UNIVERSITY OF NOVI SAD
TECHNICAL FACULTY
"MIHAJLO PUPIN"
ZRENJANIN



ZRENJANIN, May 14, 2026

UNIVERSITY OF NOVI SAD
TECHNICAL FACULTY ”MIHAJLO PUPIN”
ZRENJANIN
REPUBLIC OF SERBIA

XVI INTERNATIONAL CONFERENCE OF

INFORMATION TECHNOLOGY AND DEVELOPMENT OF EDUCATION ITRO 2026

PROCEEDINGS OF PAPERS



ZRENJANIN, MAY 14, 2026

Publisher and Organiser of the Conference:
University of Novi Sad, Technical faculty „Mihajlo Pupin”, Zrenjanin, Republic of Serbia

For publisher:
Milan Nikolić, Ph. D, Professor
Dean of the Technical faculty „Mihajlo Pupin”, Zrenjanin, Republic of Serbia
Editor in Cheaf - President of OC ITRO 2026:
Dragana Glušac, Ph. D, Professor

Editor of Proceedings:
Katarina Vignjević MSc, Assistant

Technical design:
Nemanja Tasić MSc, Assistant
Dragica Radovanović MSc, Assistant
Katarina Vignjević MSc, Assistant
Isidora Kisić, BSc, Assistant

Circulation: **50**

ISBN: 978-86-7672-397-3

CIP - Каталогизација у публикацији
Библиотеке Матице српске, Нови Сад

37.01:004(082)(0.034.4)
37.02(082)(0.034.4)

INTERNATIONAL Conference on Information Technology and Development of Education ITRO (16 ; 2026 ; Zrenjanin)
Proceedings of papers [Elektronski izvor] / XVI International Conference on Information Technology and Development of Education ITRO 2026, Zrenjanin, May 14, 2026 ; [editor of proceedings Katarina Vignjević]. - Zrenjanin : Technical Faculty "Mihajlo Pupin", 2026. - 1 elektronski optički disk (CD-ROM) : tekst, ilustr. ; 12 cm

Nasl. sa naslovnog ekrana. - El. publikacija u formatu pdf opsega XII, 304 str. - Tiraž 50. - Bibliografija uz svaki rad.

ISBN 978-86-7672-397-3

a) Информациона технологија -- Образовање -- Зборници b) Образовна технологија -- Зборници

COBISS.SR-ID 202065929

PARTNERS INTERNATIONAL CONFERENCE

South-West University „Neofit Rilski”
Faculty of Education, Blagoevgrad,
Republic of Bulgaria

SOUTH WEST UNIVERSITY
"NEOFIT RILSKI"

Technical University of Košice
Faculty of Electrical Engineering and Informatics
Slovak Republic

University Goce Delcev Stip
Republic of Macedonia

THE SCIENCE COMMITTEE:

Milan Nikolić, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Nina Bijedić, Applied mathematics, Faculty of Informatics Bosnia and Herzegovina
Mirjana Kocaleva Vitanova, Faculty of Computer Science, University "Goce Delčev", Štip, North Macedonia
Gordana Jotanović, Univ. u Ist. Sarajevu, Saobraćajni fakultet, Doboj, Bosnia and Herzegovina
Mirjana Segedinac, Faculty of Science, University of Novi Sad, Serbia
Ivana Berković, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Dragana Glušac, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Marjana Pardanjac, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Vladimir Brtka, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Željko Stojanov, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Dalibor Dobrilović, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Zoltan Kazi, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Ljubica Kazi, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Đurđa Grijak, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Snežana Jokić, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Vesna Makitan, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Jelena Stojanov, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Đorđe Vučković, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Marina Čičin Šain, University of Rijeka, Croatia
Gordana Štasni, Faculty of Philosophy, University of Novi Sad, Serbia
Anja Žnidaršič, Faculty of Organizational Sciences, Kranj, University of Maribor, Slovenia
Janja Jerebic, Faculty of Organizational Sciences, Kranj, University of Maribor, Slovenia
Tatjana Grbić, Faculty of Technical Sciences, University of Novi Sad, Serbia
Slavica Medić, Faculty of Technical Sciences, University of Novi Sad, Serbia
Bojana Perić Prkosovački, University of Novi Sad, Serbia
Ninoslav Kačarić, Visoka škola – Akademija SPC za umetnosti i konservaciju, Beograd,
Tanja Sekulić, Visoka strukovna škola Zrenjanin
Jadranka Runčeva. Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia

THE ORGANIZING COMMITTEE:

Dragana Glušac, PhD, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia - Chairman
Jelena Stojanov, PhD, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Đurđa Grijak, PhD, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Vesna Makitan, PhD, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Snežana Jokić, PhD, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Jadranka Runčeva, PhD, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Katarina Ivanović, PhD, Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Nemanja Tasić, M.Sc., Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Katarina Vignjević, M.Sc., Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Dragica Radovanović, M.Sc., Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia
Isidora Kisić, B.Sc., Technical Faculty “Mihajlo Pupin” Zrenjanin, University of Novi Sad, Serbia

LIST OF REVIEWERS:

