**UNIVERSITATEA DE STAT DIN MOLDOVA**

**Facultatea de Matematică și Informatică**

Cu titlu de manuscris

C.Z.U: 519.83

**GRIGORIU NICOLAE**

# GRAFURI TRANZITIV ORIENTABILE

**112.03 – CIBERNETICĂ MATEMATICĂ**
**ȘI CERCETĂRI OPERAȚIONALE**

**Teză de doctor în științe matematice**

**Conducător științific:** **Cataranciuc Sergiu,**

doctor habilitat în științe fizico-matematice, conferențiar universitar

**Autorul:**

**CHIȘINĂU, 2016**

**© Grigoriu Nicolae, 2016**

**CUPRINS**

**ADNOTĂRI** ..... **5**

**LISTA ABREVIERILOR** ..... **8**

**INTRODUCERE** ..... **9**

**1. EVOLUȚIA CERCETĂRILOR ÎN DOMENIUL STUDIERII GRAFURILOR TRANZITIV ORIENTABILE ȘI ASPECTELE APLICATIVE ALE ACESTORA** ..... **16**

**1.1. Rolul grafurilor în soluționarea problemelor teoretico-aplicative** ..... **16**

**1.2. Grafurile de comparabilitate a mulțimilor parțial ordonate** ..... **20**

**1.3. Clase de implicații și orientarea tranzitivă a grafurilor** ..... **23**

**1.4. Aplicații ale grafurilor tranzitiv orientabile** ..... **28**

**1.5. Concluzii la capitolul 1** ..... **39**

**2. ORIENTAREA TRANZITIVĂ A GRAFURILOR NEORIENTATE** ..... **42**

**2.1. Subgrafuri stabile și lanțuri netriangulate** ..... **43**

**2.2. Subgrafuri B-stabile** ..... **52**

**2.3. Construirea orientărilor tranzitive** ..... **58**

**2.4. Numărul de orientări tranzitive** ..... **67**

**2.5. Concluzii la capitolul 2** ..... **79**

**3. GENERALIZĂRI ALE PROBLEMEI ORIENTĂRII TRANZITIVE A GRAFULUI** **81**

**3.1. Reorientarea unei mulțimi minimale de arce într-o orientare tranzitivă a grafului** **81**

**3.2. Reorientarea tranzitivă a grafului forțată de un arc apriori dat** ..... **90**

**3.3. Reorientarea tranzitivă a grafului forțată de o mulțime de arce apriori dată** ..... **94**

**3.4. Construirea unei orientări tranzitive având un arc dat** ..... **100**

**3.5. Construirea unei orientări tranzitive având o mulțime de arce date** ..... **106**

**3.6. Concluzii la capitolul 3** ..... **110**

**CONCLUZII GENERALE ȘI RECOMANDĂRI** ..... **112**

**BIBLIOGRAFIE** ..... **114**

**ANEXE ..... 124**

**Anexa 1. Implementarea algoritmilor elaborați în limbajul Javascript ..... 124**

**Anexa 2. Codul sursă pentru reprezentarea grafică a algoritmilor propuși ..... 135**

**DECLARAȚIA PRIVIND ASUMAREA RĂSPUNDERII ..... 141**

**CURRICULUM VITAE ..... 142**

## ADNOTARE

la teza de doctor „***Grafuri tranzitiv orientabile***”,

înaintată de către Grigoriu Nicolae pentru obținerea titlului de doctor în științe matematice la specialitatea 112.03 – Cibernetică Matematică și Cercetări Operaționale

Teza a fost elaborată la Universitatea de Stat din Moldova, Chișinău în anul 2016.

**Structura tezei:** Teza este scrisă în limba română și conține introducere, trei capitole, concluzii generale și recomandări, bibliografie ce cuprinde 119 de titluri. Lucrarea conține 113 pagini text de bază. Rezultatele obținute sunt publicate în 14 lucrări științifice.

**Cuvintele cheie:** graf tranzitiv orientabil, subgraf stabil, lanț netriangulat, graf factor, orientare tranzitivă, graf de comparabilitate.

**Domeniul de studiu al tezei:** Teoria grafurilor

**Scopul și obiectivele lucrării.** Caracterizarea structurală a grafurilor tranzitiv orientabile și elaborarea în baza acesteia a algoritmilor de construire a orientărilor tranzitive pentru grafurile neorientate cu restricții asupra muchiilor. Obiective: determinarea rolului lanțurilor netriangulate în construirea orientărilor tranzitive a grafurilor; examinarea proprietăților subgrafurilor B-stabile și rolul acestora la construirea grafurilor factor; studierea șirului complet de grafuri factor pentru descrierea problemei orientărilor tranzitive ale unui graf neorientat; determinarea formulei recurente de calcul a numărului de orientări tranzitive ale grafului; elaborarea algoritmilor pentru construirea orientărilor tranzitive cu restricții asupra muchiilor orientabile.

**Noutatea și originalitatea științifică** constă în studierea unei clase noi de grafuri stabile și construirea în baza proprietăților acestora a șirului complet de grafuri factor, folosit la soluționarea problemei orientării tranzitive a grafurilor neorientate, precum și la soluționarea problemei de enumerare a acestor orientări prin deducerea unei formule recurente.

**Problema științifică importantă soluționată** *constă în* determinarea unei clase de subgrafuri stabile folosite la studierea problemei de orientare tranzitivă și caracterizarea structurală a grafurilor tranzitiv orientabile, *care au condus la* obținerea unei metode eficiente pentru construirea orientărilor tranzitive și calcularea numărului acestor orientări.

**Semnificația teoretică** este determinată de obținerea unei caracterizări structurale a grafurilor tranzitiv orientabile. Se propune o metodă eficientă de studiere a orientărilor tranzitive în baza unui șir complet de grafuri factor.

**Valoare aplicativă.** Se propun algoritmi de construire a orientărilor tranzitive ale unui graf neorientat ce completează aspectul aplicativ al problemei studiate la soluționarea problemelor practice: testarea codului sursă al programelor, decompoziția rețelelor Petri, etc.

**Implementarea rezultatelor științifice.** Rezultatele obținute pot servi pentru inițierea unor cercetări în domeniu pentru studenții și masteranzii universităților, pot servi drept suport pentru unele cursuri opționale, pentru soluționarea problemelor practice. Algoritmii elaborați sunt realizați sub formă de programe în limbajul Javascript.

## АННОТАЦИЯ

диссертационной работы "***Транзитивно ориентируемые графы***", представленной автором Николае Григориу на соискание учёной степени кандидата математических наук по специальности 112.03 – Математическая Кибернетика и Исследование Операций

Диссертация выполнена в Молдавском Государственном Университете, Кишинёв, 2016.

**Структура работы:** Диссертация написана на румынском языке и содержит введение, три главы, заключение с рекомендациями, список цитированной литературы из 119 наименований. Работа содержит 113 страниц основного текста. Полученные результаты опубликованы в 14 научных работах.

**Ключевые слова:** транзитивно ориентируемый граф, стабильный подграф, нетриангулированная цепь, граф фактор, транзитивная ориентация, граф сравнения.

**Область исследования:** Теория графов.

**Цель исследования.** Структурная характеристика транзитивно ориентируемых графов и разработка алгоритмов для построения транзитивной ориентаций с ограничениями по дугам графа; определения роли нетриангулированных цепей в построении транзитивных ориентаций; изучение свойств Б-стабильных подграфов для построения графов фактор, изучение полного ряда графов фактор для описания проблемы транзитивных ориентаций неориентированного графа; определения рекурсивной формулы для подсчета количество транзитивных ориентаций графа.

**Научная новизна и оригинальность** выражается в том, что был описан новый класс стабильных подграфов и построение полного ряда графов фактор на основе их свойств, используемый для решения задачи транзитивной ориентации неориентированных графов, в том числе и для решения проблемы подсчета этих ориентаций через дедукции рекурсивной формулы.

**Решенная важная научная задача** *состоит в* определения нового класса стабильных подграфов, используемых для транзитивной ориентации и структурируемой характеристики транзитивно ориентируемых графов *которое привело к* получению эффективного метода для построения транзитивных ориентаций и подсчета их количество.

**Теоретическая ценность работы** определяется получением структурной характеристики транзитивно ориентируемых графов. Предлагается эффективный метод для изучения транзитивных ориентаций на основе полного ряда графов фактор.

**Практическая ценность работы.** Предлагаются алгоритмы построения транзитивных ориентаций неориентированного графа, которые дополняют практическую часть изучаемой задачи решением практических задач: анализ исходного кода программ, декомпозиция сетей Петри, и др.

**Внедрение научных результатов.** Полученные результаты могут лечь в основу выбора тем для обучения студентов. Разработанные алгоритмы реализованы в виде программ на программном языке Javascript.

## ANNOTATION

of the thesis "**Transitively orientable graphs**"

submitted by Nicolae Grigoriu in partial fulfillment of the requirements for degree of PhD in Mathematics, specialty 112.03 – Mathematical Cybernetics and Operational Research

The thesis has been elaborated in Chisinau, Moldova State University, in 2016.

**Thesis structure:** The thesis is written in Romanian language and contains an introduction, three chapters, general conclusions and recommendations, a bibliography of 119 titles, 113 pages of main text. Obtained results were published in 14 scientific papers.

**Keywords:** transitively orientable graph, stable subgraph, non-triangulated chain, graph factor, transitive orientation, comparability graph.

**The field of study:** Graph theory

**The aim of the research.** Structural characterization of the transitively orientable graphs and elaboration of the algorithms for the transitive orientation construction based on these characterizations. Objectives: definition of the non-triangulates chains role in construction of the transitive orientations, examination of the B-stabile subgraph properties and their role in construction of the graphs factor, study of the complete sequence of the graphs factor for the transitive orientation problem, definition of a recurrence formula for calculation of the number of transitive orientations in a graph.

**The scientific novelty and originality** is reflected in the following: there was described a new class of the stable subgraphs and based on their properties there was defined the complete sequence of graphs factor, that are used for the solution of the transitive orientation of undirected graphs, as well for the problem of the counting the number of transitive orientations of a graph.

**Important scientific problem solved in the research** *consists in* definition of a new class of stable subgraphs used for the study of the transitive orientation problem and structural characterization of the transitive orientable graphs, *which leads to* the obtaining of an efficient method for construction of transitive orientations of the graph and counting number of them.

**The theoretical significance** of the work is defined by the results which describe the structural characterization of transitive orientable subgraphs. It is proposed an efficient method of study of the transitive orientations based on a complete sequence of graphs factor.

**The applicative value of the paper.** There are proposed algorithms for construction of transitive orientations of an undirected graph, which completes the practical aspect of the studied problem for the solution of practical problems: testing of the source code of the programs, decomposition of the Petri nets, etc.

**The implementation of the scientific results:** The results might provide a basis for selecting the themes for students. The developed algorithms are implemented as a software in Javascript programming language.

## LISTA ABREVIERILOR

**SS** – subgraf stabil;

**LN** – lanț netriangulat;

**SSM** – subgraf stabil minimal;

**SSCM** – subgraf stabil complet maximal;

**SSVM** – subgraf stabil vid maximal;

**A-SBS** – subgraf B-stabil generat de mulțimea de vârfuri **A**;

**TRO** – orientare tranzitivă.

# INTRODUCERE

**Actualitatea și importanța temei de cercetare** este determinată de rolul grafurilor tranzitiv orientabile la soluționarea problemelor teoretico-aplicative din diverse domenii. Deja sunt bine cunoscute unele rezultate ce țin de analiza existenței instrucțiunilor de ieșire în codul sursă al programelor [113], decompoziția rețelelor Petri [108], analiza grafurilor moleculare [4], etc. Rezultatele obținute la moment generează la rândul sau mai multe probleme care necesită studii suplimentare pentru obținerea unei caracterizări a grafurilor tranzitiv orientabile pentru problemele legate de orientarea specială a muchiilor acestora. E de menționat faptul că grafurile tranzitiv orientabile servesc drept model eficient pentru soluționarea în timp polinomial a unor probleme importante care în caz general sunt NP-complete. De exemplu, pe clasa de grafuri menționată se rezolvă în mod eficient problema determinării clicii maximale ponderate [94], determinarea numărului de stabilitate internă al unui graf neorientat [40], precum și unele variații ale problemei de colorare a grafurilor [40], [9], [28], [32], [64].

De asemenea, este știut că problema de sortare încă rămâne a fi o problemă actuală în combinatorică și informatică. Sortarea mulțimilor parțial ordonate este un domeniu cu multe rezultate valoroase ce au multe aplicații în diverse domenii ale matematicii și informaticii. Este știut faptul că mulțimile parțial ordonate pot fi reprezentate prin clasa grafurilor tranzitiv orientabile. Astfel, reorientarea arcelor într-un graf tranzitiv orientabil duce la sortarea elementelor mulțimii parțial ordonate ce este reprezentată prin graful tranzitiv orientabil.

În prezent sunt cunoscute rezultate legate de studierea grafurilor tranzitiv orientabile, obținute de către Ghouila-Houri A. [38], Gilmore P.C., Hoffman A. J. [39], Golumbic M.C. [41], Shevrin L.N., Filipov N.D. [119], Wolk E.S. [109], ș.a. La baza acestor cercetări stau așa structuri ca subgraf stabil [116], clase de implicații [42], cu ajutorul cărora sunt obținute un șir de proprietăți speciale care permit construirea unor orientări tranzitive ale grafului. Totuși, în baza rezultatelor cunoscute nu putem obține răspunsuri la o serie de probleme importante: există oare o caracterizare structurală a grafurilor tranzitiv orientabile; obținerea unei formule simple de calcul a unei orientări tranzitive; determinarea condițiilor de existență a orientărilor tranzitive parțiale ale unui graf, etc.

**Scopul și obiectivele lucrării.**

Realizarea prezentei teze a pretins implicit atingerea următorului scop: caracterizarea structurală a grafurilor tranzitiv orientabile; obținerea formulei pentru determinarea numărului de orientări tranzitive într-un graf; elaborarea în baza structurilor obținute a algoritmilor de construire a orientărilor tranzitive pentru grafurile neorientate cu restricții asupra muchiilor.

În conformitate cu scopul enunțat au fost stabilite următoarele obiective ale cercetării:

- determinarea rolului lanțurilor netriangulate în construirea orientărilor tranzitive ale grafurilor;
- examinarea proprietăților subgrafurilor B-stabile și rolul acestora la construirea grafurilor factor;
- studierea șirului complet de grafuri factor pentru descrierea problemei orientărilor tranzitive ale unui graf neorientat;
- determinarea formulei recurente de calculare a numărului de orientări tranzitive ale grafului;
- elaborarea algoritmilor pentru construirea orientărilor tranzitive cu restricții asupra muchiilor orientabile;
- propunerea algoritmilor pentru editarea unei orientări tranzitive cu restricții impuse arcelor în orientarea existentă;
- implementarea algoritmilor elaborați.

Suportul metodologic al cercetărilor include unele noțiuni și metode din teoria grafurilor, metode de optimizare, teoria algoritmilor, etc. Aceste noțiuni sunt descrise în majoritatea surselor bibliografice. Totuși pot fi remarcate lucrările: [7], [59], [115].

**Noutatea științifică a rezultatelor obținute.** Gradul de noutate privind cercetările efectuate în lucrare îl constituie următoarele rezultate:

- a fost studiată o nouă clasă de subgrafuri stabile, utile pentru soluționarea problemei orientării tranzitive a grafurilor neorientate;
- au fost descrise proprietățile lanțurilor netriangulate în raport cu clasa subgrafurilor stabile într-un graf tranzitiv orientabil;
- în baza clasei de subgrafuri stabile au fost formulate condițiile necesare și suficiente pentru ca un graf să fie tranzitiv orientabil;
- în baza șirului complet de grafuri factor a fost dedusă formula recurentă de calcul a numărului de orientări tranzitive;
- a fost elaborată o serie de algoritmi în baza structurilor menționate pentru construirea unei orientări tranzitive cu restricții asupra muchiilor orientabile;
- au fost elaborați algoritmi pentru modificarea unei orientări tranzitive prin inversarea unei mulțimi de arce.

**Problema științifică importantă soluționată** *constă în* determinarea unei clase de subgrafuri stabile folosite la studierea problemei de orientare tranzitivă și caracterizarea structurală a grafurilor tranzitiv orientabile, *care au condus la* obținerea unei metode eficiente pentru construirea orientărilor tranzitive și calcularea numărului acestor orientări.

**Importanța teoretică.** Este determinată de obținerea unei caracterizări structurale a grafurilor tranzitiv orientabile. Se propune o metodă eficientă de studiere a orientărilor tranzitive în baza unui șir complet de grafuri factor. Rezultatele obținute pot servi pentru inițierea unor cercetări în domeniu pentru studenți și masteranzii universităților pot servi drept suport pentru unele cursuri opționale pentru soluționarea problemelor practice.

**Valoarea aplicativă a lucrării.** Rezultatele propuse au valoare științifică datorită gradelor lor de noutate și originalitate. Subgrafurile descrise în lucrare și algoritmii propuși au importanță deosebită nu doar din punct de vedere teoretic, dar și aplicativ cum ar fi analiza și optimizarea codului sursă a programelor de calculator, dirijarea sistemului de transport, analiza grafurilor moleculare.

**Aprobarea rezultatelor științifice.** Rezultatele științifice de bază, obținute de către autor și reflectate în prezenta lucrare, au fost prezentate sub formă de:

**a) Teze la conferințe științifice de rang internațional:**

- *The Conference "Mathematics & Information Technologies: Research and Education" (MITRE – 2013), August 18-22, 2013, Chișinău, Republic of Moldova,* (Minimal stable subgraphs in undirected graphs. [48]);
- *Conferința științifică internațională a doctoranzilor „Tendințe contemporane ale dezvoltării științei: viziuni ale tinerilor cercetători.", 10 martie 2015 Chișinău,* (Subgrafurile B-stabile în orientarea tranzitivă a grafurilor. [53]);
- *The Conference "Mathematics & Information Technologies: Research and Education" (MITRE – 2015), July 2-5, 2015, Chișinău, Republic of Moldova,* (Number of transitive orientations in an undirected graph. [49]);
- *The 23rd conference on applied and industrial mathematics (CAIM-2015), September 17-20, 2015, Suceava, Romania.* (Transitive orientation of graph using B-stable subgraphs. [54])

**b) Teze la conferințe științifice de rang național:**

- *Conferința interuniversitară. „Educație prin cercetare – garant al performanței învățământului superior”, Chișinău, 3-4 mai 2012*, (Lanțuri netriangulate și mulțimi stabile într-un graf neorientat. [17]);
- *The 20th conference on applied and industrial mathematics dedicated to academician Mitrofan M. Ciobanu, Chișinău, 22-25 august 2012*, (Stable subgraphs and non-triangulated chains. [51]);
- *Conferința Științifică „Integrare prin Cercetare și Inovare”, Chișinău 10-11 noiembrie 2015,* (Asupra numărului de orientări tranzitive într-un graf. [45])

**c) Comunicări la seminare științifice:**

- *Seminarul de cercetare al grupului „Analiză și Optimizare” din cadrul Facultății de Matematică și Informatică, Universitatea Babeș-Bolyai, Cluj-Napoca, 29 noiembrie 2012,* (Grafuri tranzitiv orientabile.);
- *Seminarul științific „Probleme Actuale de Matematică și Informatică” în cadrul laboratorului “Modelare Matematică și Optimizare” de pe lângă Universitatea de Stat din Moldova, Facultatea de Matematică și Informatică, conducător acad. P. Soltan, 20 martie 2013,* (Grafuri tranzitiv orientabile).
- *Seminarul științific „Probleme Actuale de Matematică și Informatică” în cadrul laboratorului “Modelare Matematică și Optimizare” de pe lângă Universitatea de Stat din Moldova, Facultatea de Matematică și Informatică, conducător acad. P. Soltan, 5 noiembrie 2014,* (B-Stabilitatea și orientarea tranzitivă a grafurilor).
- *Seminarul științific „Probleme Actuale de Matematică și Informatică” în cadrul laboratorului “Modelare Matematică și Optimizare” de pe lângă Universitatea de Stat din Moldova, Facultatea de Matematică și Informatică, conducător acad. P. Soltan, 8 aprilie 2015,* (Subgrafuri B-stabile în orientarea tranzitivă a grafurilor).

**d) Comunicări în cadrul școlilor doctorale de vară:**

- *„Doctoral Intensive Summer School on Evolutionary Computing in Optimisation and Data Mining (ECODAM)”, România, Iași, 17-24 iunie 2012,* (Comunicare: Transitive orientations on undirected graphs.);
- *„Summer School Marktoberdorf 2013” Software Systems Safety, Germany, Marktoberdorf, July 30 to August 11, 2013,* (Comunicare: Algotihmic approach of transitively orientable graphs);

e) **Articole științifice publicate în reviste de specialitate din țară și de peste hotare, precum și în analele (proceedings) ale conferințelor:**

- *Conferința științifică internațională "Modelarea Matematică, Optimizare și Tehnologii Informaționale", Ediția a III-a, Academia de Transporturi, Informatică și Comunicații, Chișinău, 19-23 martie, 2012,* (Subgrafuri stabile într-un graf neorientat [52]);
- *„Doctoral Intensive Summer School on Evolutionary Computing in Optimisation and Data Mining (ECODAM)", România, Iași, 17-24 iunie 2012,* (Transitive orientations on undirected graphs. [18]);
- *Conferința științifică internațională "Modelare* Matematică*, Optimizare și Tehnologii Informaționale", Ediția a IV-a, Academia de Transporturi, Informatică și Comunicații, Chișinău, 25-28 martie 2014,* (Orientarea tranzitivă a grafurilor ce nu conțin subgrafuri stabile minimale [50]);
- *The third conference of mathematical society of the Republic of Moldova: dedicated to the 50th anniversary of the foundation of the Institute of Mathematics and Computer Science - IMCS-50, Chisinau, August 19-23, 2014,* (B-stable subgraphs in undirected graphs [46]);
- *Computer Science Journal of Moldova* (Construction of a transitive orientation using B-stable subgraphs [47]), Vol. 23, Nr. 1(67), 2015;
- *Studia Universitatis Moldaviae, Revistă științifică a Universității de Stat din Moldova, Seria „Științe exacte și economice"* (Clase de Subgrafuri Stabile în Orientarea Tranzitivă a Grafurilor [16]), Nr. 2(82), 2015
- *Studia Universitatis Babeş-Bolyai, Informatica,* (Algorithmic Approach in Reorientation of Comparability Graphs [15]), Vol. LX, Nr. 2, 2015;

În total la tema tezei au fost publicate 14 lucrări științifice dintre care 7 articole(3 în reviste științifice diferite și 4 fără coautori), 7 teze ale comunicărilor la conferințe.

**Sumarul compartimentelor tezei.** Teza este structurată în trei capitole, în care se oferă o caracterizare structurală a grafurilor tranzitiv orientabile, exprimată prin prisma clase de subgrafuri stabile, ce conduc la soluționarea mai multor probleme cu caracter teoretico-aplicativ. Pe lângă cele trei capitole menționate, lucrarea conține concluzii generale și recomandări, introducere, adnotările în limbile română, rusă și engleză, precum și o listă bibliografică ce cuprinde 119 titluri, două anexe și CV-ul autorului.

În introducere, sunt formulate scopul și obiectivele tezei, se argumentează actualitatea temei de cercetare. Se formulează problema științifică cu menționarea importanței teoretice și a valorii aplicative a lucrării. Este dată o analiză succintă a publicațiilor la tema tezei. Se încheie acest compartiment cu o sinteză a conținutului lucrării.

**Primul capitol** al tezei poartă un caracter introductiv și are drept scop examinarea situației în domeniul de studiu cu cercetarea ulterioară a principalelor structuri folosite pentru descrierea grafurilor tranzitiv orientabile. În acest capitol se descriu succint evoluția și metodele de cercetare caracteristice teoriei grafurilor, în mod special a clasei de grafuri tranzitiv orientabile, utile la soluționarea problemelor teoretico-aplicative. Sunt analizate primele rezultate ce țin de studierea grafurilor tranzitiv orientabile obținute de Shevrin L.N. și Filipov N.D. bazate pe mulțimile parțial ordonate [119]. Se apelează în mod special la rezultatele lui Wolk E.S. [109] care descriu orientările tranzitive ale arborilor. Este analizat aportul și altor matematicieni la dezvoltarea direcției menționate de cercetare. Se examinează principalele structuri cu ajutorul cărora sunt caracterizate grafurile tranzitiv orientabile. Aceste structuri reprezintă clase de implicații [41] și subgrafuri stabile [116]. Sunt descrise proprietățile de bază ale subgrafurilor stabile cunoscute în literatura de specialitate care urmează a fi utilizate în lucrare. Proprietățile menționate urmează a fi generalizate în capitolul 2 al tezei. Este argumentată valoarea aplicativă a grafurilor tranzitiv orientabile prin soluționarea unor probleme practice. La sfârșitul capitolului sunt menționate mai multe probleme deschise legate de studierea grafurilor tranzitiv orientabile și soluționate în capitolele ce urmează. Sunt formulate: problema de cercetare, direcțiile de soluționare ale ei, scopul, obiectivele pentru atingerea scopului propus și metodologiile utilizate în teză.

Partea centrală a tezei o constituie **Capitolul al doilea**, în care este descrisă caracterizarea structurală a grafurilor tranzitiv orientabile. Pentru caracterizarea structurală se folosesc rezultate ce țin de studierea proprietăților unor noțiuni importante, precum ar fi subgrafurile B-stabile, grafurile factor, lanțurile netriangulate. Aceste noțiuni stau la baza studierii și caracterizării grafurilor tranzitiv orientabile. Este determinată relația dintre lanț netriangulat și subgraful stabil. În special se demonstrează că mulțimea tuturor vârfurilor unui subgraf stabil minimal poate fi acoperită cu un lanț netriangulat, chestiune importantă pentru cercetările fundamentale menționate. Rolul lanțurilor netriangulate în construirea orientărilor tranzitive ale grafurilor este reflectat prin examinarea condițiilor necesare și suficiente pentru ca un graf să fie tranzitiv orientabil. În calitate de subclasă a grafurilor stabile sunt examinate subgrafurile B-stabile. Este menționat rolul acestora la construirea șirului complet de grafuri factor, care reprezintă momentul central în procesul de construire a tuturor orientărilor tranzitive ale unui graf. Se demonstrează că într-un graf toate

șirurile complete de grafuri factor au aceeași lungime. Această proprietate permite determinarea formulei recurente de calcul a numărului de orientări tranzitive ale grafului neorientat. Sunt examinate diverse metode de construire a orientărilor tranzitive pentru clase speciale de grafuri, cum ar fi: grafurile complete, grafurile neorientate ce nu conțin subgrafuri stabile, etc. Se examinează problema construirii orientării tranzitive a grafului în baza șirului complet de grafuri factor, cu indicarea complexității acesteia.

În cel de-al **treilea Capitol** sunt examinate metode de soluționare a problemelor ce țin de construirea orientării tranzitive cu restricții speciale asupra mulțimii de muchii a grafurilor neorientate. Și anume: construirea orientării tranzitive în condițiile când este predeterminată orientarea unei submulțimi de muchii. Dat fiind faptul că mulțimile parțial ordonate pot fi reprezentate prin grafuri tranzitiv orientabile, acest rezultat poate fi aplicat la sortarea a mulțimilor menționate prin fixarea poziției itemilor. Acest rezultat permite sortarea mulțimilor de arce date în timp eficient, $O(pk\Delta)$, unde $p$ este mulțimea arcelor fixate, $k$ – lungimea șirului complet de grafuri factor, iar $\Delta$ – gradul maximal al grafului. În acest capitol sunt examinați algoritmi de soluționare a problemei de reconstruire a unei orientări tranzitive în dependență de condițiile inițiale impuse asupra arcelor. Este descris algoritmul de reconstruire a orientării tranzitive prin reorientarea unui număr minim de arce precum și algoritmi pentru reconstruirea unei orientări tranzitive în condițiile când se impune inversarea orientărilor unor arce fixate. Revenind la reprezentarea mulțimilor parțial ordonate, schimbarea parțială a orientării în graf semnifică rearanjarea itemilor în mulțime cu păstrarea proprietăților inițiale. Acest fapt indică la aceea că pentru a obține o nouă ordonare parțială a mulțimii este necesar de parcurs doar o parte din itemi. Este cunoscut faptul că mulțimile parțial ordonate pot fi reprezentate cu ajutorul grafurilor tranzitiv orientabile. Prin construirea și reconstruirea orientărilor tranzitive unui graf, se obține de fapt o sortare a mulțimii date, în timpul $O(pk\Delta)$, acest rezultat oferă un avantaj numeric important.

În compartimentul **Concluzii generale și recomandări** sunt expuse concluziile generale ale autorului asupra rezultatelor obținute în cadrul tezei. Sunt, de asemenea, expuse impactul și valoarea elaborărilor acestor rezultate în dezvoltarea domeniului dat. Se prezintă recomandările autorului în formă de sugestii privind cercetările de perspectivă.

În **Anexa 1** este prezentat codul sursă a procedurilor $Potential$ și $SBS$ pentru determinarea unui subgraf B-stabil implementat în limbajul Javascript. Aceste proceduri sunt descrise detaliat în paragraful 2.2.

În **Anexa 2** este prezentat codul sursă implementat în limbajul Javascript al algoritmilor de construire și reconstruire a unei orientări tranzitive prezentate în capitolul 3.

# 1. EVOLUȚIA CERCETĂRILOR ÎN DOMENIUL STUDIERII GRAFURILOR TRANZITIV ORIENTABILE ȘI ASPECTELE APLICATIVE ALE ACESTORA

În primul capitol se prezintă descrierea în ordine cronologică a cercetărilor legate de studierea grafurilor tranzitiv orientabile care își au originea în cercetările matematicienilor Wolk E.S. [109], Ghuila-Houri A. [38], Gilmore P.C. și Hoffman A.J. [39], Shverin L.N. și Filipov N.D. [119]. Sunt descrise rezultatele principale ce țin de studierea diferitor aspecte de cercetare ale grafurilor tranzitiv orientabile în legătură cu necesitatea de soluționare a unor probleme atât de ordin teoretic cât și de ordin practic.

Se analizează aportul diferitor matematicieni la dezvoltarea direcției menționate de cercetare și se accentuează momente importante nesoluționate la moment și utile pentru diverse aplicații. Dezvoltarea direcției de cercetare legată de grafurile tranzitiv orientabile a avut loc datorită folosirii diferitor structuri matematice, fapt menționat în acest capitol.

La sfârșitul capitolului sunt menționate mai multe probleme deschise legate de studierea grafurilor tranzitiv orientabile și soluționate în capitolele următoare.

## 1.1. Rolul grafurilor în soluționarea problemelor teoretico-aplicative

Teoria grafurilor reprezintă un domeniu al matematicii moderne ce oferă metode eficiente pentru soluționarea diverselor probleme teoretico-aplicative. Însăși graful s-a dovedit a fi un model ce descrie adecvat multe procese din diverse domenii de activitate (transport, economie [91], servicii urbane, genetică [111], sisteme electrice [75] etc), ceea ce simplifică esențial soluționarea problemelor prin aplicarea rezultatelor teoretice cunoscute din teoria grafurilor.

**Problema secvenței polimorfismului uninucleotidic**

În biochimia computațională sunt multe situații în care este necesar de a soluționa conflictele dintre catenele de ADN prin excluderea unor secvențe de gene. În asemenea cazuri poate fi construit graful de conflicte, unde fiecare secvență de gene este reprezentată printr-un vârf, iar două vârfuri sunt adiacente dacă între secvențele corespunzătoare există conflict. Problema constă în extragerea a cât mai puține secvențe pentru a rezolva conflictul [70]. În termenii teoriei grafurilor această problemă constă în determinarea acoperii minimale cu vârfuri (în caz general această problemă este NP-completă). Reamintim că, într-un graf $G = (X; U)$ o mulțime de vârfuri $C \subset X$ se numește acoperire de vârfuri, dacă orice muchie din $U$ este incidentă cu cel puțin un vârf din $C$. Polimorfismul uninucleotidic (abreviat din engleză SNP) este o variație în secvența ADN-ului genomic ce interesează un singur nucleotid - A, T, C sau G - observată la doi indivizi diferiți

ai unei specii sau pe cei doi cromozomi omologi ai unui individ. Frecvența acestui tip de polimorfism este de aproximativ o variație pentru 1000 perechi de baze. SNP și variațiile numărului de copii reprezintă tipurile cele mai importante de variație întâlnite în genomul uman. Problema secvenței SNP poate fi definită în modul următor: o secvență SNP este tripletul $(S, F, R)$, unde $S = \{s_1, s_2, \dots, s_n\}$ este o mulțime de $n$ polimorfisme uninucleotidice, $F = \{f_1, f_2, \dots, f_m\}$ este o mulțime din $m$ fragmente și $R$ este relația $R: S \times F \to \{0, A, B\}$ care indică dacă un element din mulțimea de polimorfisme $s_i \in S$ nu poate fi regăsit în fragmentul de genom $f_i \in F$ (indicat prin 0) sau dacă se regăsește, atunci obține valoarea $A$ sau $B$. Două elemente $s_i$ și $s_j$ sunt în conflict dacă există două fragmente $f_k$ și $f_l$ astfel încât exact trei relații din $R(s_i, f_k)$, $R(s_i, f_l)$, $R(s_j, f_k)$, $R(s_j, f_l)$ au același element nenul și exact una are valoarea opusă nenulă. Problema constă în eliminarea a cât mai puține elemente din mulțimea $S$, astfel ca să nu existe conflictele între elementele din $S$. După cum a fost menționat, poate fi construit graful conflictelor pentru secvențele de polimorfisme uninucleotidice, unde două vârfuri sunt adiacente dacă între elementele $s_i$ și $s_j$ există conflict. În baza acestui graf se rezolvă problema de acoperire minimală cu vârfuri.

**Rețelele de telefonie mobilă GSM**

Este cunoscut faptul că rețeaua GSM este celulară, astfel încât toată suprafața de acoperire este divizată în hexagoane. Fiecare celulă hexagonală are un turn de comunicare ce unește toate telefoanele mobile din celulă. Toate telefoanele mobile se conectează la rețeaua GSM căutând celulele din vecinătatea lor. Problema de diferențiere a celulelor în interpretarea teoriei grafurilor se reduce la colorarea grafului ce reprezintă rețeaua GSM: Anume din aceste considerente rețelele GSM operează doar cu patru diapazoane de frecvențe [87]. Conceptul este următorul: având o hartă în plan sau pe o suprafață sferică, conform teoremei celor patru culori [1] regiunile hărții pot fi oricând colorate cu cel mult patru culori astfel încât două regiuni învecinate să nu aibă aceeași culoare. Graful corespunzător hărții poate fi construit prin marcarea fiecărei regiuni cu un vârf iar două vârfuri sunt adiacente dacă regiunile respective sunt învecinate. Coloarea grafului implică și colorarea hărții asociate lui.

**Clasificarea amprentelor digitale**

În ultimii ani identificarea persoanelor cu ajutorul sistemelor automatizate pentru recunoașterea amprentelor digitale are o creștere a importanței în multe aplicații, cum ar fi identificarea răufăcătorilor în urma amprentelor depistate de la locul crimei. Timpul de identificare depinde în mod direct de numărul de amprente ce se conțin în baza de date, deoarece identificarea

se face prin compararea amprentei căutate cu fiecare din cele prezente în baza de date. Deoarece în realitate numărul de amprente în bazele de date reale este foarte mare, procesul de identificare este foarte anevoios. De exemplu, dacă o bază de date conține 200 de milioane de amprente (cum ar fi cazul agenției FBI) și compararea a două amprente durează 0,2 secunde, atunci timpul mediu pentru identificarea persoanei cu amprentele căutate este de aproximativ 246 de zile. Pentru a reduce timpul de căutare este important ca fiecare amprentă să fie clasată în una din cele cinci categorii propuse de E. Henry, astfel limitând căutarea doar în cadrul clasei respective. Având exemplul de mai sus și asumând că toate amprentele sunt uniform distribuite în cele cinci clase, atunci timpul necesar pentru identificare poate fi redus la 49 de zile.

