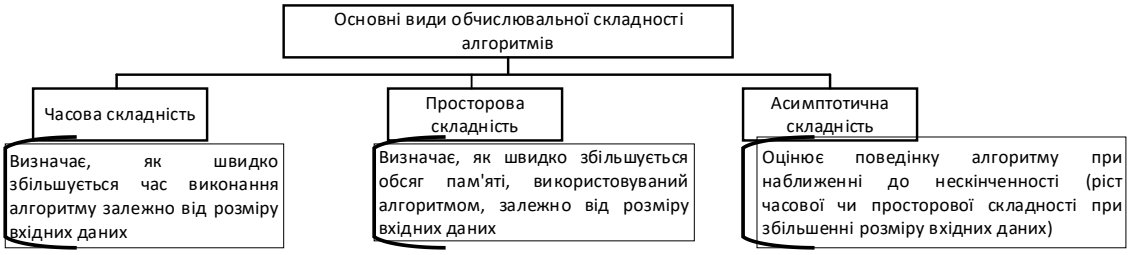

Рис. 1. Основні види обчислювальної складності алгоритмів

Асимптотична складність являє собою О-нотацію, позначається як O та застосовується для опису верхньої межі часової або просторової складності алгоритму. Вона дозволяє оцінити швидкість зростання часу виконання або використання ресурсів при зміні розмірності ВД.

Виділяють п’ять основних правил для розрахунку асимптотичної складності:

1. Якщо для алгоритмічної функції f потрібно виконати певні дії $f(N)$ раз, то для цього алгоритму потрібно виконати $O(f(N))$ кроків.

2. Якщо алгоритм виконує одну операцію, яка складається із $O(f(N))$ кроків, після чого виконує наступну операцію із $O(k(N))$ кроків, то загальна результативність f і k буде сумуватись: $O(f(N)+k(N))$.

3. Якщо потрібно здійснити $O(f(N)+k(N))$ кроків і область значень N функції $f(N)$ більша, ніж у $k(N)$, то обчислювальну складність можна спростити до $f(N)$.

4. Якщо алгоритму всередині кожного кроку $O(f(N))$ однієї операції потрібно виконати ще $O(k(N))$ кроків іншої операції, то загальна результативність буде $O(f(N)\times k(N))$.

5. Постійними множниками (константами) можна знехтувати.

3. Математичний апарат обчислювальної складності генетичного алгоритму

Здійснимо оцінку асимптотичної складності ГА, що вказує на те, як змінюється продуктивність алгоритму при збільшенні розміру популяції, розміру хромосоми та інших параметрів, а також при наближенні до великих розмірів вхідних даних. Саме асимптотична складність найчастіше використовується для визначення загальної тенденції алгоритму при збільшенні розміру вхідних даних, а не конкретних значень для певного розміру.

Обчислювальна складність для ГА визначається витратами на формування початкової популяції та витратами на виконання еволюції:

$$T_{sum} = P \cdot T_{sol} + \sum_{i=1}^{G} T_i, \tag{1}$$

де P - розмір популяції, T_{sol} - середній час побудови рішень, T_i - час виконання еволюції для i-го покоління, G - загальна кількість поколінь.

Еволюція ГА відбувається через ітерації, під час яких покоління індивідів піддаються певним операціям з метою знаходження оптимального рішення . Розрахунок T_i проводиться на основі часу,

витраченого на схрещування хромосом, часу виконання мутації, а також часу декодування хромосом, яке по суті полягає в формуванні рішень для розглянутої множини даних. Таким чином, T_i можна визначити як:

$$T_i = P \cdot p_{cross} \cdot 2T_{cross_i} + P(1 + p_{cross} \cdot 2) \cdot p_{mut} \cdot T_{mut_i} + P(1 + p_{cross} \cdot 2) \cdot T_{decod\_i}, \quad (2)$$

де p_{cross} – імовірність виконання схрещування для кожної особини (оскільки при цьому утворюється два нащадки, ймовірність у формулі теж подвоюється), p_{mut} – імовірність мутації особин у популяції (застосовується не тільки для вихідної популяції, а й для нових утворених особин), T_{cross_i} – час схрещування однієї пари хромосом, T_{mut_i} – час мутації однієї хромосоми, T_{decod_i} – час декодування хромосоми для розглянутої множини даних.