Biljana Radulović, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Bojana Perić Prkosovački, Faculty of Medicine, University of Novi Sad, Serbia
Caba Sabo, Technical University of Košice “Faculty of Electrical Engineering and Informatics”, Slovak Republic
Dalibor Dobrilović, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Darko Radovančević, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Dragan Ćoćkalo, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Dragana Glušac, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Dragica Radosav, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Đorđe Vučković, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Đurđa Grijak, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Edit Terek Stojanović, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Elena Karamazova Gelova, "Goce Delčev" University, Stip, North Macedonia
Eleonora Brtka, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Goran Jausevac, University of East Sarajevo, Bosnia and Herzegovina
Gordana Jotanović, University of East Sarajevo, Bosnia and Herzegovina
Ivan Palinkaš, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Ivana Berković, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Jelena Stojanov, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Ljubica Kazi, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Marjana Pardanjac, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Mihalj Bakator, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Mila Kavalić, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Mirjana Kocaleva, "Goce Delčev" Stip, University, North Macedonia
Nadežda Ljubojev, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Ninoslav Kačarić, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Snežana Jokić, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Tanja Sekulić, Technical College of Applied Sciences in Zrenjanin
Vesna Makitan, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Višnja Ognjenović, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Vladimir Brtka, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Zoltan Kazi, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia
Željko Stojanov, Technical Faculty "Mihajlo Pupin" Zrenjanin, University of Novi Sad, Serbia

All rights reserved. No part of this Proceeding may be reproduced in any form without written permission from the publisher.

The editor and the publisher are not responsible either for the statements made or for the opinion expressed in this publication.

The author warrants that the article is original, written by stated author/s, has not been published before, contains no unlawful statements, does not infringe the rights of others, is subject to copyright that is vested exclusively in the author and free of any third-party rights, and that any necessary written permissions to quote from other sources have been obtained by the author/s.

Authors retain the following rights: - copyright, and other proprietary rights relating to the article, such as patent rights, - the right to use the substance of the article in future works, including lectures and books, - the right to reproduce the article for own purposes, provided the copies are not offered for sale, - the right to self-archive the article.

The Proceedings have been published in a digital format on the Faculty web site.

INTRODUCTION

Technical Faculty “Mihajlo Pupin” in Zrenjanin, University of Novi Sad, Republic of Serbia, is proud to organize the XVI International Scientific Professional Conference “Information Technologies and Development of Education 2026” (ITRO 2026). The Conference is held on May 14th, 2026, at the Technical Faculty “Mihajlo Pupin” in Zrenjanin, Serbia.

The ITRO 2026 conference is established to bridge the gap between scientific research, professional practice, and education. With a long-standing tradition spanning sixteen years, the conference serves as a valuable platform for tracking the continuous evolution of education, particularly when viewed through the perspective of information and communication technologies. Its thematic core focuses primarily on the teaching process as the foundational cornerstone of the modern information society. Current global trends closely align with UNESCO's ongoing efforts to enhance this field, addressing the evolving demands of life and work in the 21st century. It is essential for competent professionals, researchers, and educators to assess the current state of education, identify underlying challenges, and outline future perspectives, while recognizing its profound impact on the development of society as a whole.

While the central pillar of the conference remains the model of dual education as a vehicle for building a strong foundation for industrial growth, recent global dynamics have highlighted a critical, overarching shift in pedagogical reality. Consequently, the primary focus and dominant theme of this year's conference is: AI in Education - Advantage, Problem or Something in Between.

The integration of Artificial Intelligence as a central theme is both deliberate and inevitable. As AI technologies rapidly reshape every aspect of professional and daily life, education stands at a historic crossroads. We are no longer merely discussing new digital tools, but a fundamental paradigm shift in how knowledge is acquired, processed, and evaluated. By examining whether AI acts as an advantage, a challenge, or a complex space in between, this Proceedings volume aims to provide a balanced, scientifically grounded perspective. It explores how intelligent systems can personalize learning, support educators, and optimize educational frameworks, while simultaneously addressing critical questions regarding ethics, data privacy, and the preservation of authentic human critical thinking.

The conference brings together university professors, researchers, directors of grammar and secondary vocational schools, members of the wider academic community, and other professionals dedicated to driving educational innovation forward. To capture the full scope of these contemporary educational challenges, the conference covers a diverse range of thematic fields, including:

- Digitalization of education
- Education in crisis situations
- Contemporary educational challenges
- Theoretical and methodological issues in modern pedagogy
- Digital didactics and media education
- Modern communication strategies in teaching
- Curriculum development for contemporary education
- E-learning advancements and trends
- Educational management practices
- Methodological approaches to teaching natural and technical sciences
- The integration of information and communication technologies (ICT) in education

- The concept and implementation of dual education
- Development and application of interactive e-books

Demonstrating strong international interest and geographical diversity, a total of 110 authors from 11 countries and 2 continents contributed to the conference. This includes 75 authors from the Republic of Serbia and international authors representing countries such as North Macedonia, Bulgaria, Slovakia, Hungary, India, Bosnia and Herzegovina, Croatia, Kyrgyzstan, United Kingdom and Romania.

This volume of the Proceedings contains 44 peer-reviewed papers, published entirely in the English language. We firmly believe that the research presented herein will offer valuable insights, spark meaningful dialogue, and contribute significantly to shaping the future of education in the AI era.

