

Lecture Notes in Networks and Systems 1156

Kohei Arai *Editor*

Proceedings of the Future Technologies Conference (FTC) 2024, Volume 3

 Springer

Series Editor

Janusz Kacprzyk , *Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland*

Advisory Editors

Fernando Gomide, *Department of Computer Engineering and Automation—DCA, School of Electrical and Computer Engineering—FEEC, University of Campinas—UNICAMP, São Paulo, Brazil*

Okyay Kaynak, *Department of Electrical and Electronic Engineering, Bogazici University, Istanbul, Türkiye*

Derong Liu, *Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, USA*

Institute of Automation, Chinese Academy of Sciences, Beijing, China

Witold Pedrycz, *Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada*

Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

Marios M. Polycarpou, *Department of Electrical and Computer Engineering, KIOS Research Center for Intelligent Systems and Networks, University of Cyprus, Nicosia, Cyprus*

Imre J. Rudas, *Óbuda University, Budapest, Hungary*

Jun Wang, *Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong*

The series “Lecture Notes in Networks and Systems” publishes the latest developments in Networks and Systems—quickly, informally and with high quality. Original research reported in proceedings and post-proceedings represents the core of LNNS.

Volumes published in LNNS embrace all aspects and subfields of, as well as new challenges in, Networks and Systems.

The series contains proceedings and edited volumes in systems and networks, spanning the areas of Cyber-Physical Systems, Autonomous Systems, Sensor Networks, Control Systems, Energy Systems, Automotive Systems, Biological Systems, Vehicular Networking and Connected Vehicles, Aerospace Systems, Automation, Manufacturing, Smart Grids, Nonlinear Systems, Power Systems, Robotics, Social Systems, Economic Systems and other. Of particular value to both the contributors and the readership are the short publication timeframe and the worldwide distribution and exposure which enable both a wide and rapid dissemination of research output.

The series covers the theory, applications, and perspectives on the state of the art and future developments relevant to systems and networks, decision making, control, complex processes and related areas, as embedded in the fields of interdisciplinary and applied sciences, engineering, computer science, physics, economics, social, and life sciences, as well as the paradigms and methodologies behind them.

Indexed by SCOPUS, EI Compendex, INSPEC, WTI Frankfurt eG, zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

For proposals from Asia please contact Aninda Bose (aninda.bose@springer.com).

Kohei Arai
Editor

Proceedings of the Future Technologies Conference (FTC) 2024, Volume 3

 Springer

Editor
Kohei Arai
Saga University
Saga, Japan

ISSN 2367-3370 ISSN 2367-3389 (electronic)
Lecture Notes in Networks and Systems
ISBN 978-3-031-73124-2 ISBN 978-3-031-73125-9 (eBook)
<https://doi.org/10.1007/978-3-031-73125-9>

© The Editor(s) (if applicable) and The Author(s), under exclusive license
to Springer Nature Switzerland AG 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Editor's Preface

Welcome to the ninth edition of the Future Technologies Conference (FTC 2024), held on the 14th and 15th of November 2024 in the vibrant and historic city of London, United Kingdom. We are delighted to host this premier event, which has become the world's foremost forum for reporting and discussing groundbreaking research in Artificial Intelligence, Computer Vision, Data Science, Computing, Ambient Intelligence, and related fields.

Since its inception, FTC has grown to be the pre-eminent global gathering for professors, academic researchers, Ph.D. and graduate students, research institutions, and industry developers. Each year, the conference brings together a diverse and dynamic community dedicated to advancing the frontiers of technology and innovation. This ninth edition of FTC promises to be an exceptional event, filled with insightful presentations, stimulating discussions, and unparalleled networking opportunities.

This year, we are pleased to report that FTC 2024 received an impressive total of 476 paper submissions from across the globe. After a rigorous and meticulous peer review process, 173 papers have been selected for publication. These papers represent the cutting edge of research and significant advancements in their respective fields. The high quality and diversity of the accepted papers are a testament to the vibrant and dynamic nature of the research community that FTC fosters.

We extend our deepest gratitude to all the authors who submitted their work to FTC 2024. Your contributions are the lifeblood of this conference and are crucial to its success. We also want to express our sincere appreciation to our reviewers, who dedicated their time and expertise to ensure the high quality of the conference proceedings. Your efforts have been invaluable in maintaining the rigorous standards of FTC. Special thanks go to our Technical Program Committee (TPC), whose commitment and hard work have been instrumental in shaping the conference program. We are also deeply grateful to the session chairs for their leadership and to the distinguished speakers for sharing their invaluable insights and knowledge with us.

FTC 2024 is not just about presenting research; it is about creating a platform for knowledge exchange, collaboration, and inspiration. We are confident that the presentations and discussions over these two days will spark new ideas, foster collaborations, and drive future innovations in the ever-evolving fields of technology.

Thank you for joining us at FTC 2024. We look forward to an inspiring and productive conference and hope that you find it as stimulating and rewarding as we have in organizing it.

Kind Regards,
Kohei Arai

Contents

Automatic Detection of Common Gastroenterological Diseases Using a Small Dataset: A Two-Phase Image Processing Method	1
<i>Rafael Neujahr Copstein, Vincenzo Abichequer Sangalli, Renan Magalhães Trévia, Leonardo Rosa Amado, Vinicius Chrisosthemos Teixeira, and Márcio Sarroglia Pinho</i>	
Times Square: A Time Series Dataset for Semi-supervised Crowd Counting ...	11
<i>Salma Alghamdi, Lama Al Khuzayem, and Ohoud Al-Zamzami</i>	
A Machine Learning-Based Household Robot	32
<i>Razan Al-Hamed, Rawan Al-Hamed, Aya Karam, Fatima Al-Qattan, Fatmah Al-Nnaimy, and Soraia Oueida</i>	
HAT: Homography-Based Alternate Training for Pose-Invariant Face Recognition	47
<i>Bowen Sun, Hoi-Sim Wong, and Shibao Zheng</i>	
Object Detection for Autonomous Vehicles in Urban Areas Using Deep Learning	60
<i>Muhammad Arslan, Muhammad Mubeen, and Syed Muhammad Usman</i>	
Utilizing Transfer Learning, Graph Matching, and Spatial Attention with CARLA Pre-trained Models	76
<i>Vasanth Iyer and Igor Ternovski</i>	
Towards Digital Zen: A Systematic Review of Emerging Digital Interventions for Mental Wellness	93
<i>Amir Reza Asadi and Annu Sible Prabhakar</i>	
A Neuromarketing Approach to Identify Consumer's Ties and Preferences Through Multivariate Data	115
<i>Verónica de Jesús Pérez Franco, Ana Lilia Coria Páez, Jesús Jaime Moreno Escobar, Oswaldo Morales Matamoras, Erika Yolanda Aguilar del Villar, and Mauro Daniel Castillo Pérez</i>	
Analyzing E-Commerce Dynamics: Customer Satisfaction, Revenue Prediction, and Sentiment Analysis in Retail	145
<i>Ashutosh Sagar, Ishan Makadia, Meet Sinojia, Zahra Sadeghi, and Stan Matwin</i>	