Часова складність оператора схрещування залежить від кількості вузлів обраних хромосом. Оператор мутації полягає в зміні значення деяких евристик (перемішування, вставки, заміни, тощо).

Обидві перелічені операції мають часову складність, нехтувано малу порівняно з часом побудови рішень, тому її можна розглядати як константу, а час виконання етапу (покоління) еволюції можна оцінити за часом декодування хромосоми:

$$T_i = O(P \cdot T_{decod\_i}). \quad (3)$$

Час декодування хромосоми залежить від часу обчислення відстаней між вузлами, побудови маршруту та при наявності – оптимізаційних технік (застосування попередньо обчислених відстаней, оптимізованих структур даних тощо). Час декодування T_{decod_i} зростає пропорційно зі збільшенням кількості вузлів, відповідно можна визначити як

$$T_{decod\_i} = O(T_{dist} \cdot N_{numb\_n\_r} + C), \quad (4)$$

де T_{dist} – час обчислення відстані між вузлами (для задачі комівояжера під вузлами розуміються міста), $N_{numb\_n\_r}$ – кількість вузлів у маршруті, C – константа, що представляє додаткові обчислення і операції, які можуть залежати від методу декодування, оптимізаційних технік тощо.

Декодування хромосоми в контексті задачі комівояжера означає перетворення генетичного коду (хромосоми) в фактичний маршрут, який проходить через всі міста без повторень. Хромосома складається з генів, які відповідають індексам міст в послідовності. Процес декодування полягає в перетворенні послідовності генів в послідовність міст, яка відповідає оптимальному маршруту комівояжера.

Декодування може включати розрахунок відстаней між містами та визначення послідовності міст, що дає найкоротший шлях. Чим більше міст у задачі комівояжера, тим більше часу може зайняти декодування хромосоми. Це пов'язано зі збільшенням кількості операцій порівняння та доступу до даних при відтворенні маршруту.

Варто зауважити, що (4) є спрощеною моделлю і не враховує всі можливі фактори, які впливають на час декодування. Додаткові параметри, такі як швидкодія алгоритмів обчислення відстаней та наявність оптимізаційних методів, також можуть бути включені в формулу для більш точного визначення часу декодування [7,8]. Для ГА час декодування хромосоми залежить від кількості міст у задачі комівояжера, відповідно можна оцінити цей час як лінійний.

Підсумовуючи формули (1-4) та дані, представлені у роботах [9-12], можна зробити висновок, що час виконання ГА має лінійну залежність від розміру вхідних даних .

4. Оцінка обчислювальної складності генетичного алгоритму для задачі комівояжера

Задача комівояжера (TSP) при навіть невеликій кількості міст може мати велику кількість можливих розв'язків, що робить її важкою для повного перебору. TSP має просторово-географічний характер, і ГА, в якості алгоритму глобального пошуку, може використовуватись для знаходження оптимального шляху без застрягання в локальних мінімумах. TSP має багато практичних застосувань, таких як оптимізація маршрутів доставки, розкладання задач обслуговування, планування маршрутів транспортних засобів та багато інших. Ці особливості роблять дану задачу цікавою для подальших досліджень .

Для оцінки обчислювальної складності ГА розглянуто розв’язання TSP для 28 міст України з використанням модифікованої бібліотеки TSPLIB та платформи DEAP, яка створена на мові програмування Python [13]. Було сформовано файл u28.tsp, в якому вказано координати розглянутих міст України. Даний файл використовується на етапі вхідних даних відповідно до блок-схеми алгоритму, представленої на рис. 2.

Також на етапі вхідних даних задаються такі початкові параметри ГА як імовірність схрещування та імовірність мутації, значення яких не змінюються в процесі досліджень.