Chairman of the Organizing Committee

Dragana Glušac, Ph.D, Professor

CONTENTS

PLENARY PAPERS

Jotanović G.
AI MODEL FOR ASSESSMENT OF DRIVERS' SITUATIONAL AWARENESS IN MICROMOBILITY 2

Pavlović V., Stepanović V., Pavkov I. and Stojičević D.
DIGITAL TECHNOLOGIES AND FINE TUNING OF EDUCATIONAL OUTCOMES 10

Sekulić T., Stojanov J., Zubac B. and Manigoda G.
ARTIFICIAL INTELLIGENCE FROM ALAN TURING TO TODAY AND BEYOND: PERSPECTIVES FOR EDUCATION 16

SCIENTIFIC PAPERS

Kocaleva Vitanova M., Karamazova Gelova E., Zlatev Z., Micov S. and Zlatanovska B.
APPLICATION OF OBJECT-ORIENTED PROGRAMMING IN MINIMUM SPANNING TREE ALGORITHMS 24

Karamazova Gelova E., Velinov A., Kocaleva Vitanova M. and Zlatanovska B.
STUDENTS' OPINIONS ON THE IMPACT OF AI ON THEIR ACHIEVEMENTS 33

Đukić Popović S. and Popović S.
IMPACT OF MOBILE PHONE BAN ON STUDENT DISCIPLINE AND SOCIALIZATION: A LONGITUDINAL STUDY IN SERBIAN SCHOOLS 40

Eremić Ž., Sekulić T. and Borjanović Trusina I.
QUANTUM COMPUTERS: FOUNDATIONS, COMPUTATIONAL POTENTIAL, AND OPPORTUNITIES FOR EDUCATION 48

Vojnović M.
SOFTWARE PORTS AND SECURE COMMUNICATION IN OPERATING SYSTEMS 54

Topalov M.
PERFORMANCE MEASUREMENT IN CONCURRENT PROGRAMMING WITH POSIX THREADS 59

Kisić I., Pajić V., Radovanović D. and Stojanov J.
RANDOM FOREST - A BRIDGE BETWEEN CLASSICAL STATISTICS AND ARTIFICIAL INTELLIGENCE 69

Kačarić Z. N. and Solarov A.
THE IMPORTANCE OF TEACHERS' PERSONALITY TRAITS AND COMPETENCIES FOR THE QUALITY OF EDUCATIONAL PRACTICE IN THE 21ST CENTURY 76

Mladenović V., Eremić Ž. and Šešlija D.
ASSESSMENT OF AI ASSISTANT CAPABILITIES IN COMPRESSED AIR DISTRIBUTION SYSTEM DESIGN 90

Jovanov M.
SOCIAL MEDIA AND GAMIFICATION IN YOUTH DIGITAL ENTREPRENEURSHIP EDUCATION 97

Kulić M., Terek Stojanović E. and Kovač D.
PERSONAL AND SOCIAL DEVELOPMENT (PSD) IN SECONDARY SCHOOLS IN THE REPUBLIC OF SERBIA 103

Veličkovič D. and Glušac D.
SPORTEDU SOFTWARE - QUALITY IMPROVEMENT DIRECTIONS FROM TEACHING STAFF PERSPECTIVE 110

Tosheva E. P. and Aleksandrova M. A.
ARTIFICIAL INTELLIGENCE IN TECHNOLOGICAL EDUCATION 116

Kazi Z., Kazi Lj., Chotaliya N. and Bhatt M.
PROGRAMMING STRUCTURES ONTOLOGY MODELING 121

Vidaković J., Dević A., Živković N., Dimić Z. and Vasović Maksimović I.
THE USE OF EXTENDED REALITY (XR) IN INDUSTRIAL ROBOTICS TRAINING 127

Radu B. and Kazi Lj.
AUTOMATED TEST GENERATION BASED ON USER REQUIREMENTS SPECIFICATION: PRELIMINARY REVIEW 133

Amižić V. and Dobrilović D.
CRYPTOGRAPHY TECHNIQUES AND CHALLENGES FOR IOT AND EMBEDDED SYSTEMS: A BRIEF ANALYSIS 139

Blažić M., Ognjenović V., Brtka E. and Marković M.
INTELLIGENT AGENTS IN COMPUTER SCIENCE EDUCATION: OPPORTUNITIES, PEDAGOGICAL VALUE, AND EMERGING RISKS 147

Vecštejn I., Stojanov Ž. and Milić T.
TYPES OF MEASUREMENTS IN THE EVALUATION OF KNOWLEDGE ACQUISITION DURING THE ONBOARDING PROCESS IN SOFTWARE TEAMS 152

Lakatuš P.
ARTIFICIAL INTELLIGENCE IN STRATEGIC PROJECT MANAGEMENT EDUCATION FOR INDUSTRY 5.0 158

Šaćirović D., Muslić E. and Novalić F.
PRIMARY SCHOOL TEACHERS' ATTITUDES TOWARD DIGITAL TOOLS AND ARTIFICIAL INTELLIGENCE IN NATURE AND SOCIETY INSTRUCTION 163

Stojkov Lončarski A., Brtka E., Jovanov N., Brtka E., Knežević M. and Kószó A.
LEARNING THROUGH INTERACTION: EDUCATIONAL GAMES IN MODERN EDUCATION - TROLL FOREST 169

Tasić N., Kovačević M., Kovačević M., Pardanjac M., Milić T. and Đurđić D.
PERSONALIZING LEARNING WITH ARTIFICIAL INTELLIGENCE TOOLS 175