Cu părere de rău, sunt o serie de factori ce complică încadrarea unor amprente într-o anumită clasă. Acești factori pot fi calitatea nesatisfăcătoare a imaginii amprentei, existența amprentelor ambigue care nu pot fi bine clasificate chiar și de experți. În caz particular, problema ambiguității amprentelor apare din cauza variației largi a elementelor dintr-o clasă și a puținelor elemente dintre clase. În unele cazuri amprentele care nu pot fi atribuite unei singure clase chiar și de experți sunt încadrate în toate clasele în care se pare că pot fi încadrate, astfel de amprente se numesc amprente cu referințe încrucișate. Sunt utilizate câteva metode de clasificare a amprentelor, metode structurale și metode bazate pe grafuri. Clasificarea amprentelor bazate pe grafuri poate fi atinsă prin mai multe căi, de la construirea grafurilor relaționale pentru potrivirea inexactă, până la grafurile aciclice pozițional bazate pe rețele neuronale recursive [73].

Una din metodele naturale de reprezentare structurală a partiționării imaginii amprentei digitale este utilizarea grafurilor relaționale. Astfel, fiecare nod corespunde unei regiuni a imaginii amprentei obținută în urma aplicării algoritmului de segmentare, iar două vârfuri sunt adiacente dacă regiunile corespunzătoare sunt învecinate. Pentru compararea grafurilor relaționale se folosește graful pentru corectarea erorii de potrivire. Această metodă generează o valoare a disimilarității dintre graful introdus și un graf prototip. Valoarea obținută se numește distanță de editare. Fie $G_I$ și $G_P$ grafurile introdus și prototip respectiv. Fie $T(G_I, F_P)$ mulțimea tuturor secvențelor operațiilor de editare a grafului $G_I$ pentru a obține $G_P$. Fie $S = (o_1, o_2, \dots, o_n) \in T(G_I, G_P)$ o secvență a operațiilor de editare $(o_1, o_2, \dots, o_n)$. Fie $C(o_i) \in R^+ \cup \{0\}$ o valoare nenegativă numită costul asociat operației $o_i$. Deci, problema de clasificare a amprentelor digitale poate fi redusă la determinarea secvenței de operații $S^*$ cu costul minim. În acest scop se construiește arborele de căutare care conține toate operațiile de editare din mulțimea $T(G_I, F_P)$. Fiecare lanț ce pornește de la rădăcina arborelui reprezintă o secvență a operațiilor de editare. Deci, determinarea secvenței minime $S^*$ se reduce la determinarea celui mai scurt drum. Această problemă poate fi ușor soluționată aplicând algoritmul Dijkstra.

Algoritmica grafurilor de mult timp este parte integrală a vizualizării computerizate. Aplicarea algoritmilor menționați permite segmentarea și gruparea formelor, fapt ce permite abstractizarea eficientă a imaginilor. Aceste abstractizări în mod natural sunt reprezentate cu ajutorul grafurilor, care mai apoi sunt indexate și potrivite pentru păstrarea modelului grafic [34].

Pornind de la primul articol publicat de către L. Euler [35] apărut în anul 1758. Pe parcursul secolelor teoria grafurilor a cunoscut o dezvoltare impunătoare prin apariția unui șir de manuale și monografii care au intrat în patrimoniul de aur al matematicii: C. Berge [9], F. Harary [58], O. Ore [85], N. Christofides [23], C. Marshall [27], R. Wilson [106], A.A. Zâkov [116], etc. Evoluția teoriei grafurilor a fost marcată de rezultatele lui A. F. Mobius, care în 1840 a dat ideea de graf complet și graf bipartit. Conceptul de arbore (graf complet fără cicluri) a fost descris pentru prima dată de către Gustav Kirchhoff în 1845, ale cărui rezultate teoretice au fost aplicate în estimarea curentului în circuite sau rețele electrice. În 1852 Thomas Gutherie a formulat vestita problemă a celor patru culori. Cu toate acestea problema dată a putut fi soluționată abia peste un secol de către Kenneth Appel și Wolfgang Haken. Anume această perioadă din istoria matematicii fiind considerată nașterea teoriei grafurilor. În lucrări strâns legate de domeniul chimiei, Sylvester pentru prima dată, în 1878 introduce noțiunea de graf făcând analogie dintre invarianți cuantici și covarianți din algebră și diagrame moleculare. Ca și bază a multor lucrări în compartimentul ce ține de algoritmica grafurilor au constituit rezultatele lui, Kruskal D, renumit prin construirea algoritmului pentru determinarea învelitoarei convexe din graf, Dijsktra E., matematician ale cărui rezultate au avut un aport deosebit în dezvoltarea teoretică și practică a teoriei grafurilor, în special prin elaborarea vestitului algoritm pentru determinarea celui mai scurt lanț într-un graf.

Trecerea în revistă a rezultatelor remarcabile din teoria grafurilor ar fi incompletă fără evidențierea contribuției aduse teoriei grafurilor de către cercetătorii sovietici și cei din actuala Federație Rusă. În Uniunea Sovietică studierea teoriei grafurilor a început la sfârșitul anilor ’60 ai secolului XX prin lucrările lui Белов В. В. [112], Евстигнеев В.А. [113], Емеличев В.А. [114], Зыков А. А. [116], ș.a. Numeroase rezultate au fost obținute în lucrările Зыков А. А. și discipolii săi ce țin de grafurile Berge, și alte structuri din teoria grafurilor.

De asemenea este important de menționat rezultatele matematicienilor din România, Moldovan G. [80], Roșu A. [89] Toadere T. [99], Tomescu I. [101], ș.a. De remarcat faptul că multe din rezultate în teoria grafurilor au fost obținute prin colaborarea matematicienilor din România și Republica Moldova [100].

În mod special, este necesar de menționat că pe parcursul ultimilor 50 de ani în Republica Moldova s-a format un grup de specialiști care au obținut rezultate științifice importante care au

contribuit în mod esențial la dezvoltarea teoriei grafurilor și în special la dezvoltarea aspectului algoritmic al acestei teorii. Aici, în primul rând este necesar de menționat rezultatele obținute de către academicianul Petru Soltan în legătură cu aplicarea metodei teoriei grafurilor la soluționarea unor probleme aplicative importante. Astfel, au fost puse bazele teoriei d-convexității în grafurile neorientate [21], [117], soluționarea problemelor de amplasare a centrelor de deservire [20], [118]. Rezultate importante au fost obținute și de către discipolii domnului academician P. Soltan prin studierea unor clase de grafuri cu proprietăți metrice speciale (S. Cataranciuc), studierea problemei Weber pe grafuri (A. Poștaru), examinarea mulțimilor stelare în diferite spații metrice (O. Topală). Contribuții la dezvoltarea domeniului teoriei grafurilor au adus și V. Cepoi, A. Niculiță, C. Prisăcaru, D. Zambițchi, ș.a.

## 1.2. Grafurile de comparabilitate a mulțimilor parțial ordonate

Un aport important în dezvoltarea teoriei grafurilor a avut matematicianul francez Claude Berge, printr-o serie de lucrări: [7], [10], ș.a. Printre multitudinea de rezultate pot fi menționate și grafurile perfecte [8]. Grafurile perfecte au avut un impact major în dezvoltarea teoriei grafurilor în ultimii patruzeci de ani. Aceasta a dus la definirea și studierea a multor clase de grafuri. Această clasă de grafuri a fost descrisă ulterior de către alți matematicieni, precum: Chudnovsky M. [24], [25], Golumbic M. C. [40], László L. [71], Зыков A. A. [115] ș.a.

Amintim că un graf se numește perfect dacă numărul cromatic și densitatea coincid pentru orice subgraf indus al grafului [13], [26]. Din anii 1960, anii când a fost introdusă noțiunea de graf perfect, multe clase de grafuri s-au dovedit a fi parte din această clasă. Între timp, cercetările în acest domeniu pot fi divizate în două compartimente. Primul compartiment ținea de demonstrația teoremei grafului perfect, demonstrația conjuncturii grafului tare perfect, studierea grafurilor imperfecte, și alte aspecte ale grafurilor perfecte. Al doilea compartiment ținea de studierea proprietăților matematice și algoritmice a unor clase speciale de grafuri perfecte: grafuri de comparabilitate, numite și grafuri tranzitiv orientabile, grafuri triangulate, grafuri intervale, etc. Multe din aceste grafuri pot fi exprimate destul de natural prin elementele lumii reale. Cum ar fi, optimizarea spațiului în memoria calculatorului, analiza structurilor genetice, sincronizarea proceselor paralele, ș.a. Din categoria grafurilor perfecte fac parte mai multe clase de grafuri:

- Grafurile bipartite
- Graf al muchiilor
- Graf trapezoidal
- Grafuri tranzitiv orientabile
- ș.a. [44]

Grafurile perfecte sunt importante prin aceea că multe probleme care sunt NP- complete pe grafuri arbitrare, cum ar fi colorarea grafului sau determinarea clicii maximale au o complexitate liniară pe grafurile perfecte [61], [66], [74], [79], [105].

După cum a fost menționat mai sus, grafurile tranzitiv orientabile, cunoscute și sub denumirea de grafuri de comparabilitate, au fost studiate din perspectiva clasei grafurilor perfecte. Primele rezultate ce țin de caracterizarea grafurilor tranzitiv orientabile au fost elaborate în [119] de către L. N. Shevrin și N. D. Filipov folosind aparatul algebric. Astfel grafurile tranzitiv orientabile fiind descrise sub formă de structuri parțial ordonate. O ordonare parțială constă dintr-o mulțime $X$ și o relație binară $\prec$ care este tranzitivă (aceasta însemnă că $x \prec y,\ y \prec z$ implică $x \prec z$) [3]. În acest context, E.S. Wolk inițiind ciclul de cercetări în domeniu a examinat grafurile de comparabilitate a mulțimilor parțial ordonate, care sunt arbori [109]. Astfel, au fost formulate condițiile necesare și suficiente pentru ca un graf $G$ care este arbore să fie și graf tranzitiv orientabil. A fost introdusă noțiunea de proprietate diagonală a unui graf $G = (X; U)$, și anume că pentru orice patru elemente $\{x_1, x_2, x_3, x_4\}$ din $X_G$ cu proprietatea $x_1 \prec x_2 \prec x_3 \prec x_4$ implică $x_1 \prec x_3$ sau $x_2 \prec x_4$. Deci, conform rezultatelor lui E.S. Wolk condiția necesară și suficientă pentru ca un arbore să fie tranzitiv orientabil, este ca acest arbore să posede proprietatea diagonală. În continuarea rezultatelor obținute A. Ghouila-Houri are o tentativă reușită de a caracteriza grafurile tranzitiv orientabile finite [38], iar P.C. Gilmore și A.J. Hoffman obțin caracterizarea grafurilor infinite cu proprietatea de comparabilitate. Totuși, fiecare din aceste clase de grafuri au fost caracterizate în cazul unor condiții speciale. Pe de altă parte L. N. Shevrin și N. D. Filipov reușesc să obțină o caracterizare generală a grafurilor tranzitiv orientabile, și anume formulează condițiile necesare și suficiente cu ajutorul noțiunilor de subgraf stabil și lanț netriangulat în modul următor: pentru ca un graf să fie parțial ordonat este necesar și suficient ca orice ciclu netriangulat al său să fie de lungime pară.

În urma studierii mulțimilor parțial ordonate în contextul grafurilor tranzitiv orientabile, a fost observat că pentru multe grafuri de comparabilitate mulțimile asociate pot fi ordonate doar în două moduri. Această proprietate a dus la determinarea clasei de mulțimi unic parțial ordonabile.

Fie $H_0$ un graf cu $n$ vârfuri: $X_{H_0} = \{x_1, x_2, \dots, x_n\}$ și fie $H_1, H_2, \dots, H_n$ $n$ grafuri distincte. Graful compus $H = H_0[H_1, H_2, \dots, H_n]$ este format în modul următor: pentru toți indici $1 \leq i, j \leq n$, înlocuim vârful $x_i$ cu graful $H_i$. Dacă orice vârf din $H_i$ este adiacent cu fiecare vârf al grafului $H_j$, atunci vârfurile $x_i$ și $x_j$ sunt adiacente [59]. În mod formal având grafurile $H_i = (X_i, U_i)$, putem defini graful compus $H = (X, U)$ în modul următor:

$$X = \bigcup_{i \geq 1} X_i ;$$

$$U = \bigcup_{i \geq 1} U_i \cup \{[xy] | x \in X_i, y \in X_j \,\&[x_i x_j] \in U_0\}.$$

Graful $H_0$ se va numi factor extern, iar grafurile $H_1, \dots, H_n$ se vor numi factori interni. [102] Dacă $G = G_0[G_1, \dots, G_n]$ este graf compus din n grafuri distincte $G_i$, atunci G este tranzitiv orientabil dacă și numai dacă $G_i$ unde $1 \leq i \leq n$, sunt tranzitiv orientabile [41]**.** Iar graful G se va numi decompozabil dacă acesta poate fi exprimat sub formă netrivială a unor subgrafuri induse; în caz contrar graful se va numi nedecompozabil.

Evident că orice graf posedă decompozițiile triviale: $G = K_1[G]$ și $G = G[K_1, \dots, K_1]$. Altfel spus, un graf $G = (X, U)$ este decompozabil, dacă există o partiție de submulțimi nevide de vârfuri distincte două câte două $X = X_1 \cup X_2 \cup \dots \cup X_r$, unde $1 < r < |X|$, astfel încât

$$G = G_R[G_{X_1}, G_{X_2}, \dots, G_{X_r}]$$

pentru orice mulțime reprezentativă $R = \{x_1, x_2, \dots, x_r\}, x_i \in X_i$. O astfel de partiție induce o decompoziție proprie a grafului $G$. Din cele de mai sus, poate fi observat că: dacă $G = G_R[G_{X_1}, G_{X_2}, \dots, G_{X_r}]$ este decompoziția proprie a grafului tranzitiv orientabil G, atunci orientarea tranzitivă $\vec{G}$ poate fi definită astfel: $\vec{G} = \overrightarrow{G_R}[\overrightarrow{G_{X_1}}, \overrightarrow{G_{X_2}}, \dots, \overrightarrow{G_{X_r}}]$.

Există o legătură strânsă între decompoziția grafului și lanțurile netriangulate ale acestui graf. Un lanț netriangulat poate fi definit în întregime pe un factor intern sau doar pe muchiile externe ale grafului. Fie $G = G_0[G_1, \dots, G_n]$ este decompoziția a $n$ grafuri distincte $G_i = (X_i, U_i)$, $i = 0,1, \dots, n$. [40] Dacă $U_l$ este mulțimea de muchii a unui lanț netriangulat din graful $G$, atunci doar una din următoarele relații are loc:

1. $U_l \subseteq U_j$, *pentru exact un indice* $j \geq 1$, *sau*
2. $U_l \cap U_j = \emptyset$, *pentru* $\forall j \geq 1$.

Graf tranzitiv orientabil $G$ se va numi graf unic parțial ordonabil, dacă acesta conține exact două orientări tranzitive, una fiind opusă celeilalte. Dacă $G$ este un graf tranzitiv orientabil conex, atunci $G$ este graf unic parțial ordonabil și orice mulțime stabilă proprie din G este o mulțime stabilă interior, iar pentru orice decompoziție proprie a grafului G, orice factor intern este o mulțime stabilă interior (cu alte cuvinte, toate muchiile sunt externe). [103] În contextul celor menționate, un graf $G$ conține mulțimi stabile, dacă și numai dacă este decompozabil. Iar dacă $G$ este un graf nedecompozabil, atunci $G$ este graf unic parțial ordonabil.

În contextul grafurilor unic parțial ordonabile a fost descris comportamentul asimptotic al acestei clase de grafuri, și anume că aproape toate grafurile de comparabilitate sunt unic parțial ordonabile. [78] Fie $G(n)$ numărul grafurilor tranzitiv orientabile, iar $G_{UPO}(n)$ numărul de grafuri unic parțial ordonabile, atunci rezultatul menționat mai sus poate fi descris de următoarea formulă:

$$\lim_{n\to\infty} \frac{G_{UPO}(n)}{G(n)} = 1$$

Multe cercetări au fost efectuate pentru a elabora noi algoritmi de determinare a condițiilor necesare și suficiente pentru ca un graf să fie tranzitiv orientabil, sau determinarea numărului orientărilor tranzitive într-un graf, de asemenea au fost obținute o serie de rezultate pentru a construi o orientare tranzitivă în timp polinomial. Această direcție de cercetare a fost dezvoltată intensiv în ultimii 20 de ani de către Berge C, Ghouila-Houri, A., Golumbic M.C., McConnell, R., Mohring, R., Olariu, S. ș.a. în lucrările, [77], [86], [29], [65], [60], [84], [110], [33]. În Republica Moldova rezultate importante în studierea grafurilor tranzitiv orientabile au obținut Зыков А. А. [116], [115] și Cataranciuc S. [19], [14]

La momentul actual sunt cunoscuți doi algoritmi care soluționează problema unui graf tranzitiv orientabil. Primul algoritm fiind descris de Golumbic M.C. [40], care calculează $G-$ decompoziția în mod recursiv. În așa mod se formează o partiție a grafului în așa numitele clase de implicații care determină o orientare tranzitivă. Acest algoritm soluționează problema în timpul $O(n \cdot m)$. Orientarea finală este tranzitivă dacă algoritmul finisează procedura de $G-$ decompoziție cu succes. Al doilea algoritm care rulează în timpul $O(n^2)$ a fost descris de Spinrad [97]. Acest algoritm construiește o orientare tranzitivă, dacă o asemenea orientare în genere există.

Există o diferență fundamentală dintre acești algoritmi. Primul algoritm determină o orientare tranzitivă atunci când pentru orice muchie poate fi aplicată procedura de $G-$ decompoziție, pe când algoritmul al doilea în orice situație returnează o orientare a unui graf. Această orientare este tranzitivă, dacă în genere există o astfel de orientare în acest graf, dar acest lucru trebuie verificat în mod separat.

### 1.3. Clase de implicații și orientarea tranzitivă a grafurilor

Datorită aplicabilității grafurilor tranzitiv orientabile acestea se întâlnesc în lucrările mai multor matematicieni [29], [33], [65], [77], [84], [86], [115] etc. Prezintă un oarecare interes rezultatele recente obținute de către M.C. Golumbic în legătură cu necesitatea soluționării unor probleme practice, pe care acesta le menționează în lucrarea [40], și care la rândul său au contribuit la dezvoltarea aspectului teoretic al acestei direcții de cercetare. În cercetările lui Golumbic se

pune accent pe niște structuri speciale numite multiplexe. Pentru a formula rezultatele respective sunt necesare niște noțiuni speciale, unele dintre care sunt cunoscute în literatura de specialitate, iar altele sunt introduse de către Golumbic.

Relația Γ pe mulțimea de muchii ale grafului $G = (X; U)$ este definită în modul următor: $[xy]\Gamma[x'y']$ , unde $x = x'$ și $yy' \notin U$, sau $y = y'$ și $xx' \notin U$. Vom spune că muchia $[xy]$ forțează direct muchia $[x'y']$. Închiderea reflexiv tranzitivă $\Gamma^*$ a relației Γ formează o clasă de echivalență pe mulțimea de muchii $U$, deci, împarte mulțimea $U$ în partiții numite clase de implicații ale grafului $G$. Muchiile $[xy]$ și $[zw]$ fac parte din aceeași clasă de implicații, dacă și numai dacă există o secvență de muchii $[xy] = [x_0y_0]\Gamma[x_1y_1]\Gamma \dots \Gamma[x_ky_k] = [zw]$, unde $k \geq 0$. Această secvență este numită **lanț-Γ** de la muchia $[xy]$ până la $[zw]$. Fie $\mathcal{F}(G)$ o mulțime de clase de implicații din $G$. Vom defini $\hat{\mathcal{F}}(G) = \{\hat{A} | A \in \mathcal{F}(G)\}$, unde $\hat{A} = A \cup A^{-1}$, este închiderea simetrică a mulțimii $A$. Elementele mulțimii $\hat{\mathcal{F}}(G)$ sunt numite clase de culori ale grafului $G$.

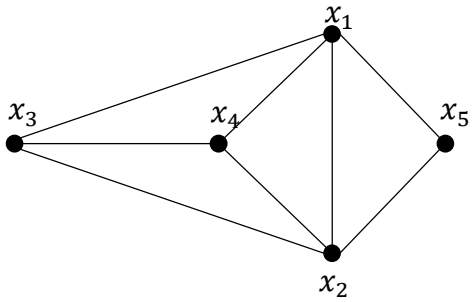



Fig. 1.1. *Clasele de implicații*

**Exemplu 1.1.** Graful din figura 1.1 conține opt clase de implicații:

$A_1 = \{[x_1, x_2]\}$, $A_2 = \{[x_3, x_4]\}$, $A_3 = \{[x_1, x_3], [x_1, x_4], [x_1, x_5]\}$,

$A_4 = \{[x_2, x_3], [x_2, x_4], [x_2, x_5]\}$.

$A_1^{-1} = \{[x_2, x_1]\}$, $A_2^{-1} = \{[x_4, x_3]\}$, $A_3^{-1} = \{[x_3, x_1], [x_4, x_1], [x_5, x_1]\}$,

$A_4^{-1} = \{[x_3, x_2], [x_4, x_2], [x_5, x_2]\}$.

În cazul când G posedă orientarea tranzitivă $\vec{G}$, iar $A$ este o clasă de implicații, atunci una din relațiile următoare este adevărată: $\vec{G} \cap \hat{A} = A$ sau $\vec{G} \cap \hat{A} = A^{-1}$; dar în ambele cazuri $A \cap \hat{A} = \emptyset$. Însă nu orice reuniune arbitrară a claselor de implicații $\vec{G} = \bigcup_i A_i$, ce satisface condițiilor: $\vec{G} \cap \overleftarrow{G} = \emptyset$ și $\vec{G} \cap \overleftarrow{G} = U_G$, formează o orientare tranzitivă. *Dacă A este o clasă de implicații a unui graf neorientat $G = (X; U)$, atunci are loc una din relațiile:*

1. $A = A^{-1} = \hat{A}$;
2. $A \cap A^{-1} = \emptyset$, $A$ și $A^{-1}$ generează orientări tranzitive, și sunt unicele orientări tranzitive ale închiderii $\hat{A}$.

Precum s-a adeverit, studierea grafurilor tranzitiv orientabile a condus la obținerea unor rezultate neașteptate, la prima vedere. Astfel, a fost obținută legătura dintre aceasta cu problema colorării unui graf neorientat. S-a demonstrat [94] că orice clasă de culori a unui graf neorientat $G$ are exact două orientări tranzitive, una fiind opusa celeilalte, sau nu conține nici o orientare tranzitivă. Dacă în graful $G$ există o clasă de culori ce nu conține nici o orientare tranzitivă, atunci graful nu mai este tranzitiv orientabil.

Vom menționa că în lucrările [38], [119], [66] ușor putem deduce legătura dintre problema studiată în prezenta lucrare și problema de ordonare liniară a elementelor unei mulțimi.

**Exemplu 1.2.** O orientare tranzitivă a unui graf ordonează parțial toate vârfurile acestuia. Considerând o orientare tranzitivă a unui graf complet $K_n$ cu $n$ vârfuri, obținem că fiecare pereche de vârfuri din $K_n$ este comparabilă, deci această ordonare parțială este de fapt o ordonare liniară (ordonare totală) a vârfurilor grafului. În schimb, orice ordonare liniară a vârfurilor grafului $K_n$ produce o orientare tranzitivă prin orientarea muchiilor de la vârfurile cu indice mai mic, către vârfuri cu indice mai mare. Astfel, $\tau(K_n) =$ numărul de ordonări liniare a $n+1$ elemente $= n!$.

Precum s-a menționat, M.C. Golumbic a reușit să obțină rezultate interesante legate de orientarea tranzitivă a grafurilor definind noțiunea de clasă de implicații și multiplexe. În baza proprietăților multiplexului au fost obținute rezultate ce conduc într-un final la stabilirea formulei de calcul a orientărilor tranzitive într-un graf, care însă, spre regret, sunt greu de aplicat în practică și nu pot conduce la obținerea unui algoritm polinomial.

Fie $G = (X; U)$ este un graf neorientat. Un subgraf complet $S = (X_S; U_S)$, unde $|X_S| = r + 1$, se va numi simplex de ordinul $r$ dacă fiecare muchie $[x, y] \in X_S$ aparține unei clase de culori diferite din $G$. De exemplu, orice muchie din $G$ reprezintă un simplex de ordinul 1. Un simplex va fi maximal, dacă acesta nu se conține într-un simplex mai mare.

Un multiplex definit de simplexul $S$ de ordinul $r$, poate fi definit ca subgraful parțial $M = (X_M; U_M)$, unde $U_M = \{[x, y] \in U | [x, y]\Gamma^*[a, b]$ pentru o anumită muchie $[a, b] \in U_S\}$, sau notarea alternativă $U_M = \cup \hat{A}$, unde $\cup \hat{A}$ este reuniunea tuturor claselor de culori $\hat{A} \in \hat{\mathcal{F}}(G)$ ce satisface condiția că $\hat{A} \cap S \neq \emptyset$. Astfel $M$ este reuniunea a $\frac{1}{2}r(r+1)$ clase de culori reprezentate de muchiile simplexului $S$. Un multiplex este maximal dacă acesta nu se conține într-un multiplex mai mare al grafului $G$.

Un izomorfism între simplexele $S_1 = (X_{S_1}; U_{S_1})$ și $S_2 = (X_{S_2}; U_{X_2})$ al unui graf neorientat este o bijecție $f: X_{S_1} \longrightarrow X_{S_2}$ astfel încât $[xy]\Gamma^*[f(x)F(y)]$ pentru orice pereche de vârfuri distincte din $x, y \in X_{S_1}$. Dacă $T = (X_T; U_T)$ este un simplex care generează un multiplex $M$, iar $S = (X_S; U_S)$ un alt simplex care se conține în $M$. Atunci S este izomorf cu subsimplexul $T$. Simplexele care generează același multiplex sunt izomorfe. Fie $S = (X_S; U_S)$ un multiplex al unui graf neorientat $G = (X; U)$ care generează un multiplex $M = (X_M; U_M)$. Următoarea lemă arată cum poate fi construit un simplex. Dacă $G$ conține triunghi tricolor cu vârfurile $x, y, z$ astfel încât $[xy] \notin U_M$ dar muchia $[yz] \in U_M$, atunci putem adăuga vârful $x$ simplexului $S$ astfel încât să obținem un simplex nou $T = (X_T; U_T)$ care conține pe $S$, unde*:*

$$X_T = X_S \cup \{x\};$$

$$U_T = U_S \cup \left\{\widehat{[xt]} \middle| t \in X_S\right\}.$$

Dacă $S$ este un simplex ce se conține în multiplexul $M$, atunci există un simplex $S_M$ care generează $M$ astfel încât $S \subseteq S_M$. În consecință d*acă* $M_1$ *și* $M_2$ sunt multiplexe astfel încât $M_1 \subseteq M_2$, atunci se verifică relațiile*:*

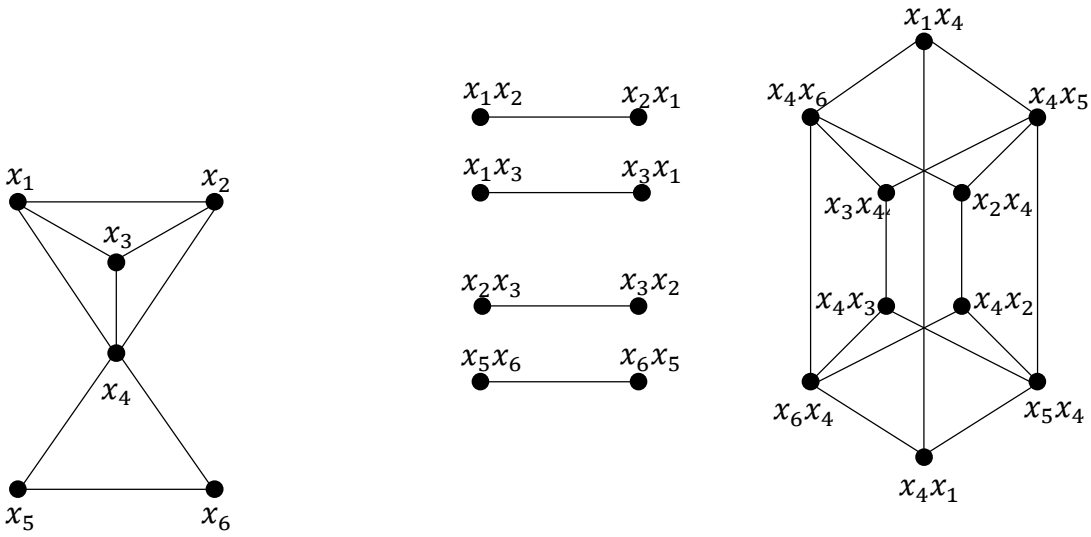
1. *Orice simplex care generează* $M_1$ *se conține în simplexul care generează* $M_2$*;*
2. *Orice simplex care generează* $M_2$ *conține un subsimplex care generează* $M_2$*.*

Relația dintre un multiplex maximal $M$ și un simplex maximal $S$ este că $M$ *e*ste multiplex maximal, dacă și numai dacă $S$ este simplex maximal. În așa caz poate fi construit multiplexul maximal prin căutarea locală pe muchii. Se alege o muchie în mod aleatoriu și se construiesc simplexe mai mari care conțin predecesorul, până când simplexul obținut va fi maximal. Acesta la rândul său va genera un multiplex maximal. Mulțimea de simplexe pe de altă parte formează o partiție pe mulțimea de muchii a grafului $G$. Dacă $M_1$ și $M_2$ sunt multiplexe maximale ale unui graf neorientat $G$, atunci, sau $M_1 \cap M_2 = \emptyset$, sau $M_1 = M_2$. Legătura dintre clasele de implicații și multiplex este că: dacă $A$ este o clasă de implicații a unui graf neorientat $G = (X; U)$, astfel încât $A = \hat{A}$, atunci $A$ este un multiplex maximal de ordinul 1 [95].

Un simplex de ordinul $r$ are $(r + 1)!$ orientări tranzitive, așa cum a fost prezentat la începutul paragrafului curent. Mai mult, orientarea tranzitivă a unui simplex extinde în mod unic orientarea tranzitivă a multiplexului generat de acesta, cu excepția cazului când multiplexul însăși formează o clasă de implicații și astfel nu este tranzitiv orientabil. Invers, orientarea tranzitivă a unui multiplex restricționează la o orientare tranzitivă unică a oricărui simplex care se conține în el.

Dacă $M$ este un multiplex de ordinul $r$ și este tranzitiv orientabil, atunci $t(M) = (r + 1)!$.

Partiția unui graf neorientat $G = (X; U)$ în multiplexe maximale $U = M_1 + M_2 + \cdots + M_k$ se va referi la $M-$decompoziție. Aceasta este unică, până la ordinul multiplexului $M_i$. După investigarea orientării tranzitive a multiplexelor, putem trece la determinarea orientării tranzitive a tuturor muchiilor din $U$.




$G$ $\qquad\qquad$ $G_{qtd}$

Fig. 1.2. *Graful tranzitiv orientabil G împreună cu graful bipartit $G_{qtd}$ asociat lui.*

Următorul rezultat arată corespondența biunivocă dintre orientările tranzitive ale multiplexelor $M_i$ și ale muchiilor din $U_G$. Dacă $G = (X; U)$ este un graf neorientat, și $U = M_1 + M_2 + \cdots + M_k$, unde $M_i$ este multiplex maximal al mulțimii de muchii $U_G$, atunci următoarele afirmații sunt echivalente:

1. Dacă $\overrightarrow{U_G}$ este orientarea tranzitivă a grafului $G$, atunci $\overrightarrow{U_G} \cap M_i$ este orientarea tranzitivă a multiplexului $M_i$;
2. Dacă $\overrightarrow{U_{M_1}}, \overrightarrow{U_{M_2}}, \dots, \overrightarrow{U_{M_k}}$ sunt orientări tranzitive ale multiplexelor $M_1, M_2, \dots, M_k$ respectiv, atunci $\overrightarrow{U_{M_1}} + \overrightarrow{U_{M_2}} + \cdots + \overrightarrow{U_{M_k}}$ este orientarea tranzitivă a lui $G$;
3. $t(G) = t(M_1)t(M_2) \dots t(M_k)$;
4. Dacă $G$ este un graf tranzitiv orientabil și $r_i$ este ordinul multiplexului $M_i$, atunci numărul de orientări tranzitive al grafului $G$ este $t(G) = \prod_{i=1}^{k}(r+1)!$.

Fie $G = (X; U)$ un graf neorientat. Din graful $G$ poate fi construit graful $G_{qtd}$ după următoarea procedură: $X_{G_{qtd}} = \bigcup_{[uv]\in U_G}\{x_{uv}, x_{vu}\}$ și a) există o muchie între vârfurile $x_{uv}$ și $x_{wz}$ în caz că $w = v$ și $[uz] \notin U_G$, sau $u = z$ și $[vw] \notin U_G$, b) există o muchie $[x_{uv}, x_{vu}]$ în $U_{G_{qtd}}$

pentru orice muchie $[uv] \in U_G$; c) dacă în $G_{qtd}$ există muchia $[x_{uv}, x_{vw}]$, atunci și muchia $[x_{wv}, x_{vu}]$ aparține mulțimii $U_{G_{qtd}}$.

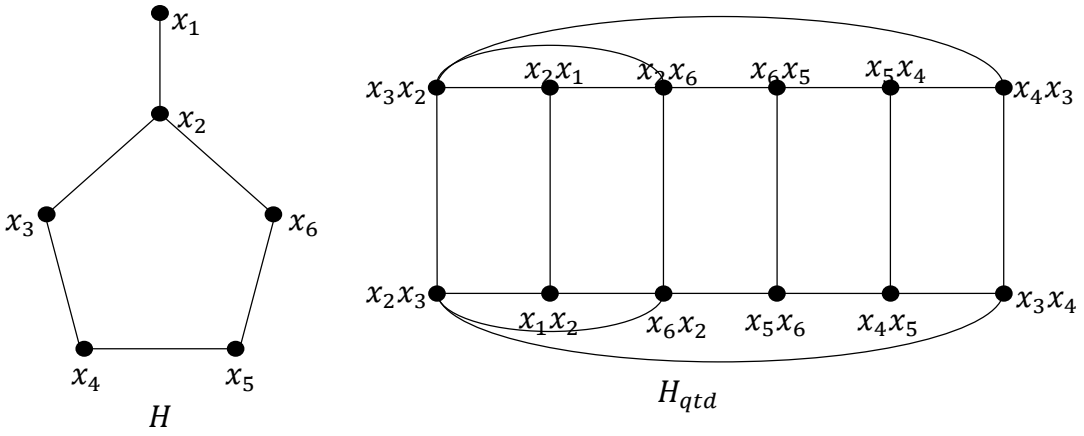



Fig. 1.3. *Graf ce nu este tranzitiv orientabil $H$ împreună cu graful $H_{qtd}$ asociat lui, unde $H_{qtd}$ nu este bipartit.*

Folosind procedura descrisă mai sus de construire a grafului special $G_{qtd}$, ce va mai înainte, mai exact în anul 1962, matematicianul Ghouila-Houri Alain încercând să studieze unele aspecte ale grafurilor tranzitiv orientabile a obținut un rezultat prin care se spune că un graf arbitrar $G$ este tranzitiv orientabil dacă și numai dacă$G_{qtd}$ este bipartit (a se vedea lucrarea [38]). Acest rezultat poat fi privit ca o caracterizare a grafurilor tranzitiv orientabile care practic nu se întâlnește în lucrările apărute mai târziu ale altor matematicieni [29], [33], [39], [42] datorită faptului că problema orientării tranzitive se reduce la o problemă nu mai simple ce constă în verificarea dacă un graf este bipartit. Din aceste considerente cercetările în domeniul respectiv au continuat în lucrările matematicienilor Pinar Heggernes, Federico Mancini, Charis Papadopoulos [60], Michael Jean [65], Rolf H Möhring [77]. Totuși rezultatul obținut de Ghouila-Houri a permis ulterior soluționarea problemei de recunoaștere dacă un graf neorientat $G$ este tranzitiv orientabil. În mod special, aceasta se examinează în lucrarea  lui Golumbic [41]. Din rezultatele respective ușor se deduce un algoritm care în timp $O(\Delta m)$ determină dacă graful $G$ este tranzitiv orientabil ($m$ este numărul de muchii ale grafului $G$, iar $\Delta$ – gradul maxim al vârfurilor din $G$). Rezultatul merită atenție deoarece, evident, nu orice graf este tranzitiv orientabil. Drept exemplu putem aduce graful $H$ din figura 1.3. Precum ușor se verică $H_{qtd}$ nu este bipartit și prin urmare graful inițial $H$ nu este tranzitiv orientabil.

