УДК 004.021
DOI https://doi.org/10.32782/2521-6643-2026-2-72.24

Тиш Є. В., кандидат технічних наук,
доцент кафедри комп'ютерних систем та мереж
Тернопільського національного технічного університету
імені Івана Пулюя
ORCID: 0000-0001-6420-7623

ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ МІНІМІЗАЦІЇ ДЕТЕРМІНОВАНИХ СКІНЧЕННИХ АВТОМАТІВ ЗА ОЦІНКОЮ ЕФЕКТИВНОСТІ РЕАЛІЗАЦІЇ

В статті досліджено задачу мінімізації детермінованих скінченних автоматів, яка відноситься до основних проблем теорії автоматів та формальних мов. Практична важливість задачі мінімізації має значення для синтаксичного аналізу, оптимізації цифрових схем, проєктування компіляторів та побудови пошукових, а також лексичних аналізаторів. Основну увагу статті приділено порівнянню ефективності реалізації найпоширеніших алгоритмів мінімізації автоматів: методу розмітки таблиці, алгоритму Хопкрофта та алгоритму Бржозовського.

Експериментальне дослідження виконано на кількох типах детермінованих автоматах, зокрема, випадкових автоматах, лексичних автоматах, автоматах із високою щільністю переходів, а також автоматах з великою надмірністю станів. Для кожного алгоритму визначено час виконання та обсяг використаної оперативної пам'яті залежно від кількості станів, розміру алфавіту та характеру переходів між станами. На основі експериментальних результатів доведено, що алгоритм Хопкрофта володіє найкращою або близькою до найкращої продуктивністю у більшості досліджених випадків та характеризується високою стабільністю й хорошою масштабованістю при зростанні розміру автоматів. Показано значну варіативність ефективності алгоритму Бржозовського, його переваги для автоматів з великою кількістю станів, однак суттєве погіршення показників для автоматів із щільними переходами. Експериментально встановлено, що метод розмітки таблиці характеризується найгіршими показниками масштабування, тому він є доцільним переважно для автоматів невеликого розміру. Отримані результати узгоджуються з теоретичними оцінками складності розглянутих алгоритмів та можуть бути використані для обґрунтованого вибору методу мінімізації детермінованих скінченних автоматів у практичних застосуваннях.

Ключові слова: *детерміновані скінченні автомати, мінімізація, еквівалентні стани, метод розмітки таблиці, алгоритм Хопкрофта, алгоритм Бржозовського, час виконання алгоритму, обчислювальна складність.*

Tysh Ie. V. Comparative analysis of deterministic finite automata minimization algorithms with respect to implementation efficiency

The problem of minimizing deterministic finite automata, which belongs to the fundamental problems of automata theory and formal languages, is investigated. The practical importance of the minimization problem is associated with syntactic analysis, digital circuit optimization, compiler design, and the construction of search and lexical analyzers. Particular attention is paid to the comparison of the efficiency of implementations of the most common automata minimization algorithms, namely the table-filling method, Hopcroft's algorithm, and Brzozowski's algorithm.

An experimental study is carried out on several types of deterministic automata, including random automata, lexical automata, automata with high transition density, and automata with a large number of redundant states. For each algorithm, execution time and memory consumption are determined depending on the number of states, the alphabet size, and the nature of transitions between states. Based on the experimental results, it is demonstrated that Hopcroft's algorithm exhibits the best or near-best performance in most of the considered cases and is characterized by high stability and good scalability as the size of automata increases. The analysis of Brzozowski's algorithm reveals significant variability in its efficiency, as well as its advantages for automata with a large number of states, alongside a substantial degradation in performance for automata with dense transitions. It is experimentally established that the table-filling method demonstrates the worst scalability and is therefore mainly suitable for automata of small size. The obtained results are consistent with theoretical complexity estimates of the considered algorithms and can be used to justify the selection of a minimization method for deterministic finite automata in practical applications.

Key words: *deterministic finite automata, minimization, equivalent states, table-filling method, Hopcroft's algorithm, Brzozowski's algorithm, algorithm execution time, computational complexity.*

Постановка проблеми. Детерміновані скінченні автомати (ДСА) є базовими моделями обчислень, які застосовуються у різних галузях інформатики. Їх використовують у компіляторах, цифрових системах, мережевих протоколах, апаратному проєктуванні та інших програмно-апаратних системах [1–4]. Однією з центральних задач теорії скінченних автоматів є їх мінімізація – процес побудови еквівалентного автомата

з мінімальною кількістю станів. Зменшення кількості станів дозволяє знизити витрати пам'яті, прискорити роботу систем та спростити подальший аналіз структури автоматів.