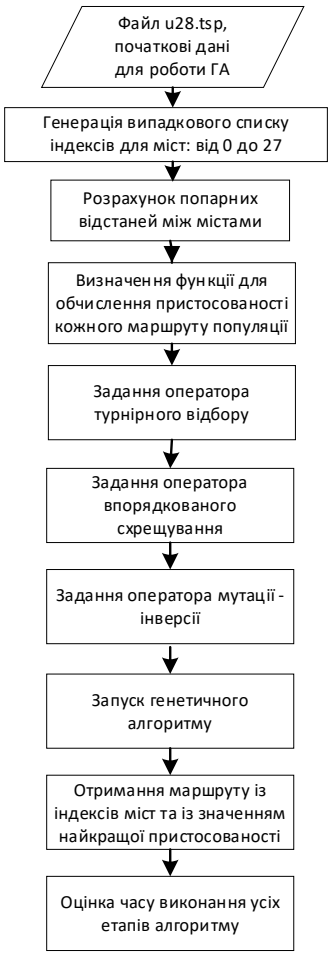

Рис. 2. Блок-схема ГА, використовуваного в процесі дослідження [13]

Наведена блок-схема демонструє оператори ГА, вибір яких обгрунтовано у [14-22], а саме оператор турнірного відбору, оператор впорядкованого схрещування та інверсний оператор мутації. Використання різноманітних операторів допомагає забезпечити баланс між експлуатацією та дослідженням простору параметрів, забезпечуючи швидке створення ефективних генетичних конфігурацій для розв'язання задач оптимізації.

У таблиці 1 представлено отримані результати.

Таблиця 1

Результати залежності часу виконання генетичного алгоритму від розміру популяції та кількості поколінь

Розмір популяції	К-сть поколінь	Час виконання алгоритму, с
50	50	0.33848
	100	0.435387
	150	0.535934
	200	0.632839
100	50	0.414877
	100	0.608575
	150	0.795506
	200	0.999659
150	50	0.50919
	100	0.774426
	150	1.024876
	200	1.308046
200	50	0.63162
	100	0.98206
	150	1.332859
	200	1.678956
250	50	0.686229
	100	1.198397
	150	1.570081
	200	1.971478
300	50	0.795662
	100	1.431955
	150	1.80948
	200	2.374368
350	50	0.86436
	100	1.441972
	150	2.102198
	200	2.880971
400	50	1.032986
	100	1.693255
	150	2.593768
	200	3.371781
450	50	1.065609
	100	1.855762
	150	2.834478
	200	3.399076
500	50	1.112597
	100	2.018632
	150	2.857986
	200	3.752734

Для більш наглядного представлення отриманих результатів їх проілюстровано на рисунку 3. Наведено лінії з різними графічними елементами, які відповідають наступному:

– ⬟ - 200 поколінь,
– ■ - 150 поколінь,
– ▲ - 100 поколінь,
– ● - 50 поколінь.

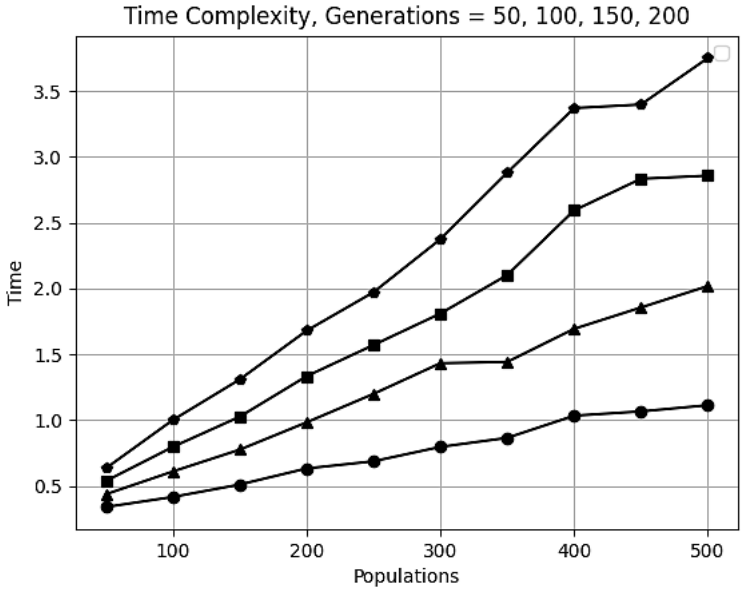

Рис. 3. Залежність часу виконання алгоритму від розміру популяції та кількості поколінь

На основі представлених результатів імітаційного моделювання, показано, що час виконання алгоритму зростає лінійно при збільшенні розміру популяції та кількості поколінь.

Найменша часова складність представленого ГА для наведеної задачі TSP становить 0.33848 секунди при розмірі популяції 50 та аналогічній кількості поколінь, тоді як найбільше значення – 3.752734 секунди при розмірі популяції 500 та кількості поколінь 200.