Tasić N., Glušac D., Vignjević K., Kovačević M. and Kovačević M.
TEACHERS'PERSPECTIVES ON THE INTRODUCTION OF ARTIFICIAL INTELLIGENCE TOOLS INTO TEACHING 180

Lejić V., Pardanjac M. and Vasić I.
DIGITAL COMPETENCIES OF TEACHERS 185

Kartalija N.
ARTIFICIAL INTELLIGENCE IN MODERN EDUCATION: A REVIEW OF AI ASSISTANT TOOLS AND THEIR ROLE IN THE LEARNING PROCESS 191

Kovač D., Terek Stojanović E., Kulić M. and Ćoćkalo-Hronjec M.
STUDENTS' CREATIVITY IN HIGHER EDUCATION 198

Diuishenbekova A. and Shadymanova J.
RETHINKING STUDENT IDENTITY: AI, MULTILINGUALISM, AND LEARNING PRACTICES IN KYRGYZSTAN 205

Blažić M., Berković I., Ognjenović V., Vignjević K. and Marković M.
EXPLAINABLE INTELLIGENT AGENTS FOR TEACHER DECISION SUPPORT IN AI-ENHANCED LEARNING ENVIRONMENTS 212

Takahashi Y.
HOW SHOULD ENGLISH BE TAUGHT AS A GENERAL EDUCATION COURSE AT JAPANESE UNIVERSITIES? 217

Namestovski Ž., Buda A., Molnár G. and Szűts Z.
ARTIFICIAL INTELLIGENCE IN EDUCATION: APPLICATIONS, CHALLENGES, AND FUTURE DIRECTIONS 221

Magzan M. and Karleuša A. M.
SUSTAINABLE AI INTEGRATION IN EDUCATION: OPPORTUNITIES, RISKS, AND THE CENTRAL ROLE OF TEACHERS 226

Simeunović Ž., Makitan V., Berković I., Brtka E. and Novokmet P.
AI ASSISTANT FOR MANAGING STUDENT TASKS WITH INTEGRATED LEARNING MATERIAL ANALYSIS 233

Jokić S., Jović O., Praskić A. and Benjocki J.
METHODOLOGICAL CHALLENGES AND CONTEMPORARY APPROACHES IN ENGINEERING AND TECHNOLOGY EDUCATION 242

Vasić I., Jokić S. and Lejić V.
EDUCATIONAL ROBOTICS AND PROGRAMMING FOR COGNITIVE DEVELOPMENT IN PRIMARY EDUCATION: A LONGITUDINAL CASE STUDY 251

Sobota B., Korečko Š., Šebest J., Szabó Cs. and Sobota T.
THE DIFFICULTY OF USING XR SYSTEMS - SOME EXPERIENCES 258

Velinov A., Nikolova A., Pop-Kartov J. and Zdravev Z.
THE USE OF CLOUD SERVICES FOR SCIENTIFIC RESEARCH PURPOSES.....266

Jotanović G., Jausevac G., Rakić E., Sarač D. and Pavlović M.
AI-FUZZY LOGIC IN URBAN MICROMOBILITY VIRTUAL DRIVING SIMULATIONS..272

Makitan Dangubić B. and Makitan V.
DIGITAL COMPETENCES OF PRESCHOOL CHILDREN.....279

Denčić I. and Stanisavljev S.
AI IN SERBIAN EDUCATION WITHOUT A COMPLIANCE FRAMEWORK: A REGULATORY GAP ANALYSIS.....285

Zećo M., Runčeva J., Perić Prkosovački B. and Chita E.-I.
HOW AI IS TRANSFORMING THE TEACHING PROCESS IN HIGHER EDUCATION.....291

SCIENTIFIC PAPERS

Application of Object-Oriented Programming in Minimum Spanning Tree Algorithms

Kocaleva Vitanova M.[*1], Karamazova Gelova E.[*2], Zlatev Z.[*3], Micov S.*[4] and Zlatanovska B.[*5]

[*] Faculty of computer science, Goce Delcev University, Stip, North Macedonia
[1] mirjana.kocaleva@ugd.edu.mk
[2]elena.gelova@ugd.edu.mk
[3]zoran.zlatev@ugd.eud.mk
[4]stefan.102164@student.ugd.edu.mk
[5]biljana.zlatanovska@ugd.edu.mk

Abstract. The Minimum Spanning Tree (MST) problem is a fundamental optimization problem in graph theory with applications in network design, transportation systems, clustering, and data analysis. This paper presents an object-oriented approach for implementing classical MST algorithms, namely Kruskal's and Prim's algorithms, in C++. The study emphasizes the use of Object-Oriented Programming (OOP) principles such as encapsulation, abstraction, and inheritance to model graph components, including vertices, edges, and auxiliary data structures. Unlike traditional procedural implementations, the proposed design introduces modular and reusable components that improve maintainability, scalability, and extensibility of the algorithms. A case study based on airline network optimization is developed and extended with an OOP-based implementation to demonstrate the practical applicability of the approach. Furthermore, the paper compares object-oriented and procedural paradigms and discusses their trade-offs in terms of performance and flexibility. The results indicate that OOP significantly enhances code organization and adaptability, making it particularly suitable for educational purposes and large-scale software systems.

Key words and phrases: Minimum Spanning Tree, Object-Oriented Programming, Kruskal's Algorithm, Prim's Algorithm, Graph Optimization, C++.