Попри добре розроблену теорію, різні алгоритми мінімізації демонструють різні показники ефективності (часову складність – час виконання алгоритму та просторову складність – використання пам'яті) залежно від структури автомата, розміру алфавіту, щільності переходів та особливостей реалізації. Тому питання порівняльного дослідження продуктивності різних методів мінімізації залишається актуальним.

Аналіз останніх досліджень і публікацій. Класичними підходами вирішення задачі мінімізації ДСА є описані в [5–7]: метод розмітки таблиці, що був запропонований Майхіллом та Нероде; алгоритм Хопкрофта, відомий як один із найефективніших методів мінімізації кількості станів; а також алгоритм Бржозовського, в основі якого лежить подвійне реверсування та детермінізація.

Сучасні роботи по мінімізації ДСА зосереджуються на оптимізованих структурах даних, паралельних реалізаціях [8], а також застосуванні алгоритмів на спеціалізованих апаратних архітектурах [9], алгоритмах подвійного обертання та розділеної мінімізації [10] тощо. Особлива увага приділяється аналізу ефективності методів на великих автоматах [11], що виникають, наприклад, у задачах аналізу потоків даних або формальної верифікації.

Метою дослідження є проведення комплексного теоретичного та експериментального порівняння найпоширеніших алгоритмів мінімізації ДСА. Основний внесок роботи полягає в дослідженні методу розмітки таблиці, алгоритму Хопкрофта та алгоритму Бржозовського для різних структур вхідних даних та оцінки їх часової й просторової ефективності.

Виклад основного матеріалу. Детермінований скінченний автомат – це математична модель обчислень, яка може перебувати в одному зі скінченної кількості станів [4, 5]. Для кожного стану та кожного вхідного символу в детермінованому автоматі існує лише один чітко визначений перехід в наступний стан. ДСА описується п'ятіркою

$$A = (Q, \Sigma, \delta, q_0, F),$$

де Q – скінченна множина станів, Σ – вхідний алфавіт (скінченна множина символів), $\delta : Q \times \Sigma \to Q$ – функція переходів (зазначає, що для кожної пари (стан, вхідний символ) є тільки один наступний стан), $q_0 \in Q$ – початковий стан (один визначений початковий стан, з якого починається робота), $F \subseteq Q$ – множина кінцевих (приймальних) станів.

Два автомати є еквівалентні, якщо вони приймають однакову мову, тобто для будь-якого рядка з алфавіту обидва автомати дають однаковий результат (приймають або відхиляють рядок). Мінімальний ДСА – це автомат з мінімальною кількістю станів, еквівалентний початковому.

Розглянемо три класичні алгоритми мінімізації.

Метод розмітки таблиці застосовується для мінімізації ДСА шляхом виявлення та об'єднання еквівалентних станів. Алгоритм ґрунтується на побудові таблиці попарних станів та послідовному маркуванні нееквівалентних пар. Основними кроками алгоритму є:

Крок 1. Побудова таблиці всіх пар станів (p, q), де $p, q \in Q, p \neq q$.

Крок 2. Пари станів (p, q) маркуються як нееквівалентні, якщо $(p \in F \wedge q \notin F) \vee (p \notin F \wedge q \in F)$.

Крок 3. Для кожної немаркованої пари (p, q) та кожного символу $a \in \Sigma$ проводиться перевірка переходів $(\delta(p, a), \delta(q, a))$. Якщо відповідна пара вже маркована, то пара (p, q) також маркується.

Крок 4. Процес повторюється до досягнення фіксованої точки, коли маркування не з'являються. Немарковані пари вважаються еквівалентними станами.

Крок 5. Побудова мінімізованого автомата шляхом об'єднання еквівалентних станів.

Метод розмітки таблиці має часову складність, що рівна $O(|Q|^2 \cdot |\Sigma|)$, та просторову складність $O(|Q|^2)$. До переваг цього методу мінімізації слід віднести наочність та простоту реалізації, а до недоліків – неефективність для автоматів з великою кількістю станів [5, 6].