Fast and Scalable Multi-Kernel Encoder Classifier	161
<i>Cencheng Shen</i>	
Security Analysis in Ecuador: Advanced Integration of Geo-Positioning and Named Entity Recognition (G-NER) in X Platform Publications	178
<i>Henry N. Roa, Diego Quisi-Peralta, Jordan Murillo-Valarezo, Daniel Pulla-Sánchez, Ruth Reategui-Rojas, and Janneth Chicaiza</i>	
Lake Water Level Forecasting Using LSTM and GRU: A Deep Learning Approach	197
<i>Yuxin Du, Jing Fan, Ari Happonen, Dassan Paulraj, and Micheal Tuape</i>	
Optimization of Few-Shot Learning NER Models Through Grammatical Conditioning of Training Data	217
<i>Tobias Dorrn and Almuth Müller</i>	
Training-Testing Data Ratio Selection for Accurate Time Series Forecasting: A COVID-19 Case Study	227
<i>Wisam Bukaita, Guillermo Garcia de Celis, and Manaswi Gurram</i>	
Emotions and Customer Satisfaction in the Mobile Banking Era: An Empirical Analysis	247
<i>Yeferson Torres Berru, Santiago Jimenez, Lander Chicaiza, and Viviana Espinoza Loayza</i>	
A Comparative Study of Machine Learning Algorithms on Datasets of Varying Sizes	261
<i>Xiaoting Huang, Xuelian Xi, Siqi Wang, Zahra Sadeghi, Asif Samir, and Stan Matwin</i>	
City Recommender System: A Comparative Study of AI-Driven Approaches	275
<i>Dorsa Soleymani, Mahsa Mousavi Diva, Lovelyn Uzoma Ozougwu, Riasat Mahbub, Zahra Sadeghi, Asif Samir, and Stan Matwin</i>	
Global Warming's Influence on Temperature Increase	291
<i>Wisam Bukaita</i>	
An Approach for House Price Prediction Using Bayesian Regression	307
<i>Vinh Vo, Nhan Vuong, Linh Do, Hao Do, Thai Le, and Anh Le</i>	
Pricing Data Based on Value: A Systematic Literature Review	319
<i>Marija Radić, Philipp Herrmann, Theresa Stein, Nicolas Heirich, and Dubravko Radić</i>	

Using Mixed Exponentials for Unsupervised Discretization 340
Dulan S. Dias, Adele H. Marshall, and Aleksandar Novakovic

Strategic Selection of ITIL V4 Services for Cybersecurity Defense:
 An AHP and TOPSIS Approach for Moroccan Universities 361
*Abdelilah Chahid, Souad Ahriz, Kamal El Guemmat,
 and Khalifa Mansouri*

GenDatr: Visualizing Probabilistic Data Generation in Medical Data
 Science 373
*ZongPoh Loo, AngKang Tan, Clement JunKai Choo, Shaun JiaLe Tay,
 XueEr Lim, Jeremiah XianMing Loh, Andrew Ge-Hall,
 Peter ChunYu Yau, and Dennis Wong*

Systematic Literature Review and Bibliometric Analysis of Low-Resource
 Speech-to-Text Translation 379
D. Fortuné Kponou, Fréjus A. A. Laleye, and Eugène C. Ezin

Data Property Mitigator: Empowering Pneumonia Detection Using Deep
 Learning Models and Chest X-Rays 399
Hanan J. Farhat, Georges E. Sakr, Rima Kilany, and Ayman Al Osta

Speech Enhancement: Data Manipulation Techniques for Augmenting
 Existing Datasets 410
Yixuan Zhu and Olga Scrivner

A Deep Learning Framework for Classifying and Mitigating Bias in News
 Reporting 424
*Abhijit Mitra, Rivujit Das, Jayanta Paul, Rajdeep Majumder,
 Arghyadeep Saha, Nandita Sengupta, and Jaya Sil*

Improving Analytic Approximation of Log-Normal Interest Rate Model
 with Neural Network 444
Anna Knezevic

Exploring the Performance of Deep Learning Models for Neutrino
 Direction Prediction in High-Energy Astrophysics 452
Tawanda Blessing Chiyangwa and Sandile Thamie Mhlanga

Simulating Extreme Precipitation Phenomena Through Generative
 Adversarial Networks: Advancing Hydroclimatic Understanding 468
*Yiyang Luo, V. I. Lutsenko, S. M. Shulga, I. V. Lutsenko, O. V. Soboliak,
 and M. B. Shevelev*

Reinforcement Learning of Emerging Swarm Technologies: A Literature Review	478
<i>Afnan M. Alharbi, Ghaida Alshehri, and Salma Elhag</i>	
Advanced Traffic Safety Analysis: Leveraging Deep Learning and Large Language Models for Near-Crash Detection in Crowdsourced Videos	495
<i>Shadi Jaradat, Richi Nayak, and Mohammad Elhenawy</i>	
ChatGPT and ChatGPT API: An Experiment with Evaluating ChatGPT Answers	514
<i>Ahmad Rawashdeh, Omar Rawashdeh, and Mohammad Rawashdeh</i>	
Building Blocks to Empower Cognitive Internet with Hybrid Edge Cloud	534
<i>Fay Arjomandi, Siavash Alamouti, Michel Burger, and Bashar Altakrouri</i>	
Spectrum Serenade: OPNET Expedition Unveiling WLAN 802.11e Performance Evaluation	558
<i>Ali Mohd Ali, Hamza Abu Owida, and Ahmad Al-Qerem</i>	
Information Retrieval Systems: A Methodological Review	572
<i>Nathaniel Jay Maña, Johniel Babiera, Kriziah Lynn Bayloces, Xavier-Lewis Palmer, Lucas Potter, Rabby Lavilles, and Lemuel Clark Velasco</i>	
Augmented Reality Storybook for Color-Blind Children: Enhancing Reading Experience	592
<i>John Ivan C. Maurat, Elsie V. Isip, and Jensen A. Santillan</i>	
WIRE: Write Energy Reduction via Encoding in Phase Change Main Memories (PCM)	599
<i>Mahek Desai, Apoorva Rumale, Marjan Asadinia, and Sherrene Bogle</i>	
A Novel OO-Based Code Complexity Metric	616
<i>J. H. Aluthwaththage and H. A. N. N. Thathsarani</i>	
Tamil-Based Mobile Application for the Identification of Anthurium Plant Diseases	629
<i>Dilshan Indraraj De Silva and Selveraja Rasikadevi</i>	
TCAD Electrothermal Analysis of 3D GAAFET Structures for Future VLSI Circuits	643
<i>Konstantin O. Petrosyants, Denis S. Silkin, and Dmitriy A. Popov</i>	

Practical Quantum Combinatorial String Matching 653
*Domenico Cantone, Claudio Caudullo, Simone Faro,
 Francesco Pio Marino, Arianna Pavone, and Caterina Viola*

Generation of Student’s Programming Exercises Using SCT Generator 670
Damir Vusić, Danijel Radošević, and Andrija Bernik

The Design and Development of School Cooperative System 681
*Check-Yee Law, Yong-Wee Sek, Choo-Chuan Tay, Tze-Hui Liew,
 Leonard-Chi-Boon Yew, and Nur Sarah Najihah Binti Nor Alahyadi*