Висновки

У роботі розглянуто теоретичні аспекти обчислювальної складності алгоритмів. Проаналізовано взаємозв'язок елементів генетичного алгоритму, на основі чого представлено математичний апарат для оцінки його асимптотичної складності. Здійснено дослідження впливу розміру популяції та кількості поколінь на асимптотичну складність генетичного алгоритму при розв'язанні задачі комівояжера. На основі отриманих результатів показано лінійну залежність часу виконання ГА від розміру розглянутих вхідних даних.

Список використаних літературних джерел

[1] *Čoric, R., Dumic, M., & Jakobovic, D. (2017). "Complexity comparison of integer programming and genetic algorithms for resource constrained scheduling problems," 2017 40th Int. Convention on Information and Communication Technology, Electronics and Microelectronics, Croatia, pp. 1182-1188.*

[2] *Marappan, R., & Sethumadhavan, G. (2020). Complexity Analysis and Stochastic Convergence of Some Well-known Evolutionary Operators for Solving Graph Coloring Problem. Mathematics, 8(3):303. https://doi.org/10.3390/math8030303.*

[3] *Hafiiak A. (2018). Application of genetic programming tools as a means of solving optimization. Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава, Т. 6 (52). – С. 58-60.*

[4] *Коваль, В.С., & Струбицький, П.Р. (2017). Алгоритми і структури даних. – Навчальний посібник – Тернопіль: ФОП Шпак В. Б., 74 с.*

[5] *Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. // Omega. Volume 34, Issue 3, p. 209-219.*

[6] *Sabbah, T. (2020). "Enhanced Genetic Algorithm for Optimized Classification," 2020 International Conference on Promising Electronic Technologies (ICPET), Jerusalem, Palestine, pp. 161-166, doi: 10.1109/ICPET51420.2020.00039.*

[7] *Muh-Cherng, W., Chi-Shiuan, L., Chia-Hui, L., & Chen-Fu, C. (2017). Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems, Computers & Operations Research, Volume 80, pp. 101-112, https://doi.org/10.1016/j.cor.2016.11.021.*

[8] Ashraf, M., Gola, A., AlArjani, A., Hasan, F. (2022). Optimization of a Can Size Problem Using Real Encoded Chromosome in Genetic Algorithm. Journal of Physics: Conference Series, vol. 2198, https://dx.doi.org/10.1088/1742-6596/2198/1/012004.

[9] Lissovoi, A. & Oliveto, P.S. (2019) On the time and space complexity of genetic programming for evolving Boolean conjunctions. Journal of Artificial Intelligence Research, 66. pp. 655-689.

[10] Durrett, G., Neumann, F., & O'Reilly, U. M. (2011). "Computational complexity analysis of simple genetic programming on two problems modeling isolated program semantics". In Proceedings of the Foundations of Genetic Algorithms workshop (FOGA 2011), pp.69–80.

[11] Volovyk, A., Pyrih, Y., Urikova ,O., Masiuk, A., Shubyn, B., & Maksymyuk, T. (2024). Dynamic System State Estimation with a Resilience to Observation Data Anomalies. Contemporary Mathematics (Singapore), 5 (1).

[12] Oliveto, P. S., He, J., & Yao, X. (2017). Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing, 4 (3), 281–293.

[13] Pyrih, Y., Klymash, M., Kaidan, M., & Strykhalyuk, B. (2023). "Investigating the Efficiency of Tournament Selection Operator in Genetic Algorithm for Solving TSP", IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT-2023), 21-25 November, Lviv, Ukraine.

[14] Пиріг, Я., Климаш, М., Пиріг, Ю., & Лаврів, О.(2023). Генетичний алгоритм як засіб розв'язання оптимізаційних задач. Інфокомунікаційні технології та електронна інженерія, №3(2), С. 95-107. https://doi.org/10.23939/ictee2023.02.

[15] Maha Ata Al-Omeer & Zakir Hussain Ahmed (2019). "Comparative study of crossover operators for the mtsp", 2019 International Conference on Computer and Information Sciences (ICCIS), pp. 1–6, doi: 10.1109/ICCISci.2019.8716483.