Метод Хопкрофта є найбільш ефективним методом мінімізації, що базується на ітеративному розбитті множини станів на еквівалентні класи [5]. Алгоритм містить наступні кроки:

Крок 1. Множина станів Q розбивається на два блоки: приймальні стани F та неприймальні стани $Q \setminus F$. Отримане розбиття є початковим наближенням класів еквівалентності.

Крок 2. Формується робоча множина блоків W, яка містить блоки початкового розбиття. Для підвищення ефективності алгоритму до W може бути включений лише менший із двох блоків.

Крок 3. Вибирається активний блок. Тобто з робочої множини W обирається лише один блок $A \subseteq Q$, який використовується для подальшого уточнення розбиття.

Крок 4. Уточняється розбиття за вхідними символами, а саме для кожного символу $a \in \Sigma$ визначається множина станів

$$X = \{q \in Q | \ \delta(q, a) \in A\}.$$

Кожен блок Y поточного розбиття перевіряється на можливість поділу. Якщо виконуються умови $X \cap Y \neq \varnothing$ та $X \setminus Y \neq \varnothing$, блок Y розбивається на два нові блоки $X \cap Y$ та $X \setminus Y$.

Крок 5. У разі розбиття блоку Y здійснюється оновлення робочої множини W. Якщо блок Y належав до W, до робочої множини додаються обидва нові блоки; інакше до W додається менший з них.

Крок 6. Кроки 3–5 повторюються поки робоча множина W стає порожньою, що свідчить про досягнення стабільного розбиття множини станів.

Крок 7. Фінальне розбиття визначає класи еквівалентності станів. Кожному класу відповідає окремий стан мінімізованого автомата, а функція переходів формується відповідно до переходів між класами в початковому автоматі.

Крок 8. Отримання мінімального автомату, що є еквівалентним до початкового ДСА.

Метод Хопкрофта має часову складність, що рівна $O(|\Sigma| \cdot |Q| \cdot \log|Q|)$, та просторову складність $O(|\Sigma| \cdot |Q|)$. Перевага даного алгоритму полягає у високій швидкодії на реальних автоматах, а недоліком є складніша реалізація порівняно з методом розмітки таблиці [7].

Алгоритм Бржозовського використовує операції реверсування та детермінізацію для побудови мінімального ДСА та ґрунтується на теорії регулярних мов [6]. Мінімальний автомат отримується шляхом:

$$ДСА_{\min} = Det(Rev(Det(Rev(A)))),$$

що відповідає проходженню наступних кроків:

Крок 1. Реверсування автомата, а саме інвертуються напрямки всіх переходів, при цьому початковий стан стає приймальним, приймальні стани стають початковими.

Крок 2. Детермінізація (побудова за методом підмножин), отриманий недетермінований автомат перетворюється на ДСА.

Крок 3. Повторне реверсування отриманого на попередньому кроці автомата.

Крок 4. Повторна детермінізація для отримання мінімального детермінованого автомата.

Часова та просторова складність методу складає $O(2^{|Q|})$. До переваг такого методу мінімізації слід віднести концептуальну простоту, що є корисно для теоретичного аналізу, а до недоліків – метод є мало ефективний для великих автоматів.

Методологія експериментального дослідження. Експериментальне дослідження по оцінці ефективності наведених вище методів було виконано в однаковому програмно-апаратному середовищі з метою забезпечення коректності порівняння результатів: мова реалізації – Python 3.11, структури даних – списки переходів, множини та словники. Усі алгоритми мінімізації реалізовані з використанням однакового представлення ДСА, що усуває вплив структурних відмінностей реалізації на результати вимірювань. Аналіз трьох методів мінімізації було проведено на чотирьох сформованих наборах тестових ДСА, що представлені в табл. 1.

Методика вимірювання часу виконання мінімізації ДСА кожним алгоритмом полягала у використанні функції time.perf_counter() стандартної Python бібліотеки time. Кожен експеримент повторювався десять разів, після чого обчислювалося середнє значення часу виконання, відповідні результати подано на рис. 1–4.

Як видно з рис. 1, для випадкових детермінованих автоматів (набір 1) алгоритм Хопкрофта демонструє найменший час виконання мінімізації. Зі збільшенням кількості станів спостерігається зростання часу, близьке до лінійного в дослідженому діапазоні. Метод розмітки таблиці характеризується квадратичним характером зростання, що призводить до суттєвого збільшення часу при 500 станах (113 мс). Алгоритм Бржозовського показав проміжні значення між алгоритмом Хопкрофта та методом розмітки таблиці.