**Autoadaptive Lattices of Magnetic Vortices in Coherent Domains
 of Condensed Matter for “Intelligent” Quantum Computing and Storing** 693
Luigi Maxmilian Caligiuri

Author Index 713



Practical Quantum Combinatorial String Matching

Domenico Cantone¹, Claudio Caudullo¹, Simone Faro¹,
Francesco Pio Marino^{1,2}(✉), Arianna Pavone³, and Caterina Viola¹

¹ Dipartimento di Matematica e Informatica, Università di Catania,
viale A.Doria n.6, 95125 Catania, Italy
francesco.marino@phd.unict.it

² Univ Rouen Normandie, INSA Rouen Normandie, Université Le Havre Normandie,
Normandie Univ, LITIS UR 4108, CNRS NormaSTIC FR 3638, IRIB,
76000 Rouen, France

³ Dipartimento di Matematica e Informatica, Università di Palermo,
Via Archirafi n.34, 90123 Palermo, Italy

Abstract. This paper focuses on the practical implementation of quantum computation for various combinatorial problems in strings. We provide a detailed description of all the operators involved in solving the problems addressed in this paper, along with practical implementation details of our solution. The algorithms developed for quantum computation provide polylogarithmic solutions, showcasing a complexity improvement compared to classical solutions, with the best results achieving linear complexity.

Keywords: String algorithms · Quantum computing · Combinatorial algorithms

1 Introduction

Quantum computing, positioned at the convergence of computer science and quantum mechanics, proposes a groundbreaking methodology for data processing. Utilizing *quantum bits*, or *qubits*, which can simultaneously exist in multiple states and become entangled to execute joint operations, quantum computers operate on principles fundamentally different from those of classical computers. These unique characteristics enable faster and more efficient operations, particularly advantageous in domains requiring complex computations such as cryptography and optimization.

Significant advancements in quantum computing include algorithms like Shor's for integer factorization [21] and Grover's for unstructured search [11], both of which surpass their classical equivalents. These developments not only highlight the theoretical potential of quantum computing but have also spurred extensive research aimed at leveraging this potential across various computer science applications.

One particularly intriguing application of quantum computing is in text-processing, where the efficiency of string matching algorithms is crucial due to the vast amounts of textual data processed in fields such as data mining, bioinformatics, and text retrieval. Classical string matching solutions [8], like the Boyer-Moore-Horspool algorithm [3, 13], though efficient, still face limitations in speed and scalability. Quantum computing presents a pathway to surpass these limitations.

Quantum solutions to the exact string matching (ESM) problem, which involve verifying or locating a pattern within a text, have shown promise. Initially examined in the quantum query complexity model [17, 20], these approaches utilized Grover's algorithm [11] to achieve substantial reductions in query complexity and operational time. The pioneering work by Ramesh and Vinay [20] introduced a quantum solution to the ESM problem, significantly accelerating the verification process to a complexity of $\tilde{O}(\sqrt{n})$ queries, and identifying all occurrences within $\tilde{O}(\sqrt{\rho n})$ time where ρ is the density of pattern occurrences.

Recent advancements have extended these principles into the quantum circuit model [23], with researchers like Niroula and Nam [4, 9, 18] developing circuits that operate with similar time complexity. This approach is particularly relevant given the prominence of the circuit-based quantum model, supported by modern quantum programming environments such as IBM's Qiskit, Microsoft's Q#, and Google's Cirq.

Other quantum solutions to string problems have focused mainly on the edit distance [16]. Nonetheless, the question of whether a quantum algorithm can outperform classical algorithms for the edit distance remains open [6]. Further research has also concentrated on the Longest Common Substring (LCS) and Longest Palindromic Substring (LPS) problems [10, 15], which are pivotal in string processing.

In this paper, we aim to explore the practical implementation of quantum circuits for addressing combinatorial problems in string comparison, building on the theoretical foundations laid by earlier research [5].

Given two strings, x and y , composed by characters drawn from the same alphabet Σ of size σ , having both length n , and given d , with $0 < d \leq n$, we address the following problems:

- the *Fixed Prefix Matching* (FPM) problem, which verifies the existence of a prefix of fixed length d common to both strings. This problem can be generalized to cases where x has length n and y has length m and $n > m$, by solving the FPM problem for the strings $x[0\dots m-1]$ and y .
- the *Fixed Factor Matching* (FFM) problem, starting from an index i , with $0 \leq i < n-d$, verifies that the two strings share a substring of length d starting at position i . More formally, it checks whether $x[i\dots i+d-1] = y[i\dots i+d-1]$. This problem can also be generalized to strings of different lengths. Notably, the FPM problem is equivalent to the FFM problem with index $i = 0$.
- the *Shared Fixed Substring Checking* (SFC) problem, which checks for the existence of a common substring between x and y starting at the same position

in both strings. Formally, SFC verifies that

$$\exists j : 0 \leq j < n - d \text{ and } x[j..j + d - 1] = y[j..j + d - 1].$$

Note that the SFC problem corresponds to the logical OR of the solutions returned by FFM for indices i varying from 0 to $n - d - 1$. Thus, the SFC problem is computationally more challenging than the previous ones.

These problems, though less explored, offer valuable insights and applications across various domains. Therefore, solving these problems is of paramount importance in this domain.

Traditionally, these problems have been efficiently addressed using classical computation techniques, often exploiting simple data structures such as generalized suffix trees. However, Cantone *et al.* [5] propose quantum computation-based solutions for these problems, enabling solutions to be achieved in polylogarithmic time. Specifically, their approach, grounded in the circuit-based computational model, adeptly addresses this challenge, yielding a computational time complexity of $O(\log^3(n))$ for binary strings and $O(\log^4(n))$ in the general case.

In this paper, we present a practical implementation of their quantum algorithm to solve the three different problems, providing also the pseudo-code and some experimental results.

The paper is organized as follows. In Sect. 2 we provide some basic notions of Quantum Computing, while in Sect. 3, we detail our implementation using `Qiskit`, discussing the algorithm implementation of all the operators involved.

In Sect. 4 we demonstrate the practical approach for each of the three combinatorial problems discussed in the paper, solving each of them using the operators previously defined. Finally, in Sect. 5, we draw our conclusions.

2 Basics of Quantum Computation

In this paper, we assume the reader has some familiarity with quantum computing. However, to make the paper self-contained, we introduce the essential notions of quantum computation useful for understanding our presentation.

We employ the model of *reversible quantum circuits* for our computational framework. Quantum circuits consist of wires that transmit qubit values to and from gates, which perform elementary quantum operations and enable the linear transfer of qubits throughout the circuit. Input values are fed into the circuit from the left, and outputs are collected on the right. In this model, each wire may interact with no more than one gate at any given time step. To address complex computational tasks, *ancillary qubits*, or *ancilla*, are often integrated into the circuit architecture.

The complexity of a quantum circuit in this model can be quantified by its *size*, i.e., the number of gates it contains. However, a more nuanced metric is the circuit's *depth*, defined by the number of discrete time steps required for the circuit's execution. During each time step, a single gate operation can be conducted. It is important to note that the circuit's depth is distinct from the

total count of gates, as multiple gates that operate on non-intersecting sets of qubits may execute concurrently, allowing for parallelism within the circuit operations.