## 1.4. Aplicații ale grafurilor tranzitiv orientabile

După cum a fost menționat în secțiunile precedente grafurile tranzitiv orientabile reprezintă o subclasă a grafurilor perfecte și au proprietăți unice. Dacă un graf aparține clasei de grafuri

tranzitiv orientabile, atunci o serie de probleme teoretice (colorarea grafurilor, determinarea clicii maximale ponderate, determinarea numărului de stabilitate internă) pot fi soluționate în timp polinomial. Aceste probleme la rândul lor pot duce la soluționarea unor probleme practice importante. De asemenea, cu ajutorul modelelor grafurilor tranzitiv orientabile pot fi soluționate multe probleme practice, cum ar fi:

**Decompoziția rețelelor Petri**

O rețea Petri este o formă populară a reprezentării grafice și specifice a unui sistem concurent. Aceasta este efectivă în analiza, proiectarea, verificarea logicii binare a controllerelor. Un pas important în proiectarea și analiza unui controller este decompoziția acestuia. O rețea Petri inițială este împărțită în module paralele separate (SMC). Fiecare componentă reprezintă un sub-proces independent care ulterior este marcat secvențial. Cu părere de rău, întregul proces de decompoziție al unei rețele Petri necesită un timp exponențial. Aplicarea grafurilor tranzitiv orientabile în acest proces, poate reduce timpul de decompoziție al rețelei până la $O(n^3)$.

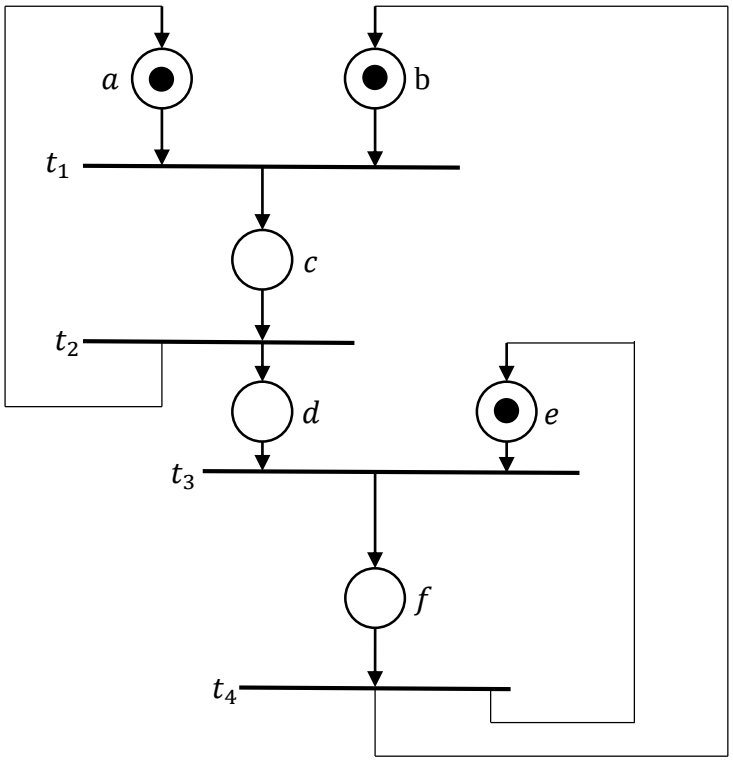



Fig. 1.4. *Rețea Petri* $PN_1$

Reamintim că o rețea Petri este un cuadruplu de forma $PN = (P, T, F, M_0)$, unde $P$ este o mulțime finită de poziții, $T$ – o mulțime finită de tranziții, $F \subseteq (P \times T) \cup (T \times P)$ – o mulțime de arce, $M_0$ – marcajul inițial. Starea a unei rețele Petri se numește marcaj dacă aceasta poate fi reprezentată ca distribuție a jetoanelor în rețeaua de poziții. Dacă o poziție conține unul sau mai

multe jetoane, atunci aceasta se numește poziție marcată. Un marcaj poate fi schimbat prin execuția unei tranziții. O tranziție $t$ poate fi executată, dacă orice poziție de intrare a acesteia $p$: $(p, t) \in F$ conține un jeton. Execuția unei tranziții are loc prin extragerea unui jeton din poziția inițială și adăugarea acestuia în poziția finală $p'$: $(t, p') \in F$. Rețeaua extinsă de liberă alegere (EFC) este o rețea Petri pentru care orice două tranziții au o poziție de intrare comună, și au mulțimi egale de poziții de intrare: $\forall p \in P$: $(t_1, t_2 \in T, p \in \bullet t_1, p \in \bullet t_2) \Longrightarrow (\bullet t_1 = \bullet t_2)$. O componentă a stării de mașină (SMC) a unei rețele Petri $PN$ poate fi definită ca o subrețea $PN'$ generată de pozițiile din $PN$ astfel încât toate tranzițiile de intrare și ieșire din $PN'$ și arcele de conexiune aparțin $PN'$, și fiecare tranziție a subrețelei are un arc de intrare și un arc de ieșire. Mai mult $PN'$ conține exact un singur jeton în marcajul inițial. După cum a fost menționat rețelele Petri pot fi reprezentate prin graful de concurență. Un graf de concurență $G_C = (X; U)$ al unei rețele Petri $PN$ este un graf neorientat care reprezintă relația structurală de concurență a rețelei. Mulțimea de vârfuri $X$ corespunde pozițiilor din $PN$. Două vârfuri sunt adiacente, dacă pozițiile corespunzătoare sunt marcate simultan cu unul din marcajele rețelei.

Metoda pentru decompoziția rețelei EFC poate fi descrisă în următorii pași:

1. Determinarea grafului concurențial $G_C$ pentru o rețea Petri inițială $PN$. Această structură reprezintă relația structural concurențială a rețelei. Pentru $n$ poziții ale rețelei EFC, determinarea grafului $G_C$ se efectuează în timpul $O(n^3)$ [69]. Pentru rețeaua $PN_1$ din figura 1.4 graful de comparabilitate $G_C$ din figura 1.5 conține șase vârfuri care se referă la pozițiile rețelei și șapte muchii care reprezintă relațiile structurale de concurență.

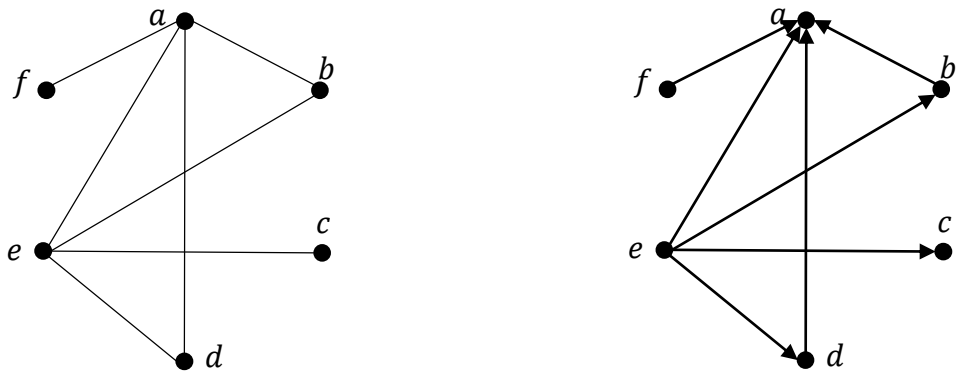
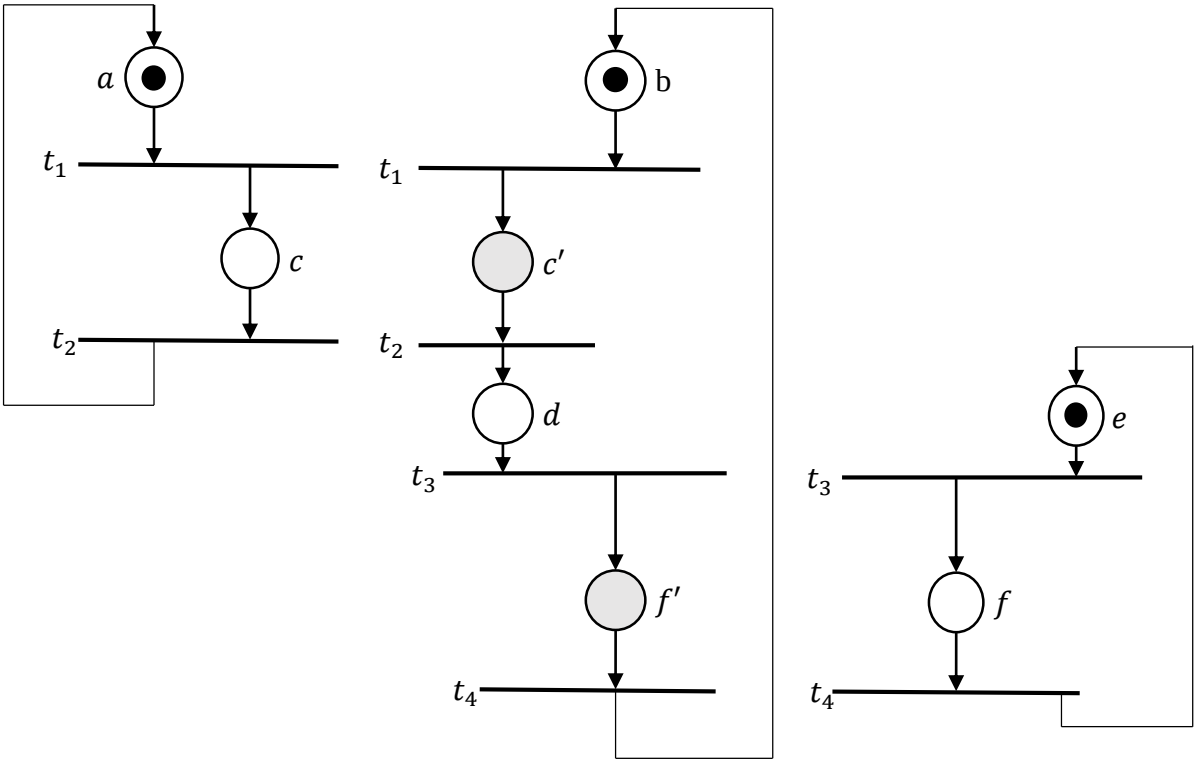



Fig. 1.5. *Graful de concurență a rețelei $PN_1$ și orientarea tranzitivă a sa.*

2. Recunoașterea dacă graful de concurență face parte din clasa grafurilor tranzitiv orientabile este punctul cheie a metodei. Dacă graful de concurență a rețelei Petri este și graf tranzitiv orientabil atunci acesta poate fi descompus cu aplicarea metodei menționate, în caz contrar sunt aleși alți algoritmi pentru descompunerea rețelei. Recunoașterea grafului tranzitiv orientabil într-

un graf de concurență poate fi efectuată în timpul $O(n^2)$, unde $n$ este numărul de poziții în rețea [41]. Dacă graful de concurență este și tranzitiv orientabil atunci poate fi aplicată metoda de colorare a grafurilor, unde soluția optimală poate fi obținută în timp polinomial.

3. Fiecare culoare a grafului de concurență reprezintă la un singur SMC [94]. Dacă un modul SMC nu conține o poziție specifică, atunci este adăugată o poziție neoperațională. Având exemplul din figura 1.4 graful de concurență a rețelei $PN_1$ poate fi colorat cu 3 culori: $S_1 = \{a, c\}$, $S_2 = \{b, d\}$, $S_3 = \{e, f\}$. Deoarece modulul $S_2$ nu formează o stare de mașină adecvată, atunci sunt adăugate două poziții neoperaționale $c', f'$. Rețeaua descompusă în modulele SMC $S_1 = \{a, c\}$, $S_2 = \{b, c', d, f'\}$, $S_3 = \{e, f\}$ este prezentată în figura 1.6.




4. Fig. 1.6. *Descompunerea rețelei Petri* $PN_1$

Deoarece fiecare pas în parte poate fi executat în timp polinomial, prin urmare, întregul proces de descompunere poate fi efectuat în timp polinomial. Deoarece cel mai costisitor pas este descompunerea grafului de concurență $O(n^3)$, rezultă că timpul necesar pentru descompunerea rețelei Petri este de $O(n^3)$. Deoarece mai mult de 94% din grafurile concurențiale ce caracterizează rețelele EFC sunt grafuri tranzitiv orientabile, rezultă că majoritatea rețelelor Petri de tipul EFC pot fi optimal descompuse în timp polinomial.

**Gestionarea fișierelor de înregistrare pentru procesoarele stream**

Pentru accesarea rapidă a informației din memoria internă a calculatorului, procesoarele moderne utilizează memoria cache a procesorului. Această metodă s-a dovedit a fi eficientă în

dezvoltarea performanței lucrului calculatorului, totuși există o serie de probleme care nu pot fi soluționate prin această metodă. Complexitatea arhitecturii memoriei cache duce la supraîncălzirea procesorului și mărirea suprafeței acestuia. Latența nedefinită de accesare a memoriei cache nu garantează performanța dorită în timp real.

Spre deosebire de memoria cache a procesului, memoria administrată pe chip are avantaje în suprafață, preț, și viteza de accesare [2]. Această metodă este pe larg răspândită în sisteme încorporate, arhitecturi stream (cunoscute ca și fișiere de înregistrare a fluxului sau memorie locală), procesoare a cartelelor video. Cât ține de construcția super calculatoarelor memoria administrată pe chip este foarte des utilizată, în special în accelerarea procesoarelor. Astfel de metode sunt folosite la dezvoltarea arhitecturilor Merrimac [31], Cyclops64 [30], Grape-DR [72], Roadrunner și super calculatoarele din seria Tianhe printre care și Tianhe-2 (cel mai rapid super calculator potrivit site-ului www.top500.org, lista realizată în iunie 2015).

Procesoarele stream reprezintă o alternativă promițătoare procesoarelor tradiționale pentru a atinge performanțe înalte în aplicații stream. În modelul de programare stream pentru procesoare un program este descompus într-o secvență de nuclee ce operează cu fluxuri de date. În timpul execuției unui nucleu pe un procesor stream toate fluxurile accesate sunt conectate prin memorie internă a procesorului, numită fișier de înregistrare a fluxului (SRF). Optimizarea administrării memoriei este crucială în dezvoltarea performanței înalte a aplicațiilor. Este cunoscut că fluxurile în aplicațiile stream pot fi reprezentate prin grafuri de interferență. În aplicațiile stream grafurile de interferență au proprietate de tranzitivitate, fie că sunt grafuri tranzitiv orientabile, fie că pot fi descompuse în mulțimi de subgrafuri tranzitiv orientabile. Această proprietate a grafurilor de interferență în aplicațiile stream permite optimizarea alocării memoriei în SRF.

Procesoarele stream, așa ca Imagine, Raw, Cell, Merrimac și cele grafice reprezintă o alternativă promițătoare ce ține de atingerea performanțelor mari în aplicații media. În plus programarea stream poate fi ușor implementată în rezultate de cercetare științifică. Tehnologia compilatoarelor pentru limbajele stream și arhitecturile respective sunt încă la nivel incipient. Fișierele pentru înregistrarea fluxului sunt introduse pentru a capta răspândirea producerii și răspândirii locale, precum și pentru reducerea traficului de memorie din afara procesului. Spre deosebire de registrele convenționale, fișierele pentru înregistrarea fluxului nu pot fi ocolite, cu alte cuvinte, datele de intrare și ieșire din nucleu sunt păstrate în fișierele SRF atunci când nucleul este în proces de execuție. Dacă informația cu care operează nucleul este prea mare pentru a memora SRF, atunci se aplică procedura de segmentare a datelor astfel încât nucleul să fie solicitat pentru fiecare segment doar o singură dată. O metodă alternativă de administrare a datelor de prelucrare a procesorului este de a tampona o parte din fluxuri sau divizarea fluxurilor până când

capacitatea SRF depășește mărimea datelor. În prezent datele sunt prelucrate prin intermediul SRF în dependență de ordinea pătrunderii informației, această metodă nu este optimală pentru volum mare de date.

După cum a fost menționat mai sus fluxurile de date ce sunt prelucrate de o secvență de nuclee pot fi prezentate prin grafuri ponderate de interferență. Observația de bază este că în multe aplicații media grafurile de interferență formează grafuri tranzitiv orientabile. Procesul de compilare a unei aplicații poate fi prezentat prin colorarea grafului de interferență. După cum însă este cunoscut, grafurile tranzitiv orientabile pot fi colorate în timp polinomial.

Modelul de programare stream este de tipul *StreamC/KernelC*. Ideea centrală a acestei paradigme este de a diviza aplicația în nuclee și fluxuri. Ca rezultat aplicația este împărțită în două programe, programul stream ce rulează pe unitatea centrală, iar programul nucleu rulează pe procesorul stream. Programul stream are rolul de a specifica cursul fluxurilor între nuclee și inițiază execuția acestora. Programul nucleu execută aceste nuclee pe rând. Un program stream constă dintr-o secvență de cicluri astfel încât fiecare ciclu include o secvență de nuclee ce operează pe fluxuri. În compilatoarele stream toate ciclurile sunt separate în procesul de alocare de memorie în SRF. Alocarea fluxurilor de memorie pe ciclu în fișierele SRF este o reprezentare clasică a grafurilor de interferență. Toate fluxurile accesate pe ciclu sunt numite diapazoane ce urmează a fi plasate în SRF. Dacă două diapazoane se suprapun, atunci pentru ele trebuie de alocat unități de memorie separată în SRF. După ce toate diapazoanele au fost definite, poate fi construit graful de interferență, unde vârfurile reprezintă diapazoanele, iar mărimea acestora este ponderea lor, iar două vârfuri ale grafului sunt adiacente dacă diapazoanele se suprapun. După cum a fost menționat anterior, procesul de alocare a memorie în SRF poate fi reprezentat prin colorarea grafului de interferență. În cele mai multe cazuri s-a dovedit că aceste grafuri de interferență sunt și grafuri tranzitiv orientabile. Astfel, este cunoscut faptul că, colorarea unui graf tranzitiv orientabil poate fi efectuată în timp polinomial. Problema optimizării de alocare a memoriei în SRF ține de minimizarea spațiului total ocupat de fluxuri. Deci, o colorare corectă a grafului de interferență asociat modelului oferă o alocare optimă de memorie în SRF.

Revenind la modelul de lucru al proceselor stream, un ciclu reprezintă un program ce constă dintr-o serie de nuclee, fiecare din ele produce fluxuri intermediare care sunt consumate de următoarea secvență de nuclee. Astfel, toate diapazoanele din fiecare flux nu acoperă mai mult de două nuclee. În așa caz graful de interferență este graf tranzitiv orientabil. De exemplu, în figura 1.7 este prezentat graful de interferență pentru o serie de trei nuclee, unde nici un domeniu de memorie nu necesită mai mult de două nuclee pentru accesare.

Vârful $x_2$ ce reprezintă un flux, este parte din nucleul 1 și 2. Vârfurile $x_6, x_7, x_8$ sunt active

pentru nucleul 2 și 3. Celelalte vârfuri sunt active doar pentru nucleele din care fac parte.

```
Load(..., ,x1);
Kernel('1', x1, x2);
Load(...,x3);
Load(...,x4);
Load(...,x5);
Kernel('2', x2, x3, x4, x5, x6, x7, x8);
Load(...,x9);
Kernel('3', x6, x7, x8, x9, x10);
```

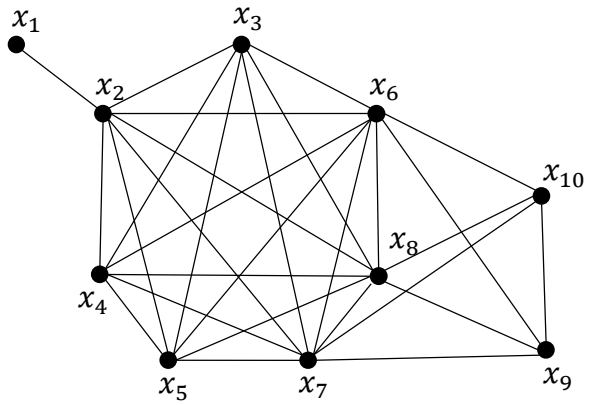



Fig. 1.7. *Program stream și graful de interferență asociat*

În baza exemplului menționat poate fi formulată problema de optimizare. Fie $G_{CG}$ graful de interferență ce conține $N_{CG}$ nuclee, astfel încât fiecare domeniu activ din $G_{CG}$ nu este parte componentă a mai mult de două nuclee. Prin construirea partiției de domenii active pot fi formate $2N_{CG}$ mulțimi:

$$K_1, K_{12}, K_2, K_{23}, K_3, \dots, K_{(N_{CG}-1)N_{CG}}, K_{N_{CG}}, K_{N_{CG}1} \tag{1.1}$$

Unde, $K_i$ constă din fluxurile accesate dar care sunt active doar în nucleul $i$, iar $K_{i(i\oplus 1)}$ reprezintă toate fluxurile active în nucleele $i$ și $i \oplus 1$. Operația $i \oplus c$ poate fi definită ca $(i + c - 1)\%N_{CG} + 1$, iar $i \ominus c = (i - c - 1)\%N_{CG} + 1$.

În baza exemplului din figura 1.7 toate fluxurile accesate într-un nucleu formează clici maximale în fluxul grafului de interferență. Mai mult, fluxurile din nucleul $K_{(i\ominus 1)i} \cup K_i \cup K_{i(i\oplus i)}$ formează clică maximală pentru orice nucleu $i$.

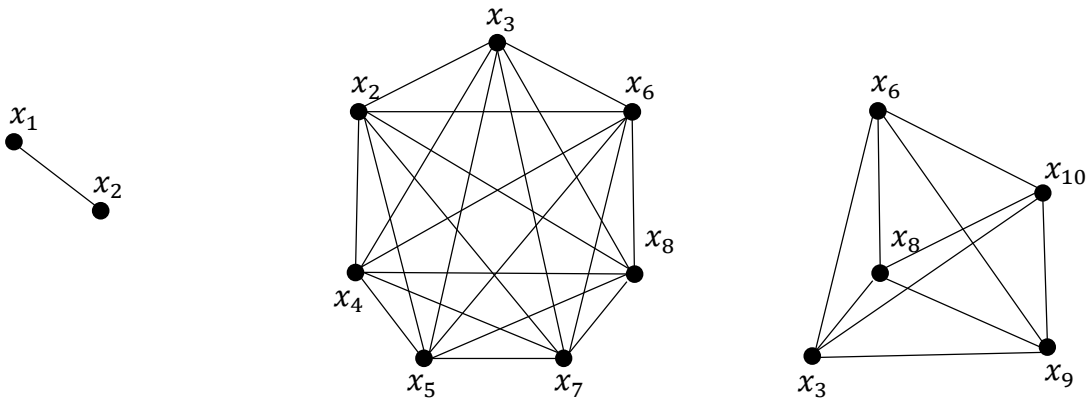



Fig. 1.8. *Clicile maximale induse de nucleele din figura 1.7.*

În baza celor menționate mai sus este demonstrat faptul că dacă $N_{CG}$ este număr par, atunci graful $G_{CG}$ este tranzitiv orientabil. De asemenea, dacă $K_{N_{CG}1} = \emptyset$, atunci $G_{CG}$ este graf tranzitiv orientabil. Dacă nu sunt respectate nici una din condițiile menționate pentru amplasarea informației, atunci se mai face o iterație de obținere a mai multor nuclee în graf. Această operație se repetă până când va fi satisfăcută cel puțin o condiție. Această operație este executată la nivel de procesor, astfel că ciclul menționat nu influențează performanța de lucru a procesorului.

În procesul de amplasare a informației dacă graful de interferență $G$ este tranzitiv orientabil, iar $G'$ este un subgraf indus, atunci $G'$ are cel mult opt orientări tranzitive. În baza partițiilor de domenii active este cunoscut următorul rezultat: dacă într-un graf tranzitiv orientabil $G$ toate mulțimile din partiția (1.1) sunt nevide, atunci graful $G$ admite exact două orientări tranzitive.

Din cele menționate, poate fi concluzionat că o colorare a grafului de interferență $G$ duce la amplasarea optimă a informației în fișierul de înregistrare a fluxului SRF. Dacă acest graf este tranzitiv orientabil atunci procesul dat poate fi efectuat în timp polinomial.

**Analiza codului sursă a programelor**

Multe probleme de analiza programelor, optimizarea, translarea, verificarea corectitudinii, testarea programelor pot fi simplificate prin reprezentarea sub formă de grafuri. La baza modelelor menționate stau așa numitele grafuri de control al fluxurilor. Un graf orientat $\vec{G} = (X; U)$ se numește graf de control al fluxurilor sau graf de trecere, dacă îndeplinește următoarele condiții:

1. Graful $\vec{G}$ nu conține arce paralele;
2. În mulțimea de vârfuri $X$ poate fi identificat un vârf $s$ numit intrarea grafului;
3. În mulțimea de vârfuri $X$ poate fi identificat un vârf $t$ numit ieșirea din graf;
4. Pentru orice vârf $x \in X \setminus \{s\}$ poate fi construit un lanț de forma $[s, \dots, x]$;
5. Pentru orice vârf $x \in X \setminus \{s\}$ poate fi construit un lanț de forma $[x, \dots, t]$.

Poate fi considerat și cazul când graful $\vec{G}$ are mai multe vârfuri de ieșire, atunci poate fi introdus un vârf fictiv care este unit cu arce de la fiecare vârf de ieșire.

De exemplu, în figura 1.9 este dat pseudocodul algoritmului de sortare prin metoda bulelor. Graful de trecere redus, asociat acestui program este ilustrat în figura 1.10

Construirea grafului de trecere pentru un program concret se face prin reguli simple, cu un grad de detaliere cerut inițial. În cel mai simplu caz, soluția problemei analizate constă în reprezentarea fiecărui operator (sau a unei comenzi, sau orice alt bloc de cod ce poate fi prezentat ca o componentă independentă a limbajului). Două vârfuri sunt adiacente dacă între operatorii

respectivi există transmitere de control. Mai exact, operatorul care transmite unitatea de control este vârful inițial al arcului, iar operatorul care primește controlul este vârful final al arcului. În asemenea caz dimensiunea grafului crește foarte mult din cauza că în graf se formează lanțuri lungi de vârfuri ce corespund secvenței liniare de cod. Aceste secvențe de cod pot fi substituite printr-un singur vârf.

```
1. Procedura Sortare(A, N)
2. Begin
3.     For i: = 2 step S_1 until N do:
4.     Begin
5.         If A(i) ≥ A(i − 1) then goto Next:
6.         j ≔ i
7.         Loop: If j ≤ 1 then goto Next:
8.             If A(j) ≥ A(j − 1) then goto Next:
9.             temp ≔ A(j)
10.            A(j) = A(j − 1)
11.            A(j − 1) = temp
12.            j ≔ j − 1
13.        Goto Loop
14.        Next: null
15.    End
16. End.
```

Fig. 1.9. *Pseudo codul algoritmului de sortare prin metoda bulelor.*

În cazul când este necesar să determinăm dacă între două vârfuri dintr-un graf există lanț atunci apare problema de construire a închiderii tranzitive a grafului orientat. În termenii de analiză a programelor, se verifică dacă un graf nu are cicluri infinite și are instrucțiuni de ieșire. Reamintim că închiderea tranzitivă a unui graf $G = (X; U)$ este graful orientat $G^*$ ce conține aceeași mulțime de vârfuri $X_G$, iar două vârfuri $x_i$ și $x_j$ sunt adiacente în $G^*$ dacă în $G$ vârfurile $x_i$ și $x_j$ pot fi unite printr-un lanț.

Matricea de adiacență $A(G^*)$ a închiderii tranzitive $G^*$ este echivalentă cu matricea de atingere $R(G)$ a grafului $G$. Construirea matricei de atingere $R(G)$ necesită timpul $O(n^3)$ [36]. Dacă însă graful $G$ este tranzitiv orientat atunci matricea atingerii coincide cu matricea de adiacență, deoarece închiderea tranzitivă a grafului coincide cu însăși graful $G$, deci determinarea matricei de atingere necesită timpul $O(n + m)$. Analizarea unui program la prezența ciclurilor infinite în cazul când acesta poate fi reprezentat printr-un graf tranzitiv orientabil necesită resurse mai puține, anume această diferență de timp prezintă interes de a studia această clasă de grafuri.

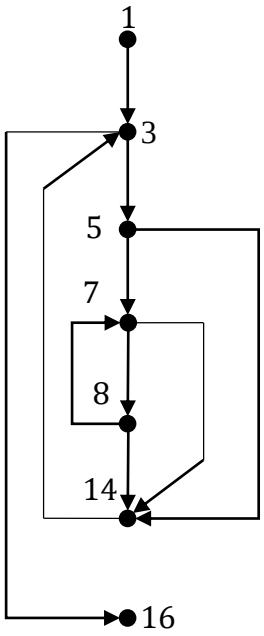



Fig. 1.10. *Graful de trecere al algoritmului de sortare prin metoda bulelor.*

**Grafurile tranzitiv orientabile sunt grafuri perfecte**

După cum a fost menționat mai sus soluționarea problemelor practice care pot fi reprezentate cu ajutorul grafurilor tranzitiv orientabile poate fi efectuată în timp polinomial. Aceasta se datorează faptului că o serie de probleme teoretice implementate asupra grafurilor tranzitiv orientabile pot fi soluționate în timp polinomial. Printre problemele teoretice care pot fi soluționate în timp polinomial sunt colorarea optimă a grafului, determinarea clicii maximale ponderate, determinarea numărului de stabilitate internă.

Pe mulțimea de vârfuri a unui graf orientat fără cicluri $\vec{G} = (X; \vec{U})$ se definește o ordonare parțială strictă, notată $x < y$, dacă și numai dacă în $\vec{U}$ există un lanț simplu ce unește vârfurile $x$ și $y$. Funcția de înălțime $h$ se definește în modul următor: $h(v) = 0$ dacă gradul de ieșire al vârfului $v$ este 0, $h(v) = 1 + \max\{h(w) | [v, w] \in \overrightarrow{U_G}\}$. Numărul $h(v)$ poate fi calculat în timp linear aplicând algoritmul de căutare în adâncime. Pentru grafurile tranzitiv orientabile funcția $h$ oferă o colorare corectă a grafului, deoarece pentru orice două vârfuri $x$ și $y$ dacă $[x, y] \in \overrightarrow{U_G}$, atunci $h(x) < h(y)$, dar, pentru aceasta nu este necesară o colorare minimală. În cazul când funcția $h$ definește o colorare optimă atunci graful $G$ este tranzitiv orientabil.

Dacă $(X, \leq)$ este o mulțime parțial ordonată, atunci mulțimea strict ordonată $X$ se numește lanț. Submulțimea de vârfuri din $X$ ce nu conține elemente comparabile cu operația „ $\geq$ ” se

numește antilanț. O diagramă Hasse este o reprezentare compactă a ordonării tranzitive. Pentru mulțimile parțial ordonate $\left(X, \vec{U}\right)$ diagrama Hasse se definește în modul următor $(X, \vec{V})$, unde $\vec{V} \subseteq \vec{U}$ și $[x, y] \in \vec{U}$ dacă și numai dacă $[x, y] \in \vec{V}$ și nu există nici un vârf $z \in X$ astfel încât $[x, z] \in \vec{V}$ și $[z, y] \in \vec{V}$. În baza celor menționate poate fi formulat următorul rezultat, numărul de lanțuri necesar pentru a forma o partiție pe mulțimea parțial ordonată $(X, \leq)$ este egal cu cardinalul antilanțului din mulțimea $X$. Acest rezultat poate fi aplicat la calcularea complexității pentru determinarea colorării optime a grafului tranzitiv orientabil și determinarea clicii maximale. Aceste operații pot fi efectuate în timpul $O(m + n)$, unde $m$ este cardinalul mulțimii de muchii a grafului $G$, iar $n$ este cardinalul mulțimii de vârfuri a grafului $G$.

**Determinarea clicii maximale ponderate**

Multe problemele practice pot fi reduse la determinarea clicii maximale ponderate, în caz general este cunoscut faptul că această problemă este NP-completă, totuși dacă graful asociat este tranzitiv orientabil aceasta poate fi soluționată în timpul $O(m + n)$, unde $m$ este cardinalul mulțimii de muchii a grafului $G$, iar $n$ este cardinalul mulțimii de vârfuri a grafului $G$.

Problema determinării clicii maximale poate fi formulată în următorul mod. Fie dat un graf neorientat $G = (X; U)$, fiecărui vârf $x_i$ i se atribuie o pondere $p(x_i)$. Să se determine clica grafului $G$ cu suma maximă a ponderilor vârfurilor.

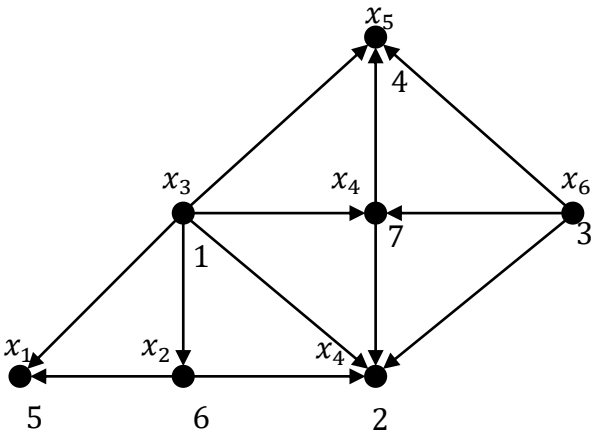



Fig. 1.11. *Graf tranzitiv orientat.*

Ideea algoritmului constă în construirea unei orientări tranzitive a grafului $G$. În graful orientat $\vec{G}$ se construiesc drumurile elementare $d[x, y]$ ce leagă vârfurile $x, y \in X_G$. Prin $P(d[x, y])$ se notează suma ponderilor vârfurilor drumului $d[x, y]$. Numărul $P(d[x, y])$ se numește ponderea drumului $d[x, y]$. Pentru un vârf arbitrar $v \in X_G$ se introduce un parametru suplimentar $W(v) = \max_z\{P(d[v, z])\}$. Deci, pentru orice vârf $x \in X_G$ se calculează valoarea $W(x)$, iar în drumul

$d[x, z]$ cu ponderea maximă se atribuie un indice succesorului vârfului $x$.

În graful din figura 1.11 este prezentată o orientare tranzitivă împreună cu ponderile vârfurilor. Aplicând algoritmul pentru determinarea clicii maximale ponderate asupra grafului dat, se obține clica maximală determinată de mulțimea de vârfuri $\{x_4, x_5, x_6\}$ cu valoarea 14.

**Calcularea numărului de stabilitate internă $\alpha(G)$**

Ca și problema determinării clicii maximale ponderate, calcularea numărului de stabilitate internă $\alpha(G)$ este o problemă NP-completă în cazul general al grafurilor. Dacă în graful $G$ este tranzitiv orientabil, atunci $\alpha(G)$ se determină în timp polinomial.

Graful $\vec{G}$ se transformă într-o rețea de transport prin adăugarea vârfului sursă $s$ și vârfului destinație $t$ și respectiv a arcelor $[s, x]$ și $[y, t]$. Atribuind fiecărui vârf limita de jos 1, se calculează fluxul minimal. Această problemă se rezolvă în timp polinomial. Valoarea fluxului obținut va fi $k(G)$, unde $k(G)$ este numărul de acoperire cu clici. Deoarece graful $G$ este tranzitiv orientabil, după cum a fost menționat mai sus, $G$ este și graf perfect. Conform definiției grafurilor perfecte, rezultă că $k(G) = \alpha(G)$.

O importanță deosebită în estimarea timpului de lucru al unui algoritm îl reprezintă și reprezentarea grafului în memoria calculatorului. Cele mai des utilizate reprezentări sunt sub formă de matrice de adiacență, matrice de incidență, datorită simplității acestora. Totuși, sunt multe probleme în care este necesară o reprezentare mai eficientă, cum ar fi reprezentarea grafului sub formă de listă de adiacență a vârfurilor,

**Lista vârfurilor adiacente.** Este formată din $n$ liste, unde lista cu ordinul $i$ conține indicele vârfurilor adiacente vârfului $x_i$. Memorarea se realizează într-o matrice astfel încât linia $i$ conține vârfurile adiacente cu vârful $x_i$. Pentru fiecare vârf $x_i$ se memorează de asemenea și numărul de vârfuri din lista. O altă variantă a listei vârfurilor adiacente este aceea de a folosi listele liniare memorate static sau dinamic.