Таблиця 1

Досліджувані детерміновані скінченні автомати

Набір	Тип автоматів	Кількість автоматів	Діапазон станів	Алфавіт	Характеристика
1	Випадкові ДСА	50	100–500	3 символи	Середня щільність переходів
2	Автомати з реальних мов програмування	10	200–700	8 символів	Типові для лексичних аналізаторів
3	Автомати з високою щільністю переходів	20	100–300	10 символів	Максимально повні переходи
4	Автомати з великою надмірністю	20	200–1000	4 символи	До 60 % еквівалентних станів

Для лексичних аналізаторів (набір 2) алгоритм Хопкрофта має найменший час виконання мінімізації (рис. 2). Однак розрив між ним та алгоритмом Бржозовського є меншим, ніж для випадкових автоматів. Це пояснюється структурними властивостями лексичних автоматів, зокрема наявністю великої кількості еквівалентних та недосяжних станів, що зменшує негативний вплив подвійного реверсування в алгоритмі Бржозовського. Метод розмітки таблиці для набору 2 характеризується різким зростанням часу із збільшенням кількості станів ДСА.

Для набору 3, автоматів із високою щільністю переходів (рис. 3), алгоритм Бржозовського демонструє різко лінійне зростання часу виконання: при 500 станах час становить 425 мс, що майже вдвічі перевищує відповідне значення для 300 станів. Алгоритм Хопкрофта залишається стабільним і показує лише помірне зростання часу (7.9 мс при 500 станах). Метод розмітки таблиці зберігає квадратичний характер зростання. Отримані результати свідчать, що алгоритм Бржозовського є чутливим до густини переходів та може бути непридатним для автоматів із великою кількістю переходів.

Для набору 4, автоматів з великою надмірністю (рис. 4), алгоритм Бржозовського демонструє найменший час виконання для контрольних точок 200 та 500 станів (1.9 мс та 4.3 мс відповідно), незначно випереджаючи алгоритм Хопкрофта. Це пояснюється тим, що під час реверсування швидко усувається значна частина недосяжних та еквівалентних станів. Як видно з рисунку 4 метод розмітки таблиці у цьому наборі є обчислювально неефективним.

Таким чином, експериментальні результати підтверджують, що ефективність алгоритмів мінімізації ДСА визначається не лише кількістю станів, але й структурними характеристиками автоматів. Алгоритм Хопкрофта демонструє стабільно найкращу або близьку до найкращої продуктивність у всіх наборах даних. Алгоритм Бржозовського характеризується високою варіативністю – від найкращих результатів для надмірних автоматів до суттєвої деградації для автоматів із високою щільністю переходів. Метод розмітки таблиці демонструє найгірше масштабування серед розглянутих підходів.

Методика дослідження використання оперативної пам'яті здійснювалося за допомогою стандартних засобів Python sys.getsizeof та tracemalloc для оцінки об'єму виділених об'єктів, що зберігають структури автоматів та допоміжні дані алгоритмів. Результати часу виконання подано на рис. 5–8.

Отримані експериментальні дані однозначно свідчать про квадратичну залежність обсягу використаної пам'яті методу розмітки таблиці від кількості станів автомата, що зумовлює його обмежену застосовність для задач мінімізації скінченних автоматів великої розмірності.

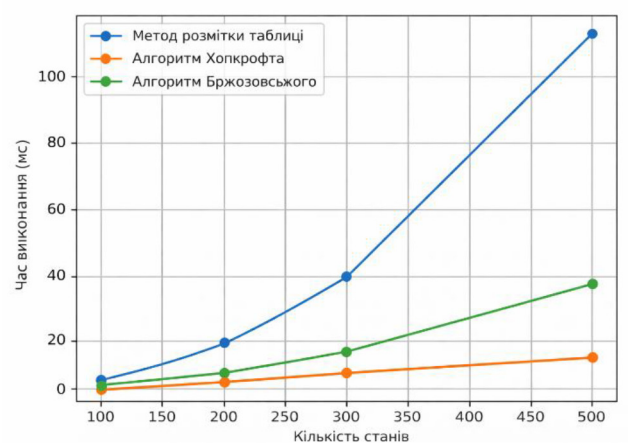

Рис. 1. Залежність часу виконання алгоритмів мінімізації від кількості станів для набору 1