[16] Bernardino, R., & Paias, A .(2018). Metaheuristics based on decision hierarchies for the traveling purchaser problem. International Transactions in Operational Research, 25(4):1269–1295, doi: 10.1111/itor.12330.

[17] Pavlenko, O., Tymoshenko, A., Tymoshenko, O., Luntovskyy, A., Pyrih, Y., & Melnyk, I. (2023). Searching Extreme Paths Based on Travelling Salesman's Problem for Wireless Emerging Networking. In: Klymash, M., Luntovskyy, A., Beshley, M., Melnyk, I., Schill, A. (eds) Emerging Networking in the Digital Transformation Age. Lecture Notes in Electrical Engineering, vol 965. Springer, Cham. https://doi.org/10.1007/978-3-031-24963-1_16

[18] Pyrih, Y., Masiuk, A., Pyrih, Y., & Urikova, O. (2024). Investigation of a Genetic Algorithm for Solving the Travelling Salesman Problem. In: Luntovskyy, A., Klymash, M., Melnyk, I., Beshley, M., Schill, A. (eds) Digital Ecosystems: Interconnecting Advanced Networks with AI Applications. Springer.

[19] Lakshmi, R., & Vivekanandan, K. (2013). "An analysis of recombination operator in genetic algorithms," 2013 Fifth International Conference on Advanced Computing (ICoAC), Chennai, India, pp. 223-226, doi: 10.1109/ICoAC.2013.6921954.

[20] Pravesjit, S., & Kantawong, K. (2017). "An improvement of genetic algorithm for optimization problem," 2017 International Conference on Digital Arts, Media and Technology (ICDAMT), Chiang Mai, Thailand, pp. 226-229, doi: 10.1109/ICDAMT.2017.7904966.

[21] Roeva, O., Shannon, A., & Pencheva, T.(2012) "Description of simple genetic algorithm modifications using Generalized Nets," 2012 6th IEEE International Conference Intelligent Systems, Sofia, Bulgaria, 2012, pp. 178-183, doi: 10.1109/IS.2012.6335212.

[22] Hildayanti, I.K., Soesanti, I. & Permanasari, A.E.(2018). "Performance Comparison of Genetic Algorithm Operator Combinations for optimization Problems," 2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, pp. 43-48, doi: 10.1109/ISRITI.2018.8864469.

COMPUTATIONAL COMPLEXITY EVALUATION OF A GENETIC ALGORITHM

Yaroslav Pyrih

Lviv Polytechnic National University, S. Bandery Str., 12, 79013, Lviv, Ukraine

The article is devoted to the estimation of computational complexity of a genetic algorithm as one of the key tools for solving optimisation problems. The theoretical aspects of computational complexity of algorithms and the interrelation of elements of a genetic algorithm are considered. The main types of computational complexity of algorithms are described: time, simple and asymptotic. Five basic rules for calculating the

asymptotic complexity are given. A mathematical apparatus for estimating the asymptotic complexity of a genetic algorithm is presented, which takes into account the costs of forming the initial population and performing evolution. Evolution occurs through iterations, during which generations of individuals are subjected to certain operations in order to find an optimal solution (crossing, mutation, chromosome decoding, etc.). GA, as a global search algorithm, is considered to find the optimal path without getting stuck in local minima. To assess the computational complexity of GA, we consider solving the traveling salesman problem (TSP) for 28 cities of Ukraine using a modified TSPLIB library and the DEAP platform created in the Python programming language. A block diagram of the GA is presented, the main elements of which are the tournament selection operator, the ordered crossover operator, and the inversion mutation operator. The influence of the population size and the number of generations on the asymptotic complexity of the genetic algorithm in solving the TSP problem is studied. The study considered changing the size of the GA population from 50 to 500 with a step of 50, while for each such value four sets of the number of generations were modelled: from 50 to 200 with a step of 50. Based on the obtained results, we show a linear dependence of the GA execution time on the size of the considered input data. It is shown that the smallest time complexity of the presented GA for the given TSP problem is 0.33848 seconds with a population size of 50 and a similar number of generations, while the largest value is 3.752734 seconds with a population size of 500 and a number of generations of 200. The obtained results can be used to optimise the performance of a GA in the TSP problem.

Keywords: *computational complexity, genetic algorithm, traveling salesman problem, population, generation.*