În exemplele prezentate în această lucrare vom folosi în deosebi reprezentarea geometrică a grafurilor, iar pentru prelucrarea datelor în algoritmii descriși vom folosi reprezentarea sub formă de liste de adiacență. Reprezentarea sub formă de listă de adiacență a fost aleasă anume din considerentul că operațiile de căutare în adâncime oferă un timp rezonabil pentru grafurile cu o reprezentare de acest tip.

## 1.5. Concluzii la capitolul 1

În baza celor expuse, putem constata că necesitatea soluționării problemei de ordin teoretico-aplicativ a condus la apariția unei clase speciale de grafuri numite grafuri tranzitiv orientabile.

Studierea proprietăților acestora a îmbogățit teoria generală a grafurilor și a permis soluționarea unor probleme practice. Astfel a devenit posibilă soluționarea unor cazuri speciale a problemelor de utilizare a managementului fișierelor de înregistrare pentru procesoarele stream, decompoziția rețelelor Petri, analiza codului sursă a programelor.

În primul capitol sunt descrise în ordine cronologică principalele rezultate cu referire la grafurile tranzitiv orientabile și este menționat aportul mai multor matematicieni la dezvoltarea cercetărilor legate de studierea structurilor descrise în lucrare. În mod special se menționează rolul pe care l-au avut în dezvoltarea teoriei respective matematicienii, Ghouila-Houri A., Gilmore P.C., Hoffman A. J., Golumbic M.C., Shevrin L.N., Filipov N.D., Wolk E.S. Fiecare din ei a studiat diferite aspecte ale clasei de grafuri tranzitiv orientabile.

Wolk E.S a făcut o caracterizare abstractă a grafurilor tranzitiv orientabile care sunt arbori. Totodată, Shevrin L.N. și Filipov N.D. au oferit descrierea clasei repective de grafuri prin intermediul unor structuri abstracte. În lucrările lui Ghouila-Houri A. au fost formulate condițiile necesare și suficiente pentru ca un graf să fie tranzitiv orientabil. Iar Gilmore P.C. și Hoffman A. J. au descris grafurile tranzitiv orientabile prin intermediul mulțimilor parțial ordonate. R H. Möhring a demonstrat că aproape toate grafurile tranzitiv orientabile au doar două orientări tranzitive opuse una alteia. În sfârșit Golumbic M.C. a caracterizat grafurile tranzitiv orientabile prin intermediul claselor de implicații. Cu ajutorul acestor structuri a fost posibilă elaborarea unui algoritm pentru construirea unei singure orientări tranzitive.

În prezenta lucrare se continuă studierea grafurilor tranzitiv orientabile prin elaborarea și studierea metodelor de construire a orientărilor tranzitive, chestiune neîntâlnită la alți autori și importantă pentru unele probleme, cum ar fi: sortarea mulțimilor parțial ordonate. În baza metodelor menționate se elaborează algoritmi de construire a orientărilor tranzitive cu analiza eficienții acestora și estimarea vitezei de lucru.

Pe lângă examinarea problemei de bază au fost obținute și o serie de rezultate auxiliare, importante pentru soluționarea în ansamblu a grafurilor: determinarea formulei recurente de calcul a numărului de orientări tranzitive, caracterizarea clasei de grafuri factor pentru construirea orientărilor tranzitive ale grafului, obținerea unor proprietăți speciale pentru subgrafurile stabile pentru determinarea șirului de grafuri factor.

Realizarea prezentei teze a pretins implicit atingerea următorului scop: caracterizarea structurală a grafurilor tranzitiv orientabile; obținerea formulei pentru determinarea numărului de orientări tranzitive într-un graf; elaborarea în baza structurilor obținute a algoritmilor de construire a orientărilor tranzitive pentru grafurile neorientate cu restricții asupra muchiilor.

În conformitate cu scopul enunțat au fost stabilite următoarele obiective ale cercetării:

- determinarea rolului lanțurilor netriangulate în construirea orientărilor tranzitive ale grafurilor;
- examinarea proprietăților subgrafurilor B-stabile și rolul acestora la construirea grafurilor factor;
- studierea șirului complet de grafuri factor pentru descrierea problemei orientărilor tranzitive ale unui graf neorientat;
- determinarea formulei recurente de calcul a numărului de orientări tranzitive ale grafului;
- elaborarea algoritmilor pentru construirea orientărilor tranzitive cu restricții asupra muchiilor orientabile;
- propunerea algoritmilor pentru editarea unei orientări tranzitive cu restricții impuse arcelor în orientarea existentă;
- implementarea algoritmilor elaborați.

Totuși, precum s-a adeverit au mai rămas probleme neclarificate până la urmă, care urmează a fi studiate în cercetările ulterioare. În baza analizei situației în domeniul studierii grafurilor tranzitiv orientabile și aplicării acestora la soluționarea problemelor teoretic-aplicative putem face următoarele concluzii.

**1.** Examinarea grafurilor tranzitiv orientabile a condus la obținerea unor rezultate importante pentru soluționarea diverselor probleme atât de ordin teoretic, cât și de ordin aplicativ care stimulează interesul pentru continuarea cercetărilor cu scopul extinderii acestora asupra soluționării unor probleme complexe.

**2.** Rezultatele obținute la moment de către alți autori permit soluționarea parțială a problemei orientării tranzitive pentru grafurile neorientate, din care însă nu rezultă mecanismul construirii tuturor orientărilor tranzitive și legătura dintre diferite orientări.

**3.** Din cercetările cunoscute la moment nu se cunoaște mecanismul găsirii soluției orientării tranzitive a grafului în condițiile restricției asupra orientărilor unor arce.

**4.** Apare necesitatea studierii suplimentare a grafurilor tranzitiv orientabile cu scopul determinării caracterizării structurale ale acestora și determinării unei formule recurente de calcul a orientărilor tranzitive.

**5.** Apare necesitatea elaborării și implementării unor algoritmi cu referire la grafurile tranzitiv orientabile, aplicabili pentru soluționarea eficientă a problemelor practice.

# 2. ORIENTAREA TRANZITIVĂ A GRAFURILOR NEORIENTATE

În capitolul doi sunt expuse principalele rezultate care conduc la soluționarea problemei orientării tranzitive a grafurilor. Studiul respectiv este generat, în mare parte, de importanța unor probleme practice, soluționarea cărora se reduce la examinarea modelelor matematice reprezentate prin grafuri tranzitive. În mod special printre aceste probleme pot fi menționate: decompoziția rețelelor Petri, optimizarea managementului fișierelor de înregistrare pentru procesoarele stream, analiza codului sursă a programelor, etc.

Soluționarea problemei orientării tranzitive a unui graf neorientat se obține prin utilizarea proprietăților speciale ale unor structuri examinate în acest capitol: subgrafurile B-stabile, graful factor, lanțurile netriangulate ale unui graf neorientat, etc.

Lanțurilor netriangulate le revine un rol aparte în studierea problemei menționate. Această structură apare la studierea proprietăților subgrafurilor stabile minimale, folosite la elaborarea metodelor de construire a orientărilor tranzitive. În legătură cu studierea lanțurilor netriangulate au fost obținute un șir de rezultate, expuse în secțiunea 2.1, printre care se evidențiază rezultatul formulat în teorema 2.1 ce conține condițiile de existență a unui lanț netriangulat ce trece prin toate muchiile unui graf neorientat.

Construirea orientărilor tranzitive ale grafului neorientat se bazează pe generarea unui șir finit de grafuri factor. „Cărămizile" principale la construirea unui graf factor sunt subgrafurile stabile (minimale, complete maximale, vide maximale). Prin teorema 2.5 este obținută caracterizarea subgrafurilor stabile minimale: un subgraf generat de o mulțime de vârfuri $X_F \in X_G$ este stabil minimal dacă și numai dacă în graful $G$ există lanț netriangulat ce conține doar vârfurile din $X_F$. Subgrafurile stabile studiate în secțiunea 2.2 se includ în clasa subgrafurilor B-stabile ale unui graf neorientat, caracterizate prin teorema 2.7.

Cu ajutorul structurilor menționate, pentru un graf stabil se construiește șirul complet de grafuri factor, în baza cărora se deduce formula recurentă de calculare a orientărilor tranzitive ale unui graf neorientat (a se vedea formula 2.13).

Rezultatele teoretice descrise în capitolul doi stau la baza elaborării metodelor de orientare a grafului sau a unei părți a acestora: construirea orientării tranzitive a unui subgraf stabil minimal (descrierea algoritmului cu analiza complexității este în secțiunea 2.3); construirea orientării tranzitive a subgrafului stabil complet maximal (descrierea algoritmului cu analiza complexității este în secțiunea 2.4), etc.

Rezultatele obținute în acest capitol servesc drept suport pentru unele cercetări adiționale, descrise în capitolul următor al tezei.

## 2.1. Subgrafuri stabile și lanțuri netriangulate

Elementele de bază în procesul de studiere a grafurilor tranzitiv orientabile sunt lanțurile netriangulate și subgrafurile stabile. În paragraful ce urmează vor fi expuse principalele proprietăți ale acestor structuri, care vor fi utilizate în caracterizarea grafurilor tranzitiv orientabile. Rezultatele de bază în care sunt descrise subgrafurile stabile sunt formulate în lemele 2.1 – 2.4 și au fost publicate în lucrările [17] și [52]. Proprietățile lanțurilor netriangulate și relația dintre lanțurile netriangulate și subgrafurile stabile au fost formulate în lemele 2.3 – 2.6 și teoremele 2.1 – 2.3 publicate în lucrările [16], [18], [51].

Fie $G = (X; U)$ un graf neorientat şi $F \subset X$ o submulţime arbitrară de vârfuri a acestui graf. Vom nota prin $G(F)$ subgraful grafului $G$, generat de $F$.

De rând cu notația $x \sim y$ de adiacență a vârfurilor $x, y \in X$, în cele ce urmează vom mai folosi următoarele notații:

$x \sim F$ — vârful *x* este adiacent cu toate vârfurile din $F$.

$x \nsim F$ — vârful *x* nu este adiacent cu nici unul din vârfurile din $F$.

**Definiția 2.1.** [115] *Subgraful $G(F)$ se numeşte subgraf stabil al grafului $G$, dacă pentru $\forall x \in X \setminus F$ are loc una dintre următoarele relații:*

$$x \sim F \text{ sau } x \nsim F.$$

Mulțimea $F$ se numește mulțime stabilă. În prezenta lucrare se studiază subgrafurile stabile proprii ale grafului $G$, adică subgrafurile generate de submulțimea de vârfuri $X_F \in X_G$ ce respectă următoarea condiție:

$$2 \leq |X_F| \leq n - 1.$$

Pentru simplitatea expunerii ulterioare, și din considerente de comoditate, astfel de subgrafuri le vom numi subgrafuri stabile, omițând cuvântul proprii.

Ușor ne putem convinge că nu orice graf conține subgrafuri stabile proprii. Drept exemplu poate servi graful din figura 2.1 a). În cazul grafului din figura 2.1 b), în calitate de subgraf stabil poate fi considerat subgraful generat de submulțimea de vârfuri $A = \{x_2, x_3, x_4\}$.

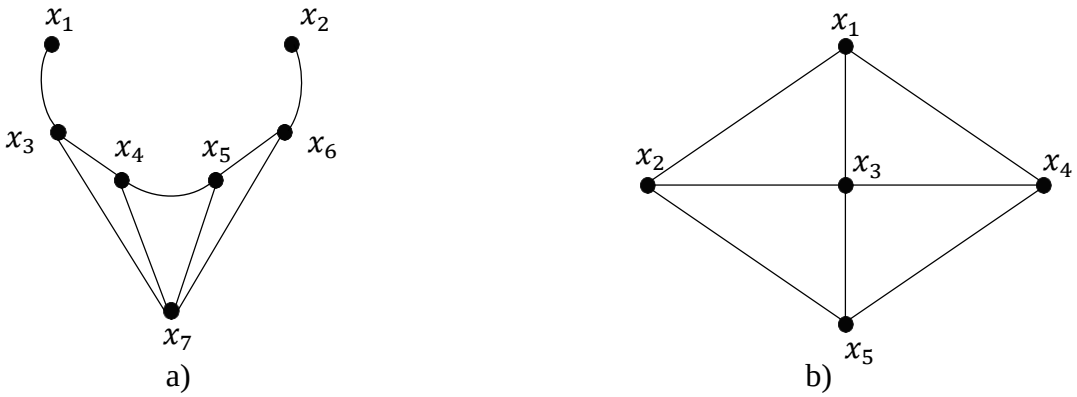



Fig. 2.1. *Graf neorientat ce nu conține subgraf stabil (cazul a)) și graf neorientat în care astfel de graf există (cazul b)).*

În continuare vor fi examinate o serie de proprietăți ale subgrafurilor stabile, necesare pentru soluționarea problemei orientării tranzitive ale unui graf neorientat.

**Lema 2.1.** *Dacă $G(A)$ şi $G(B)$ sunt subgrafuri stabile ale grafului $G$, atunci intersecţia lor $G(A \cap B)$ de asemenea este un subgraf stabil.*

**Demonstrație:** Deoarece $G(A) \cap G(B) = G(A \cap B)$, pentru a demonstra afirmația lemei este suficient de demonstrat că pentru $\forall x \in X \setminus (A \cap B)$ are loc una din relațiile:

$$x \sim (A \cap B) \text{ sau } x \nsim (A \cap B).$$

Fie $x$ un vârf arbitrar ce nu aparține intersecției $A \cap B$, ceea ce este echivalent cu afirmația: $x \in X \setminus A$ sau $x \in X \setminus B$. Vom examina cazul când vârful $x \in X \setminus A$. (Cel de-al doilea caz se examinează în mod similar).

Conform definiției 2.1, deoarece $G(A)$ este subgraf stabil, rezultă că pentru orice vârf $x \in X \setminus A$ are loc una din relațiile: $x \sim A$ sau $x \nsim A$. Aceasta, la rândul său implică următoarele:

$$x \sim (A \cap B) \text{ sau } x \nsim (A \cap B),$$

ceea ce și demonstrează afirmația lemei.

Spre deosebire de intersecția subgrafurilor stabile, reuniunea acestora nu totdeauna generează un subgraf stabil. Să examinăm graful din figura 2.2. Subgrafurile generate de mulțimile de vârfuri $A = \{x_1, x_2\}$ și $B = \{x_6, x_7\}$ sunt subgrafuri stabile, pe când reuniunea lor nu mai posedă această proprietate.

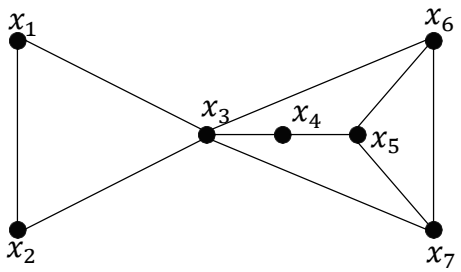



Fig. 2.2. *Graf cu subgrafuri stabile, generate de mulțimile* $A = \{x_1, x_2\}$ *și* $B = \{x_6, x_7\}$.

**Lema 2.2.** *Dacă* $G(A)$ *şi* $G(B)$ *sunt subgrafuri stabile ale unui graf neorientat* $G$ *şi* $A \cap B \neq \emptyset$*, atunci reuniunea lor* $G(A \cup B)$ *de asemenea este un subgraf stabil.*

**Demonstrație:** Fie $G(A)$ *şi* $G(B)$ două subgrafuri stabile ale grafului $G = (X; U)$, $A \cap B \neq \emptyset$, şi $z$ - unul dintre vârfurile ce aparține intersecției $A \cap B$. Vom cerceta două cazuri:

a) Fie $x$ un vârf arbitrar din $X \backslash A$. Deoarece $G(A)$ *şi* $G(B)$ sunt subgrafuri stabile, obținem următorul șir de implicații:

$$x \sim A \Rightarrow x \sim z \Rightarrow z \sim B \Rightarrow x \sim A \cup B$$

sau

$$x \nsim A \Rightarrow x \nsim z \Rightarrow z \nsim B \Rightarrow x \nsim A \cup B$$

b) Fie $x$ un vârf arbitrar din $X \backslash B$. Cazul acesta este identic cazului a).

Prin urmare subgraful $G(A \cup B)$ este stabil.

**Lema 2.3.** Pentru orice două subgrafuri stabile $G(A)$ și $G(B)$ ale unui graf neorientat $G = (X; U)$ ce respectă condiția $A \cap B \neq \emptyset$ sunt adevărate afirmațiile:

a) dacă există două vârfuri $x \in A$ și $y \in B$ care sunt adiacente, se respectă cel puțin una din condițiile $x \in A \setminus B$ sau $y \in B \setminus A$, atunci $A \sim B$.
b) Dacă $G(A)$ și $G(B)$ sunt grafuri conexe pentru care există două vârfuri adiacente $x, y \in A \cap B$, atunci $A \sim B$.

**Demonstrație:** Să analizăm fiecare caz în parte.

a) Fie $x \in A$ și $y \in B$. Fără a pierde din generalitate considerăm că $x \in A \setminus B$. Deoarece $y \in B$ și $[x, y] \in U_G$, condiția că $B$ este mulțime stabilă implică faptul că $x \sim B$. Deoarece $x \sim B$ în particular $[x, w] \in U_G$, unde $w \in B \setminus A$ și $w \neq y$. Din aceste considerente și din faptul că $A$ este mulțime stabilă obținem că $w \sim A$. Deci, obținem că $A \sim B$.

b) Fie $s \in A \setminus B$ încât $[s, x] \in U_G$. Deoarece $G(A)$ și $G(B)$ sunt grafuri conexe rezultă că $s \sim B$. Situație similară se obține dacă alegem un vârf $t \in B \setminus A$. Astfel, obținem că $A \sim B$.

**Consecință:** Dacă $G(A)$ și $G(B)$ sunt subgrafuri stabile conexe și $A \cap B \neq \emptyset$, atunci mulțimile $A$ și $B$ sunt adiacente $A \sim B$.

**Definiția 2.2.** Vom numi diferență a două subgrafuri stabile $G(A)$ și $G(B)$, notată prin $G(A) \backslash G(B)$, subgraful generat de mulțimea de vârfuri $A \backslash B$.

**Lema 2.4.** Dacă $G(A)$ și $G(B)$ sunt subgrafuri stabile ale uni graf $G = (X; U)$ și $A \cap B \neq \emptyset$, atunci diferența lor $G(A) \backslash G(B) = G(A \backslash B)$ de asemenea este un subgraf stabil.

Demonstrația acestei leme se face în mod analog cu demonstrația lemei 2. 3.

Notăm prin $\bar{G}(A)$ subgraful $G \backslash G(A)$ și îl vom numi subgraf complementar lui $G(A)$.

Observăm, că graful în care orice subgraf este stabil este graful complet $K_n$, sau graful vid $O_n$. Ba mai mult: pentru orice două submulțimi de vârfuri din $K_n$ sau $O_n$ subgrafurile $G(A \cap B)$, $G(A \cup B)$, $\bar{G}(A)$ sunt subgrafuri stabile.

**Definiția 2.3.** *Subgraful stabil ce conține un subgraf oarecare $G(A)$ al grafului $G$ se numește închidere stabilă a lui $G(A)$ și se notează prin $S(A)$.*

Evident, dacă $G(A)$ este subgraf stabil, atunci închiderea stabilă S(A) a acestuia coincide cu însuși subgraful $G(A)$.

Pot fi menționate câteva proprietăți simple ale închiderii stabile:

P1. $S\big(S(A)\big) = S(A)$

**Demonstrație:** Conform definiției închiderii stabile, $S(A)$ generează un subgraf. La rândul său $S(A)$ este subgraf stabil, deci închiderea stabilă va fi însăși subgraful stabil $S(A)$.

P2. Dacă $A \subseteq B$, atunci $S(A) \subseteq S(B)$

**Demonstrație:** Deoarece $A \subseteq B \Rightarrow A \subseteq S(B) \Rightarrow S(A) \subseteq S\big(S(B)\big)$, conform celor demonstrate mai sus, rezultă că: $S(A) \subseteq S(B)$.

P3. Dacă $A \cap B \neq \emptyset$, atunci $S(A) \cup S(B) = S(A \cup B)$

**Demonstrație:**

$$\begin{matrix} A \subseteq A \cup B \Rightarrow S(A) \subseteq S(A \cup B) \\ B \subseteq A \cup B \Rightarrow S(B) \subseteq S(A \cup B) \end{matrix} \Rightarrow S(A) \cup S(B) \subseteq S(A \cup B) \qquad (2.1)$$

Deoarece $S(A)$ și $S(B)$ sunt stabile, și $A \cap B \neq \emptyset$, conform proprietății P2 a subgrafurilor stabile rezultă că $S(A) \cup S(B)$ este subgraf stabil. E clar că $A \cup B \subseteq S(A) \cup S(B)$, de unde rezultă:

$$S(A \cup B) \subseteq S(A) \cup S(B) \tag{2.2}$$

Din (2.1) și (2.2) rezultă $S(A) \cup S(B) = S(A \cup B)$.

**Afirmația 2.1.** *Dacă graful $G = (X; U)$ nu conține subgrafuri stabile proprii, atunci $G$ este graf conex.*

**Demonstrație:** Presupunem că $G$ nu conține subgrafuri stabile, dar nu este conex. Atunci în $G$ există o submulțime de vârfuri care generează o componentă conexă. Și dacă $G$ nu conține subgrafuri stabile, atunci pentru orice submulțime de vârfuri afirmația: $A \subset X, \forall x \in X \backslash A: x \sim A$ sau $x \nsim A$ este falsă. Dar pentru submulțimea de vârfuri care generează o componentă conexă este adevărată următoarea relația: $x \nsim A$.

**Afirmația 2.2.** *Într-un graf complet $K_n$ există $2^{n-1} - 3$ subgrafuri stabile proprii.*

**Demonstrație:** Deoarece $K_n$ este graf complet, este clar că orice submulțime de vârfuri din el generează subgraf stabil. Dacă vom lua pe rând numărul tuturor subgrafurilor formate din 2 vârfuri, atunci avem $C_n^2$ subgrafuri stabile formate din 2 vârfuri, numărul subgrafurilor formate din 3 vârfuri va fi $C_n^3$, continuând logica până la $n-1$ obţinem relaţia: $\nu = \sum_{i=2}^{n-1} C_n^i$ ceea ce este egal cu $2^{n-1} - 3$.

În cele ce urmează vom introduce noțiunea de lanț netriangulat, această structură poate fi folosită în descrierea subgrafurilor stabile și respectiv, în caracterizarea orientărilor tranzitive.

**Definiția 2.4.** [115] *Consecutivitatea de vârfuri $l = [x_1, x_2, \dots, x_p]$ pentru care în graful $G = (X; U)$ nu există muchii de tipul: $[x_i, x_{i+2}]$ se va numi lanț netriangulat.*

**Definiția 2.5.** [119] *Lanțul netriangulat închis $\mu = [x_1, x_2, \dots, x_p, x_1]$ pentru care în graful $G = (X; U)$ nu există muchii de tipul $[x_i, x_{i+2}]$ și $[x_2, x_p]$ se numește ciclu netriangulat.*

De exemplu, în figura 2.3 $l_1 = [x_1, x_3, x_2, x_5, x_6, x_7, x_4]$ nu este netriangulat deoarece vârfurile $x_3$ și $x_5$ sunt adiacente (precum și vârfurile $x_4$ și $x_6$ sunt vârfuri adiacente). Totodată, $l_2 = [x_1, x_3, x_4, x_3, x_2, x_3, x_6]$ este netriangulat.

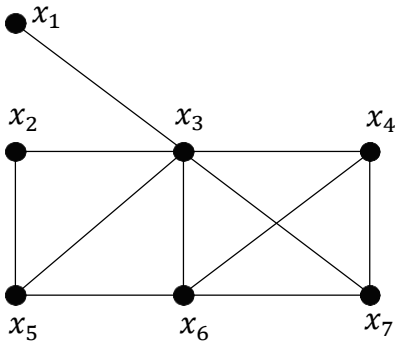



Fig. 2.3. *Graf cu lanț netriangulat* $[x_1, x_3, x_4, x_3, x_2, x_3, x_6]$.

**Lema 2.5.** *Dacă* $a \sim b$ *atunci* $S(\{a, b\})$ *coincide cu mulțimea tuturor vârfurilor* $x$ *din* $G$ *pentru care există lanț netriangulat* $l = [a, b, \dots, x]$.

**Demonstrație:** Demonstrația afirmației date este echivalentă cu demonstrația următoarelor două afirmații descrise mai jos:

1. Dacă $x \in X_G, x \neq a, b$, astfel încât în $G$ există lanț netriangulat

$$[a = y_1, b = y_2, y_3, \dots, y_i, y_{i+1}, \dots, y_{p-1}, y_p = x] \tag{2.3}$$

atunci $x \in S(\{a, b\})$. Mai mult ca atât $y_i \in S(\{a, b\}), i = \overline{3, p}$

2. Dacă $x \in X_G, x \neq a, b$ astfel încât orice lanț netriangulat este de tipul (2.3),

atunci vârful $x \notin S(\{a, b\})$.

Să demonstrăm afirmația 1:

Fie în $G$ există lanțul netriangulat (2.3). Evident $a, b \in S(\{a, b\})$. Presupunem că există $2 < i \leq p$ astfel încât $y_1, y_2, \dots, y_i \in S(\{a, b\})$, iar $y_{i+1} \notin S(\{a, b\})$.

Deoarece $S(\{a, b\})$ generează un subgraf stabil și $[y_i, y_{i+1}] \in U_G$ rezultă că $y_{i+1} \sim S(\{a, b\})$. Prin urmare $y_{i+1} \sim y_{i-1}$, ceea ce înseamnă că lanțul (2.3) nu este netriangulat.

Contradicția obținută demonstrează afirmația 1.

Să demonstrăm afirmația 2:

Presupunem că $x \in S(\{a, b\})$. Fie un vârf $z \neq a, b, x$ , $z \in S(\{a, b\})$, atunci pentru $z$ are loc unul din următoarele patru cazuri:

a) $z \sim a, b$

b) $z \nsim a, b$

c) $z \sim a,\ z \nsim b$

d) $z \nsim a,\ z \sim b$

Dar cazurile c) și d) nu pot fi adevărate în condițiile afirmației de mai sus, deoarece în asemenea situație va putea fi construit lanțul netriangulat de forma: $[b, a, z, a, x]$ pentru cazul c), sau $[a, b, z, b, x]$ pentru cazul d).

Deci pot avea loc doar cazurile a) și b). Dar din definiția închiderii stabile și din condițiile afirmației 2 rezultă că $z \notin S(\{a, b\})$, deci presupunerea că $x \in S(\{a, b\})$ este greșită, fapt ce demonstrează a doua afirmație.

Ținând cont de cele două afirmații demonstrate, rezultă că $S(\{a, b\})$ coincide cu mulțimea tuturor vârfurilor grafului $G$ pentru care există lanț netriangulat de forma (2.3).

**Lema 2.6.** *Dacă în garful $G$ există lanț netriangulat ce conține toate muchiile grafului, atunci în acest graf putem construi și un ciclu netriangulat ce conține toate muchiile din $G$.*

Demonstrația lemei devine evidentă dacă observăm că având lanțul netriangulat menționat ce pornește dintr-un anumit vârf $z$, parcurgându-l consecutiv în ordine inversă, ajungem în $z$.

**Teorema 2.1.** *Dacă $G = (X; U)$ este un graf neorientat conex ce nu conține cicluri de lungimea 3, atunci pentru $G$ există lanț netriangulat ce trece prin toate muchiile sale.*

**Demonstrație:** Aplicăm inducția matematică în raport cu numărul de vârfuri $n$ ale garfului $G$. Ușor ne convingem că pentru orice graf cu numărul de vârfuri $n = 1, 2, 3, 4$ afirmația teoremei este adevărată. În cele ce urmează vom analiza cazul când $n \geq 5$.

Presupunem că afirmația este adevărată pentru orice graf conex cu $n \leq k$ vârfuri, unde $k \geq 4$. Să studiem un graf conex arbitrar cu $n = k + 1$ vârfuri. Alegem un vârf oarecare $x$, fie $deg(x) = p$ și $\Gamma(x) = \{y_1, y_2, \dots, y_p\}$. Admitem că în graful obținut $G - x$ obținem $t - 1$ componente conexe: $G_1, G_2, \dots, G_t$.

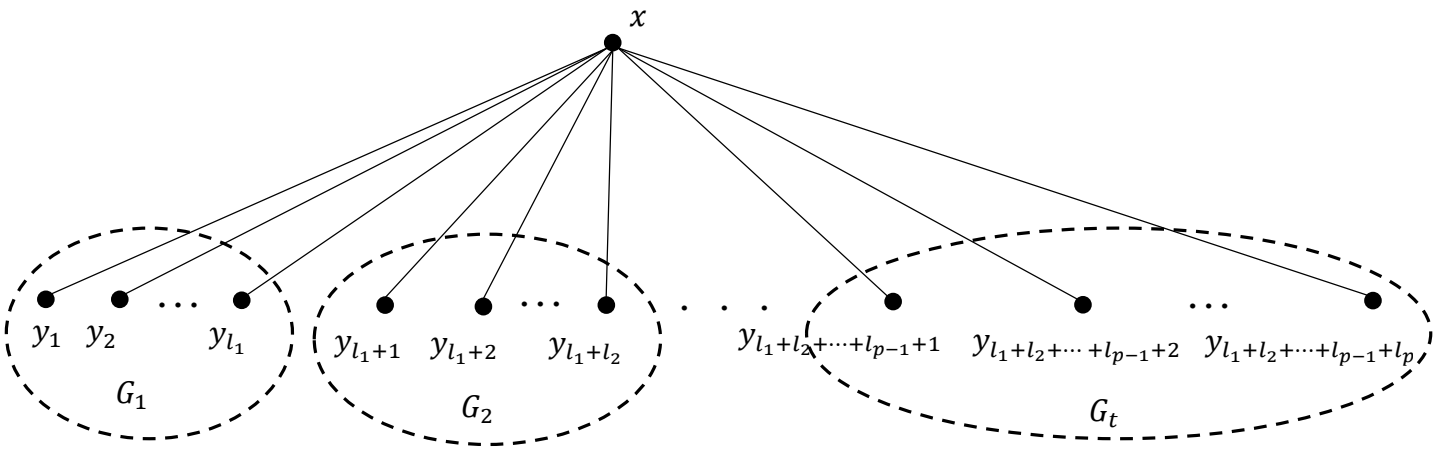



Fig. 2.4. *Lanț netriangulat construit în baza a* $t-1$ *componente conexe ale grafului* $G$.

Notăm prin $X_i -$ mulțimea de vârfuri ale componentei conexe $G_i$ și considerăm $|X_i \cap \Gamma(x)| = l_i$. Deoarece graful $G$ nu conține triunghiuri, atunci $y_i \nsim y_j$ pentru $\forall i, j = \overline{1,p}, i \neq j$. Conform inducției matematice și lemei 2.6 pentru fiecare componentă conexă există lanț netriangulat ce conține toate muchiile grafului. Atunci putem construi lanțul:

$$x, \overline{y_1, C_1, y_1}, x, y_2, x, \dots, x, y_{l_1}, x, \overline{y_{l_1+1}, C_2, y_{l_1+1}}, x, y_{l_1+2}, x, \dots, y_{l_1+l_2}, x, \dots,$$

$$x, \overline{y_{l_1+l_2+\cdots+l_{p-1}+1}, C_{l_1+l_2+\cdots+l_{p-1}+1}, y_{l_1+l_2+\cdots+l_{p-1}+1}}, x, y_{l_1+l_2+\cdots+l_{p-1}+2}, x, \dots, x, y_p.$$

Unde $\overline{y_\iota, C_\iota, y_\iota}$ este lanțul netriangulat din componenta conexă $i$ ce pornește din vârful $y_i$, trece prin toate muchiile ciclului netriangulat $C_i$ și ultimul vârf din lanț rămâne a fi $y_i$. Lanțul descris în demonstrație poate fi ilustrat în exemplul din figura 2.4.

**Consecință:** Într-un graf neorientat $G = (X; U)$ ce posedă proprietățile:

a) $G$ nu conține subgrafuri proprii stabile;

b) Orice ciclu din $G$ este netriangulat și are lungime pară

există lanț netriangulat ce trece prin toate vârfurile din $G$.

**Definiția 2.6.** *Dacă pentru muchiile* $[a, b]$ *și* $[c, d]$ *există lanț netriangulat* $[a, b, \dots, c, d]$, *atunci vom spune că muchia* $[a, b]$ *este legată triangulat de muchia* $[c, d]$.

Definim pe mulțimea de muchii a grafului $G = (X; U)$ o relația binară $\tau$. Vom spune că două muchii $[a, b]$ și $[c, d]$ se află în relația $\tau$, dacă $[a, b]$ este legată triangulat de $[c, d]$.

Menționăm că relația $\tau$ este o relație de echivalență, adică verifică următoarele proprietăți:

*Reflexivitate*: $\boldsymbol{u\tau u}$ pentru orice muchie $u = [a, b]$ din $G$.

*Simetrie:* $\boldsymbol{u\tau v \Rightarrow v\tau u}$, pentru oricare două muchii $u = [a, b]$ și $v = [c, d]$. Deoarece $u$ este legată triangulat de $v$, adică există lanțul netriangulat $[a, b, \dots, c, d]$, este evident că la schimbarea ordinii de numerotare a muchiilor vom obține un lanț netriangulat $[d, c, \dots, a, b]$.

*Tranzitivitate:* $\boldsymbol{u\tau v \& v\tau t \Rightarrow u\tau t}$, pentru oricare trei muchii $u = [a, b], v = [c, d]$ și $t = [e, f]$. Deoarece există lanț netriangulat din muchia $u$ până în muchia $v$, și din muchia $v$ până în muchia $t$, putem construi lanțul netriangulat ce este reuniunea lanțurilor din muchia $u$ până în muchia $v$, și din muchia $v$ până în muchia $t$, astfel $[ab, \dots, cd, \dots, ef]$ este lanț netriangulat.

**Lema 2.7.** *Dacă vârfurile* $a, b, c$ *sunt adiacente 2 câte 2 și* $S(\{a, b\}) = S(\{b, c\}) = S(\{a, c\})$, *atunci* $[a, b]\tau[b, c]$.

**Demonstrație:** Este suficient să arătăm că dacă sunt satisfăcute condițiile lemei, atunci muchiile $[a, b]$ și $[b, c]$ aparțin aceluiași lanț netriangulat. Din condițiile lemei rezultă că există lanțul netriangulat $[a, b, x_1, x_2, \dots, x_p, c]$. Fie $x_i$ primul vârf al lanțului netriangulat care nu este adiacent cu $c$. Dacă $i$ este număr impar, atunci consecutivitatea: $[a, b, x_1, \dots, x_i, x_{i-1}, c, x_{i-3}, c, \dots, x_1, c, b, c]$ formează lanț netriangulat. Dacă însă $i$ este număr par, atunci consecutivitatea: $[a, b, x_1, \dots, x_i, x_{i-1}, c, x_{i-3}, c, \dots, x_1, c, a, c]$ formează lanț netriangulat. Atunci disjuncția $[a, b]\tau[b, c] \vee [a, b]\tau[a, c]$ este adevărată. Din aceasta și din faptul că $\tau$ este relație de echivalență rezultă expresia $[a, b]\tau[a, c] \vee [a, c]\tau[b, c]$ este adevărată.

**Teorema 2.2.** *Dacă* $G$ *nu conține subgrafuri proprii conexe stabile, atunci oricare două muchii ale sale sunt legate printr-un lanț netriangulat.*

**Demonstrație:** Deoarece graful $G$ nu conține subgrafuri conexe stabile rezultă că $G$ este conex. Ba mai mult, $S([a, b]) = S([c, d]) = G$. Fie $x_1$ un vârf care este adiacent fie cu $a$, sau cu $b$, sau cu muchia $[a, b]$, deoarece $G$ este conex, atunci $S([a, b]) = S(a, x_1) = S(b, x_1)$, atunci conform lemei 2.6 rezultă că există lanț $[a, b, \dots, x_1]$ netriangulat. Fie un alt vârf $x_2$ care este adiacent cu $x_1$ și cu vârful precedent lui, iarăși aplicând condițiile lemei obținem un nou lanț netriangulat. Deoarece $\tau$ este relație de echivalență, în particular tranzitivă, rezultă că $[a, b]\tau[x_1, x_2]$. Continuând procesul până la muchia $[c, d]$, obținem $[a, b]\tau[c, d]$.

**Teorema 2.3.** *Pentru orice subgraf conex* $G(A)$ *al unui graf neorientat, închiderea stabilă este egală cu reuniunea închiderilor stabile ale tuturor muchiilor* $[x, y]$ *din* $G(A)$.