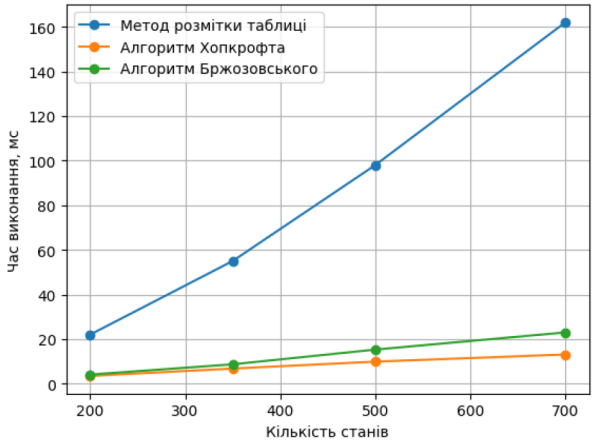

Рис. 2. Залежність часу виконання алгоритмів мінімізації від кількості станів для набору 2

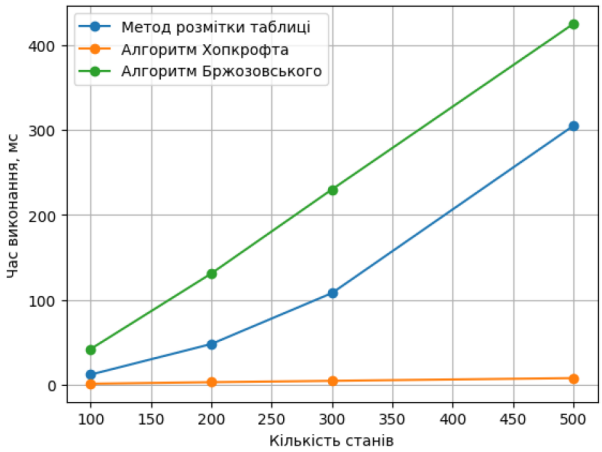

Рис. 3. Залежність часу виконання алгоритмів мінімізації від кількості станів для набору 3

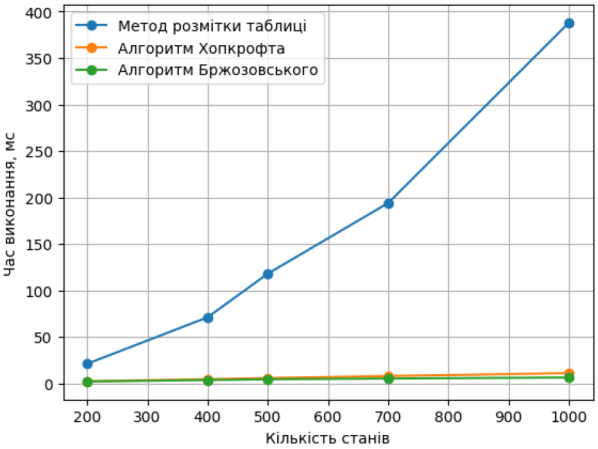

Рис. 4. Залежність часу виконання алгоритмів мінімізації від кількості станів для набору 4