A simple yet effective example is the initialization of a quantum register of n qubits to a quantum state characterized by the superposition of all possible 2^n values. This initialization, achievable through n Hadamard operators, each acting on a different qubit, requires constant time as the Hadamard gates are executed in parallel.

In this field, operators are constructs that describe functional processes altering the state of quantum registers. Implementing operators that functions in constant time on a fixed-size quantum register poses no technical challenge. However, operators that handle quantum registers of variable size necessitate the use of a composite of elementary gates.

In this study, we specifically focus on a selection of gates central to our analysis. The **Pauli-X** gate, also known as the **X** or **NOT** gate, serves as the quantum analogue of the classical negation gate within the standard basis states $|0\rangle$ and $|1\rangle$. This gate alters a single qubit's state, flipping $|0\rangle$ to $|1\rangle$ and vice versa.

Another pivotal gate in our discussion is the Hadamard gate, denoted as **H**. This single-qubit operation transforms the state $|0\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. By executing this transformation, the Hadamard gate generates a superposition of the two basis states, each with equal amplitude.

The controlled **NOT** gate (or **CNOT**) is a quantum logic gate operating on a register of two qubits $|q_0, q_1\rangle$. If the control qubit $|q_0\rangle$ is set to 1, it inverts the target qubit $|q_1\rangle$; otherwise, all qubits remain the same. Formally, it maps $|q_0, q_1\rangle$ to $|q_0, q_0 \oplus q_1\rangle$.

The Toffoli gate, also known as the **CCNOT** gate, is a fundamental reversible logic gate that operates on three qubits. When the first two qubits are both in the state 1, the third qubit is flipped; if not, the state of all qubits remains unchanged. Mathematically, it transforms a three-qubit register $|q_0, q_1, q_2\rangle$ into $|q_0, q_1, q_0q_1 \oplus q_2\rangle$.

The **Swap** gate is a two-qubit operator. Expressed in basis states, it swaps the state of the two qubits $|q_0, q_1\rangle$ involved in the operation, mapping them to $|q_1, q_0\rangle$.

Generalizations of the **CNOT** gate include the n -ary fanout operator and the multiple-**CNOT** gate. The n -ary fanout operator utilizes a single control qubit to influence $n - 1$ target qubits, whereas the multiple-**CNOT** uses $n - 1$ control qubits to act upon a single target qubit.

While theoretically, a constant time fanout could be achieved using a series of n controlled-not gates, practical limitations arise due to the no-cloning theorem, which restricts the direct duplication of qubit states in constant depth [14]. Nevertheless, if the target qubits are initialized to $|0\rangle$, a fanout operation can be efficiently computed in logarithmic depth, $\Theta(\log(n))$, through a divide-and-conquer strategy employing controlled-not gates and 0 ancillary qubits [7].

For implementing a multi-controlled NOT gate, foundational strategies from classical Boolean logic are often adapted, as first illustrated in classical circuit contexts [22] and later in quantum settings [2]. This approach, however, necessitates the use of $n - 2$ ancillary qubits for storing intermediate results. Such configurations generally result in a circuit whose depth is linear relative to the number of control qubits, as parallel processing opportunities are limited under this scheme. Recent advancements in circuit design have demonstrated that rearranging gate sequences to promote parallel execution can reduce the circuit depth to logarithmic scales, necessitating either $n - 2$ or $n/2$ ancillary qubits depending on the specific implementation strategy [1, 12].

3 A Practical Implementation

In this section, we outline the quantum circuits involved in the computation of the Fixed Substrat MAtching Problems listed in Sect. 1 and provide a comprehensive practical implementation of all the operators necessary to solve our combinatorial problem, using `Qiskit`, an open source toolkit developed by IBM and based on the Python language.

In addition to showing the `Qiskit` code useful for the implementation of quantum algorithms, during our discussion we will show the circuits generated by our code and, where necessary, we will show the result of the simulation of these circuits to verify their correct functioning. Furthermore, we will discuss, where appropriate, the complexity of the circuits obtained, both in terms of size and depth, adding useful information to what has already been discussed in [5]. The codes shown in this paper can be consulted and executed online in this [Colab Tutorial](#).

Initially, we introduce the Parameterized Cyclic Rotation Operator (Sect. 3.1), subsequently we show the implementation of the Matching Substring Vector (Sect. 3.2) made by the Match operator and the Extension operator. Afterwards we show how to implement the Register Reversal Operator (Sect. 3.3) and the Copy Operator with Reversal Control (Sect. 3.4). Finally, we propose an implementation of the two circuits for the bitwise conjunction operator (\wedge) (Sect. 3.5) and the disjunction operator (\vee) (Sect. 3.6).

3.1 The Parameterized Cyclic Rotation Operator

The first operator whose implementation we describe is the Circular Rotation Operator. In this work we will use the implementation carefully described by Pavone and Viola in [19].

A circular shift operator, denoted as R_s , is employed in quantum computing to perform a shift of s positions on a register comprising n qubits. In this work we are interested in applying left rotations to our quantum registers, both for the pattern register and for the text register. This operation ensures that the element at position $(i + s) \bmod n$ is shifted to position i , implementing a circular

arrangement for elements beyond the register size n . Formally, the action of the operator R_s is represented by the permutation:

$$|q_0, q_1, \dots, q_{n-1}\rangle \rightarrow |q_s, q_{s+1}, \dots, q_{n-1}, q_0, q_2, \dots, q_{s-1}\rangle$$

For example, rotating a quantum register representing the state $|10110001\rangle$ 3 positions to the left would obtain the new configuration $|10001101\rangle$.

In quantum computing, a rotation operation of a register can be viewed as a permutation of its elements, allowing it to be decomposed into a series of Swap operations. This decomposition typically requires at most n Swaps to transform a sequence into the desired permutation. However, within a quantum circuit, the application of multiple Swap operators can occur concurrently, particularly when dealing with mutually disjoint pairs of qubits.

For a register comprising n qubits, it becomes feasible to execute up to $n/2$ Swap operations simultaneously. This parallel execution facilitates the efficient movement of $n/2$ qubits to their final positions within a single time step. The optimal scenario arises when the register requires rotation precisely by $n/2$ positions, necessitating only $n/2$ parallel Swaps. Specifically, this entails exchanging q_i with $q_{n/2+i}$, for $i = 0..n/2 - 1$. Such an operation exhibits a time complexity of $O(1)$, signifying constant-time execution and exemplifying the efficiency achievable within quantum circuits.

Pavone and Viola present in their work [19] a comprehensive procedure for constructing the circular rotation operator for registers of any size n and any shift amount s , where both n and s are powers of 2. They prove that the resulting circuit exhibits a depth of $O(\log(n))$. Although we refrain from providing a formal proof of this assertion in our current study, it is evident that, in the worst-case scenario, employing $n/2$ swap operations effectively positions a maximum of $n/2$ elements, thus leaving the remaining $n/2$ elements for subsequent positioning. With each subsequent step, the number of qubits requiring accurate positioning decreases by half, enabling a complete rotation in $O(\log(n))$ steps.