**Demonstrație:** Fie $G(A)$ un subgraf conex al unui graf neorientat $G = (X; U)$ determinat de submulțimea de vârfuri $[x_1, x_2, \dots, x_p] = A$. Alegem o muchie arbitrară $[x_{i_1}, x_{i_2}]$ din $A$. Conform lemei 2.5 putem construi închiderea stabilă a vârfurilor $x_{i_1}, x_{i_2}$. Deoarece $G(A)$ este conex rezultă

că vârful $x_{i_2}$ este adiacent cu un anumit vârf $x_{i_3} \neq x_{i_1}$, ceea ce înseamnă că putem construi închiderea stabilă și pentru muchia $[x_{i_2}, x_{i_3}]$. Conform proprietății P3 a închiderii stabile este adevărată relația: $(\{x_{i_1}, x_{i_2}\}) \cup S(\{x_{i_2}, x_{i_3}\}) = S(\{x_{i_1}, x_{i_2}, x_{i_3}\})$. Continuăm operația anterioară până includem toate vârfurile din G(A) obținem închiderea stabilă $S(A)$. Deoarece se studiază doar grafurile finite, procedura descrisă este finită.

## 2.2. Subgrafuri B-stabile

Determinarea orientărilor tranzitive ale unui graf neorientat se bazează pe existența unui șir de grafuri factor, la construirea cărora un rol decisiv revine subgrafurilor speciale, numite subgrafuri B-stabile. Clasa subgrafurilor B-stabile conține la rândul său, subgrafuri cu diverse structuri pe care le vom examina în continuare. Rezultatele menționate în paragraful dat au fost publicate în diverse lucrări: [16], [46], [48].

**Definiția 2.7.** *Subgraful stabil vid $Q$ se numește maximal dacă acesta nu se conține într-un oarecare alt subgraf stabil vid din graful $G = (X; U)$.*

**Definiția 2.8.** *Subgraful stabil propriu conex $M$ se numește subgraf stabil minimal dacă nu este complet și nu conține alte subgrafuri stabile din graful $G = (X; U)$.*

**Observația 2.1.** *Dacă $M$ este subgraf stabil minimal, atunci $|X_M| \geq 4$.*

În cazul $|X_M| = 2$, subgraful $M$ generează un subgraf complet. Dacă $|X_M| = 3$ în $M$, atunci există subgraf propriu stabil vid sau subgraf complet, ceea ce vine în contradicție cu noțiunea de subgraf stabil minimal.

**Definiția 2.9.** *Subgraful stabil complet $H$ se numește maximal, dacă acesta nu se conține în alt subgraf stabil complet din $G$.*

**Definiția 2.10.** *Subgraful stabil $F$ se numește subgraf B-stabil dacă pentru orice subgraf stabil $M$ din $G = (X; U)$ are loc una din relațiile:*

$$F \subseteq M \text{ sau } F \cap M = \emptyset.$$

De exemplu, în figura 2.5 subgraful stabil determinat de mulțimea de vârfuri $\{x_5, x_6\}$ este subgraf B-stabil, pe când, subgraful determinat de mulțimea de vârfuri $\{x_2, x_3, x_4\}$ nu posedă această proprietate, deoarece în $G$ există subgraful stabil determinat de mulțimea de vârfuri $\{x_1, x_3, x_5, x_6\}$ ce conține o submulțime formată din vârful $\{x_3\}$.

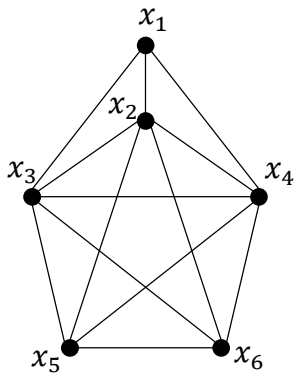



Fig. 2.5. *Graf ce conține subgraf B-stabil, generat de mulțimea de vârfuri* $\{x_5, x_6\}$.

Fie $F$ un careva subgraf stabil din $G = (X; U)$.

**Observația 2.2.** *Dacă* $F \neq M$ *pentru orice subgraf stabil* $M$ *din* $G$ *și se verifică relația* $F \cap M = \emptyset$, *atunci* $F$ *este subgraf B-stabil.*

**Observația 2.3.** *Graful complet* $K_n, n > 2$ *nu conține subgrafuri B-stabile.*

Deoarece în orice graf complet $K_n$, unde $n > 2$ pot fi alese cel puțin două subgrafuri stabile astfel încât intersecția lor nu este vidă, rezultă că nici un subgraf al grafului $K_n$ nu satisface condițiile noțiunii de subgraf B-stabil.

**Lema 2.8.** *Dacă graful neorientat* $G = (X; U)$ *conține subgrafuri stabile, atunci* $G$ *conține cel puțin un subgraf B-stabil.*

**Demonstrație:** Fie $F$ un subgraf stabil al grafului $G = (X; U)$. Dacă în $G$ nu mai există alte subgrafuri stabile, în afară de $F$, atunci sunt respectate toate condițiile pentru ca $F$ să fie subgraf B-stabil. Presupunem că există un alt subgraf stabil $M$, astfel încât $F \neq M$. Sunt posibile următoarele cazuri:

1. $X_F \cap X_M = \emptyset$;
2. $X_F \cap X_M \neq \emptyset$.

Vom analiza fiecare caz în parte:

1. Fie $X_F \cap X_M = \emptyset$. Atunci, conform definiției 2.10, $F$ este subgraf B-stabil în graful $G$.

2. Fie $X_F \cap X_M \neq \emptyset$. Conform lemei 2.1, subgraful generat de mulțimea de vârfuri $X_F \cap X_M$ generează un subgraf stabil. Să examinăm acest subgraf. Dacă acesta nu conține subgraf stabil atunci el va fi subgraful B-stabil căutat. În caz contrar analizăm subgraful obținut la intersecția

celorlalte subgrafuri până vom obține subgraful B-stabil căutat. Deoarece se examinează grafuri finite, procedura menționată va conduce la obținerea rezultatului căutat.

Din cele examinate rezultă că afirmația lemei este adevărată.

Vom nota prin $\mathfrak{S}_G$ mulțimea tuturor subgrafurilor stabile ale grafului $G = (X; U)$, iar prin $\mathfrak{B}_G$ – mulțimea subgrafurilor B-stabile a grafului $G$. Evident, în baza definițiilor 2.1 și 2.10 rezultă relația $\mathfrak{B}_G \subseteq \mathfrak{S}_G$. Ușor ne putem convinge că într-un graf $G$ există subgraf stabil, care însă nu este și B-stabil. De exemplu, în grafurile complete $K_n$ orice mulțime din două vârfuri generează subgraf stabil, dar $K_n$ nu conține subgrafuri B-stabile.

Fie $F$ un subgraf stabil vid maximal al grafului $G$, atunci este adevărată următoarea afirmație.

**Afirmația 2.3.** *Subgraful stabil vid maximal $F$ al grafului $G$ este subgraf B-stabil.*

Demonstrația afirmației se obține ușor, dacă ținem cont de definițiile noțiunilor 2.1 și 2.10, precum și de faptul că intersecția unui subgraf stabil vid maximal cu un alt subgraf stabil este vidă. Aceasta din urmă și înseamnă că orice subgraf stabil vid maximal este B-stabil.

**Lema 2.9.** *Orice subgraf stabil minimal este și subgraf B-stabil.*

**Demonstrație:** Fie $M$ un subgraf stabil minimal. Presupunem contrariul. Fie $H$ un subgraf stabil astfel încât $X_M \cap X_H = A$ și $A$ este o mulțime nevidă. Presupunem că $x \in A$, iar $y$ un vârf din mulțimea $X_M \setminus A$ și $z \in X_H \setminus A$. Sunt posibile două cazuri: a) $z \nsim x$ și b) $z \sim x$. Vom analiza fiecare caz în parte.

Dacă $z \nsim x$, atunci din faptul că $x \in X_M$ rezultă $z \nsim y$. Conform definiției subgrafului stabil minimal $M$ este conex. Deoarece $x \in X_M \cap X_H$, în particular $x$ aparține mulțimii $X_H$ și $z \nsim x$, obținem că $x \nsim X_H$. Deci $H$ nu este subgraf stabil. Contradicție cu presupunerea inițială.

Dacă însă $z \sim x$ și $H$ este subgraf stabil, atunci $z \sim y$. Totuși, conform definiției subgrafului stabil minimal, $M$ nu este subgraf complet. Deci, există un vârf $v$ în mulțimea $X_M$ încât $v \nsim x$. Dar această condiție reproduce situația a). Și în acest caz obținem contradicție cu faptul că $H$ este subgraf stabil. Deci, intersecția mulțimilor $X_M$ și $X_H$ este o mulțime vidă. Deoarece $M$ nu conține subgrafuri proprii stabile și $X_M \cap X_H = \emptyset$ rezultă că $M$ este subgraf B-stabil.

**Afirmația 2.4.** *Dacă $F$ este un subgraf stabil al grafului neorientat $G$, atunci pentru orice vârf $x \in X_G \setminus X_F$ și orice vârf $y \in X_F$, adiacent lui $x$, este adevărată relația:*

$$\deg(x) \geq \deg(y).$$

Deoarece subgraful $F$ este stabil, iar $x \in X_G \setminus X_F$ este adiacent mulțimii $X_F$, rezultă că $X_F \subseteq \Gamma(x)$. Acest fapt demonstrează afirmația de mai sus.

**Observația 2.4.** *Dacă $F$ este subgraf B-stabil atunci:*

$$\deg(x) > \deg(y).$$

**Observația 2.5.** *Dacă $F$ este un subgraf stabil vid maximal, iar $x, y \in X_F$ atunci:*

$$\Gamma(x) = \Gamma(y).$$

**Observația 2.6.** *Dacă subgraful B-stabil $F$ este un subgraf complet maximal atunci*

$$\Gamma(x_i) \cup \{x_i\} = \Gamma(x_j) \cup \{x_j\},$$

*unde* $1 \leq i, j \leq |X_F|$.

Rezultatele descrise mai sus permit elaborarea unui algoritm eficient de complexitate polinomială în raport cu numărul de vârfuri $n$, pentru determinarea subgrafurilor stabile într-un graf neorientat. Căutarea subgrafului B-stabil se face prin aplicarea algoritmului de parcurgere în adâncime a grafului $G$. Cu fiecare nivel în jos vom construi un subgraf potențial candidat la titlul de subgraf B-stabil. Astfel, vom utiliza procedura recursivă $POTENTIAL(x, y)$. Având graful $G$ sub formă de listă de adiacențe a vârfurilor sortate în ordine descrescătoare, se aleg primele două vârfuri adiacente. Se parcurge vecinătatea fiecărui vârf și se analizează dacă vecinătățile vârfurilor parcurse coincid pentru a determina dacă vârfurile fac parte din același subgraf stabil. De asemenea fiecărui vârf procesat i se va atribui o clasă numită $procesat$ ce va avea valorile booleene *TRUE* sau *FALSE*. Dacă unui vârf $x$ pe parcursul rulării procedurii $POTENTIAL$ i se atribuie clasa $procesat\ TRUE$ atunci acesta nu mai este considerat în următoarele etape de rulare a procedurii. În rezultat se returnează subgraful stabil obținut reprezentat în formă de listă de adiacență a vârfurilor.

**Date de intrare:** *Două vârfuri arbitrare $x, y \in X_G$.*

**Date de ieșire:** *Mulțimea de vârfuri ce definește un subgraf B-stabil potențial.*

*Procedura* $POTENTIAL(x, y)$
    *Pentru fiecare* $z \notin \Gamma(x)$:
        *Dacă* $\Gamma(z) = \Gamma(x)$:
            $E \longleftarrow z$;
    *Dacă* $E \neq \emptyset$:
        $E \longleftarrow y$;

*Pentru fiecare* $z \in \Gamma(y)$*:*

*Dacă* $procesat(z) = FALSE$ & $\Gamma(z) \cup \{z\} \subseteq \Gamma(x) \cup \{x\}$:

$procesat(z) \longleftarrow TRUE$;

$P \longleftarrow z$;

$POTENTIAL(x, z)$;

*Dacă* $E = \emptyset$:

*Returnează* $P$;

*Returnează* $E$;

Deoarece în procedura $POTENTIAL(x, y)$ sunt două cicluri, ce sunt parcurse în ordine consecutivă, care parcurg mulțimea de adiacență a unui singur vârf, ușor se observă că timpul de lucru al procedurii este $O(\Delta)$. În caz general procedura dată necesită timpul $O(n)$.

În cele ce urmează va fi descris algoritmul de prelucrare a subgrafurilor potențiale B-stabile prin procedura recursivă $SBS(G)$. Această procedură are la bază algoritmul de parcurgere în adâncime a grafului. În baza fiecărui vârf al grafului reprezentat sub formă de listă de adiacență sortată după gradul fiecărui vârf descrescător se construiește subgraful potențial B-stabil. Pentru fiecare subgraf potențial se verifică dacă acesta nu se include în alt subgraf potențial B-stabil, în caz afirmativ se returnează primul subgraf obținut sub formă de listă de adiacență a vârfurilor.

**Date de intrare:** *Graful* $G$ *sub formă de listă de adiacență.*

**Date de ieșire:** *Subgraful B-stabil* $F$ *sau graful* $G$ *dacă* $G$ *nu conține subgraf stabil.*

*Procedura* $SBS(G)$

*Dacă* $G$ *nu este complet:*

$P \longleftarrow G$;

$SORT(P)$;

*Pentru fiecare* $x \in X_P$:

$procesat(x) \longleftarrow TRUE$;

$E \longleftarrow \emptyset$;

$G \longleftarrow POTENTIAL(x, x)$:

*Dacă* $G \neq P$:

$SBS(G)$;

*Returnează* $G$;

**Teorema 2.4.** *Algoritmul pentru determinarea unui subgraf B-stabil al unui graf neorientat* $G = (X; U)$ *necesită în timpul* $O(n\Delta)$*, unde* $n$ *este numărul de vârfuri în* $X_G$*, iar* $\Delta$ *este gradul maximal al grafului* $G$.

**Demonstrație:** Procedura $SBS(G)$ parcurge consecutiv toate vârfurile grafului $G$ sortat în ordine descrescătoare după gradul vârfurilor. Astfel, la fiecare iterație se apelează procedura $POTENTIAL(x, x)$ care poate fi executată în timpul $O(\Delta)$. Deci, obținem că în total procedura $SBS(G)$ determină un subgraf B-stabil în timpul $O(n\Delta)$.

**Observația 2.7.** Deoarece într-un graf arbitrar $G = (X; U)$, $|X_G| = n$, are loc relația $\Delta(G) \leq n - 1$, obținem că algoritmul descris are complexitate polinomială, de gradul doi $O(n\Delta) = O(n^2)$.

**Observația 2.8.** Pentru grafurile k-regulate algoritmul descris mai sus are o complexitate liniară în raport cu $n$, $O(kn) = O(n)$.

Fie $X_F = \{x_{i_1}, x_{i_2}, \dots, x_{i_p}\}$ o mulțime de vârfuri a grafului neorientat $G = (X; U)$, dacă vârfurile $x_{i_j}$, unde $1 \leq i \leq |X_G|$, iar $1 \leq j \leq p$ și $p < |X_G|$ pot fi ordonate astfel încât să formeze un lanț netriangulat, atunci spunem că mulțimea $X_F$ poate fi acoperită cu un lanț netriangulat.

**Teorema 2.5.** *Un subgraf* $F$ *al grafului neorientat* $G = (X; U)$*, generat de mulțimea de vârfuri* $X_F = \{x_{i_1}, x_{i_2}, \dots, x_{i_p}\}$*, este stabil minimal, dacă și numai dacă mulțimea* $X_F$ *poate fi acoperită cu un lanț netriangulat.*

**Demonstrație:** *Necesitatea*. Conform definiției subgrafului stabil minimal, subgraful $F$ nu conține alte subgrafuri proprii stabile. În baza celor demonstrate în teorema 2.1 (a se vedea consecința 1), rezultă că mulțimea $X_F$ poate fi acoperită cu un lanț netriangulat.

*Suficiența. Fie* $F = (X_F; U_F)$ un subgraf astfel încât mulțimea de vârfuri $X_F$ poate fi acoperită cu un lanț netriangulat. Trebuie să demonstrăm că $F$ este subgraf stabil minimal. Conform propoziției 5.10 din [40] rezultă că $F$ este subgraf stabil. Să demonstrăm că $F$ nu este complet și nu conține alte subgrafuri stabile. Deoarece $X_F$ poate fi acoperită cu un lanț netriangulat, ușor poate fi observat că $F$ nu este subgraf complet.

Să demonstrăm acum că $F$ nu conține alt subgraf stabil. Presupunem contrariul. Fie $M$ este un subgraf propriu stabil al lui $F$. Conform teoremei 2.1, în $M$ există un lanț netriangulat care parcurge toate vârfurile subgrafului. În asemenea caz, lanțul netriangulat ce acoperă subgraful $F$ capătă următoarea formă: $l_F = [x_{i_1}, x_{i_2}, \dots, x_{i_s}, x_i, \dots, x_p]$, unde $s < p, x_{i_j} \in X_M, 1 \leq j \leq s$, iar $s = |X_M|$. Din faptul că $M$ este subgraf stabil și din structura lanțului netriangulat $l_F$ rezultă că

vârful $x_i$ este adiacent cu toate vârfurile $x_{i_j} \in X_M$, unde $1 \leq j \leq s$, $1 \leq i \leq p$, iar $s = |X_M|$. Deci în lanțul $l_F$ obținem un ciclu de lungimea trei format din vârfurile $x_{i_{s-1}}, x_{i_s}, x_i$. Astfel, lanțul ce acoperă mulțimea de vârfuri a subgrafului $F$ nu este triangulat. În baza contradicției obținute rezultă că $F$ nu conține subgrafuri proprii stabile. Deci, $F$ este subgraf stabil minimal.

Rezultatele obținute în paragraful dat ne oferă baza teoretică necesară pentru a formula condițiile necesare și suficiente pentru construirea unei orientări tranzitive, precum și pentru a elabora un algoritm cu ajutorul căruia putem determina numărul de orientări tranzitive.

## 2.3. Construirea orientărilor tranzitive

În baza rezultatelor obținute în paragrafele anterioare, în cele ce urmează vor fi prezentate proprietățile și condițiile necesare pentru orientarea tranzitivă a unui graf neorientat, în special poate fi evidențiată teorema 2.6 [47] prin care se menționează că în orice orientare tranzitivă a unui graf toate arcele adiacente unui subgraf B-stabil sunt orientate într-o singură direcție, fie spre subgraful B-stabil, fie de la subgraf. Un rol important îl are teorema 2.7 [53] prin care se argumentează că orientarea oricărui subgraf B-stabil se face în mod independent de orientarea întregului graf $G$. În baza rezultatelor teoretice menționate în lucrare, au fost formulați algoritmi pentru construirea orientării tranzitive pentru diferite tipuri de subgrafuri B-stabile. Rezultatele obținute au fost publicate într-un șir de lucrări: [18], [47], [50], [53], [54].

**Lema 2.10.** *Dacă $Q$ este un subgraf stabil vid maximal al grafului $G = (X; U)$, iar $z \in X_G \setminus X_Q$ un vârf adiacent mulțimii $X_Q$, atunci în orice orientare tranzitivă $\vec{G} = (X_G; \overrightarrow{U_G})$ a grafului $G$ se respectă una din condițiile:*

1. $[z, x] \in \overrightarrow{U_G}, \forall x \in X_Q;$
2. $[x, z] \in \overrightarrow{U_G}, \forall x \in X_Q.$

**Demonstrație:** Fie $\vec{G}$ o orientare tranzitivă a grafului $G$, și un vârf $z \in X_G \setminus X_Q$ adiacent mulțimii $X_Q$ pentru care există vârfurile $x, y \in X_Q$ astfel încât $[x, z], [z, y] \in \overrightarrow{U_G}$.

Deoarece $\vec{G}$ este o orientare tranzitivă în asemenea caz rezultă că $[x, y] \in \overrightarrow{U_G}$. Deci vârfurile $x$ și $y$ sunt adiacente în graful $G$. Contradicție cu faptul că $Q$ este subgraf vid. Prin urmare presupunerea că, într-o orientare tranzitivă muchiile ce unesc un vârf oarecare $x$ cu toate vârfurile din $Q$ pot fi orientate în direcții diferite față de subgraful stabil vid maximal, este falsă.

**Lema 2.11.** *Dacă $M$ este un subgraf stabil minimal al unui graf neorientat $G = (X; U)$, iar $x \in X_G \setminus X_M$ un vârf adiacent mulțimii $X_M$, atunci în orice orientare tranzitivă $\vec{G} = (X_G; \vec{U}_G)$ se respectă una din condițiile:*

1. $[x, y] \in \vec{U}_G$, pentru orice vârf $y \in X_M$;
2. $[y, x] \in \vec{U}_G$, , pentru orice vârf $y \in X_M$.

**Demonstrație:** Presupunem contrariul. Fie în subgraful stabil $M$ sunt două submulțimi nevide de vârfuri: A= $\{x_{j_1}, x_{j_2}, \dots, x_{j_s}\}$ și $B = \{x_{k_1}, x_{k_2}, \dots, x_{k_t}\}$, unde $s + t = |X_M|$ și $x_{j_v} \neq x_{k_w}$ $1 \leq v \leq s$, $1 \leq w \leq t$. Presupunem că au loc relațiile $[x_{j_v}, z] \in \overrightarrow{U_G}$ și $[z, x_{k_w}] \in \overrightarrow{U_G}$, unde vârful $z \in X_G \setminus X_M$ este adiacent cu toate vârfurile din mulțimea $X_M$. Acest caz este posibil doar atunci când vârfurile $x_{j_v}$ și $x_{k_w}$ sunt adiacente, deoarece în caz contrar orientarea $\vec{G}$ nu va fi tranzitivă.

Deoarece $A$ generează un subgraf al grafului stabil $M$, rezultă că pentru el se verifică una din următoarele două relații:

1) $y \sim x_{j_v}$;
2) $y \nsim x_{j_v}$. (2.4)

pentru orice vârf $y \in X_G \setminus M$, $1 \leq v \leq s$.

Din faptul că vârfurile $x_{j_v}$ și $x_{k_w}$ sunt adiacente rezultă că, pentru subgraful generat de mulțimea de vârfuri A și un vârf $y$ din mulțimea $X_G \setminus X_B$ este adevărată una din afirmațiile din relația (2.4) pentru orice indice $v$, $1 \leq v \leq s$. În asemenea caz obținem că subgraful generat de mulțimea $A \subseteq X_M$ este stabil. Deci, subgraful $M$ conține un alt subgraf stabil $A$. Contradicție cu faptul că $M$ este subgraf stabil minimal.

Rezultatele formulate în lema 2.10 și 2.11 au loc și în cazul general, când avem un subgraf B-stabil al grafului neorientat $G = (X; U)$.

**Teorema 2.6.** *Dacă $F$ este un subgraf B-stabil al unui graf neorientat $G = (X; U)$, iar $x \in X_G \setminus X_F$ un vârf adiacent mulțimii $X_F$, atunci în orice orientare tranzitivă $\vec{G}$ se respectă una din următoarele două condiții:*

1. $[x, y] \in \overrightarrow{U_G}, \forall y \in X_F$;
2. $[y, x] \in \overrightarrow{U_G}, \forall y \in X_F$.

**Demonstrație:** Presupunem contrariul, fie vârfurile $y, z \in X_F$ astfel încât în orientarea tranzitivă $\vec{G}$ are loc relația

$$[\mathrm{y}, \mathrm{x}] \in \overrightarrow{\mathrm{U_G}} \,\&[x, z] \in \overrightarrow{U_G}. \qquad (2.5)$$

În asemenea caz obținem că $[y, z] \in \overrightarrow{U_G}$, deci există muchia $[y, z]$ în $U_G$. Putem observa că pentru orice vârf $t \in X_G \setminus X_F$, $t \neq x$, pentru care $[t, x] \in U_G$ implică existența muchiilor $[t, w], \forall w \in X_F$. Conform observației 2.3 graful $G$ nu este complet, vom presupune că pentru un vârf $s \in X_G$ există $v \in X_G$, unde $v \neq s$, iar $[v, s] \notin \overrightarrow{U_G}$.

Sunt posibile următoarele situații:

a) $s \in X_F, v \in X_F$;

b) $s \in X_F, v \notin X_F$;

c) $s \notin X_F, v \in X_F$;

d) $s \notin X_F, v \notin X_F$.

Să o analizăm pe fiecare în parte:

a) Deoarece $s \in X_F$ rezultă că $[x, s] \in U_G$. Atunci, conform relației (2.5) în graful $G$ se verifică cel puțin una din condițiile $[z, s] \in U_G$ sau $[y, s] \in U_G$. Fără a pierde din generalitate presupunem

$$[z, s] \in U_G. \qquad (2.6)$$

Dacă $v = y$, datorită faptului că orice vârf $t \in X_G \setminus X_F$ adiacent cu $x$ este adiacent și cu orice vârf din $X_F$, inclusiv $z$, rezultă că mulțimea de vârfuri $\{x, z\}$ generează subgraf stabil. Acest fapt contrazice presupunerea că $F$ este subgraf B-stabil.

Dacă însă $v \neq y$, atunci conform relației (2.5) și (2.6) obținem că $[v, x] \in U_G$, ceea ce implică faptul că $[v, z] \in \setminus U_G$. Aceasta la rândul său arată că mulțimea $\{x, z\}$ generează sugraf stabil. Ceea ce de asemenea contrazice că $F$ este subgraf B-stabil.

b) Dacă $x \in X_G$ și $[v, s] \notin U_G$, unde $s \in X_F$, deoarece se respectă relația (2.5) rezultă că $[v, x] \notin U_G$. Dacă însă există un vârf $w \in X_G \setminus X_G$ adiacent cu mulțimea $X_F$, atunci ușor se observă că mulțimea de vârfuri $X_F \setminus \{y, s\} \cup \{x, w\}$ formează subgraf stabil. Este evident că intersecția acestei mulțimi de vârfuri $X_F$ este o mulțime nevidă, ceea ce contrazice presupunerea că $F$ este subgraf B-stabil.

c) Cazul când $s \in X_F$, iar $v \in X_F$ este identic cu b), prin schimbarea notațiilor vârfurilor $s$ și $v$.

d) Deoarece $s, v \in X_G \setminus X_F$, iar $[s, v] \notin U_G$, sunt posibile următoarele cazuri:

1. $s \sim u, \forall u \in X_F$;

2. $s \nsim u, \forall u \in X_F$.

1. Dacă $s \sim u, \forall u \in X_F$, atunci din relația (2.5) obținem că $[y, s] \in \overrightarrow{U_G} \& [s, x] \in \overrightarrow{U_G}$. Astfel, orice vârf adiacent cu $s$ va fi adiacent și cu $u, \forall u \in X_F$.

2. Dacă însă $s \nsim u, \forall u \in X_F$ atunci, conform (2.5), putem observa că $[s, u] \notin U_G$, unde $u \in \Gamma(w), \forall\, w \in X_F$.

Deoarece $[v, s] \notin U_G$, putem aplica aceeași procedură și asupra vârfului $v$. Analizând aceste posibilități obținem că $X \setminus y \cup A$, unde $A = \{x | [x, y] \in U_G, \forall y \in X_F\}$ formează subgraf stabil.

Contradicțiile obținute demonstrează afirmația lemei.

Fie $G = (X; U)$ un graf tranzitiv orientabil și $F$ un subgraf din $G$. Construim o orientare tranzitivă $\vec{G} = (X; \vec{U})$ a grafului $G$. Notăm prin $\vec{F} = (X_F; \overrightarrow{U_F})$ subgraful orientat din $\vec{G}$, determinat de $F$. Evident, are loc egalitatea: $\vec{U}_{\vec{G}} = \vec{U}_{\vec{G}/\vec{F}} \cup \vec{U}_{\vec{F}}$, unde $\vec{U}_{\vec{G}/\vec{F}}$ reprezintă mulțimea tuturor arcelor din orientarea $\vec{G}$ ce nu aparțin subgrafului $\vec{F}$. Vom spune că $F$ este un subgraf independent tranzitiv orientabil în $G$, dacă pentru orice orientare tranzitivă $\vec{F}^*$ a lui $F$, mulțimea de arce $\vec{U}_{\vec{G}/\vec{F}} \cup \vec{U}_{\vec{F}^*}$ determină o orientare tranzitivă $\vec{G}^*$ a grafului $G$. În cele ce urmează, subgraful independent tranzitiv orientabil se va numi ITO-subgraf. Din cele spuse rezultă că, dacă într-o orientare tranzitivă $\vec{G}$ a grafului neorientat $G = (X; U)$ efectuăm o altă orientare tranzitivă $\vec{G}$ a subgrafului independent tranzitiv orientabil $F$, atunci în rezultat obținem o nouă orientare a grafului $G$, care este, de asemenea, orientare tranzitivă.

**Teorema 2.7.** *Subgraful $F$ al grafului tranzitiv orientabil $G = (X; U)$ este B-stabil dacă și numai dacă $F$ este ITO-subgraf al lui $G$.*

**Demonstrație:** *Necesitatea.* Fie $F$ un subgraf B-stabil. Să demonstrăm că F este ITO-subgraf. Conform teoremei 2.5, dacă $x \in X_G \setminus X_F$ și $x \sim X_F$, atunci în orice orientare tranzitivă a grafului $G$ are loc una din următoarele două relații:

a) $[x, y] \in \overrightarrow{U_G}$, pentru orice vârf $y \in X_F$;

b) $[y, x] \in \overrightarrow{U_G}$, pentru orice vârf $y \in X_F$.

Din această situație rezultă că schimbarea direcției arcelor în $\vec{U}_{\vec{F}}$ determină o orientare tranzitivă nouă a grafului $G$, fără a forța schimbarea orientării arcelor în $\vec{U}_{\vec{G}/\vec{F}}$. Deci, F este ITO-subgraf.

*Suficiența.* Fie $F$ este ITO-subgraf. Să demonstrăm că $F$ este B-stabil. Deoarece $F$ este ITO-subgraf rezultă că în orice orientare tranzitivă are loc una din următoarele relații:

a) $[x, y] \in \overrightarrow{U_G}$, pentru orice vârf $y \in X_F$;

b) $[y, x] \in \overrightarrow{U_G}$., pentru orice vârf $y \in X_F$.

unde $x \in X_G \setminus X_F$. Astfel, observăm că $y \sim x$. Deci, subgraful $F$ este stabil. Fie $M$ un subgraf stabil din $G$, încât $X_M \cap X_F \neq \emptyset$. Deoarece $F$ este ITO-subgraf, rezultă că orientarea subgrafului obținut la intersecția $X_M \cap X_F$ nu influențează orientarea $\vec{F}$. Aceasta este posibil doar dacă $F \subseteq M$. Prin urmare, obținem că $F$ este subgraf B-stabil.

Fie $x$ un vârf arbitrar al unui graf orientat, notăm prin $g^+(x)$ numărul de arce ce pornesc din $x$.

**Lema 2.12.** *Pentru orice orientare tranzitivă a grafului complet $K_n$, vârfurile acestuia pot fi renotate astfel încât este adevărată relația:*

$$g^+(x_{i_1}) > g^+(x_{i_2}) > \cdots > g^+(x_{i_n}). \qquad (2.7)$$

**Demonstrație:** Presupunem contrariul. Fie există o orientare tranzitivă $\overrightarrow{K_n}$ a grafului $K_n$ astfel încât $g^+\left(x_{i_j}\right) = g^+(x_{i_k})$. Deoarece graful $K_n$ este complet, fără a pierde din generalitate presupunem că în $\overrightarrow{K_n}$ există arcul $\left[x_{i_j}, x_{i_k}\right] \in \overrightarrow{U_{K_n}}$. Deoarece, conform presupunerii $g^+\left(x_{i_j}\right) = g^+(x_{i_k})$ rezultă că există un vârf $x_{i_s}$ astfel încât $\left[x_{i_k}, x_{i_s}\right] \in \overrightarrow{U_{K_n}}$ și $\left[x_{i_s}, x_{i_j}\right] \in \overrightarrow{U_{K_n}}$. Dar atunci obținem că orientarea $\overrightarrow{K_n}$ nu este tranzitivă. Contradicția obținută demonstrează afirmația lemei.

**Consecința 1.** *În orice orientare tranzitivă a grafului complet $K_n$ este adevărată relația:*

$$g^+(x_{i_1}) = n - 1,\ g^+(x_{i_2}) = n - 2, \ldots,\ g^+(x_{i_n}) = 0. \qquad (2.8)$$

**Consecința 2.** *Dacă $F$ este subgraf stabil complet maximal, atunci conform lemei 2.12 este adevărată relația* (2.8).

**Observația 2.9.** *Dacă un subgraf B-stabil $F$ al grafului neorientat $G$ este complet maximal, atunci conform teoremei 2.6 este adevărată relația* (2.7).

**Lema 2.13.** *Dacă $G = (X; U)$ este un graf tranzitiv orientabil, atunci orice ciclu de lungime impară din $G$ generează cel puțin un triunghi.*

**Demonstrație:** Presupunem că pentru un anumit ciclu de lungime impară $G$, nu generează triunghi. Fie $[x_1, x_2, \ldots, x_p]$ acest ciclu de lungime impară. Deoarece în el nu sunt triunghiuri, orientarea tranzitivă trebuie construită în așa mod încât pentru fiecare vârf $x_i$ al lui ciclului, arcele incidente fie sunt îndreptate spre vârful $x_i$, fie de la vârful $x_i$. Dar urmând această procedură ne

vom confrunta cu situația când muchia $[x_1, x_p]$ oricum nu o vom orienta, orientarea obținută nu va fi tranzitivă. Este necesar ca să existe cel puțin o coardă pentru fiecare ciclu de lungime impară.

**Lema 2.14.** *Dacă $F$ este un subgraf B-stabil al unui graf neorientat $G = (X; U)$, atunci doar una din următoare afirmații este adevărată:*

1) *$F$ este subgraf stabil complet maximal;*
2) *$F$ este subgraf stabil vid maximal;*
3) *$F$ este subgraf stabil minimal.*

**Demonstrație:** Presupunem că $F$ este subgraf complet. Conform definiției 2.10, intersecția mulțimii $X_F$ cu mulțimea de vârfuri a unui subgraf $M$ al lui $G$, astfel încât $X_F \neq X_M$, este vidă. Rezultă că $F$ nu poate fi subgraf propriu stabil al altui subgraf complet. Deci, dacă $F$ este subgraf complet, atunci conform definiției 2.9 acesta este un subgraf stabil complet maximal.

Același principiu îl vom aplica și în cazul când subgraful $F$ este vid. Obținem că $F$ este subgraf stabil vid maximal. Fie subgraful $F$ este conex, nu este complet și nu conține subgrafuri stabile. Conform definiției 2.8 observăm că $F$ este subgraf stabil minimal.

Afirmația inversă lemei demonstrate nu este adevărată. Subgraful stabil complet maximal $F$ determinat de vârfurile $\{x_2, x_3, x_4\}$ nu este B-stabil, deoarece există un alt subgraf stabil care conține vârfuri din $X_F$.

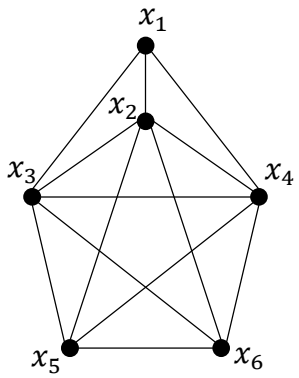



Fig. 2.6. *Graf ce conține subgraf stabil complet maximal determinat de mulțimea de vârfuri $\{x_2, x_3, x_4\}$ care nu este și subgraf B-stabil.*

**Lema 2.15.** *Dacă graful $G$ este tranzitiv orientabil, și nu conține subgrafuri stabile, atunci el conține exact două orientări tranzitive opuse una alteia.*

**Demonstrație:** Deoarece graful $G$ este tranzitiv orientabil rezultă că el conține cel puțin două orientări tranzitive opuse una alteia. Să demonstrăm că el conține cel mult două orientări tranzitive.

Deoarece $G$ nu conține subgraf stabil, este evident că $G$ nu conține nici subgraf B-stabil. Conform teoremei 2.5 avem că orientarea subgrafurilor B-stabile se face în mod independent de orientarea grafurilor care le conține. Putem face concluzia că în $G$ nu există subgraf care ar genera o orientare tranzitivă independentă de $G$. Astfel, orice inversare a orientării unui arc din $\vec{G}$ duce la inversarea orientării tuturor arcelor din $\vec{G}$. Ceea ce duce la construirea orientării $\overleftarrow{G}$.