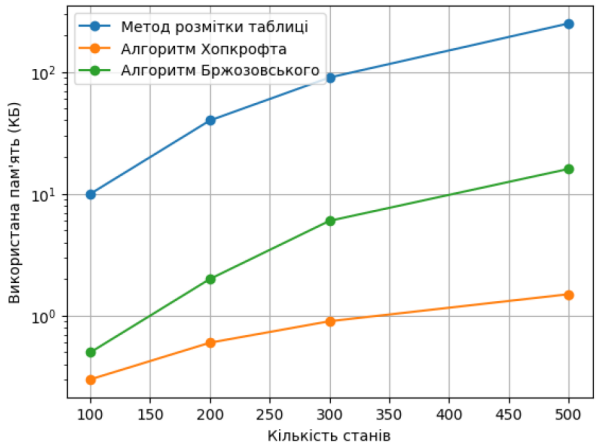

Рис. 5. Залежність використаної пам’яті від кількості станів для набору 1

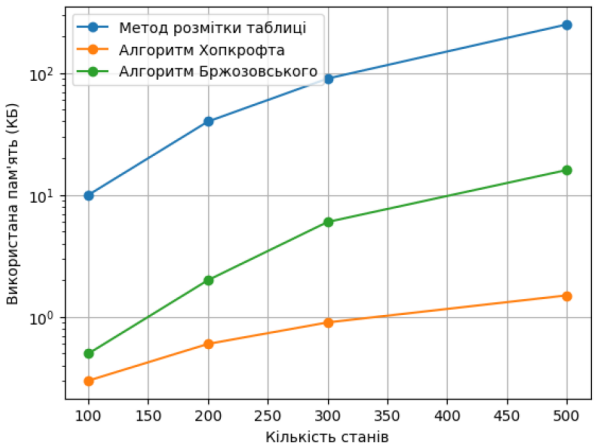

Рис. 6. Залежність використаної пам’яті від кількості станів для набору 2

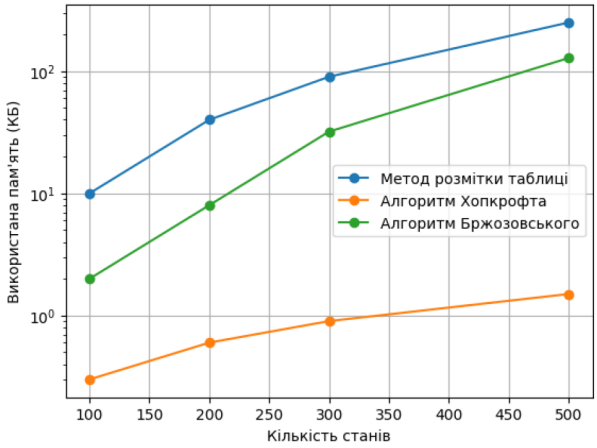

Рис. 7. Залежність використаної пам’яті від кількості станів для набору 3

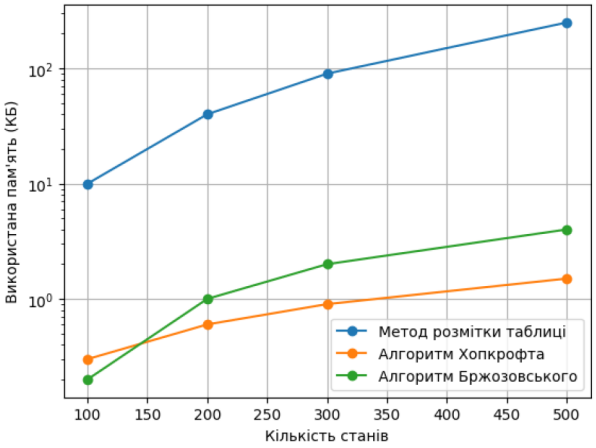

Рис. 8. Залежність використаної пам’яті від кількості станів для набору 4

Алгоритм Хопкрофта демонструє найменші витрати пам’яті серед розглянутих підходів та майже лінійне зростання споживання пам’яті при збільшенні числа станів, незалежно від типу вхідних наборів. Це підтверджує його високу масштабованість та стійкість до структурних особливостей автоматів.

Алгоритм Бржозовського характеризується суттєвою відмінністю використання пам’яті залежно від структури вхідних даних. Для автоматів із надмірною щільністю переходів спостерігається експоненціальне зростання використання пам’яті, а для випадкових, лексичних та надмірних автоматів зростання має помірніший характер.

Висновки. Проведене експериментальне дослідження показало, що алгоритм Хопкрофта є найефективнішим загальним методом мінімізації ДСА як за часом виконання процедури мінімізації, так й за використанням пам’яті. Алгоритм розмітки таблиці доцільно використовувати лише для мінімізації автоматів невеликого розміру. Алгоритм Бржозовського є ефективним у випадках, коли автомати мають надмірну кількість станів. Але застосування цього алгоритму має обмеження через можливе експоненційне зростання витрат ресурсів.

Таким чином, на основі отриманих результатів можна зробити висновок про відповідність між експериментальною поведінкою алгоритмів та їх теоретичною асимптотичною складністю. Це дозволяє вважати результати репрезентативними для практичного порівняння алгоритмів мінімізації ДСА.