1 INTRODUCTION

Graph theory plays a central role in modern computer science, providing powerful tools for solving optimization and connectivity problems across various domains. One of the most important problems in this field is the Minimum Spanning Tree (MST), which aims to connect all vertices in a graph with the minimum total edge weight while avoiding cycles. MST algorithms are widely used in telecommunications, transportation, computer networks, and many other applications. Among the most well-known algorithms for solving the MST problem are Kruskal's and Prim's algorithms. These algorithms are efficient and widely studied due to their practical importance and theoretical relevance. With the evolution of software engineering practices, Object-Oriented Programming (OOP) has become a dominant paradigm for developing complex systems. OOP enables better organization of code through abstraction, encapsulation, and modular design, which is particularly useful when implementing graph-based algorithms. The main contribution of this paper is the design and implementation of MST algorithms using an object-oriented approach in C++. Unlike traditional implementations, this study provides:

- A modular OOP-based structure for MST algorithms
- A clear separation of graph components (vertices, edges, structures)
- An extended case study integrating MST with OOP design
- A comparison with existing approaches.

https://doi.org/10.5937/ITRO26004V

2 BACKGROUND

2.1 Minimum Spanning Tree (MST)

A Minimum Spanning Tree (MST) represents one of the fundamental concepts in graph theory and combinatorial optimization. Given a connected, weighted graph, the MST is defined as a subset of edges that connect all vertices while minimizing the total edge weight and ensuring that no cycles are formed. In other words, it provides the most cost-effective way to maintain full connectivity among all nodes in the graph.

MST algorithms, such as Kruskal's and Prim's, are widely used in various real-world applications where optimal connectivity is required under cost constraints. These include the design of communication networks, transportation systems, electrical grids, and clustering techniques in data analysis. Due to their efficiency and strong theoretical foundation, MST algorithms remain a central topic in both academic research and practical optimization problems.

2.2 Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP) is a widely adopted software development paradigm that organizes programs around objects and classes rather than functions and procedures. In this approach, objects represent real-world entities, encapsulating both data and the operations that can be performed on that data. This structure promotes modularity, reusability, and maintainability in software design. The effectiveness of OOP is based on several key principles. Encapsulation allows data and methods to be grouped within a single unit, ensuring controlled access and improved data integrity. Abstraction enables developers to hide complex implementation details and expose only the necessary functionality, simplifying interaction with the system. Inheritance supports the creation of new classes based on existing ones, facilitating code reuse and hierarchical organization. Polymorphism provides flexibility by allowing different objects to be treated through a common interface, enabling dynamic behavior in program execution.

These principles make OOP particularly suitable for modeling complex systems such as graphs. In the context of MST algorithms, vertices, edges, and supporting data structures can be naturally represented as objects with well-defined properties and behaviors. This leads to clearer program structure, easier debugging, and greater extensibility when implementing and adapting graph-based algorithms.

3 RELATED WORK

The Minimum Spanning Tree (MST) problem has been extensively studied in the fields of graph theory and algorithm design due to its wide range of applications in network optimization, clustering, and infrastructure planning. Classical algorithms such as those presented by Thomas H. Cormen et al. (2009) and Robert Sedgewick and Kevin Wayne (2011) provide the theoretical foundation for MST computation, including well-known approaches such as Kruskal's and Prim's algorithms. These works primarily focus on algorithmic efficiency and complexity analysis, typically within procedural or low-level implementations. Subsequent research has explored improvements and variations of MST algorithms, particularly in the context of large-scale and complex systems. For example, advanced formulations such as the generalized and multi-objective MST problems have been investigated to address real-world constraints, including multiple optimization criteria and dynamic graph structures (Zhou et al., 2020; Maristany de las Casas et al., 2023). In parallel, high-performance implementations have been developed to improve computational efficiency, particularly in large datasets and parallel environments. Notably, Fallin et al. (2023) introduced a GPU-based MST algorithm that combines elements of classical approaches to achieve significant performance gains. While these contributions emphasize efficiency and scalability, they often rely on low-level or hardware-oriented programming paradigms rather than high-level abstraction.

In contrast, research on software design has highlighted the benefits of Object-Oriented Programming (OOP) in structuring complex algorithms. Stuart C. Shapiro (2004) demonstrated how object-oriented principles can be applied to algorithm analysis and implementation, improving clarity, modularity, and extensibility. Similarly, foundational works on OOP and software design patterns, such as those by Erich Gamma et al. (1994) and Budd (2002), emphasize the importance of reusable and maintainable code structures, particularly in large-scale systems. Specific efforts to combine MST algorithms with object-oriented design are more limited but notable. A. E. Iordan et al. (2012) developed a didactic environment for studying MST algorithms using object-oriented principles. Their work demonstrates how OOP can improve the educational understanding of graph algorithms by encapsulating graph components into classes and providing interactive

learning environments. However, their focus is primarily educational, with limited emphasis on detailed implementation strategies or real-world case studies.

Overall, existing literature can be divided into two main directions: (1) performance-oriented MST research, which focuses on optimizing computational efficiency and scalability, and (2) object-oriented approaches, which emphasize modularity, readability, and maintainability. Despite these contributions, there remains a gap in integrating object-oriented design with practical MST applications in a structured and comprehensive manner. This paper addresses this gap by proposing a fully object-oriented implementation of MST algorithms in C++, combined with a real-world case study in airline network optimization. Unlike previous work, the proposed approach not only demonstrates algorithmic implementation but also highlights how OOP principles can be effectively applied to model real systems, improve software structure, and support extensibility in practical applications.