Următoarea teoremă completează rezultatele din [40], și servește drept suport pentru caracterizarea grafurilor tranzitiv orientabile.

Fie $G$ un graf tranzitiv orientabil, iar $\vec{G} = (X; \vec{U}_G)$ o orientare a muchiilor din $G$. Vom nota prin $\vec{F} = (X_F; \overrightarrow{U_F})$ o orientare tranzitivă a unui subgraf stabil minimal $F$ din $G$, iar prin $\overleftarrow{F} = (X_F; \overleftarrow{U_F})$ orientarea obținută prin inversarea tuturor arcelor din a $\overrightarrow{U_F}$.

Orice graf $G$ poate fi divizat doar în trei tipuri de subgrafuri: subgraf stabil vid maximal, subgraf stabil complet maximal și subgraf stabil minimal. Astfel, subgraful stabil vid maximal nu influențează numărul de orientări tranzitive al grafului $G$, iar subgraf stabil complet maximal este tranzitiv orientabil, rezultă că proprietatea de orientare tranzitivă este legată doar de subgrafurile stabile minimale. În asemenea caz, observăm că teorema 2.7 conturează condițiile necesare și suficiente pentru ca un graf să fie tranzitiv orientabil.

Pornind de la rezultatele obținute și generalizând rezultatul teoremei 5.4 din [40] și în baza teoremei 2.5, putem elabora algoritmul pentru construirea unei orientări tranzitive a unui subgraf stabil minimal dacă acesta este tranzitiv orientabil.

Fie $G = (X; U)$ un graf neorientat cu mulțimea de vârfuri $X_G = \{x_1, x_2, \dots, x_n\}$. Algoritmul prezentat este recursiv, de aceea va fi descris sub formă de pseudocod. În algoritmul de mai jos va fi utilizată funcția $procesat(i, j)$ descrisă în modul următor:

$$procesat(x_i, x_j) = \begin{cases} 0, & dacă\ [x_i, x_j] \notin U_G \\ 1, & dacă\ [x_i, x_j]\ aparține\ orientării\ tranzitive\ \overrightarrow{U_G} \\ -1, & dacă\ [x_i, x_j]\ aparține\ orientării\ tranzitive\ \overleftarrow{U_G} \\ nedefinită, & dacă\ [x_i, x_j] \in U_G, dar\ încă\ nu\ a\ fost\ procesată \end{cases}$$

iar $|procesat(i, j)|$ reprezintă valoarea absolută a funcției $procesat(i, j)$. Graful $G$ este prezentat ca listă de adiacență, astfel încât fiecărui vârf îi este atribuit un șir de indici care arată vârfurile adiacente. Vom nota prin $\deg(x_i)$ gradul vârfului $x_i$. Astfel, $|U_G| = \sum_{i=1}^{n} \deg(x_i)$. Conform teoremei 2.5 orice subgraf stabil minimal poate fi acoperit cu un lanț netriangulat. Mulțimea de

vârfuri a subgrafului stabil minimal se parcurge în ordinea formării unui lanț netriangulat, la fiecare pas muchia obținută se orientează în așa mod ca să fie obținută o orientare tranzitivă.

**Algoritmul de construire a unei orientări tranzitive a subgrafului stabil minimal**

**Date de intrare:** Subgraful $F = (X_F; U_F)$ sub formă de listă de adiacență sortată crescător după gradul fiecărui vârf, de asemenea se indică indicele primei muchii în lista de adiacență.

**Date de ieșire:** O orientare tranzitivă a subgrafului $F$ dacă graful este tranzitiv orientabil, dacă $F$ nu este tranzitiv orientabil, atunci se returnează un indice BLOCAJ cu valoarea 1.

*Procedura* $OrientareSSMin(x_i, x_j)$

*pentru orice* $x_m \in \Gamma(x_i)$ *astfel încât* $[x_m \notin \Gamma(x_j) \vee |procesat(x_j, x_m)| \neq 1]$*:*

*dacă* $procesat(x_i, x_m)$ *este nedefinită:*

$proceast(x_i, x_m) \leftarrow 1;$

$procesat(x_m, x_i) \leftarrow -1;$

$OrientareSSMin(x_i, x_m);$

*altfel*

*dacă* $procesat(x_i, x_m) = -1$:

$procesat(x_i, x_m) \leftarrow 1;$

$BLOCAJ \leftarrow 1;$

$OrientareSSMin(x_i, x_m);$

*pentru orice* $x_m \in \Gamma(x_j)$ *astfel încât* $[x_m \notin \Gamma(x_i) \vee |procesat(x_i, x_m)| <> 1]$*:*

*dacă* $procesat(x_m, x_j)$ *este nedefinită:*

$procesat(x_m, x_j) \leftarrow 1;$

$procesat(x_j, x_m) \leftarrow -1;$

$OrientareSSMin(x_m, x_j);$

*altfel*

*dacă* $procesat(x_m, x_j) = -1$:

$procesat(x_m, x_j) \leftarrow 1;$

$BLOCAJ \leftarrow 1;$

$OrientareSSMin(x_m, x_j);$

*STOP.*

**Teorema 2.8.** *Orientarea tranzitivă a unui subgraf stabil minimal poate fi construită în timpul* $O(\Delta)$*, unde* $\Delta$ *este gradul maximal al grafului* $G$.

**Demonstrație:** Un factor crucial în analiza timpului de lucru al algoritmului de mai sus îl reprezintă timpul necesar pentru a accesa valoarea din funcția $procesat(x_i, x_j)$. În caz general timpul necesar pentru a determina valoarea $procesat(x_i, x_j)$ este de $O(ged(x_i))$ prin parcurgerea mulțimii $\Gamma(x_i)$. Dar dacă indicele temporar în recursie este în vecinătatea vârfului $x_i$, atunci timpul pentru stabilirea valorii $procesat(x_i, x_j)$ este o valoarea constantă. Vom considera prima iterație a procedurii $OrientareSSMin(x_i, x_j)$. Deoarece procedura se împarte în două cicluri, iar valoarea funcției $procesat(x_i, x_j)$ se calculează în timp constant, rezultă că timpul execuției procedurii este de $O(\deg(x_i) + \deg(x_j))$.

Deci, în caz general timpul necesar pentru a construi o orientare tranzitivă a unui subgraf stabil minimal este $O(\Delta)$, unde $\Delta$ este gradul maximal al grafului $G$.

Este cunoscut faptul că orice graf tranzitiv orientabil este graf de comparabilitate [88], deci în așa caz orice graf complet $K_p$ este tranzitiv orientabil [41], și se cunoaște că $\tau(K_p) = p!$. În așa caz avem că numărul de orientări tranzitive ale unui subgraf stabil complet maximal $H$ este $|X_H|!$.

În baza lemei 2.10 poate fi elaborat un algoritm de construire a unei orientări tranzitive a unui subgraf B-stabil complet maximal. Lema 2.12 oferă suportul teoretic pentru orientarea subgrafului complet maximal. Pentru obținerea unei orientări tranzitive este necesar să se construiască o permutare a vârfurilor subgrafului stabil complet maximal. Pentru economisirea timpului de procesare și a memoriei, subgraful B-stabil cercetat va fi prezentat sub formă de matrice de adiacență. Se parcurge doar partea superioară a diagonalei principale a matricei de adiacență, și toate celulele obținute la intersecția liniei și coloanei matricei vor primi valoarea 1. Pentru obținerea orientării opuse putem atribui valoarea -1.

**Algoritmul de construire a orientării tranzitive a subgrafului stabil complet maximal.**

**Date de intrare:** Subgraful B-stabil complet $F$ reprezentat în formă de matrice de adiacență sortată după indicele vârfurilor.

**Date de ieșire:** O orientare tranzitivă $\vec{F}$ a subgrafului $F$.

Procedura $K_n TRO(F)$

$\vec{F} = \emptyset$

$For\ i \coloneqq 1\ to\ n-1\ step\ 1\ do$:

$For\ j \coloneqq i+1\ to\ n\ step\ 1\ do$:

$[x_i, x_j] \in \overrightarrow{U_F}$;

$EndFor$;

$EndFor$;

Returnează $\vec{F}$;

**Teorema 2.9.** *Construirea unei orientări tranzitive a unui subgraf stabil complet maximal $F$ poate fi efectuată în timpul $O(n\Delta)$, unde $n$ este numărul de vârfuri ale subgrafului $F$, iar $\Delta$ este gradul maximal al subgrafului.*

**Demonstrație:** Algoritmul dat parcurge consecutiv toate vârfurile subgrafului $F$. Astfel, la fiecare iterație se orientează toate muchiile de forma $[x_i x_j]$, unde $x_j \in \Gamma(x_i)$. Această operație poate fi executată în timpul $O(\Delta)$. Deci, obținem că în total algoritmul construiește o orientare tranzitivă în timpul $O(n\Delta)$.

**Observația 2.10.** *Conform lemei 2.10, acest algoritm garantează construirea unei orientări tranzitive. Pentru a obține o altă orientare tranzitivă este necesar să alegem o altă permutare a vârfurilor subgrafului $B$-stabil $F$.*

## 2.4. Numărul de orientări tranzitive

Determinarea numărului de orientări tranzitive ale unui graf are un rol important în studierea clasei de grafuri tranzitiv orientabile. În paragraful dat va fi descrisă o clasă specială de grafuri numită grafuri factor. Grafurile factor reprezintă ideea principală în procesul de determinare a numărului de orientări tranzitive precum și construirea unei astfel de orientări. O importanță deosebită în procesul de construire a numărului de orientări tranzitive îl are lema 2.23, prin care se menționează că într-un graf neorientat, toate șirurile complete de grafuri factor au aceeași lungime. În cele ce urmează în lema 2.26 este descrisă formula pentru determinarea numărului de orientări tranzitive într-un graf. În baza rezultatelor teoretice din acest paragraf, precum și cele din secțiunile anterioare este formulat algoritmul pentru construirea orientării tranzitive a unui graf. Rezultatele obținute au fost publicate în [18], [45], [47], [49], [50], [54].

**Lema 2.16.** *Dacă graful $\vec{G}$ este tranzitiv orientat, atunci și graful $\overleftarrow{G}$ este tranzitiv orientat*

**Demonstrație:** Fie graful $\vec{G} = (X; \vec{U})$ este tranzitiv orientat, și graful $\overleftarrow{G} = (X; \overleftarrow{U})$ care se obține la schimbarea orientării tuturor arcelor din $\vec{G}$. Presupunem că $\overleftarrow{G}$ nu este tranzitiv. Aceasta înseamnă că există vârfurile $x, y, z \in X$ astfel încât $[x, y] \in \overleftarrow{U}$ & $[y, z] \in \overleftarrow{U}$, dar $[x, y] \notin \overleftarrow{U}$. Atunci în $\vec{G}$ am avea: $[z, y] \in \vec{U}$&$[y, x] \in \vec{U}$, dar $[z, x] \notin \vec{U}$. Aceasta contrazice faptul că $\vec{G}$ este tranzitiv. Ceea ce demonstrează lema.

**Consecința 1:** *Dacă $G = (X; U)$ este un graf tranzitiv orientabil, atunci pentru el există cel puțin două orientări tranzitive.*

**Consecința 2:** *Dacă $G = (X; U)$ este un graf tranzitiv orientabil, atunci numărul tuturor orientărilor tranzitive ale lui $G$ este par.*

**Demonstrație:** Fie $\overrightarrow{G_1} = (X_G; \vec{U}_G)$ o orientare tranzitivă a grafului $G$. Vom nota prin $\overleftarrow{G_1}$ orientarea tranzitivă obținută prin inversarea tuturor arcelor din $\overrightarrow{G_1}$, iar prin $\overrightarrow{G_2}$ o altă orientare tranzitivă astfel încât $\overrightarrow{G_2} \neq \overrightarrow{G_1}$. Demonstrația consecinței este echivalentă cu demonstrația următoarelor două afirmații:

1. Dacă $\overleftarrow{G_2} = \overrightarrow{G_1}$, atunci $\overrightarrow{G_2} = \overleftarrow{G_1}$;
2. Dacă $\overrightarrow{G_2'}$ este o orientare tranzitivă obținută prin inversarea unei submulțimi de arce din $\overrightarrow{G_2}$, și $\overrightarrow{G_2'} = \overrightarrow{G_1}$, atunci $\overleftarrow{G_2'} = \overleftarrow{G_1}$.

Să demonstrăm fiecare afirmație în parte:

1. Presupunem contrariul, fie că există o orientare tranzitivă $\overrightarrow{G_3}$ astfel încât $\overrightarrow{G_2} = \overrightarrow{G_3}$, unde $\overrightarrow{G_3} \neq \overrightarrow{G_1}$ și $\overrightarrow{G_3} \neq \overleftarrow{G_1}$. Deoarece orientările $\overrightarrow{G_2}$ și $\overrightarrow{G_3}$ sunt egale, atunci și invesele lor tot sunt egale. Deci, are loc relația: $\overleftarrow{G_2} = \overleftarrow{G_3}$. Dar din presupunerea afirmației $\overleftarrow{G_2} = \overrightarrow{G_1}$ rezultă că $\overleftarrow{G_3} = \overrightarrow{G_1}$. La inversarea tuturor arcelor în orientările $\overleftarrow{G_3}$ și $\overrightarrow{G_1}$ obținem că $\overrightarrow{G_3} = \overleftarrow{G_1}$. Fapt ce contrazice presupunerea afirmației. Demonstrația afirmației respective ne permite să facem concluzia că inversarea tuturor arcelor într-o orientare tranzitivă generează doar o singură orientare.

2. Fie $\overrightarrow{G_3}$ o orientare tranzitivă astfel încât $\overleftarrow{G_2'} = \overrightarrow{G_3}$, unde $\overrightarrow{G_3} \neq \overrightarrow{G_1}$ și $\overrightarrow{G_3} \neq \overleftarrow{G_1}$. În baza afirmației de mai sus este adevărat că $\overrightarrow{G_2'} = \overleftarrow{G_3}$. În baza presupunerii afirmației rezultă că $\overleftarrow{G_3} = \overrightarrow{G_1}$. Contradicția obținută demonstrează afirmația 2.

În baza afirmațiilor 1 și 2 consecința este demonstrată.

Există clase de grafuri care admit cel mult două orientări tranzitive. Adică pentru astfel de grafuri sau nu există orientări tranzitive, sau dacă acestea există, atunci ele sunt doar două.

**Lema 2.17.** P*entru orice graf bipartit există exact două orientări tranzitive.*

**Demonstrație:** Pentru a demonstra lema dată este necesar să arătăm că pentru orice graf bipartit există 2 orientări tranzitive, și că există doar două astfel de orientări.

a) Fie graful bipartit $G = (X_1, X_2, U)$. Din definiția grafului bipartit dacă vom orienta muchiile din submulțimea de vârfuri $X_1$ spre submulțimea $X_2$, atunci vom obține o orientare tranzitivă. Deci graful $G$ este tranzitiv orientabil.

b) Din a) și din lema 2.15 rezultă că pentru graful $G$ există cel puțin două orientări tranzitive. Mai exact, putem construi orientare tranzitivă orientând toate muchiile din mulțimea de vârfuri $X_1$ spre submulțimea $X_2$ sau invers, adică din submulțimea $X_2$ spre $X_1$.

c) Presupunem că există o altă orientarea diferită de cele descrise în b), adică orientând muchiile grafului astfel încât pentru un anumit vârf $x \in X_1$ (sau $x \in X_2$) să aibă loc relația: $\mathrm{g}(x)^+ = k$ & $\mathrm{g}(x)^- = \mathrm{g}(x) - k$. Dar în asemenea caz apare necesitatea de a construi un lanț de lungimea 3, dar în grafurile bipartite acest lucru este imposibil.

În conformitate cu cele arătate mai sus, putem face concluzia că într-un graf bipartit pentru orice vârf $x$ toate muchiile fie sunt orientate spre el, sau de la el.

**Consecință:** *Orice graf ce nu conține cicluri de lungime impară admite exact două orientări tranzitive opuse una alteia.*

**Demonstrație:** Fie graful $G = (X; U)$ nu conține cicluri de lungime impară. Atunci graful $G$ sau este arbore, sau conține cicluri de lungime pară. Deoarece orice arbore poate fi prezentat ca și un graf bipartit, atunci conform teoremei 2.5, $G$ admite exact două orientări tranzitive. Dacă însă graful $G$ admite cicluri doar de lungime pară, orientând muchiile în așa mod ca pentru orice vârf $x \in G$ toate muchiile să fie orientate sau spre el, sau de la el. Deoarece graful nu admite cicluri de lungime impară, atunci o astfel de orientare este tranzitivă. O altfel de orientare pentru grafurile ce admit cicluri de lungime pară nu este posibilă, deoarece în caz contrar ar fi necesară apariția ciclurilor de lungimea 3, fapt care ar contrazice afirmația.

O problemă importantă ține de modul de orientare tranzitivă a anumitor clase de grafuri, cum ar fi orientarea tranzitivă a arborilor. Poate fi menționat faptul că orientarea tranzitivă a unui arbore poate fi determinată în mod univoc de orientarea unei muchii suspendate $[x, y] \in U$, și anume:

a) Dacă $z$ este un vârf al arborelui $T$, ce se află la o distanță pară de la $x$, atunci toate muchiile incidente lui $z$, în orientarea tranzitivă a arborelui $T$ vor primi orientare opusă față de muchia $[x, y] \in U$.

b) Dacă însă $z$ se află la o distanță impară de la $x$, atunci toate muchiile incidente lui $z$ în orientarea tranzitivă a arborelui $T$ vor primi aceeași orientare ca și muchia $[x, y] \in U$

Orientarea tranzitivă a ciclurilor de lungime pară poate fi construită în mod univoc pornind de la ideea că în aceste cicluri nu există cicluri de lungimea 3, adică pentru oricare trei vârfuri $x_1, x_2, x_3 \in C_n$ astfel încât $[x_1, x_2] \in C_n \& [x_2, x_3] \in C_n$ nu are loc implicația $[x_1, x_{23}] \notin C_n$. Pentru a exclude situația dată, orientările muchiilor se fac în așa mod ca să satisfacă una din relațiile:

1. $g(x)^+ = 0 \ \& \ g(x)^- = g(x)$;
2. $g(x)^+ = g(x) \ \& \ g(x)^- = 0$.

Fie $F$ un subgraf B-stabil al grafului $G = (X; U)$. Vom nota prin $G/F$ graful ce se obține din $G$, după următoarele reguli:

1. subgraful stabil $F$ se înlocuiește printr-un vârf $x'$;
2. toate muchiile $[x, z], \forall x \in X_F, z \in X \setminus X_F$ se înlocuiesc cu muchia $[x', z]$.

Graful $G/F$ se va numi **graf factor** corespunzător subgrafului B-stabil $F$.

Vom nota prin $G^0$ graful $G$, iar prin $G^1 = G^0/F_0$. Dacă acest graf factor mai conține un subgraf B-stabil, atunci continuăm operația descrisă mai sus până când obținem un alt graf factor ce nu conține subgrafuri B-stabile. Conform celor descrise pentru un graf arbitrar $G = (X; U)$, obținem șirul de grafuri factor neorientate:

$$G^0 = G, G^1 = G^0/F_0, G^2 = G^1/F_1, \dots, G^k = G^{k-1}/F_{k-1} \tag{2.10}$$

cu proprietățile:

1. $F_i$ este subgraf B-stabil în graful $G^i$, unde $0 \leq i \leq k-1$;
2. ultimul graf $G^k = G^{k-1}/F_{k-1}$ nu conține subgrafuri B-stabile.

Vom numi șirul (2.10) construit conform regulilor descrise mai sus, **șir complet de grafuri factor** al grafului $G$. Menționăm că șirul dat conține $k+1$ grafuri neorientate. Obținerea unui șir complet de grafuri factor este un proces iterativ. Reprezentarea schematică de construire a șirului este reprezentată în figura 2.7.

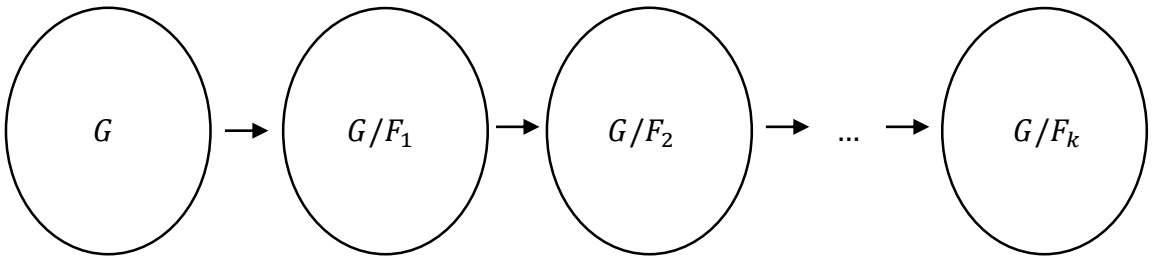



Fig. 2.7. *Reprezentarea schematică a procesului iterativ de obținere a șirului complet de grafuri factor.*

Este evident că într-un graf pot exista mai multe subgrafuri B-stabile care nu se intersectează. În graful din figura 2.8 vârfurile $x_3, x_4$ și $x_5, x_6$ generează subgrafurile B-stabile $\{x_3, x_4\} - SBS$ și $\{x_5, x_6\} - SBS$ intersecția cărora este vidă.

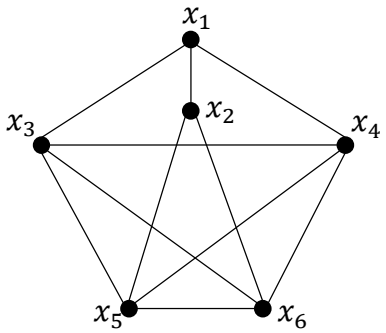



Fig. 2.8. *Graf cu subgrafuri B-stabile* $\{x_3, x_4\} - SBS$ *și* $\{x_5, x_6\} - SBS$, *intersecția cărora este vidă.*

Pentru a determina numărul de orientări tranzitive este necesar să aplicăm operația de factorizare asupra fiecărui graf factor atât timp cât graful factor va conține subgraf stabil. Astfel, vom obține un șir de grafuri factor. În mod natural apar întrebările: 1) care din aceste subgrafuri trebuie de ales pentru a obține un graf factor; 2) în ce mod alegerea unui subgraf influențează numărul de orientări tranzitive. În lemele prezentate mai jos ne propunem să dăm răspuns la aceste întrebări.

**Lema 2.18.** *Dacă*

$$G^0 = G, G^1 = G^0/A_0, G^2 = G^1/A_1, \dots, G^k = G^{k-1}/A_{k-1} \tag{2.11}$$

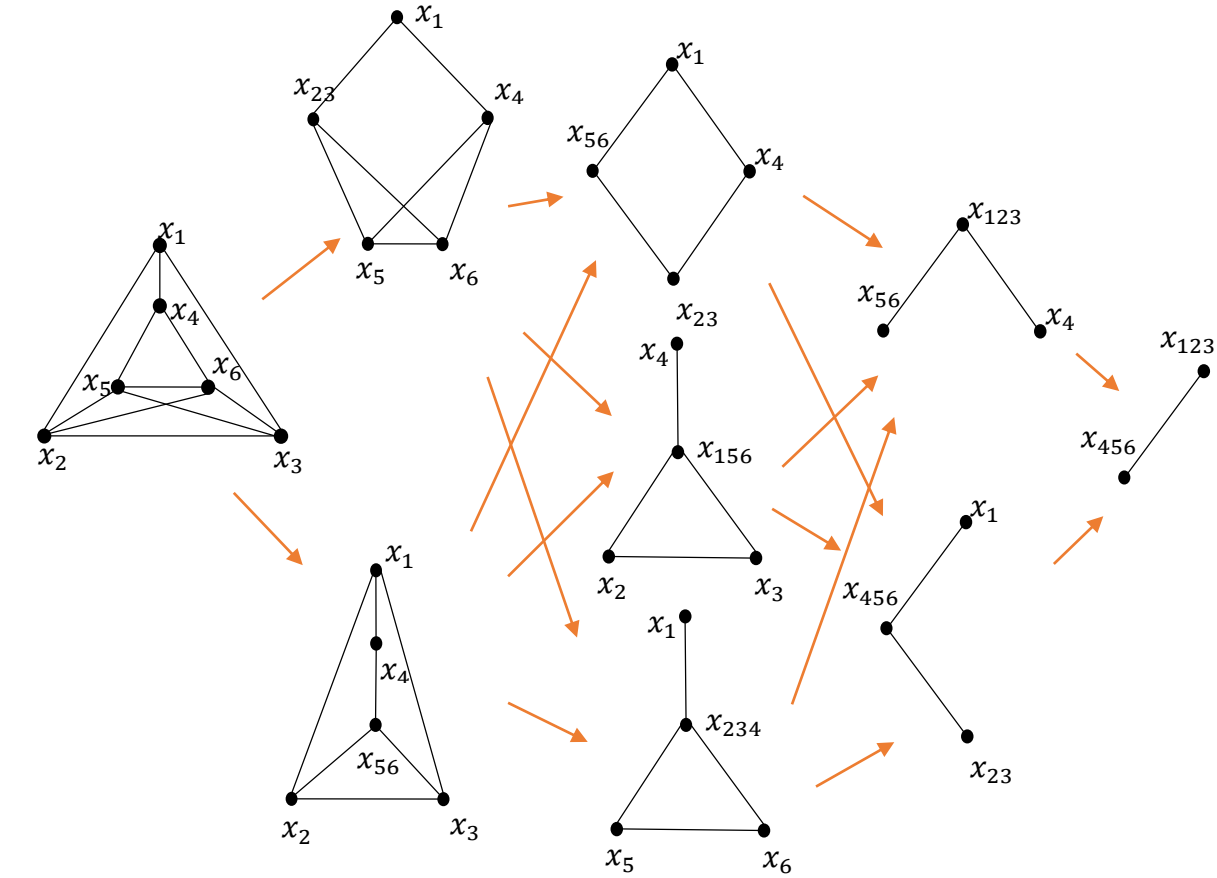
$$G^0 = G, G^1 = G^0/B_0, G^2 = G^1/B_1, \dots, G^t = G^{t-1}/B_{t-1} \tag{2.12}$$

*sunt două șiruri complete de grafuri factor ale grafului* $G$, *atunci are loc egalitatea* $k = t$.

**Demonstrație:** Pentru a demonstra afirmația lemei este suficient să arătăm că indiferent de tipul de subgrafuri B-stabile, numărul de iterații în procesul de construire a șirului complet de grafuri factor este același.

Conform teoremei 2.7 putem spune că subgraful B-stabil $A$ nu influențează operația de factorizare aplicat asupra subgrafului B-stabil $B$. Dacă în graful $G$ aplicăm operația de factorizare asupra subgrafului $A$, rezultă că în graful factor rezultant $G/A$ neapărat va exista și subgraful $B$. Aceeași logică este valabilă și atunci când aplicăm operația de factorizare asupra subgrafului $B$. Indiferent cum alegem un subgraf B-stabil care urmează a fi factorizat, neapărat celelalte subgrafuri B-stabile vor fi în șirul de grafuri factor.

Următorul lucru necesar pentru a demonstra afirmația dată este să arătăm că între grafurile $G/A$ și $G/B$ există o aplicație bijectivă. Cu alte cuvinte, trebuie să arătăm că factorizarea unui subgraf B-stabil $A$ nu generează alte subgrafuri B-stabile $B_1, B_2, \dots, B_p$ ce nu pot fi obținute la factorizarea subgrafului B-stabil $B$.




c

Fig. 2.9 *Procedura de construire a șirului complet de grafuri factor (în mod neunivoc) pentru un graf neorientat* $G$.

Deoarece în urma operației de factorizare în graful factor obținut apare un vârf nou, acesta la rândul său poate fi implicat în factorizarea unui subgraf B-stabil. Acest fapt indică la aceea că orice subgraf B-stabil obținut la o anumită etapă de construire a șirului de grafuri factor poate fi obținut la o altă etapă de construire a șirului, în dependență de ce subgraf B-stabil a fost ales pentru factorizare. Putem spune că în construcția șirului de grafuri factor poate fi schimbată ordinea apariției grafurilor nu și a numărului lor. Astfel, lungimea tuturor șirurilor complete de grafuri factor este aceeași, independent de alegerea subgrafului stabil la orice iterație pentru construirea următorului graf factor.

**Consecință:** *Numărul de orientări tranzitive ale unui graf neorientat nu depinde de modul de alegere a subgrafului B-stabil.*

Vom ilustra lema 2.18 printr-un exemplu. În graful din figura 2.9 mulțimea de vârfuri $\{x_2, x_3\}$ și $\{x_5, x_6\}$ generează subgrafuri B-stabile. Prin aplicarea procedeului de factorizare asupra subgrafurilor date obținem grafurile factor $\{x_1, x_{23}, x_4, x_5, x_6\} - SBS$ și respectiv $\{x_1, x_2, x_3, x_4, x_{56}\} - SBS$. După cum poate fi observat, subgraful care nu a fost factorizat la primul pas este prezent în graful factor obținut. Aplicând operația de factorizare, obținem același graf factor determinat de mulțimea de vârfuri $\{x_1, x_{23}, x_4, x_{56}\}$. În acest graf există subgrafurile B-stabile: $\{x_1, x_{23}\} - SBS$ și $\{x_4, x_{56}\} - SBS$. Aplicând aceeași operație asupra acestui graf factor, ajungem la un caz similar descris mai sus. În final obținem graful factor determinat de mulțimea de vârfuri $\{x_{123}, x_{456}\}$ asupra căruia nu mai poate fi aplicată operația de factorizare.

Este cunoscut faptul că un graf poate avea mau multe orientări tranzitive. Prin utilizarea șirului complet de grafuri factor poate fi descrisă metoda de construire a unei orientări tranzitive specifice sau metoda de trecere de la o orientare tranzitivă la alta. Prin orientarea muchiilor unui subgraf B-stabil dintr-un graf factor poate fi obținută o orientare tranzitivă, iar prin schimbarea orientării tranzitive a unui subgraf asupra căruia poate fi aplicată operația de factorizare determină o orientare tranzitivă nouă. În asemenea caz, fiecare șir complet de grafuri factor, generează mai multe orientări tranzitive, iar toate șirurile complete de grafuri factor permit construirea tuturor orientărilor tranzitive ale unui graf neorientat.

Următoarele rezultate se vor referi la determinarea numărului de orientări tranzitive cu ajutorul operației de factorizare.

**Lema 2.19.** *Dacă $G = (X; U)$ este un graf tranzitiv orientabil și $F$ un subgraf stabil vid maximal din $G$, iar $G/F$ graful factor corespunzător, atunci numărul de orientări tranzitive ale grafului $G$ este egal nu numărul de orientări tranzitive ale grafului $G/F$.*

**Demonstrație:** Pentru a demonstra lema este suficient să arătăm că orice orientare tranzitivă a lui $G$ determină o orientare tranzitivă în $G/F$, și invers.

*Necesitatea.* Fie $\vec{G} = (X_G; \overrightarrow{U_G})$ o orientare tranzitivă a grafului $G$. Notăm prin $z_F$ vârful grafului $G/F$ ce corespunde subgrafului stabil F. Vom orienta muchiile din $G/F$. Fie $[x, y]$ o muchie arbitrară din $G/F$. Dacă $x, y \neq z_F$, atunci orientare muchiei $[x, y]$ o păstrăm ca și în $\vec{G}$. Dacă însă $y = z_F$, atunci conform lemei 2.10, orientarea muchiei $[x, z_F]$ va coincide cu orientarea oricărui arc $[x, t], t \in X_F$ din orientarea tranzitivă $\vec{G}$.

Să demonstrăm că orientarea obținută a grafului $G/F$ este la fel tranzitivă.

Fie vârfurile $a, b, c \in X_{G/F}$, astfel încât în orientarea grafului $G/F$ construită conform operației de mai sus, există arcele $[a, b]$ și $[b, c]$. Dacă $a, b, c \neq z_F$, atunci rezultă că arcele sunt orientate în aceeași direcție precum în $\vec{G}$, deci rezultă că există și arcul $[a, c]$. Fie $a = z_F$, atunci din $F$ luăm un vârf arbitrar $x$, obținem că arcul $[x, b]$ aparține orientării tranzitive $\vec{G}$. Obținem că există și arcul $[x, c]$. Conform modelului descris mai sus ajungem la concluzia că există și arcul $[a, c]$. Aceeași operație o aplicăm și în cazul când $b = z_F$ sau $c = z_F$. Obținem că orientarea grafului $G/F$ obținută după modalitatea descrisă mai sus este tranzitivă.

Faptul că orientării tranzitive $\vec{G}$ îi corespunde o singură orientare tranzitivă a grafului $G/F$ rezultă din operația descrisă mai sus.

*Suficiența.* Fie o orientare tranzitivă $\overrightarrow{G/F}$ a grafului factor $G/F$. Să demonstrăm că pornind de la aceasta, putem obține o orientare tranzitivă a grafului $G$ și doar una singură.

Fie $z_F$ vârful grafului $G/Q$ ce corespunde subgrafului stabil $F$. Luăm o muchie arbitrară $[x, y]$ din $G/F$. Dacă $x, y \neq z_F$, atunci orientăm muchia $[x, y]$ din $G$ în aceeași direcție ca și orientarea arcului din $\overrightarrow{G/F}$. Dacă însă $y = z_F$, atunci $y$ în $G$ substituie vârfurile $\{y_1, y_2, \dots, y_p\}$ din $F$, $p = |X_F|$ și conform lemei 2.10 orientarea arcului $[x, y_i]$, $i = \overline{1, p}$ în $G$ va coincide cu orientarea $[x, y]$ din $\overrightarrow{G/F}$.

Ușor se verifică faptul că orientarea tranzitivă a grafului $G$, obținută după operația descrisă mai sus, este tranzitivă.

Faptul că unei orientări tranzitive $\overrightarrow{G/F}$ a grafului $G/F$ îi corespunde doar o singură orientare tranzitivă în $G$ rezultă din operația descrisă mai sus de construire a orientării tranzitive a lui $G$.

Vom nota prin $\tau(G)$ numărul de orientări tranzitive ale grafului $G$.

**Lema 2.20.** *Pentru graful $G$, asupra căruia nu poate fi aplicată operația de factorizare, este adevărată una din relațiile:*

1. $\tau(G) = 0$;
2. $\tau(G) = 2$;
3. $\tau(G) = |X_G|!$.

**Demonstrație:** Dacă graful $G$ nu este tranzitiv orientabil atunci evident $\tau(G) = 0$. Dacă însă $G$ este tranzitiv orientabil și pentru el nu poate fi aplicată operația de factorizare, atunci are loc una din următoarele afirmații:

1. $G$ nu conține subgrafuri proprii stabile;
2. $G$ este graf complet.

Fie graful tranzitiv orientabil $G$ nu conține subgrafuri proprii stabile. Conform lemei 2.15, obținem $\tau(G) = 2$.

Fie $G$ este graf complet. Cunoaștem că graful complet nu conține subgrafuri B-stabile. Prin urmare, asupra lui nu poate fi aplicată operația de factorizare. Conform [41], numărul de orientări tranzitive ale grafului complet cu $n$ vârfuri este $n!$.

Din cele examinate, rezultă corectitudinea afirmației lemei 2.20.

Cunoaștem că subgrafurile stabile vide maximale nu influențează numărul de orientări tranzitive (a se vedea lema 2.19). Formal vom considera că orice subgraf vid maximal $Q$ generează o orientare tranzitivă (vom scrie $\tau(Q) = 1$).

În cele ce urmează vom descrie formula recurentă pentru determinarea numărului de orientări tranzitive într-un graf neorientat. Fie $G^0 = G, G^1 = G^0/F_0, G^2 = G^1/F_1, \dots, G^{p+1} = G^p/F_p$ șirul complet de grafuri factor astfel încât se verifică relațiile:

a) $G^{p+1}$ nu conține subgrafuri B-stabile;
b) $F_i$ este un subgraf B-stabil în $G^i$, $0 \leq i \leq p$.

Conform teoremei 2.7, orientarea tranzitivă a unui subgraf B-stabil nu influențează orientarea tranzitivă a întregului graf. Aceasta este valabilă și în cazul determinării numărului de orientări tranzitive. Acesta poate fi calculat prin înmulțirea numărului de orientări tranzitive a tuturor subgrafurilor B-stabile din șirul complet de grafuri factor.