Список використаних джерел:

1. Kumar S., Kumar J., Dubey S. S., Pathak V. N. *Theory of automata and its applications in science and engineering*. Deep Science Publishing, 2025. 350 p. URL: https://doi.org/10.70593/978-93-49910-92-8 (дата звернення: 14.02.2026)

2. Morazán M. T. *Programming based formal languages and automata theory: Design, implement, validate, and prove*. Springer, 2024. 420 p.

3. Aho A. V., Sethi R., Ullman J. D. *Compilers: Principles, techniques, and tools*. 2nd ed. Addison-Wesley, 2006. 1024 p.

4. Pin J. É. (Ed.) *Handbook of automata theory*. European Mathematical Society Publishing House, 2021. 600 p.

5. Hopcroft J. E., Motwani R., Ullman J. D. *Introduction to automata theory, languages, and computation*. 3rd ed. Pearson Education, 2007. 521 p.

6. Almeida M., Moreira N., Reis R. On the performance of automata minimization algorithms. In *Proceedings of the 4th Conference on Computation in Europe: Logic and Theory of Algorithms*, 2007. P. 3–14.

7. Hopcroft J. E. An n log n algorithm for minimizing states in a finite automaton. In *Theory of machines and computations*, 1971. P. 189–196. Elsevier.

8. Martens J. J. M., Wijs A. J. An evaluation of massively parallel algorithms for DFA minimization. *Electronic Proceedings in Theoretical Computer Science*, 409, 2024. P. 138–153.

9. Heemstra J., Martens J. J. M., Wijs A. J. Evaluating massively parallel algorithms for DFA minimisation, equivalence checking and inclusion checking. *arXiv*, 2025. URL: https://arxiv.org/pdf/2508.20735 (дата звернення: 14.02.2026).

10. Daci A., Lomont C. DFA minimization: Double reversal versus split minimization algorithms. *Theoretical Computer Science*, 583, 2015. P. 78–85. https://doi.org/10.1016/j.tcs.2015.04.002

11. Slavici V., Kunkle D., Cooperman G., Linton S. Finding the minimal DFA of very large finite state automata with an application to token passing networks. *arXiv*, 2011. URL: https://arxiv.org/pdf/1103.5736 (дата звернення: 16.02.2026)

References:

1. Kumar, S., Kumar, J., Dubey, S. S., & Pathak, V. N. (2025). *Theory of Automata and Its Applications in Science and Engineering*. Deep Science Publishing. https://doi.org/10.70593/978-93-49910-92-8

2. Morazán, M. T. (2024). *Programming based formal languages and automata theory: Design, implement, validate, and prove*. Springer.

3. Aho, A. V., Sethi, R., & Ullman, J. D. (2006). *Compilers: Principles, techniques, and tools* (2nd ed.). Addison-Wesley.

4. Pin, J. É. (Ed.). (2021). *Handbook of automata theory*. European Mathematical Society Publishing House.

5. Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2007). *Introduction to automata theory, languages, and computation* (3rd ed.). Pearson Education.

6. Almeida, M., Moreira, N., & Reis, R. (2007). On the performance of automata minimization algorithms. In *Proceedings of the 4th Conference on Computation in Europe: Logic and Theory of Algorithms* (pp. 3–14).

7. Hopcroft, J. E. (1971). An n log n algorithm for minimizing states in a finite automaton. In *Theory of machines and computations* (pp. 189–196). Elsevier.

8. Martens, J. J. M., & Wijs, A. J. (2024). An evaluation of massively parallel algorithms for DFA minimization. *Electronic Proceedings in Theoretical Computer Science, 409*, 138–153.

9. Heemstra, J., Martens, J. J. M., & Wijs, A. J. (2025). Evaluating massively parallel algorithms for DFA minimisation, equivalence checking and inclusion checking. *arXiv*. https://arxiv.org/pdf/2508.20735

10. Daci, A., & Lomont, C. (2015). DFA minimization: Double reversal versus split minimization algorithms. *Theoretical Computer Science, 583*, 78–85. https://doi.org/10.1016/j.tcs.2015.04.002

11. Slavici, V., Kunkle, D., Cooperman, G., & Linton, S. (2011). Finding the minimal DFA of very large finite state automata with an application to token passing networks. *arXiv*. https://arxiv.org/pdf/1103.5736

Дата першого надходження статті до видання: 09.03.2026
Дата прийняття статті до друку після рецензування: 26.03.2026
Дата публікації (оприлюднення) статті: 30.05.2026