Figure 1 shows the procedure $\text{rot}(n, s)$ for the construction of the left rotation quantum operator together with the circuits generated by the procedure for a quantum register of 8 qubits and shift quantities equal to 1, 2 and 4 positions respectively.

In our implementation we need to apply, in superposition, all possible rotations to the quantum register of size n representing the text. Thus the circular shift operator needs to apply a rotation dependent on an input parameter j . In this context, the operation involves two quantum registers within the circuit: the first register is the parameter $|j\rangle$, which stores the input value related to the rotation amount, with a size of $\lceil \log(n) \rceil$; the second register $|q\rangle$, of size n , represents the register to be rotated. The initialization of $|j\rangle$ is expressed as $|j\rangle = \bigotimes_{i=0}^{\lceil \log(n) \rceil - 1} |j_i\rangle$, where $|j_i\rangle$ is initialized with the i -th least significant bit of the binary representation of k .

Figure 2 showcases the procedural framework, $\text{prot}(n)$, responsible for constructing the parameterized rotation circuit. In the middle, a circuit diagram instantiated by this procedure, exemplified for $n = 8$, is depicted. Notably, the

```

def rot(n, s):
    qc = QuantumCircuit(n)
    for i in range(1, log2(n)-log2(s)+2):
        for j in range(0, n/(s*(2**i))):
            for q in range(j*s*2**i, s*(j*2**i+1)):
                qc.swap(n-1-(q+s), n-1-(q+2**(i-1)*s+s))
    return qc.to_gate(label='ROT'+str(s))

def rot_gate(n,s):
    return rot(n,s).to_gate(label='Rot'+str(s))

```

Fig. 1. On top: the Qiskit procedure $rot(n, s)$ for implementing the left-rotation quantum operator which cyclically rotate a register with size n of s positions to the right. On bottom: three leftward rotation operators generated by the procedure for a quantum register of 8 qubits and shift amounts equal to 1, 2 and 4, respectively.

circuit initializes the parameter k using Hadamard operators, thereby representing all feasible shift values between 0 and $n - 1$, in superposition.

For any given value of i , such that $0 \leq i < \log(n)$, the application of the rotation operator $ROT2^i$ to register $|q\rangle$ is controlled by the qubit $|j_i\rangle$. It is noteworthy that the rotation operator $ROT2^i$ can be constructed using a circuit with a depth which is, at most, $\log(n)$.

We observe that the possibility that such an operator is controlled by a single qubit, i.e. $|j_i\rangle$, is realized through the application of a technique capable of creating multiple copies of the control qubit and which involves the use of $\log(n)$ ancillary qubits (these ancillæ qubits are implicit in the Qiskit implementation).

On the bottom of Fig. 2, we show measurements obtained from 1000 runs of the circuit delineated on the left. It's discernible that the resultant alternatives from the measurements correspond to all configurations of the rotated input string 10001110, each bearing an equiprobable likelihood of measurement.

In summary, the parameterized leftward circular shift operator, contingent on the value of an input quantum k , utilizes $n + \lceil \log(n) \rceil - 1$ input qubits, along with $\lceil \log(n) \rceil - 1$ ancillæqubits, exhibiting a depth equal to $O(\log^2(n))$.

```
def parametrized_rot(n):
    logn = int(np.log2(n))
    jr = QuantumRegister(logn, 'j')
    yr = QuantumRegister(n, 'y')
    qc = QuantumCircuit(jr, yr)
    for i in range(logn):
        crot = rot_gate(n, 2**i).control(1)
        qc = qc.compose(crot, [jr[i]]+yr[:])
    return qc
```

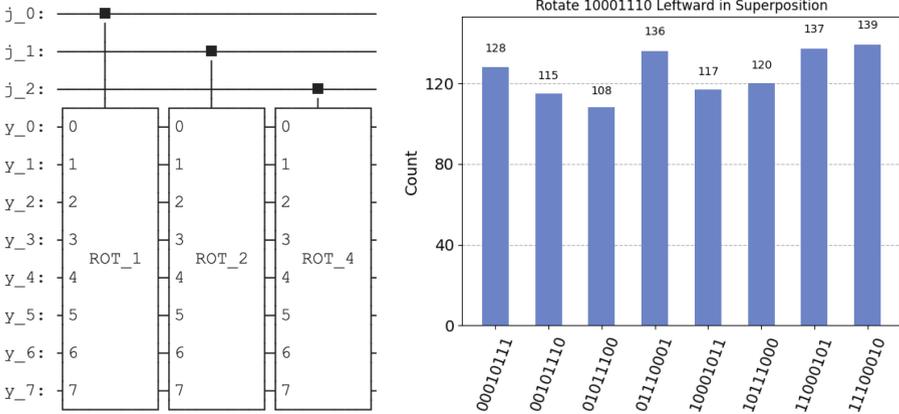


Fig. 2. The parameterized cyclic rotation operator. The Qiskit code useful for building the quantum circuit for a string of length n is shown above. Below, on the left, the circuit generated by the procedure for a string of length $n = 8$ is shown. On the right, we show the measurements obtained from 1000 runs of the circuit delineated on the left.

3.2 Constructing the Matching Substring Vector

Consider $i > 0$ representing the matching substring vectors λ_i , for $0 \leq i \leq \log(d)$, described as bit-vectors of length n such that, for $0 \leq j < n - 1$, λ_i has a bit set at position j only if the substring of x of length 2^i starting at position j matches the corresponding substring in y . In quantum computing, each vector λ_i is depicted by a quantum register $|\lambda_i\rangle$ of identical size.

To develop the *Matching Substring Vector*, two operators are crucial: the *Match* operator and the *Extension* operator.

The *Match* operator generates the initial matching substring vector λ_0 . It performs the following transformation for all $x, y \in \{0, 1\}^n$:

$$M|x\rangle|y\rangle|0^n\rangle = |x\rangle|y\rangle|\lambda_0\rangle$$

```

def M(n):
    x = QuantumRegister(n,'x')
    y = QuantumRegister(n,'y')
    λ = QuantumRegister(n,'λ')
    qc = QuantumCircuit(x,y,λ)
    for i in range(n):
        qc.mcx(x[i],y[i],λ[i])
    for i in range(n):
        qc.x(x[i])
        qc.x(y[i])
    for i in range(n):
        qc.mcx(x[i],y[i],λ[i])
    for i in range(n):
        qc.x(x[i])
        qc.x(y[i])
    return qc

def EXT(idx,n,m):
    x = QuantumRegister(n,'x')
    y = QuantumRegister(m,'y')
    qc = QuantumCircuit(x,y)
    for i in range(n-idx):
        qc.ccx(x[i],x[i+idx],y[i])
    return qc

```

Fig. 3. The two Qiskit procedures for creating the M match operator and the EXT extension operator

This transformation is based on the relation:

$$\lambda_0[j] = (x_j = 1 \wedge y_j = 1) \oplus (\neg(x_j = 1) \wedge \neg(y_j = 1))$$

The matching gate M can be constructed using two sets of n parallel Toffoli gates, with the second set interleaved with two arrays of parallel X gates. Figure 3 demonstrates the practical construction of the match operator.