Ținând cont de rezultatele formulate în lemele 2.19 și 2.20 este adevărată următoarea

**Lema 2.21.** Numărul de orientări tranzitive ale unui graf neorientat $G$ se calculează în mod recurent prin formula:

$$\tau(G) = \tau(G^{p+1}) \prod_{i=1}^{p} \tau(F_i). \qquad (2.13)$$

Conform teoremei 2.5 orientarea tranzitivă a subgrafurilor B-stabile se face în mod independent de orientarea întregului graf, rezultă că numărul de orientări ale grafului factor obținut

se înmulțește cu numărul de orientări tranzitive ale subgrafului asupra căruia a fost aplicată operația de factorizare.

În continuare vom descrie algoritmul de construire a șirului complet de grafuri factor și de calculare a numărului de orientări tranzitive într-un graf neorientat $G$. Pentru determinarea numărului de orientări tranzitive în mod iterativ se construiește șirul complet de grafuri factor. Deoarece lungimea tuturor șirurilor este identică, pentru construirea unui graf factor se alege primul subgraf B-stabil obținut în procesul de factorizare. La fiecare subgraf B-stabil determinat, în dependență de tipul acestuia, se calculează numărul de orientări tranzitive. Dacă un subgraf B-stabil nu este tranzitiv orientabil, atunci algoritmul returnează valoarea $0$. Algoritmul rulează până când subgraful B-stabil coincide cu ultimul graf factor, ceea ce indică lipsa subgrafurilor B-stabile.

**Algoritmul de construire a șirului complet de grafuri factor și determinarea numărului de orientări tranzitive ale unui graf neorientat:**

**Date de intrare:** Graful neorientat $G$.

**Date de ieșire:** Numărul de orientări tranzitive $\tau(G)$, dacă $\tau(G) = 0$, atunci graful $G$ nu este tranzitiv orientabil.

Procedura $\tau_TRO(G)$

$\tau(G) \coloneqq 1, i \coloneqq 0, G/F_0 = G, F_0 = 0;$

$While\ \boldsymbol{F_i} <> \boldsymbol{G/F_i}\ then$:

$i \coloneqq i + 1;$

$F_i \coloneqq SBS(G/F_i);$

$\tau(G) \coloneqq \tau(G) \cdot \tau(F_i);$

$if\ \boldsymbol{\tau(G) = 0}\ then$:

$Return$ – Graful $G$ nu este tranzitiv orientabil;

$EndIf$;

Se determină graful factor $G/F_i$;

$EndWhile$;

**Teorema 2.10.** *Șirul complet de grafuri factor și calcularea numărului de orientări tranzitive ale unui graf neorientat $G$ se face în timp $O(np\Delta)$, unde $n$ este numărul de vârfuri al grafului $G$, $p$ - lungimea șirului complet de grafuri factor, iar $\Delta$ - gradul maxim al grafului $G$.*

**Demonstrație:** Pasul 2 poate fi executat în timpul $O(n\Delta)$ conform procedurii $BSB$ descrisă în paragraful precedent. La rândul său, pasul 7 presupune parcurgerea tuturor vârfurilor grafului. Prin urmare acest pas este realizat în timp $O(n)$. Vom nota prin $p$ lungimea șirului complet de grafuri factor a grafului $G$. Atunci, observăm că algoritmul de mai sus se execută în $p$ pași. Astfel, timpul de prelucrare total fiind $O(np\Delta)$. În caz general numărul $p$ nu depășește $\frac{n}{2}$ ,iar $\Delta(G) \leq n - 1$. Deci, algoritmul pentru determinarea numărului de orientări tranzitive necesită timpul $O(n^3)$.

Cunoscând șirul complet de grafuri factor, cu un efort nu prea mare putem construi o oarecare orientare tranzitivă a grafului $G$. Pentru aceasta, se parcurge șirul complet de grafuri factor în ordine inversă și la fiecare iterație subgrafului B-stabil vizat i se atribuie o orientare tranzitivă, după algoritmii descriși în secțiunea 2.3. Algoritmul rulează până când se ajunge la graful $G$.

**Algoritmul de construire a unei orientări tranzitive**

**Date de intrare:** Șirul complet de grafuri factor:

$G^0 = G, G^1 = G^0/F_0, G^2 = G^1/F_1, \dots, G^p = G^{p-1}/F_{p-1}$.

**Date de ieșire:** O orientare tranzitivă $\vec{G}$ a grafului $G$.

Procedura $TRO(G^0, \dots, G^p)$

$i \coloneqq p$;

$While\ \boldsymbol{i} > \mathbf{0}\ do$:

Se obține orientarea tranzitivă a grafului factor $G^i/F_i$;

$i \coloneqq i - 1$;

Se construiește orientarea tranzitivă a subgrafului B-stabil $F_i$;

$EndWhile$;

Se returnează orientarea tranzitivă a grafului $G_0 = G$;

**Teorema 2.11.** *Construirea unei orientări tranzitive arbitrare a unui graf neorientat $G$ necesită timp $O(np\Delta)$, unde $\Delta$ este gradul maxim al grafului $G$, $p$ este lungimea șirului complet de grafuri factor al grafului $G$, iar $n$ – numărul de vârfuri al grafului $G$.*

**Demonstrație:** Deoarece algoritmul precedent oferă construirea subgrafurilor stabile, la fiecare iterație, rezultă că algoritmul curent presupune parcurgerea subgrafurilor stabile, de rând cu orientarea lor. În baza teoremelor 2.8 și 2.9 orientarea unui subgraf stabil se efectuează în timpul

$O(n\Delta)$. Parcurgerea șirului complet de grafuri factor se efectuează în timpul $O(p)$. Deci, timpul total de construire a unei orientări tranzitive arbitrare este de $O(np\Delta)$. Având estimările pentru $p \leq \frac{n}{2}$ și pentru $\Delta(G) \leq n-1$, obținem că timpul necesar pentru construirea unei orientări tranzitive este de $O(n^3)$.

În baza unui șir complet de grafului factor exemplificat în figura 2.9, vom ilustra algoritmul pentru construirea unei orientări tranzitive. După cum este descris în algoritm, vom parcurge șirul în ordine inversă până când vom obține orientarea tranzitivă a întregului graf $G$. Fie șirul complet de grafuri factor: $G^0 = G, G^1 = G^0/F_0, G^2 = G^1/F_1, G^3 = G^2/F_2, G^4 = G^3/F_3, G^5 = G^4/F_4$ determinat se subgrafurile B-stabile $G^1 = \{x_2, x_3\} - SBS$, $G^2 = \{x_5, x_6\} - SBS$, $G^3 = \{x_1, x_{23}\} - SBS$. Graful factor $G^4$ este un graf complet determinat de mulțimea de vârfuri $\{x_{123}, x_{456}\}$ asupra căruia nu mai poate fi aplicată operația de factorizare. Pornind de la acest șir de grafuri, putem construi o orientare tranzitivă. În graful factor $G^4/F_4$ vom orienta muchia $[x_{156}, x_{234}]$ de la vârful $x_{156}$ către vârful $x_{234}$, această orientare se alege în mod arbitrar. Urmărind ordinea inversă a șirului de grafuri factor, putem construi orientarea tranzitivă $\overrightarrow{G^3/F_3}$ având ca bază orientarea tranzitivă a grafului factor $G^4/F_4$. Continuând operația de construire a orientării tranzitive pentru fiecare graf factor și șirul complet de grafuri factor obținut mai sus, obținem la ultimul pas o orientare tranzitivă pentru graful neorientat $G$. Conform lemei 2.18, șirul complet de grafuri factor putea fi diferit. Totuși acest șir are aceeași lungime, mai mult, numărul de orientări tranzitive a grafului $G$ rămâne același. Atunci, $\tau(G) = 8$.

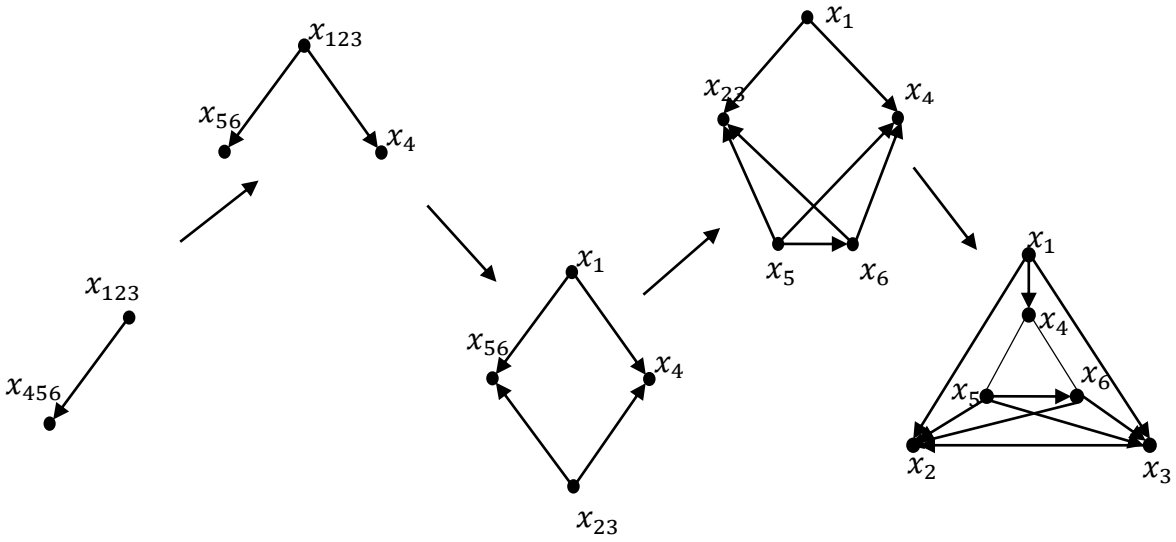



Fig. 2.10. *Procesul de construire a orientării tranzitive pentru graful* $G = (X; U)$.

## 2.5. Concluzii la capitolul 2

Capitolul doi conține un șir de rezultate importante ce țin de studierea proprietăților grafurilor tranzitiv orientabile. Cunoașterea acestor proprietăți au condus la extinderea rezultatelor obținute de către alți matematicieni în legătură cu studierea problemei construirii orientării tranzitive a unui graf neorientat. În particular, a fost obținută metoda prin care pot fi generate toate orientările tranzitive ale unui graf neorientat, datorită căreia devine clară procedura de trecere de la o orientare la alta, chestiune care lipsește în cercetările lui Golumbic M. C, care în fond examinează construirea doar a unei orientări cu condiții predeterminate. A fost obținută o formulă recurentă, pentru calcularea numărului orientărilor tranzitive. Totodată, menționăm că rezultatele obținute de Shevrin L.N. și Filipov N.D, orientate spre descrierea mulțimilor parțial ordonate prin intermediul grafurilor tranzitive au fost generalizate.

În baza celor descrise în capitolul 2, pot fi menționate principalele rezultate ce țin de problema studierii orientărilor tranzitive ale garfurilor neorientate:

- au fost studiate proprietăți noi importante ale lanțurilor netriangulate care, la rândul său, au permis extinderea rezultatelor ce țin de studierea subgrafurilor stabile;
- a fost studiată o clasă specială de subgrafuri stabile, subgrafurile B-stabile, folosite la construirea unui șir de grafuri factor, necesar soluționării problemei orientării tranzitive;
- a fost elaborată metoda de construire a șirului complet de grafuri factor cu studierea preventivă a proprietăților grafurilor tranzitiv orientabile;
- a fost obținută formula recurentă de calcul a numărului de orientări tranzitive a grafului neorientat.

***Ținând cont de rezultatele obținute în capitolul doi deducem următoarele concluzii:***

1. prin extinderea proprietăților lanțurilor netriangulate și a subgrafurilor stabile a devenit posibilă caracterizarea orientărilor tranzitive ale grafului neorientat și elaborarea metodei respective de trecere de la o orientare tranzitivă la altă orientare tranzitivă a grafului;
2. soluționarea definitivă a problemei orientării tranzitive a grafului neorientat a fost obținută prin examinarea unor probleme adiacente bazate pe cunoașterea proprietăților subgrafurilor B-stabile, a celor stabile complete maximale, vide maximale și minimale;
3. obținerea tuturor orientărilor tranzitive ale unui graf neorientat a devenit posibilă în baza generării șirului complet de grafuri factor;

4. studierea proprietăților șirului complet de grafuri factor a condus la deducerea formulei recurente pentru determinarea numărului de orientări tranzitive ale grafului;
5. cunoașterea rezultatelor teoretice ce țin de studierea proprietăților lanțurilor netriangulate, a subgrafurilor stabile, precum și a unei structuri speciale, numite graf factor, a permis elaborarea unor algoritmi ce vizează problema orientării tranzitive a grafurilor neorientate cu examinarea complexității acestora: algoritmul de orientare a subgrafurilor stabile minimale și complete maximale, algoritmul de determinare a subgrafurilor B-stabile, algoritmul de construire a șirului complet de grafuri factor și algoritmul de construire a orientării tranzitive a grafului neorientat.

# 3. GENERALIZĂRI ALE PROBLEMEI ORIENTĂRII TRANZITIVE A GRAFULUI

În acest capitol se examinează câteva probleme ce țin de orientarea tranzitivă a grafurilor, care pot fi considerate drept generalizări ale problemei examinate în capitolul precedent. Cunoaștem că deseori problemele practice se modelează cu ajutorul unor grafuri neorientate și, precum a fost menționat în capitolul întâi, soluționarea propriu zisă a problemei poate fi redusă la problema construirii orientării tranzitive a modelului respectiv. În această situație poate fi menționat că în unele cazuri modelul matematic care descrie un anumit proces se reprezintă printr-un graf mixt, adică un graf în care unele muchii sunt deja orientate (sunt arce). Desigur, prezintă interes soluționarea problemei orientării tranzitive a unor astfel de grafuri.

O altă problemă din cele expuse în capitolul precedent ar fi problema reorientării arcelor unui graf tranzitiv orientat. Cu alte cuvinte s-ar cere de examinat condițiile în care schimbarea orientărilor unor arce ale grafului tranzitiv orientat conduce la obținerea unui graf nou, care de asemenea este tranzitiv orientat. Această problemă se examinează în două aspecte:

a) determinarea submulțimii minimale de arce, reorientarea cărora păstrează proprietatea grafului de a fi tranzitiv orientat;
b) determinarea condițiilor în care reorientarea arcelor unei submulțimi oarecare de arce ale grafului nu schimbă orientarea tranzitivă în ansamblu a grafului.

Pentru problemele enunțate se face studiul teoretic respectiv, în baza căruia se elaborează algoritmi de soluționare cu estimarea complexității acestora.

## 3.1. Reorientarea unei mulțimi minimale de arce într-o orientare tranzitivă a grafului

Vom începe cu examinarea problemei de reorientare a unei submulțimi de arce ale unui graf tranzitiv orientat și vom determina condițiile în care această reorientare conduce la obținerea unui graf nou care rămâne tranzitiv orientat. Conform lemei 2.16, orice graf tranzitiv orientabil conține cel puțin două orientări tranzitive opuse una alteia. Astfel, schimbarea direcției arcelor unui graf tranzitiv orientat va oferi o nouă orientare tranzitivă. Totuși, dacă graful $G$ are mai mult de două orientări tranzitive, o altă orientare poate fi obținută prin reorientarea parțială a unui număr mai mic de arce. Să determinăm condițiile în care în graful tranzitiv orientat $\vec{G}$ există o mulțime minimală de arce $\vec{E} \subset \vec{U}_G$, reorientarea cărora păstrează proprietatea grafului de a fi tranzitiv orientat.

Conform teoremei 2.7, orice subgraf B-stabil al unui graf neorientat este ITO-subgraf. De aici ușor deducem că dacă într-un graf tranzitiv orientat $\vec{G}$ reorientăm toate arcele unui subgraf B-

stabil, atunci în rezultat se obține din nou un graf tranzitiv orientat. De asemenea, conform teoremei 2.6, dacă $x$ este un vârf adiacent cu toate vârfurile unui subgraf B-stabil, atunci în orice orientare tranzitivă arcele sunt îndreptate sau spre $x$, sau de la $x$. Deci, alternarea direcției arcelor adiacente unui subgraf B-stabil va permite construirea unei orientări tranzitive noi.

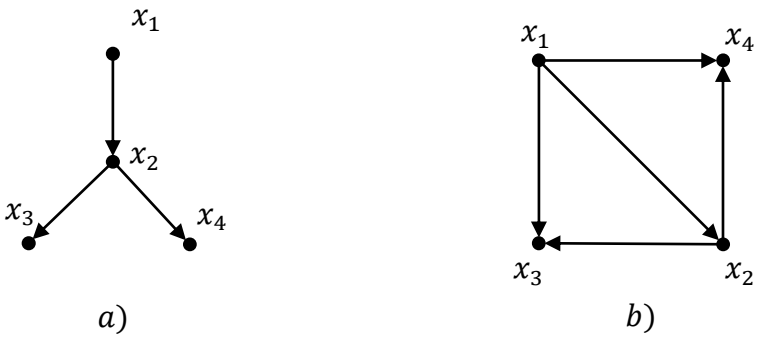



Fig. 3.1. *Diagrama Hasse și orientarea tranzitivă asociată mulțimii parțial ordonate.*

Fie $G = (X; U)$ un graf neorientat care este tranzitiv orientabil. Să admitem că pentru acest graf există mai mult decât două orientări tranzitive. Vom nota prin $\vec{G}$ una dintre orientările tranzitive ale grafului. În legătură cu problema examinată în acest paragraf este necesar mai întâi de a răspunde la întrebarea: Există oare în orientarea $\vec{G}$ o submulțime de arce $\vec{E} \neq \overrightarrow{U_G}$ la schimbarea orientării cărora obținem un graf nou care rămâne tranzitiv orientat? Răspuns la această întrebare ne dă următoarea

**Teorema 3.1.** *Dacă $\vec{G}$ este o orientare tranzitivă a unui graf neorientat $G = (X; U)$ ce admite mai mult de două orientări tranzitive atunci există o submulțime de arce $\vec{E} \neq \overrightarrow{U_G}$, $\vec{E} \neq \emptyset$ încât în rezultat schimbării orientărilor arcelor din $\vec{E}$ obținem o nouă orientare tranzitivă a grafului $G$.*

**Demonstrație:** Dacă $G$ admite mai mult de două orientări tranzitive, atunci conform lemei 2.8 rezultă că în graful $G$ există subgraf un B-stabil $F$. Conform teoremei 2.7 rezultă că orientarea tranzitivă a lui $F$ se face în mod independent de orientarea tranzitivă a întregului graf. Fie $\vec{F}$ o orientare tranzitivă astfel încât $\vec{F} \subset \vec{G}$, atunci inversarea arcelor din $\vec{F}$ generează o orientare nouă care este de asemenea tranzitivă.

Pe de altă parte dacă $F$ este subgraf B-stabil atunci conform teoremei 2.6 în orientarea tranzitivă $\vec{G}$ putem inversa direcția arcelor adiacente subgrafului $F$.

În baza alternativelor menționate rezultă demonstrația teoremei.

**Observația 3.1.** *Dacă graful neorientat $G = (X; U)$ are exact două orientări tranzitive atunci problema examinată prin teorema 3.1 nu are soluții.*

Luând în considerare teorema demonstrată vom examina doar grafurile neorientate ce conțin mai mult decât două orientări tranzitive. În acest context are sens determinarea submulțimii minime de arce la reorientarea cărora se păstrează orientarea tranzitivă a grafului.

Se cunoaște că problema reorientării tranzitive a unui graf tranzitiv orientat poate fi formulată în termenii mulțimilor parțial ordonate [93]. Ținând cont de faptul că mulțimea de vârfuri a unui graf tranzitiv orientat poate fi reprezentată ca o mulțime parțial ordonată, rezultă că soluționarea problemei formulate mai sus este legată de determinarea unei submulțimi de vârfuri ce poate fi reordonată astfel încât să obținem o nouă mulțime parțial ordonată a mulțimii de vârfuri ale grafului. Pentru a explica cele spuse vom examina următorul exemplu:

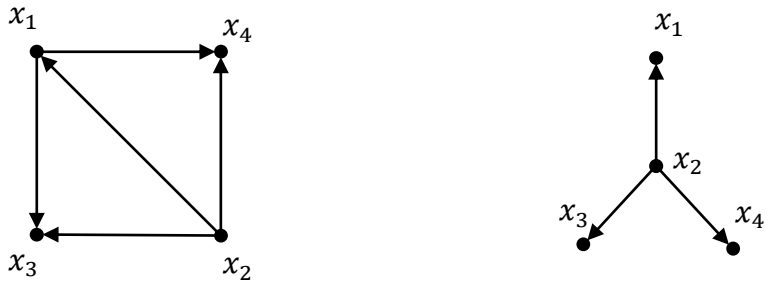



Fig. 3.2. *Orientarea tranzitivă nouă obținută prin inversarea arcului $[x_1, x_2]$ și diagrama Hasse asociată ei.*

Fie dată mulțimea parțial ordonată $\{x_1, x_2, x_3, x_4\}$ reprezentată prin diagrama Hasse în figura 3.1 a) și graful tranzitiv orientat din figura 3.1 b). În graful orientat reprezentat în figura 3.1 b) putem inversa arcul $[x_1, x_2]$, astfel încât să obținem o orientare tranzitivă nouă (reprezentată în figura 3.2 a). Pentru orientarea tranzitivă obținută se determină o mulțime parțial ordonată nouă, reprezentată în figura 3.2 b).

**Problema 3.1.** Fie $G = (X; U)$ un graf tranzitiv orientabil, iar $\vec{G}$ o orientare tranzitivă a sa. Să se determine mulțimea minimală de arce $\overrightarrow{E_G} \subset \overrightarrow{U_G}$ a căror direcție trebuie inversată, încât orientarea obținută $\overrightarrow{G'} = (X_G; \overrightarrow{U_{G'}})$ să fie la fel tranzitivă. $\overrightarrow{U_{G'}} = \left(\overrightarrow{U_G} \setminus \overrightarrow{E_G}\right) \cup \overleftarrow{E_G}$, unde $\overleftarrow{E_G}$ se obține din $\overrightarrow{E_G}$ prin inversarea direcției tuturor arcelor.

**Definiția 3.1.** *Dacă $F = (X_F; U_F)$ este un subgraf B-stabil al grafului tranzitiv orientabil $G$, atunci mulțimea de muchii $U_F$ se va numi factor intern determinat de subgraful $F$.*

Factorul intern determinat de subgraful B-stabil $F$ se va nota prin $I_F$.

**Observația 3.2.** *Dacă F este un subgraf stabil vid maximal, atunci factorul intern determinat de $F$ este o mulțime vidă.*

**Observația 3.3.** *Dacă graful tranzitiv orientabil $G = (X; U)$ nu conține subgraf B-stabil, atunci mulțimea de muchii $U_G$ determină factorul intern al grafului $G$.*

**Definiția 3.2.** *Dacă F este un subgraf B-stabil al grafului tranzitiv orientabil $G$, iar $x \in X_G \setminus X_F$ este un vârf adiacent mulțimii $X_F$, atunci, mulțimea de muchii $\{[x, y] | \forall y \in X_F\}$ se va numi factor extern determinat de subgraful $F$.*

Factorul extern determinat de subgraful B-stabil $F$ se va nota prin $E_F$.

De exemplu, în figura 3.3 muchia subgrafului B-stabil $F$ determinat de mulțimea de vârfuri $\{x_1, x_2\}$ formează un factor intern. Iar muchiile $\{[x_1, x_3], [x_2, x_3]\}$ determină un factor extern determinat de subgraful B-stabil $F$.

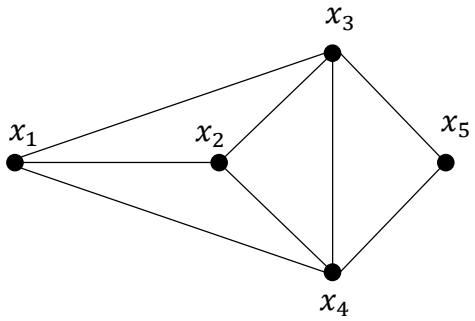



Fig. 3.3. *Factor intern $\{[x_1, x_2]\}$ determinat de subgraful B-stabil F și factorul extern $\{[x_1, x_3], [x_2, x_3]\}$.*

Factorul extern al unui subgraf B-stabil nu se determină în mod univoc. Pentru subgraful din figura 3.3 mulțimea de muchii $\{[x_1, x_4], [x_2, x_4]\}$ de asemenea formează un factor extern determinat de subgraful B-stabil $F$.

**Definiția 3.3.** *Mulțimea tuturor factorilor externi determinați de subgraful B-stabil $F$ al grafului tranzitiv orientabil $G$ se numește familie completă de factori externi corespunzători subgrafului B-stabil $F$.*

Familia completă de factori externi ce corespund subgrafului B-stabil $F$ se va nota prin $\mathcal{E}_F$. Astfel $\mathcal{E}_F = \{[x, y] \in U_G | x \in X_G \setminus X_F, y \in X_F\}$.

Dacă $M_F$ este un factor (intern sau extern) determinat de subgraful B-stabil $F$ al grafului tranzitiv orientabil $G = (X; U)$, atunci prin $|M_F|$ vom nota numărul de muchii al factorului $M_F$.

**Definiția 3.4.** *Factorul $M^*$ se va numi minim, dacă pentru orice alt factor $M$ din graful tranzitiv orientabil $G$ are loc relația* $0 < |M^*| \leq |M|$.

Din definițiile 3.1 și 3.2 problema reconstruirii orientării tranzitive poate fi formulată utilizând noțiunile de factor intern și factor extern în modul următor: *să se determine factorul minimal al grafului tranzitiv orientabil $G$, astfel încât toate arcele dintr-un subgraf tranzitiv orientat ce corespund factorului minimal pot fi inversate, astfel încât să fie păstrată proprietatea de tranzitivitate a orientării noi obținute.*

Fie $G^0 = G, G^1 = G^0/F_0, G^2 = G^1/F_1, \dots, G^k = G^{k-1}/F_{k-1}$ șirul complet de grafuri factor al grafului tranzitiv orientabil $G$, determinate de subgrafurile B-stabile $F_0, F_1, F_2, \dots, F_{k-1}$ respectiv. Este adevărată următoarea

**Teorema 3.2.** *Dacă $E_{F_i}$ este factor extern determinat de subgraful B-stabil $F_i$, atunci există factorul intern $I_{F_j}$, astfel încât $E_{F_i} \subseteq I_{F_j}$,* $0 \leq i \leq k-2$, $i+1 \leq j \leq k-1$.

**Demonstrație:** Fie $x_{F_i}$vârful obținut după operația de factorizare a grafului $G^i/F_i$, iar $[x_{F_i}, x_s]$ muchia obținută în rezultatul substituirii factorului extern $E_{F_i}$ cu vârful $x_{F_i}$. Dacă vârful $x_{F_i}$ nu aparține altui subgraf B-stabil, atunci acesta va fi parte a grafului factor obținut la ultimul pas al operației de factorizare $G^{k-1}/F_{k-1}$. Conform observației 3.2, muchia $[x_{F_i}, x_s]$ aparține mulțimi $U_{G/F_k}$. În caz contrar, muchia aparține unui alt subgraf B-stabil $F_j$, ceea ce înseamnă că ea este parte a unui factor intern $I_{F_j}$. Din cele menționate rezultă că afirmația lemei este adevărată.

Din proprietatea subgrafului B-stabil de a fi independent tranzitiv orientabil, rezultă că într-o orientarea tranzitivă toate arcele subgrafului care determină un factor intern pot fi reorientate, astfel încât orientarea nouă a întregului graf să fie la fel tranzitivă. Dacă în șirul complet de grafuri factor un vârf al subgrafului B-stabil reprezintă un alt subgraf B-stabil factorizat la un pas anterior, atunci schimbarea orientării tranzitive a lui la fel generează o orientare tranzitivă nouă. Totodată, poate fi observat, că orice muchie adiacentă unui astfel de vârf într-un graf factor precedent reprezintă un factor extern. Într-o orientare tranzitivă direcția unei astfel de muchii determină orientarea întregului subgraf B-stabil din care aceasta face parte. În asemenea caz vom spune că

muchia $[x_i, x_j]$ influențează orientarea tranzitivă a subgrafului $F$. Din cele menționate este adevărată următoarea

**Lema 3.1.** *Dacă $E_{F_i} \subset I_{F_j}$, atunci factorul extern $E_{F_i}$ influențează orientarea tranzitivă a factorului intern $I_{F_j}$.*

**Demonstrație:** Fie $x_{F_i}$vârful obținut după operația de factorizare a grafului $G^i/F_i$, iar $[x_{F_i}, x_s]$ muchia obținută în rezultatul comasării factorului extern $E_{F_i}$.

Deoarece mulțimea de vârfuri a factorului intern $I_{F_j}$ formează un subgraf B-stabil nevid, conform teoremei 2.7, orientarea $I_{F_j}$ se face în mod independent de orientarea întregului graf. În asemenea caz, orientarea fiecărei muchii din interiorul subgrafului $F_j$ influențează orientarea întregului subgraf, inclusiv și muchia $[x_{F_i}, x_s]$. Din cele menționate mai sus, pentru reconstruirea unei orientări tranzitive este suficient să inversăm arcele unui subgraf B-stabil din $\vec{G}$.

**Lema 3.2.** *Dacă subgraful B-stabil $F = (X; U_F)$ este un subgraf stabil complet maximal, atunci în orice orientare tranzitivă $\vec{G}$ poate fi ales exact un arc $[x_i, x_j] \in \vec{U}_F$ inversarea căruia generează o orientare tranzitivă nouă $\overrightarrow{G'}$.*

**Demonstrație:** Conform teoremei 2.7, orice subgraf B-stabil al unui graf neorientat este ITO-subgraf. Prin urmare, orice orientare tranzitivă a subgrafului $F$ generează o orientare tranzitivă a întregului graf. Pentru a obține o orientare tranzitivă nouă într-un graf complet, conform lemei 2.12, este necesar să găsim un arc, inversarea căruia permite ca în subgraful tranzitiv orientat $\vec{F}$ să fie satisfăcută relația 2.7. Pentru aceasta este suficient să alegem vârful $x_i \in X_F$, al cărui grad de ieșire este 0 și vârful $x_j \in X_F$, al cărui grad de ieșire este 1. Deoarece $F$ este subgraf complet, rezultă că $[x_i, x_j] \in \vec{U}_F$. Prin inversarea direcției arcului dat, relația 2.7 va fi adevărată, ceea ce garantează o orientare tranzitivă nouă a subgrafului B-stabil $F$. Prin urmare, aceasta garantează și orientarea tranzitivă nouă a întregului graf. Din cele demonstrate rezultă afirmația lemei.

**Observația 3.4.** *Dacă $F$ este subgraf stabil minimal, atunci pentru a obține o orientare tranzitivă nouă este necesar ca să fie inversate toate arcele din $\vec{U}_F$.*

Ușor poate fi observat că într-un graf există subgrafuri stabile minimale $F$ tranzitiv orientabile, astfel încât $|E_F| < |I_F|$ și invers $|I_F| < |E_F|$. Cu alte cuvinte, într-un graf pot fi alese subgrafuri stabile minimale astfel încât mulțimea de muchii adiacente subgrafului să fie mai mică

decât numărul de muchii din $F$. Totodată există subgrafuri stabile minimale pentru care numărul de muchii este mai mic decât numărul de muchii adiacente.

**Definiția 3.5.** *Dacă $F_i$ este un subgraf B-stabil iar $G/F_i$ graful factor determinat de $F_i$, atunci vârful $x_{F_i} \in X_{G/F_i}$ se numește vârf factor.*

**Lema 3.3.** *Dacă subgraful stabil $F$ conține vârf factor, atunci $F$ nu conține factor minim.*

**Demonstrație:** Fie $x_M$ un vârf factor al subgrafului $F$. Prin aplicarea procedurii inverse operației de factorizare obținem un subgraf stabil $M$, astfel încât $M \subset F$. Deci are loc relația $U_M \subset U_F$. În asemenea caz cardinalul factorului intern al subgrafului $F$ crește datorită apariției vârfurilor noi ale subgrafului $M$. Deoarece $F$ este subgraf stabil, rezultă că și cardinalul factorilor externi crește tot din aceleași considerente. Rezultă că în $F$ există un subgraf stabil cardinalul factorilor căruia este mai mic, deci $F$ nu conține factori minimi.

În baza rezultatelor obținute putem descrie un algoritm ce garantează reorientarea minimală a orientării tranzitive a unui graf. Mai întâi se parcurge șirul de grafuri factor până când este depistat un factor minimal. Dacă factorul depistat nu conține vârfuri factor, atunci se reorientează arcele factorului dat. După reorientarea factorului ales, se aplică procedura inversă operației de factorizare până când se ajunge la graful inițial cu o orientare tranzitivă nouă.

**Algoritmul de reorientare tranzitivă minimală a grafului**

**Date de intrare:** Șirul de grafuri factor $G/F_0, G/F_1, \dots, G/F_k$, orientarea tranzitivă $\vec{G}$.

**Date de ieșire:** Orientarea tranzitivă $\overrightarrow{G'} \neq \vec{G}.0$

Procedura $MinReverseTro(G^0, \dots, G^k)$

$i := 0$, $F_i = \emptyset$, $\overrightarrow{G'} = \emptyset$;

$While$ $\boldsymbol{F_i}$ $\boldsymbol{conține}$ $\boldsymbol{vârf\ factor}$ $||$ $\boldsymbol{U_{F_i}} = \emptyset$ $do$:

$i; = i + 1$;

Se obține subgraful B-stabil $F_i$;

$EndWhile$;

$While$ $\boldsymbol{i > 0}$ $do$:

$\overrightarrow{G'} := \overrightarrow{G/F_i}$

$If$ $\boldsymbol{F_i - graf\ complet}$ $do$:

Se inversează arcul $[x_i, x_j]$ unde $g^+(x_i) = 1$ și $g^+(x_j) = 0$;

$ElseIf$ $\boldsymbol{F_i - subgraf\ stabil\ minimal}$ $do$:

Se inversează arcele din $\overrightarrow{F_i}$;

$EndIf$;

$EndWhile$;

Se returnează $\overrightarrow{G'}$;

**Teorema 3.3.** *Timpul necesar pentru reconstruirea unei orientări tranzitive al unui graf tranzitiv orientabil $G$ este $O(p\Delta)$,* unde $p$ este lungimea șirului complet de grafuri factor, iar $\Delta$ este gradul maxim al grafului $G$.

**Demonstrație:** În cazul când la pasul $p$ se determină un factor intern obținem șirul complet de grafuri factor. În caz general, un factor intern poate fi depistat la pasul 1 sau 2. În asemenea caz se va lua în considerare doar timpul necesar pentru reorientarea arcelor factorului intern.

Conform observației 3.1 reconstruirea orientării tranzitive are loc doar dacă $F$ este subgraf stabil minimal sau subgraf stabil complet maximal. Deci, putem spune că reconstruirea orientării tranzitive a grafului $G$ se reduce la reconstruirea orientării tranzitive a subgrafului $F$. Dacă $F$ este subgraf stabil minimal, atunci este necesar de a inversa orientarea tuturor arcelor din $\overrightarrow{U_F}$, conform lemei 2.15. Dacă însă $F$ este subgraf stabil complet maximal, atunci schimbând ordinea numerotării oricăror două vârfuri din $X_F$, vom obține o orientare tranzitivă nouă, conform observației 2.8. Din cele menționate rezultă afirmația teoremei.

În baza algoritmului prezentat mai sus vom analiza un exemplu de reconstruire a unei orientări tranzitive.

Fie $G$ un graf tranzitiv orientabil, iar $\vec{G}$ o orientare tranzitivă a sa, prezentată în figura 3.2.

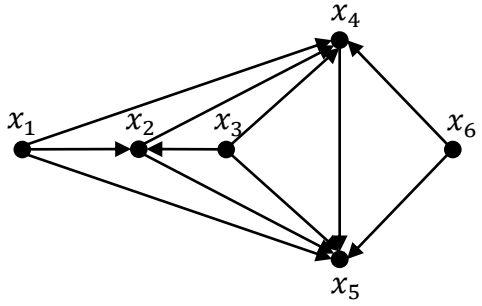



Fig. 3.4. *Orientarea tranzitivă* $\vec{G}$.

În graful de din figura 3.4, vârfurile $\{x_1, x_3\}$ generează un subgraf B-stabil. Deoarece $X_{F_1} = \{x_1, x_3\}$ generează un subgraf stabil vid maximal, rezultă că $I_{F_1} = \emptyset$. Deci, nu putem decât să aplicăm operația de factorizare. În figura 3.5 este prezentat graful factorul obținut.