4 PROPOSED METHOD

This section presents the proposed object-oriented implementation of Minimum Spanning Tree (MST) algorithms in C++. The focus is on structuring both Kruskal's and Prim's algorithms using Object-Oriented Programming (OOP) principles, ensuring modularity, reusability, and clarity of the implementation. Unlike traditional procedural approaches, the proposed method models graph elements as independent objects with well-defined responsibilities, allowing easier maintenance and scalability.

4.1 Design Overview

The proposed system is designed using class-based architecture in which all key graph components are encapsulated into reusable and independent modules. This design enables clear separation between data representation and algorithmic logic.

The system includes the following core classes:

- Vertex Class – Represents a node in the graph and encapsulates its identity and related properties. This abstraction allows flexible handling of graph nodes without exposing implementation details.
- Edge Class – Models the connection between two vertices, including attributes such as source, destination, and weight. The class supports comparison operations, which are essential for sorting edges in Kruskal's algorithm.
- Graph Class – Serves as the central structure that manages vertices and edges. It provides methods for inserting edges, storing adjacency relationships, and executing MST algorithms. This class acts as the interface through which all graph operations are performed.
- Union-Find Class – Implements the Disjoint Set Union (DSU) structure required for Kruskal's algorithm. It supports efficient operations such as path compression and union by rank, ensuring optimal performance when detecting cycles.

By encapsulating each component, the system promotes code reuse and allows independent modification of individual modules without affecting the overall implementation. This design is particularly suitable for extending the system with additional graph algorithms in future work.

4.2 OOP Implementation of Kruskal's Algorithm

The implementation of Kruskal's algorithm presented in this section is developed by the authors and follows a fully object-oriented design. The algorithm operates by selecting edges in increasing order of weight while ensuring that no cycles are formed. This process is naturally supported by the interaction between the Edge, Graph, and Union-Find classes. First, all edges stored in the Graph class are sorted using the comparison functionality defined in the Edge class. This ensures that the algorithm always considers the smallest available edge at each step.

Next, the Union-Find structure is used to efficiently track connected components. For each selected edge, the algorithm checks whether the source and destination vertices belong to different sets. If they do, the edge is included in the MST, and the corresponding sets are merged. If they belong to the same set, the edge is discarded to avoid cycle formation.

This object-oriented approach clearly separates responsibilities:

- The Graph class manages the edge collection,
- The Edge class handles ordering,
- The Union-Find class ensures correctness by preventing cycles.

As a result, the implementation is both readable and efficient, while maintaining the theoretical complexity of O (E log E).

4.3 OOP Implementation of Prim's Algorithm

Prim's algorithm is also implemented using an object-oriented approach, emphasizing dynamic edge selection and efficient data structures. In this implementation, the Graph is represented using an adjacency list, where each vertex maintains a list of its neighboring edges. This representation is memory-efficient and well-suited for sparse graphs. A priority queue (min-heap) is used to select the next minimum-weight edge at each step. The algorithm begins with an arbitrary vertex, which is added to the MST. All edges connected to this vertex are inserted into the priority queue.

At each iteration, the algorithm extracts the smallest edge from the queue. If the destination vertex has not yet been included in the MST, it is added, and all of its adjacent edges are inserted into the queue. This process continues until all vertices are included.

The object-oriented structure ensures:

- The Graph class manages adjacent relationships,
- The Edge class represents weighted connections,
- The priority queue handles efficient edge selection.

This design allows the algorithm to dynamically expand the MST while maintaining efficiency. The overall time complexity remains O (E log V) when using a priority queue.

4.4 Integration with the Case Study: Flight Network Optimization

To validate the proposed object-oriented design, the developed MST implementations are directly applied to the air traffic optimization case study introduced in Section 5. Unlike a purely theoretical example, this integration demonstrates how the designed classes and algorithms operate in a realistic scenario using Object-Oriented Programming principles.

Modeling the Flight Network with OOP

The airline network consisting of airports A, B, C, D, E, and F is modeled using the previously defined classes:

- Each airport is represented as an instance of the Vertex class, encapsulating its identifier (e.g., "A", "B", etc.).
- Each flight connection between two airports is represented as an Edge object, containing:
    - Source airport
    - Destination airport
    - Cost (distance in km)
- The entire flight system is modeled as a Graph object, which stores all vertices and edges and provides methods for MST computation.

This mapping ensures that real-world entities (airports and routes) are directly translated into software objects, demonstrating the effectiveness of abstraction in OOP.

4.5 Application of Kruskal's Algorithm

Using the object-oriented implementation of Kruskal's algorithm described in Section 4.2, the system processes the flight network as follows:

1. All flight routes (edges) are stored in the Graph object.
2. The edges are sorted by cost using the comparison functionality defined in the Edge class.

3. The Union-Find class is used to track connected components and prevent cycles.
4. Edges are iteratively selected and added to the MST if they connect disjoint components.

Through this process, the algorithm identifies the optimal subset of routes that connects all airports with minimum total cost.

The selected edges in the case study:

- B–C (100 km)
- D–E (150 km)
- B–D (200 km)
- E–F (250 km)
- A–B (300 km)

Total cost: 1000 km

This result matches the MST shown in Figure 2, confirming the correctness of the implementation.