The subsequent $|\lambda_i\rangle$ registers, for $i > 0$, are derived from $|\lambda_{i-1}\rangle$ using the extension operator, denoted as EXT_i .

The EXT_i operator can be implemented with a series of Toffoli gates. Figure 4 presents the Qiskit procedure for constructing the EXT operator.

The extension operator functions in constant time, using two sets of parallel Toffoli gates: one set for even positions and another for odd positions. As a result, the computation of all $|\lambda_i\rangle$ registers, for $0 \leq i \leq \log(n)$, has a depth of $O(\log(n))$.

The final circuit for the Matching Substring Vector is illustrated in Fig. 5, with a practical execution plot shown in Fig. 5.

3.3 The Register Reversal Operator

The register reversal operator, denoted as \doteq , transforms a single register $|q\rangle$ of size n by reversing the order of its qubits. Specifically, for any $q \in \{0, 1\}^n$, the register reversal operator performs the following transformation:

$$\doteq |q\rangle = |\bar{q}\rangle$$

This operation involves $\lfloor n/2 \rfloor$ swaps between pairs of qubits, but since these swaps can be executed in parallel, the reversal of the entire register requires only constant time.

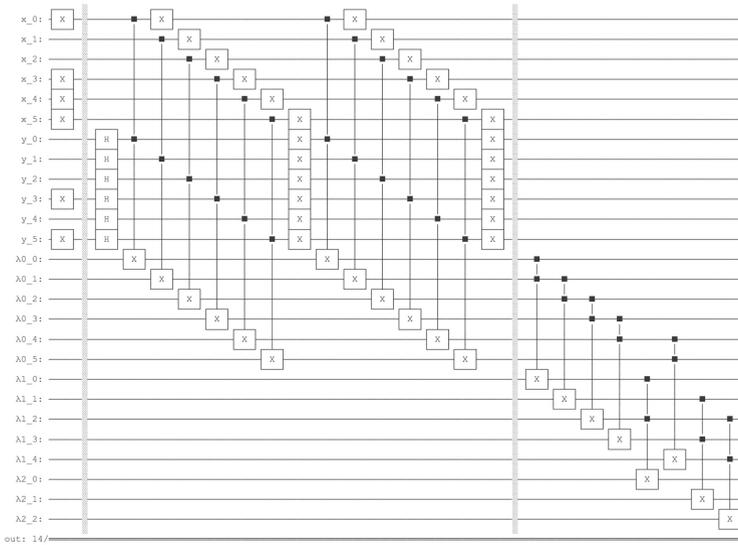


Fig. 4. Graphical depiction of the circuit used for computing the Matching Substring Vector for two strings of length 6. The Matching Substring Vector is stored in the final 3-qubit register λ_2 .

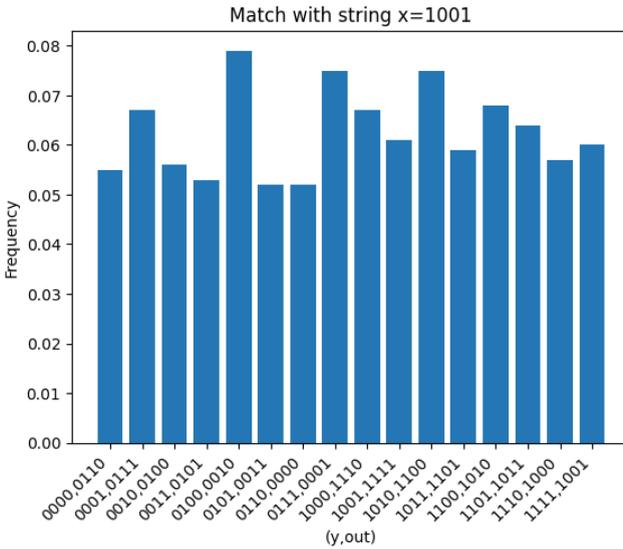


Fig. 5. Plot of the Matching Substring Vector for the string $x = '1001'$ and the string y in superposition. The x-axis shows all pairs $(y, M(x, y))$, while the y-axis indicates the frequency of the results.

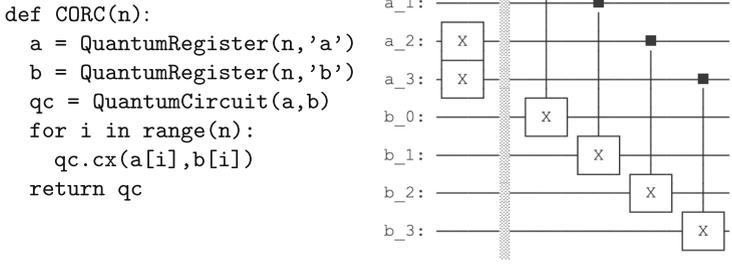


Fig. 6. Pseudocode and graphical representation of the *CORC* operator

In our algorithm, we employ the register reversal operator during the initialization phase of the parameter d , which determines the length of the sub-strings to be compared within x and y . Specifically, we derive \bar{d} from the input parameter d through the mapping $\doteq |d\rangle \rightarrow |\bar{d}\rangle$.

3.4 The Copy Operator with Reversal Control

The copy operator with reversal control (CRC operator) is a specialized copy operator designed to transfer the value of the n qubits from a source register $|a\rangle$ to a target register $|b\rangle$, which is initially set to $|0^n\rangle$. Additionally, there is a control qubit $|c\rangle$ that determines whether the operation should be executed. Specifically, this control qubit employs inverse logic: the operation is performed when $|c\rangle$ is 0 and is inhibited when $|c\rangle$ is 1.

The CRC operator is implemented using n Toffoli gates. The i -th Toffoli gate, for $0 \leq i < n$, uses $|c\rangle$ and $|a_i\rangle$ as control qubits and $|b_i\rangle$ as the target qubit. Despite having a common control qubit $|c\rangle$, which prevents parallel execution, we can use a well-known technique to copy the value of $|c\rangle$ onto n ancilla qubits. This enables the parallel application of the n fanout operators.

This technique allows us to build a circuit for this operator with a depth of $\mathcal{O}(\log(n))$. Figure 6 shows the practical implementation and the circuit.

3.5 The Controlled Bitwise Conjunction Operator

The bitwise conjunction operator performs a bit-by-bit logical AND operation between two registers, $|a\rangle$ and $|b\rangle$, each containing n qubits, and stores the result in a third register, $|q\rangle$, also of size n . However, we focus on a controlled version of this operator, where an additional register $|c\rangle$, composed of a single qubit, acts as the control qubit. This means that the bitwise logical AND operation is performed only when the control qubit is 0; otherwise, no operation is performed.

This operator can be implemented using n fanout gates, where each gate has three control qubits: $|c\rangle$, $|a_i\rangle$, and $|b_i\rangle$, and the target is $|q_i\rangle$, for $0 \leq i < n$. Figure 7 illustrates the practical implementation and the circuit.

```

def CBC(n):
    a = QuantumRegister(n,'a')
    b = QuantumRegister(n,'b')
    q = QuantumRegister(n,'q')
    qc = QuantumCircuit(a,b,q)
    for i in range(n):
        qc.mcx([a[i],b[i]],q[i])
    return qc
    
```

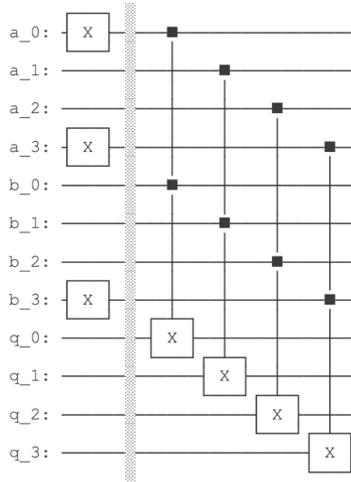


Fig. 7. Practical implementation and graphical representation of the *CBC* operator

Even though these n fanout gates share a common control qubit $|c\rangle$, preventing parallel application, a known technique allows us to copy the value of $|c\rangle$ onto n ancilla qubits. This enables the parallel execution of the n fanout gates.

Using this technique, we can construct a circuit for this operator with a depth of $\mathcal{O}(\log(n))$.

3.6 The Register Disjunction Operator

The register disjunction operator, often called the disjunction operator, performs a logical OR operation across the n qubits of the input register $|a\rangle$ and stores the result in the output qubit $|r\rangle$.

This operator can be implemented using a fanout operation on the qubits in $|a\rangle$, with $|r\rangle$ as the target. This process involves surrounding the fanout operation with two layers of X gates applied to the qubits in $|a\rangle$. The practical implementation and the circuit for the disjunction operator are illustrated in Fig. 8. The depth of the resulting circuit is $\mathcal{O}(\log(n))$.

4 Solving the Three Cases

As mentioned above, all the gates and operators defined in the previous sections will be necessary to solve three different combinatorial problems. The differences between these problems lie in the approach to the solution. In the original paper[5], the authors propose to solve the three problems using the same solution only changing the way the input registers are initialized. Their final quantum circuit has a depth equal to $\mathcal{O}(\log^3(n))$. In this paper we define two easier and more efficient solutions for the FPM and FFM problems, which requires a limited

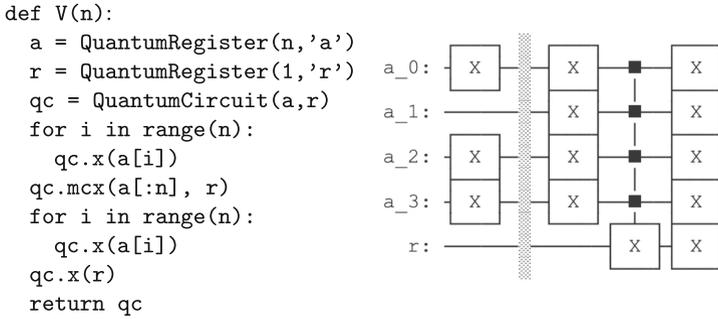


Fig. 8. Practical implementation and graphical representation of the disjunction operator

number of gates and operators compared with the previous version reducing their complexity to $O(\log(n))$. Whilst for the SFSC algorithm we use the same specific provided in literature.

```

def FPM(x,y,d):
    xr = QuantumRegister(d, 'x')
    yr = QuantumRegister(d, 'y')
    λ = QuantumRegister(d, 'λ')
    r = QuantumRegister(1, 'r')
    cr = ClassicalRegister(1, 'out')
    qc = QuantumCircuit(xr,yr,λ,r,cr)
    qc = qc.compose(M(d),xr[:d]+yr[:d]+λ[:])
    qc.mcx(λ[:d],r)
    qc.measure(r,cr)

def FFM(x,y,j,d):
    xr = QuantumRegister(n, 'x')
    yr = QuantumRegister(n, 'y')
    λ = QuantumRegister(d, 'λ')
    r = QuantumRegister(1, 'r')
    cr = ClassicalRegister(1, 'out')
    qc = QuantumCircuit(xr,yr,λ,r,cr)
    qc = qc.compose(rot(n,j),xr[:n])
    qc = qc.compose(M(d),xr[:d]+yr[:d]+λ[:])
    qc.mcx(λ[:d],r)
    qc.measure(r,cr)
    
```

Fig. 9. Practical implementation of the FPM and FFM algorithm

4.1 FPM Case

The first combinatorial problem which we take into account is the Fixed Prefix Matching (FPM) problem, which checks for the presence of a common prefix between x and y of a given length d , Formally:

$$x[0 \dots d - 1] = y[0 \dots d - 1]$$

Thus in this problem the only operator that we need to use as shown in Fig. 9 is the Match operator, which checks if exists a prefix match of length d .

4.2 FFM Case

The second problem studied in this paper is the Fixed Factor Matching (FFM) problem, which tests, for a given input parameter j , with $0 \leq j < n - d$, for the existence of a common factor between x and y , beginning at position j of both strings, formally:

$$x[j \dots j + d - 1] = y[j \dots j + d - 1]$$

This second problem stands to be really similar to the previous one, only requiring a pre-computed shift in order to align the two strings at position j , as shown in the code at Fig. 10.

```
def SFSC(x,y,d):
    n = len(x)
    xr = QuantumRegister(d, 'x')
    yr = QuantumRegister(d, 'y')
    dr = int_to_binary_vector(d)[::-1]
    Dr = [QuantumRegister(ceil(log2(n+1))2, f'D{i-1}') for i in range(len(dr)+1)]
    λ = QuantumRegister(d, 'λ0')
    r = QuantumRegister(1, 'r')
    cr = ClassicalRegister(1, 'out')
    qc = QuantumCircuit(xr,yr,λ,Dr,r,cr)
    for i in range(d):
        λ2 = QuantumRegister((len(λ[i-1])-i), 'λ'+str(i))
        λ.append(λ2)
    for i in range(len(Dr[0])):
        qc.x(Dr[0][i])
    qc = qc.compose(M(d),xr[:d]+yr[:d]+λ[:len(λ)])
    for i in range(1,len(dr)):
        qc=qc.compose(EXT(i,len(λ[i-1]),len(λ[i])),λ[i-1][:len(λ[i-1])]+λ[i][:len(λ[i])])
    for i in range(0,len(dr)):
        if dr[i] == '1':
            qc = qc.compose(CBC(len(λ[i])),λ[i][:len(λ[i])]+Dr[i][:len(λ[i])]+Dr[i+1][:len(λ[i])])
            qc = qc.compose(rot(len(Dr[i+1]),2**i,1), Dr[i+1])
        else:
            qc = qc.compose(CORC(len(Dr[i])),Dr[i][:len(Dr[i])]+Dr[i+1][:len(Dr[i+1])])
    qc = qc.compose(V(len(Dr[len(dr)]),Dr[len(dr)][:len(Dr[len(dr)])+r[:len(r)]]
    qc.measure(r,cr)
```

Fig. 10. Practical implementation of the SFSC algorithm

4.3 SFSC Case

The Shared Fixed Substring Checking SFSC differs from the FFM for the sake of the initial position of the match j is not previously defined. Formally it tests weather:

$$\exists j : 0 \leq j < n - d \text{ and } x[j..j + d - 1] = y[j..j + d - 1]$$

Although in the first two problems we just used the Rotation operator and the Match operator separately, in this problem we use the Matching Substring Vector

made by using both the Match operator and the *Ext* operator. Subsequently the *CBC* operator will perform a controlled bit-to-bit logical AND between the two registers while also a rotation is applied using the *Rot* operator. Finally the Register Disjunction Operator computes the logic *OR* operation between n qubits of the input register and deposit the result inside the output qubit. The code of the algorithm is shown in Fig. 11.