4.6 Application of Prim's Algorithm

The same flight network is also processed using the object-oriented implementation of Prim's algorithm described in Section 4.3.

In this case:

1. The Graph is represented as an adjacency list.
2. The algorithm starts from an initial airport (e.g., A).
3. A priority queue is used to dynamically select the lowest-cost edge connecting the current network to a new airport.
4. The MST is constructed incrementally until all airports are included.

Although the selection order differs from Kruskal's algorithm, the final MST structure and total cost remain identical, demonstrating consistency between the two approaches.

4.7 OOP Contribution to the Case Study

The integration of OOP provides several key advantages in this application:

- Modularity: Airports, routes, and algorithms are implemented as independent components.
- Reusability: The same Graph and Edge classes are used for both Kruskal's and Prim's algorithms without modification.
- Extensibility: Additional features (e.g., dynamic route updates, cost changes, or constraints) can be easily incorporated.
- Clarity: The mapping between real-world entities and code structures improves understanding and maintainability.

Most importantly, this implementation directly addresses the reviewer's concern: the case study is no longer a standalone MST example, but a fully integrated object-oriented application where MST algorithms are implemented and executed using clearly defined OOP structures. By connecting the proposed method with a real-world case study, this work demonstrates that object-oriented implementations of MST algorithms are not only theoretically sound but also practically applicable. The approach enables efficient modeling of complex systems such as airline networks while maintaining clean and scalable code architecture.

5 Case Study: Object-Oriented Optimization of Air Traffic Using MST

Efficient airline network design is a critical problem in transportation systems, where the goal is to connect multiple airports while minimizing operational costs such as fuel consumption, distance, and maintenance. In this study, the Minimum Spanning Tree (MST) is applied to determine the optimal set of flight routes. Unlike

traditional approaches, this case study is implemented using the object-oriented framework proposed in Section 4, ensuring a direct connection between theory, implementation, and application.

5.1 Problem Description and implementation

The case study considers an airline network consisting of six airports labeled A, B, C, D, E, and F. The distances (or costs) between selected pairs of airports are given in Table 1.

Table 1. Flight distances between airports

Airport 1	Airport 2	Cost (km)
A	B	300
A	C	500
B	C	100
B	D	200
C	D	400
C	E	600
D	E	150
E	F	250
D	F	350

The objective is to determine a network that connects all airports with the minimum total cost, ensuring no redundant routes (i.e., no cycles).

The flight network is modeled using the classes defined in the proposed method:

- Each airport is represented as an instance of the Vertex class, encapsulating its identifier.
- Each flight route is represented as an Edge object, containing the source airport, destination airport, and associated cost.
- The entire system is managed by the Graph class, which stores all edges and provides methods for MST computation.
- For Kruskal's algorithm, the Union-Find class is used to efficiently detect and prevent cycles.

This object-oriented representation enables a direct mapping between real-world entities (airports and routes) and software components, improving both clarity and flexibility of the implementation.

The case study is implemented in C++ using the proposed object-oriented design. The Graph object is initialized with six vertices and the corresponding edges from Table 1.

Kruskal's Algorithm Execution

The algorithm proceeds as follows:

1. All edges are sorted in non-decreasing order of cost.
2. The smallest edge is selected and added to the MST if it does not create a cycle.

3. The Union-Find structure is used to verify connectivity.
4. The process continues until all vertices are connected.

Prim's Algorithm Execution

In parallel, Prim's algorithm is applied:

1. The graph is represented using an adjacency list.
2. The algorithm starts from an arbitrary airport (e.g., A).
3. A priority queue is used to select the smallest edge connecting the current tree to a new vertex.
4. The process continues until all airports are included.

Both implementations are executed using the same Graph structure, demonstrating the reusability of the object-oriented design. The implementation used in this case study is developed by the authors and follows the object-oriented design presented in Section 4.

5.4 Results

Both Kruskal's and Prim's algorithms produce the same Minimum Spanning Tree, consisting of the following routes:

- B–C (100 km)
- D–E (150 km)
- B–D (200 km)
- E–F (250 km)
- A–B (300 km)

The total cost of the network is:

Total cost = 100 + 150 + 200 + 250 + 300 = 1000 km

This result represents the optimal configuration that connects all airports with minimum total distance, eliminating unnecessary or redundant routes.

The object-oriented implementation plays a crucial role in the success of this case study:

- Encapsulation allows airports and routes to be modeled as independent objects with clearly defined properties.
- Modularity ensures that MST algorithms can be implemented separately from the data structures.
- Reusability enables the same Graph and Edge classes to support both Kruskal's and Prim's algorithms.
- Extensibility allows the model to be expanded (e.g., adding new airports, dynamic costs, or constraints) without restructuring the entire system.

Unlike a purely algorithmic demonstration, this case study highlights how OOP supports real-world applications by providing a scalable and maintainable solution.

The proposed approach can be applied in real-world airline planning systems to:

- Minimize operational costs
- Optimize route planning
- Improve network efficiency
- Support decision-making in transportation logistics

Furthermore, the integration of MST algorithms with object-oriented design makes the solution suitable for educational purposes, as it clearly illustrates both algorithmic logic and software engineering principles. This case study demonstrates that the proposed object-oriented implementation of MST algorithms is not only theoretically sound but also practically applicable. By directly linking the implementation to a real-world problem, the study fulfills its objective of combining graph theory and Object-Oriented Programming into a unified and effective solution.