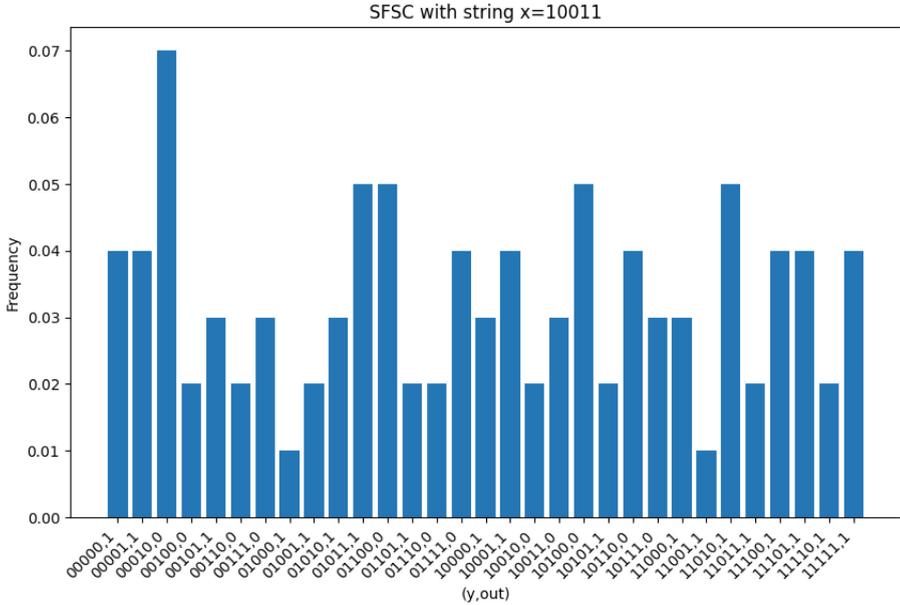


Fig. 11. Plot generated running 1000 run of the SFSC algorithm over the string $x = 10011$ and the string y in super position. In the abscissa the values y and 1 is a match is found and 0 otherwise, in the ordinates the frequency of each plot.

5 Conclusions and Future Works

This paper provides a comprehensive elucidation of several combinatorial problems in string processing within the domain of Quantum Computation. We introduce an exhaustive exposition of the required operators and gates involved in the computational circuitry. Subsequently, we present practical tutorials and implementations for all three problems discussed in this paper. We also improved the circuits of previously exposed in literature for two of the three problem presented, gaining a cubic factor. In future research, we aim to explore alternative string algorithms well-established in classical computation.

References

1. Balauca, S., Arusoia, A.: Efficient constructions for simulating multi controlled quantum gates. In: Groen, D., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) ICCS 2022. LNCS, vol. 13353, pp. 179–194. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08760-8_16
2. Barenco, A., et al.: Elementary gates for quantum computation. *Phys. Rev. A* **52**(5), 3457–3467 (1995)
3. Boyer, R.S., Moore, J.S.: A fast string searching algorithm. *Commun. ACM* **20**(10), 762–772 (1977)
4. Cantone, D., Faro, S., Pavone, A.: Quantum string matching unfolded and extended. In: Kutrib, M., Meyer, U. (eds.) RC 2023. LNCS, vol. 13960, pp. 117–133. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38100-3_9
5. Cantone, D., Faro, S., Pavone, A., Viola, C.: Quantum circuits for fixed substring matching problems. CoRR, abs/2308.11758 (2023)
6. Chakraborty, D., Das, D., Goldenberg, E., Koucký, M., Saks, M.E.: Approximating edit distance within constant factor in truly sub-quadratic time. *J. ACM* **67**(6), 36:1–36:22 (2020)
7. Fang, M., Fenner, S., Green, F., Homer, S., Zhang, Y.: Quantum lower bounds for fanout. *Quantum Inf. Comput.* **6**(1), 46–57 (2006)
8. Faro, S., Lecroq, T.: The exact online string matching problem: a review of the most recent results. *ACM Comput. Surv.* **45**(2), 13:1–13:42 (2013)
9. Faro, S., Marino, F.P., Scardace, A.: Practical implementation of a quantum string matching algorithm. In: TODO (ed.) Proceedings of Quasar. CEUR Workshop Proceedings (2024)
10. Le Gall, F., Seddighin, S.: Quantum meets fine-grained complexity: sublinear time quantum algorithms for string problems. *Algorithmica* **85**(5), 1251–1286 (2023)
11. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC 1996, pp. 212–219. Association for Computing Machinery, New York (1996)
12. He, Y., Luo, M., Zhang, E., Wang, H.-K., Wang, X.-F.: Decompositions of n -qubit Toffoli gates with linear circuit complexity. *Int. J. Theor. Phys.* **56**, 2350–2361 (2017)
13. Horspool, R.N.: Practical fast searching in strings. *Softw. Pract. Exp.* **10**(6), 501–506 (1980)
14. Høyer, P., Spalek, R.: Quantum fan-out is powerful. *Theory Comput.* **1**, 81–103 (2005)
15. Jin, C., Nogler, J.: Quantum speed-ups for string synchronizing sets, longest common substring, and k -mismatch matching. In: Bansal, N., Nagarajan, V. (eds.) Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, 22–25 January 2023, pp. 5090–5121. SIAM (2023)
16. Kaye, P., Laflamme, R., Mosca, M.: An Introduction to Quantum Computing. Oxford University Press, Oxford (2006)
17. Montanaro, A.: Quantum pattern matching fast on average (2015)
18. Niroula, P., Nam, Y.: A quantum algorithm for string matching. *npj Quantum Inf.* **7**, 37 (2021)
19. Pavone, A., Viola, C.: The quantum cyclic rotation gate. In: Castiglione, G., Sciortino, M. (eds.) Proceedings of the 24th Italian Conference on Theoretical Computer 2023. CEUR Workshop Proceedings, vol. 3587, pp. 206–218. CEUR-WS.org (2023)

20. Ramesh, H., Vinay, V.: String matching in $O(n+m)$ quantum time. *J. Discrete Algorithms* **1**(1), 103–110 (2003). *Combinatorial Algorithms*
21. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**(5), 1484–1509 (1997)
22. Toffoli, T.: Reversible computing. In: de Bakker, J., van Leeuwen, J. (eds.) *ICALP 1980*. LNCS, vol. 85, pp. 632–644. Springer, Heidelberg (1980). https://doi.org/10.1007/3-540-10003-2_104
23. Yao, A.C.-C.: The complexity of pattern matching for a random string. *SIAM J. Comput.* **8**(3), 368–387 (1979)