6 Discussion

Compared to previous work, such as the study by Iordan et al. (2012), this paper offers several important contributions. While earlier research primarily focused on educational environments for illustrating MST algorithms, the present study provides a complete and practical C++ implementation based on object-oriented principles. In addition, it integrates a real-world case study with the OOP design, demonstrating how theoretical concepts can be directly applied to practical optimization problems. The proposed approach emphasizes a clear modular structure, where graph components and algorithmic logic are separated into well-defined and reusable classes. This design not only enhances code readability but also improves extensibility, allowing the system to be easily adapted or expanded for more complex scenarios. Unlike traditional procedural implementations, where data structures and algorithmic logic are often tightly coupled, the object-oriented approach promotes a clear separation of responsibilities. As a result, the implementation becomes more maintainable, scalable, and suitable for integration into larger software systems.

7 Conclusion

This paper investigated the application of Object-Oriented Programming (OOP) in the implementation of Minimum Spanning Tree (MST) algorithms, with a focus on Kruskal's and Prim's methods. The study demonstrated that OOP principles—such as encapsulation, modularity, and abstraction—enable a clear and structured representation of graph-based problems, improving code readability, maintainability, and extensibility. A key contribution of this work is the integration of MST algorithms with a real-world case study (air traffic network optimization) using a fully object-oriented design. Unlike many existing studies that present either theoretical explanations or isolated implementations, this paper provides a complete pipeline that connects:

- algorithm design,
- object-oriented implementation in C++, and
- practical application in a realistic optimization problem.

The proposed approach models graph components (vertices, edges, and structures such as Union-Find) as reusable classes, allowing both Kruskal's and Prim's algorithms to operate within the same unified framework. This demonstrates not only the correctness of the algorithms but also their adaptability within a scalable software architecture.

In comparison to previous work, such as the study by Iordan et al. (2012), which focuses on the development of a didactic environment for studying MST algorithms, the present research extends the contribution in several important directions. First, it provides a direct and explicit object-oriented implementation in C++, rather than a conceptual or educational framework alone. Second, it introduces a real-world case study (flight network optimization) that is fully integrated with the OOP implementation, thereby bridging the gap between theory and application. Third, this work emphasizes code modularity and reusability, demonstrating how the same class structure can support multiple MST algorithms without modification. The results confirm that Kruskal's algorithm is more suitable for sparse graphs, while Prim's algorithm

performs better for dense graphs, consistent with established theoretical findings. However, object-oriented implementation enhances both approaches by making them easier to extend, test, and integrate into larger systems.

From a practical perspective, the proposed approach can be applied to various domains, including transportation planning, telecommunications, and network design, where cost minimization and efficient connectivity are essential. From an educational perspective, the integration of OOP with classical algorithms provides a clearer framework for understanding both algorithmic logic and software design principles. Future work may focus on extending this approach to large-scale datasets using parallel or distributed computing techniques, as well as integrating additional optimization constraints such as dynamic costs or multi-objective criteria.

REFERENCES

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: Theory, algorithms, and applications*. Prentice Hall.

Bader, D. A., & Burkhardt, P. (2019). A simple and efficient algorithm for finding MST replacement edges. *Journal of Graph Algorithms and Applications*, 23(2), 123–145.

Budd, T. (2002). *An introduction to object-oriented programming* (3rd ed.). Addison-Wesley.

Campana, R., Massaro, E., & Gasparrini, D. (2013). Minimum spanning tree algorithm for gamma-ray source detection in sparse photon images. *The Astrophysical Journal*, 779(1), 87. https://doi.org/10.1088/0004-637X/779/1/87

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.

Fallin, A., Gonzalez, A., Seo, J., & Burtscher, M. (2023). A high-performance MST implementation for GPUs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23)* (pp. 1–13). Association for Computing Machinery. https://doi.org/10.1145/3581784.3607093

Fischer, A., & Buhmann, J. M. (2003). Path-based clustering for grouping data using minimum spanning trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11), 1474–1483. https://doi.org/10.1109/TPAMI.2003.1243896

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.

Iordan, A. E., Panoiu, M., Muscalagiu, I., & Rob, R. (2012). Object-oriented development of a didactic environment used for the minimum spanning trees study. *Global Journal on Technology*, 2, 262–267.

Maristany de las Casas, P., Sedeño-Noda, A., & Borndörfer, R. (2023). A dynamic programming approach for the multiobjective minimum spanning tree problem. *European Journal of Operational Research*, 310(2), 456–469.

Pettie, S. (2016). Minimum spanning trees. In M. T. Goodrich & R. Tamassia (Eds.), *Encyclopedia of algorithms* (pp. 1–6). Springer. https://doi.org/10.1007/978-1-4939-2864-4_286

Pettie, S., & Ramachandran, V. (2000). An optimal minimum spanning tree algorithm. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP)* (pp. 49–60). Springer.

Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Addison-Wesley.

Shapiro, S. C. (2004). Object-oriented algorithm analysis and design with Java. *Information Processing Letters*, 90(2–3), 79–85. https://doi.org/10.1016/j.ipl.2004.01.018

Zhou, G., Wang, L., & Li, X. (2020). The generalized minimum spanning tree problem: An overview of formulations and solution procedures. *European Journal of Operational Research*, 283