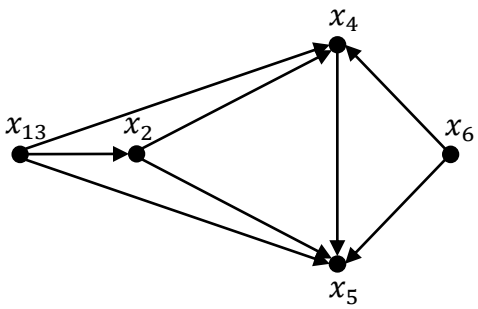



Fig. 3.5. *Orientarea tranzitivă a grafului factor* $G^2/F_1$.

Din graful $G^1/F_1$ mulțimea de vârfuri $X_{F_2} = \{x_{13}, x_2\}$ generează un alt subgraf B-stabil care este un subgraf stabil complet maximal. Deci, prin inversarea direcției unui singur arc (în cazul dat, este unicul) obținem o orientare nouă $\overrightarrow{G^1/F_1{}'}$ prezentată în figura 3.6.

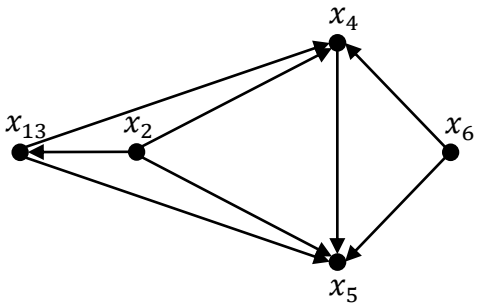



Fig. 3.6. *Orientarea tranzitivă* $\overrightarrow{G^1/F_1{}'}$.

Aplicând operația inversă operației de factorizare asupra grafului $G^1/F_1{}'$, obținem același graf $G$ cu o altă orientare tranzitivă $\overrightarrow{G'}$.

În paragraful dat am analizat metoda de reorientare a unei orientări tranzitive date. Orientarea tranzitivă obținută nu implică condiții specifice asupra sa. În continuare vom analiza cazurile când orientarea tranzitivă reconstruită trebuie să îndeplinească anumite condiții, fie asupra unui arc dar, sau a unei mulțimi de arce date în $\vec{G}$.

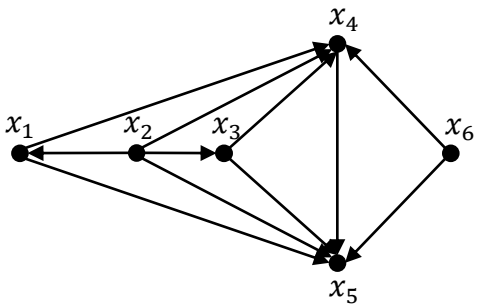



Fig. 3.7. *Orientarea tranzitivă reconstruită $\overrightarrow{G'}$ a grafului $G$.*

## 3.2. Reorientarea tranzitivă a grafului forțată de un arc apriori dat

În paragraful precedent a fost examinată problema determinării submulțimii minime de arce $\overrightarrow{M^*}$ a unui graf orientat tranzitiv $\vec{G}$ ce posedă proprietatea: schimbarea orientărilor tuturor arcelor din $\overrightarrow{M^*}$ conduce la obținerea unui graf nou care de asemenea este orientat tranzitiv. Din anumite considerente de ordin practic prezintă interes când $\left|\overrightarrow{M^*}\right| = 1$. Cu alte cuvinte ne-ar interesa situația când schimbarea orientării unui singur arc păstrează proprietatea orientării tranzitive a grafului.

Fie $[x_{i_1}, x_{i_2}, \dots, x_{i_p} = x_{i_1}]$ un ciclu oarecare al grafului neorientat $G = (X; U)$. Orice muchie $[x_{i_j}, x_{i_{j+2}}]$ unde $1 \leq j \leq p-2$, se numește **t-coardă** a ciclului respectiv.

**Teorema 3.4.** *Dacă $[x, y] \in \overrightarrow{U_G}$ este un arc al orientării tranzitive $\vec{G}$ ce aparține unui ciclu fără t-coarde atunci schimbarea orientării arcului $[x, y]$ conduce la obținerea unui graf care nu mai este orientat tranzitiv.*

**Demonstrație:** Fie $c = [x_{i_1}, x_{i_2}, \dots, x_{i_p} = x_{i_1}]$ un ciclu orientat tranzitiv. Dacă $c$ nu conține t-coarde atunci rezultă că subgraful determinat de vârfurile $x_{i_1}, x_{i_2}, \dots, x_{i_p} = x_{i_1}$ nu conține cicluri de lungimea 3. Prin urmare $c$ este un ciclu elementar, deci generează un subgraf bipartit. În baza demonstrației din lema 2.17 într-o orientare tranzitivă unui arc forțează direcția celorlalte arce. Prin urmare schimbarea direcției unui arc nu generează o orientare tranzitivă în graf.

Din cele menționate mai sus constatăm că într-o orientare tranzitivă a grafului există arce reorientarea cărora nu păstrează proprietatea de orientare tranzitivă. Totodată, ar putea exista o submulțime de arce $\overrightarrow{M^*}$ schimbarea orientării cărora împreună cu $[x, y]$ conduce la obținerea unei orientări tranzitive noi. În acest caz schimbarea orientării arcelor din $\overrightarrow{M^*}$ este impusă de arcul $[x, y]$. Vom nota această mulțime prin $\overrightarrow{M^*}([x, y])$.

**Observația 3.5.** *Problema propusă în acest paragraf nu ține de construirea unui algoritm optim de schimbare a orientării tranzitive, ci de reconstruirea unei orientări tranzitive ce îndeplinește anumite condiții impuse asupra arcelor unei orientări tranzitive existente, prin inversarea unui număr minim de arce.*

De exemplu, fie dată orientarea tranzitivă $\vec{G}$ a unui graf $G$, în figura 3.6.

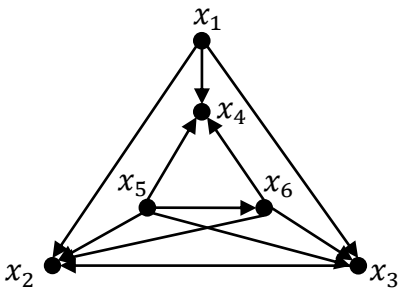



Fig. 3.8. *Orientarea tranzitivă* $\vec{G}$

Să se reconstruiască o orientare tranzitivă nouă $\overrightarrow{G'}$, în care arcul $[x_5.x_4] \in \overrightarrow{U_G}$ va fi inversat. Este evident că orientarea poate fi reconstruită prin redirecționarea tuturor arcelor din $\overrightarrow{U_G}$. Dar acest lucru nu în toate cazurile este eficient. Astfel, este necesar de a găsi acele arce care sunt dependente de orientarea arcului inițial dat. Cu alte cuvinte, trebuie să determinăm mulțimea de arce din $\overrightarrow{U_G}$ ce vor fi inversate în orientarea tranzitivă nouă $\overrightarrow{G'}$. Având exemplul de mai sus și condiția menționată, putem reconstrui orientarea tranzitivă necesară. Astfel, orientarea nou obținută este prezentată în figura 3.7.

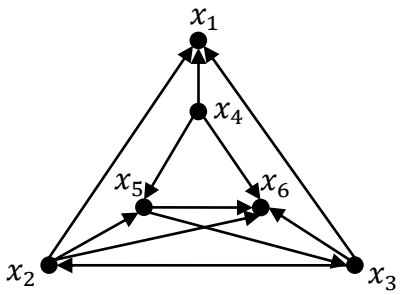



Fig. 3.9. *Orientarea tranzitivă* $\overrightarrow{G'}$

În continuare vom da o definiție strictă a problemei de reconstruire a unei orientări tranzitive ce impune anumite condiții asupra unui arc dat.

Fie $[x_0, y_0]$ un arc al unui graf tranzitiv orientat $\vec{G} = (X; \vec{U}_G)$. În caz general la schimbarea orientării acestui arc se obține un graf nou care ar putea și să nu fie tranzitiv orientat. Pentru ca

acesta să satisfacă proprietatea de tranzitivitate, este necesar de schimbat orientarea încă a unei anumite submulțimi de arce $\vec{E} \neq \vec{U}_G$.

În baza rezultatelor expuse în capitolul doi o astfel de mulțime totdeauna există. De exemplu, e suficient să reorientăm și toate celelalte arce rămase din graf. Vom numi mulțimea $\vec{E}$ mulțime satelit a arcului $[x_0, y_0]$.

**Problema 3.2.** Să se determine mulțimea satelit minimală a arcului $[x_0, y_0]$ într-un graf tranzitiv orientat $\vec{G} = (X; \vec{U}_G)$ la reorientarea arcelor căreia graful orientat obținut va păstra proprietatea de tranzitivitate.

Din cele expuse mai sus, rezultă că schimbarea orientărilor arcelor mulțimii satelit a unui arc fixat conduce la obținerea unei orientări noi a grafului, în care arcul dat inițial este de orientare inversă.

Ca și în paragraful precedent, vom folosi noțiunea de factor intern și, respectiv, factor extern determinat de un subgraf B-stabil $F$ din $G$.

Fie că $[x_0, y_0]$ este arcul inițial dat care forțează construirea unei orientări tranzitive noi, atunci sunt adevărate următoarele:

**Teorema 3.5.** *Dacă arcul inițial dat* $[x_0, y_0]$ *aparține unui factor intern* $I_F$ *determinat de un subgraf B-stabil* $F$*, atunci pentru construirea unei orientări tranzitive noi în șirul complet de grafuri factor se orientează toate arcele din graful care nu conține vârfuri factor.*

**Demonstrație:** Conform teoremei 2.7 orientarea subgrafului B-stabil $F$ se face în mod independent de orientarea întregului graf $G$. Astfel, obținem că orientarea arcului $[x_0, y_0]$ influențează doar orientarea subgrafului $F$. Dacă $U_F \neq \emptyset$ rezultă că $F$ poate fi sau subgraf stabil minimal, sau subgraf stabil complet maximal.

Dacă $F$ este subgraf stabil minimal, atunci conform lemei 2.18 reorientarea arcului $[x_0, y_0]$ din $\overrightarrow{U_G}$ va duce la inversarea direcției tuturor arcelor din $\overrightarrow{U_F}$.

Dacă însă $F$ este subgraf stabil complet maximal, atunci vom aplica aceleași procedură ca și cea descrisă în paragraful precedent. Vom reordona mulțimea de vârfuri din $X_F$ prin schimbarea cu locurile a vârfurilor $x_0$ cu $y_0$.

**Teorema 3.6.** *Dacă arcul inițial dat* $[x_0, y_0]$ *aparține unui factor extern* $E_{F_i}$ *determinat de subgraful B-stabil* $F_i$*, atunci inversarea arcelor din factorul extern* $E_{F_j}$*, unde* $j < i$ *determinat de*

*subgraful B-stabil $F_j$ adiacente subgraful B-stabil $F_j$ care nu conține vârfuri factor generează o orientarea tranzitivă.*

**Demonstrație:** După cum a fost menționat în teorema 3.2, dacă $[x_0, y_0]$ aparține unui factor extern $E_F$, atunci în șirul complet de grafuri factor $G^0 = G, G^1 = G^0/F_0, G^2 = G^1/F_1, \dots, G^k = G^{k-1}/F_{k-1}$, arcul $[x_0, y_0]$ aparține unui factor intern $I_{F'}$. Conform teoremei 2.6 în orice orientare tranzitivă orientarea tuturor arcelor dintr-un factor extern coincide. Deci reorientarea arcului $[x_0, y_0]$ implică reorientarea tuturor arcelor din $E_F$. Din cele menționate mai sus, dacă $F$ este un subgraf B-stabil, iar $x_F$ vârful asociat lui în graful factor obținut în urma operației de factorizare, atunci toate arcele de forma $[x, y]$, unde $x \in X_{F_i}$, iar $y \in X_G \setminus X_{F_i}$ vor avea aceeași direcție. Deci, arcul $[x_{F_i}, y]$ va avea orientarea menționată anterior. Dacă în graful factor obținut $G^i/F_i$ arcul $[x_{F_i}, y]$ aparține tot unui factor extern, atunci în operația de factorizare atribuim aceeași direcție ca și arcului corespunzător. Dacă însă $[x_{F_i}, y]$ aparține unui factor intern, atunci aplicăm procedura descrisă în cazul 1 descris mai sus.

**Observația 3.6.** *Dacă arcul inițial dat $[x_0, y_0]$ nu aparține nici unui factor intern și nici unui factor extern atunci orientarea tranzitivă obținută prin inversarea acestuia poate fi obținută dacă considerăm întreg graful ca și factor intern și aplicăm metoda descrisă în teorema 3.5.*

Conform teoremei 3.2 un factor extern determinat de un subgraf B-stabil $F$ în șirul de grafuri factor va fi neapărat parte a unui factor intern. Deci, dacă arcul $[x_0, y_0]$ aparține unui factor extern în unul din grafurile factor, atunci poate fi aplicată procedura din cazul 2. Dacă însă arcul $[x_0, y_0]$ nu aparține nici unui factor extern, atunci conform observației 3.2, $[x_0, y_0]$ aparține factorului intern $G/F_k$. Astfel poate fi aplicată procedura descrisă în cazul 1.

În baza rezultatelor expuse mai sus putem elabora un algoritm de reconstruire a orientării tranzitive forțată de un arc dat.

**Date de intrare:** Graful $G$, orientarea tranzitivă $\vec{G}$, arcul $[x_0, y_0]$.

**Date de ieșire:** Orientarea tranzitivă $\overrightarrow{G'}$.

Pas 1. $i := 1$.

Pas 2. Se determină orientarea tranzitivă $\overrightarrow{F_i}$ a subgrafului B-stabil $F_i$

Pas 3. Dacă $[x_0, y_0] \in \overrightarrow{U_{F_i}}$: Se construiește $\overrightarrow{F_i'}$, Se trece la Pas 6.

Pas 4. $i := i + 1$.

Pas 5.Se determină $\overrightarrow{G^{\iota}/F_{\iota}}$, se trece la Pas 2.

Pas 6. Dacă $i <> 0$: Se determină $\overrightarrow{G^{\iota}/F_{\iota}'}$.

Pas 7. Dacă $i = 0$: STOP, se returnează $\overrightarrow{G'}$.

Pas 8. $i := i - 1$: Se trece la Pas 6.

**Teorema 3.7.** *Timpul necesar pentru reorientarea tranzitivă a unui graf $\vec{G}$ forțată de un arc apriori dat este de $O(k\Delta)$, unde $k$ este numărul de grafuri factor, iar $\Delta$ este gradul maximal al grafului $G$.*

**Demonstrație:** În caz general procedura descrisă în algoritmul de mai sus parcurge șirul complet de grafuri factor care are lungimea $k$. Pentru reinversarea arcelor subgrafului B-stabil care nu conține vârfuri factor în caz general este necesar timpul $O(\Delta)$. Pornind de la graful factor arcurile căruia au fost inversate se parcurge șirul până la graful inițial, această procedură necesită timpul $O(k)$. Prin urmare timpul total este de $O(k\Delta)$.

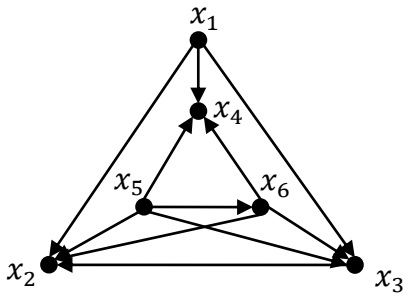



Fig. 3.10. *Orientarea tranzitivă $\vec{G}$*

În figura 3.8 este prezentată orientarea tranzitivă dată de arcul $[x_5, x_4]$. Acest arc fiind inversat în orientarea din figura 3.6.

### 3.3. Reorientarea tranzitivă a grafului forțată de o mulțime de arce apriori dată

Vom generaliza problema reorientării tranzitive a unui graf forțată de un arc apriori dat (a se vedea problema 3.2). Și anume, vom construi reorientarea tranzitivă forțată de o mulțime de arce $\overrightarrow{E_G} \subset \overrightarrow{U_G}$, $1 < |\overrightarrow{E_G}| < |\overrightarrow{U_G}|$.

Să examinăm pentru început un graf tranzitiv orientabil $G$ (a se vedea figura 3.11.a) și o orientare tranzitivă a acesteia $\vec{G}$ (a se vedea figura 3.11.b). Fixăm mulțimea $\overrightarrow{E_G} = \{[x_3, x_2], [x_1, x_4]\}$. Ușor observăm că reorientarea doar a arcelor mulțimii $\overrightarrow{E_G}$ nu ne asigură obținerea unei orientări tranzitive a grafului. Pentru aceasta, pe lângă arcele mulțimii $\overrightarrow{E_G}$ este

necesar de reorientat, probabil, și alte arce din graf. În cazul nostru, suplimentar se vor orienta arcele $\{[x_1, x_2], [x_5, x_2], [x_6, x_2], [x_5, x_4], [x_6, x_4]\}$. În rezultat se obține graful din figura 3.11.c. Vom spune că orientarea tranzitivă din figura 3.11.c reprezintă reorientarea arcelor grafului din figura 3.11.b forțată de mulțimea $\overrightarrow{E_G}$.

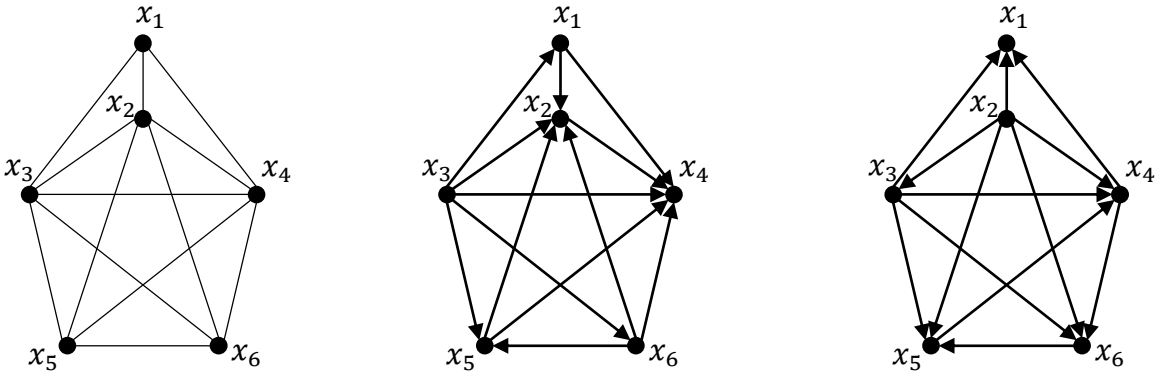



Fig. 3.11. *Graful tranzitiv orientabil(cazul a)) cu orientarea tranzitivă $\vec{G}$ a acestuia(cazul b)) și reorientarea forțată(cazul c)).*

Vom nota prin $\overrightarrow{E_0} \subset \overrightarrow{U_G}$, $\overrightarrow{E_0} \cap \vec{E} = \emptyset$ mulțimea de arce ce trebuie reorientate concomitent cu reorientarea arcelor din $\vec{E}$ pentru a obține din nou o orientare tranzitivă a grafului. Vom numi mulțimea de arce $\overrightarrow{E_0}$ **atașament** al mulțimii $\vec{E}$. Evident, în cazul $\overrightarrow{E_0} = \emptyset$ obținem problema examinată în paragraful 3.1.

**Lema 3.4.** *Pentru orice mulțime de arce $\vec{E} \subset \overrightarrow{U_G}$ a unei orientări tranzitive $\vec{G}$ astfel încât $|\vec{E}| > 1$ inversarea direcției arcelor mulțimii $\vec{E} \cup \overrightarrow{E_0}$ generează o nouă orientare tranzitivă.*

**Demonstrație:** Fie $[x, y]$ un arc arbitrar din $\vec{E}$. Aplicând algoritmul din paragraful 3.2. obținem o orientare tranzitivă. Arcele care au fost forțate pentru a obține noua orientare se adaugă la una din cele două mulțimi, fie $\vec{E}$, fie $\overrightarrow{E_0}$. Aplicând metoda descrisă în algoritmul din paragraful 3.2. pentru toate arcele din $\vec{E}$ până când va fi epuizată mulțimea $\vec{E}$ obținem orientarea tranzitivă care este diferită de $\vec{G}$.

**Teorema 3.8.** *Atașamentul mulțimii $\vec{E}$ se determină în mod univoc.*

**Demonstrație:** Fie $\vec{G}$ o orientare tranzitivă iar $\overrightarrow{E_0}$ atașamentul mulțimii $\vec{E}$. Vom nota prin $\overrightarrow{E_{I_F}}$ mulțimea de arce care aparține unui factor intern. Dacă $F$ este subgraf stabil minimal este necesar să inversăm toate arcele din $I_F$, ceea ce presupune inversarea tuturor arcelor din $\overrightarrow{E_{I_F}}$. Dacă

$F$ este subgraf stabil complet atunci inversarea arcelor din $E_{I_F}$ necesită inversarea arcelor mulțimii $\overrightarrow{E_{0I_F}}$ pentru ca să fie obținută o orientare tranzitivă, unde $\overrightarrow{E_{0I_p}}$. Astfel $\overrightarrow{E_{0I_F}}$ se obține în mod univoc conform lemei 2.12.

Fie $\overrightarrow{E_{E_F}}$ este mulțimea de arce care aparține unui factor extern. Aplicând operația de factorizare asupra $\vec{G}$ și asupra $\overrightarrow{E_{E_F}}$ obținem un arc astfel inversarea arcului obținut în graful factor duce la inversarea tuturor arcelor din $E_F$. Ușor poate fi verificat că această operație se face în mod univoc. Dacă însă arcul $[x, y]$ nu aparține nici unui factor extern și nici unui factor intern aplicând operația de factorizare asupra $\vec{G}$ și asupra $\vec{E}$ atunci în baza observației 3.6 la un pas finit se va stabili apartenența acesteia la unui din factorii externi sau interni. În baza celor menționate teorema este demonstrată.

În cele ce urmează vom da o definire strictă a problemei de mai sus.

**Problema 3.3.** Fie $G = (X; U)$ un graf tranzitiv orientabil, iar $\vec{G} = (X_G; \overrightarrow{U_G})$ o orientare tranzitivă a sa, și $\overrightarrow{E_G} \subset \overrightarrow{U_G}$, unde $2 < |\overrightarrow{E_G}| < |\overrightarrow{U_G}|$. Să se determine o orientare tranzitivă nouă $\overrightarrow{G'} = (X_G; \overrightarrow{U_G'})$, astfel încât $\overleftarrow{E_G} \subset \overrightarrow{U_G'}$, unde $\overleftarrow{E_G} = \{[x, y] | [y, x] \in \overrightarrow{E_G}\}$.

Deoarece mulțimea $\overleftarrow{E_G}$ se obține prin inversarea arcelor din $\overrightarrow{E_G}$, putem ușor observa că orientarea nou obținută va fi la fel tranzitivă. Deci, în continuare vom prezenta pașii necesari pentru a obține noua orientare tranzitivă ce conține arcele din $\overleftarrow{E_G}$.

În aceste condiții putem aplica algoritmul de reconstruire a orientării tranzitive cu unele modificări ce țin de mulțimea de arce $\overrightarrow{E_G}$. Deoarece pentru reconstruirea orientării vom folosi mulțimea $\overrightarrow{E_G}$, este necesar să aplicăm și operația de factorizare și asupra mulțimii date. Astfel, dacă un arc din mulțimea $\overrightarrow{E_G}$ este determinat de un vârf ce aparține unui subgraf B-stabil asupra căruia a fost aplicată operația de factorizare, atunci acest vârf al arcului se substituie cu vârful obținut la factorizare. Dacă întreg arcul aparține unui subgraf ce este factorizat, atunci acest arc se substituie cu vârful obținut la factorizare.

De exemplu, în figura 3.12 este prezentat șirul complet de grafuri factor al grafului $G$.

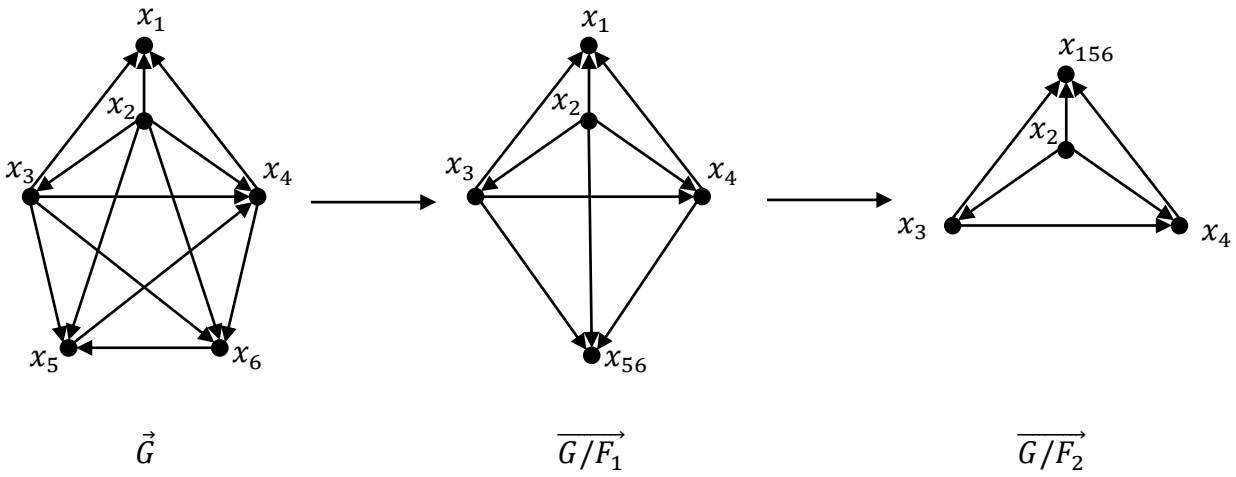



Fig. 3.12. *Șirul complet de grafuri factor al orientării* $\vec{G}$.

Fie mulțimea $\overrightarrow{E_G} = \{[x_3, x_1], [x_3, x_4], [x_6, x_5], [x_4, x_6]\}$, atunci după aplicarea operației de factorizare asupra orientării $\vec{G}$ obținem mulțimea $\overrightarrow{E_{G/F_1}} = \{[x_3, x_1], [x_3, x_4], x_{56}, [x_4, x_{56}]\}$. Repetând aceeași operație asupra orientării $\overrightarrow{G^1/F_1}$, obținem mulțimea $\overrightarrow{E_{G^2/F_2}} = \{[x_3, x_{156}], [x_3, x_4], x_{156}, [x_4, x_{156}]\}$. Dacă asupra orientări $\overrightarrow{G^2/F_2}$ nu mai poate fi aplicată operația de factorizare, atunci mulțimea $\overrightarrow{E_{G^2/F_2}}$ poate fi folosită pentru reconstruirea noii orientări tranzitive.

În urma celor menționate putem elabora un algoritm de reconstruire a orientării tranzitive, având o mulțime de arce date $\overrightarrow{E_G}$. Acest algoritm va fi divizat în două etape. La prima etapă va fi obținută orientarea asupra căreia nu mai poate fi aplicată operația de factorizare, dar și mulțimea de arce corespunzătoare ei. La etapa a doua va fi obținută orientarea tranzitivă reconstruită.

**Reorientarea tranzitivă a grafului forțată de o mulțime de arce apriori dată**

**Date de intrare:** Orientarea tranzitivă $\vec{G}$, mulțimea de arce $\overrightarrow{E_G}$.

**Date de ieșire:** Orientarea tranzitivă $\overrightarrow{G'}$.

**Etapa I**

Pas 1. $i := 1$, $\overrightarrow{G^0/F_0} := \vec{G}$, $\overrightarrow{E_{G^k/F_k}} := \overrightarrow{E_G}$.

Pas 2. Se determină subgraful B-stabil $F_i$.

Pas 3. Se construiește $\overrightarrow{G^i/F_i}$.

Pas 4. Se construiește $\overrightarrow{E_{G^i/F_i}}$.

Pas 5. Dacă $\overrightarrow{G^{\iota}/F_{\iota}} = \overrightarrow{G^{\iota-1}/F_{\iota-1}}$: **STOP**

Pas 6. $i \coloneqq i + 1$.

Pas 7. Se trece la Pas 2.

**Etapa a II-a**

Pas 1. $i \coloneqq k$.

Pas 2. Se construiește $\overleftarrow{E_{G^{\iota}/F_{\iota}}}$.

Pas 3. Se construiește $\overleftarrow{G^{\iota}/F_{\iota}}$.

Pas 4. Dacă $i \leq 0$: **STOP**

Pas 5. $i \coloneqq i - 1$.

Pas 6. Se trece la Pas 2.

**Teorema 3.9.** *Timpul necesar pentru reorientarea tranzitivă a grafului forțată de o mulțime de arce apriori dată este $O(kp\Delta)$, unde $k$ este numărul de grafuri factor, $p$ – numărul de muchii în mulțimea $\vec{E}$ iar $\Delta$ este gradul maximal al grafului.*

**Demonstrație:** În caz general procedura descrisă în algoritmul de mai sus parcurge șirul complet de grafuri factor care are lungimea $k$. Pentru reinversarea arcelor subgrafului B-stabil care nu conține vârfuri factor în caz general este necesar timpul $O(\Delta)$. Pentru determinarea mulțimii $\overrightarrow{E_0}$ se aplică operația de factorizare asupra mulțimii $\vec{E}$. Această procedură necesită timpul $O(p)$. Pornind de la graful factor arcurile căruia au fost inversate se parcurge șirul până la graful inițial, această procedură necesită timpul $O(k)$. Prin urmare timpul total este de $O(kp\Delta)$.

În cele ce urmează vom analiza lucrul algoritmului în baza exemplului din figura 3.9. Deoarece în figura 3.11 este reprodus șirul de grafuri factor. Acest șir este rezultatul lucrului din etapa I a algoritmului. Vom reproduce doar etapa a II-a.

Pornind de la orientarea $\overrightarrow{G^2/F_2}$ obținem mulțimea de arce $\overrightarrow{E_{G^2/F_2}} = \{[x_3, x_{156}],\ [x_3, x_4],\ x_{156},\ [x_4, x_{156}]\}$. Inversând orientarea tuturor arcelor din $\overrightarrow{E_{G^2/F_2}}$, obținem mulțimea $\overleftarrow{E_{G^2/F_2}} = \{[x_{156}, x_3],\ [x_4, x_3],\ x_{156},\ [x_{156}, x_4]\}$. Din această mulțime putem reconstrui orientarea $\overrightarrow{G^2/F_2}'$ reprezentată în figura 3.13.

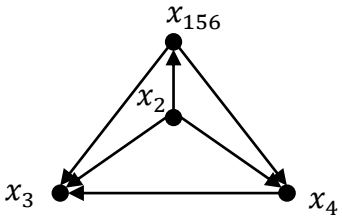



Fig. 3.13. *Orientarea tranzitivă* $\overrightarrow{G^2/F_2}'$

Având orientarea $\overrightarrow{G^2/F_2}'$, putem trece la următoarea iterație. Vom construi mulțimea de arce $\overleftarrow{E_{G^2/F_1}} = \{[x_1, x_3], [x_4, x_3], x_{56}, [x_{56}, x_4]\}$. Ușor poate fi observat că mulțimea de arce $[x_1, x_3]$ și $[x_{56}, 4]$ aparține familiei de factori externi $\mathcal{E}_{F_2}$. Acest lucru presupune reorientarea tuturor arcelor din $\mathcal{E}_{F_2}$. Astfel, obținem orientarea $\overrightarrow{G^1/F_1}'$ prezentată în figura 3.14.

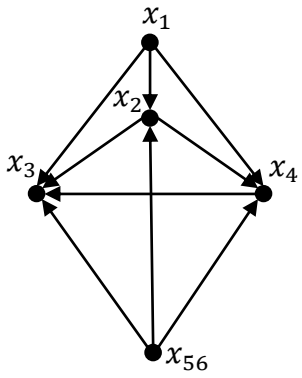



Fig. 3.14. *Orientarea tranzitivă* $\overrightarrow{G^1/F_1}'$.

Aplicând aceeași procedură descrisă mai sus, cu ușurință putem reproduce orientarea tranzitivă $\overrightarrow{G^0/F_0}' = \overrightarrow{G'}$ din figura 3.15.

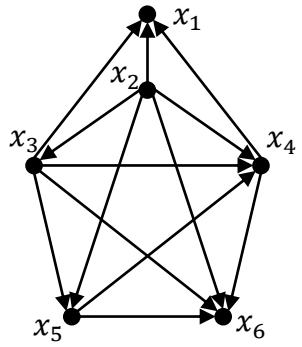



Fig. 3.15. *Orientarea tranzitivă* $\overrightarrow{G'}$.

Problema reconstruirii unei orientări tranzitive, utilizând clasa subgrafurilor stabile, poate fi soluționată în timp polinomial. Același rezultat poate fi obținut prin utilizarea claselor de implicații prin reorientarea doar a unui singur arc. La momentul de față nu sunt cunoscuți algoritmi în baza claselor de implicații. pentru reconstruirea orientării tranzitive forțată de o mulțime de arce.

### 3.4. Construirea unei orientări tranzitive având un arc dat

Deoarece orice graf tranzitiv orientabil poate avea cel puțin două orientări tranzitive, de multe ori apare necesitatea de a construi o orientare cu anumite condiții inițiale prestabilite. Prin condiții inițiale vom înțelege o mulțime de arce ce trebuie orientate într-un anumit sens.

De exemplu, în figura 3.16 este dat graful tranzitiv orientabil $G$. Se cere de a construi o orientare tranzitivă luând, în considerare faptul că muchia $[x_2, x_5]$ din $U_G$ trebuie să formeze arcul $[x_5, x_2]$ în orientarea tranzitivă $\vec{G}$.

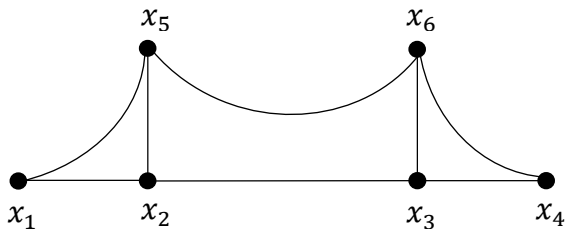



Fig. 3.16. *Graf tranzitiv orientabil*

Având exemplul din figura 3.16 și condiția menționată, putem construi orientarea tranzitivă reprezentată în figura 3.17 mai jos.

Ușor poate fi observat că graful din figura 3.14 posedă doar două orientări tranzitive. Astfel, este limpede că orientarea arcului $[x_5, x_2]$ determină orientarea întregului graf. Totuși, dacă un graf $G$ conține mai mult decât două orientări tranzitive rezultă că orientarea unor arce influențează doar subgraful B-stabil din care face parte.

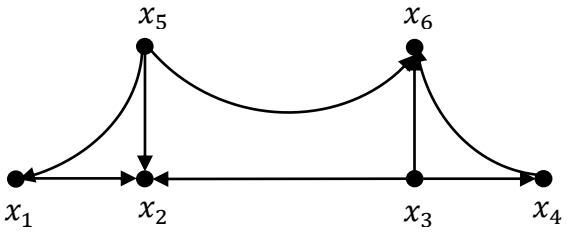



Fig. 3.17. *Orientarea tranzitivă a grafului* $G$.

În paragraful ce urmează, ne propunem să elaborăm un algoritm de construire a unei orientări tranzitive, atunci când în calitate de condiții inițiale avem doar un singur arc.

Din cele menționate mai sus, putem formula problema dată în modul următor:

Fie dat graful tranzitiv orientabil $G$, și muchia $[x, y] \in U_G$. Să se construiască o orientare tranzitivă $\vec{G} = (X_G; \overrightarrow{U_G})$ știind că muchia $[x, y]$ din $U_G$ trebuie să formeze arcul $[x, y]$ din $\overrightarrow{U_G}$.

Pentru soluționarea problemei menționate, vom folosi noțiunile de factor intern și factor extern din paragraful 3.1. Luând în considerare acest fapt, putem observa că sunt posibile trei cazuri.

1. Arcul inițial $[x, y]$ aparține unui factor intern;
2. Arcul inițial $[x, y]$ aparține unui factor extern;
3. Arcul inițial nu aparține nici factorului intern și nici factorului extern.

În cele ce urmează vom analiza fiecare caz în parte:

1. Fie că arcul inițial $[x, y]$ aparține unui factor intern $I_F$.

Conform teoremei 2.7, orientarea factorului intern nu influențează orientarea tranzitivă a întregului graf. Deci, problema construirii unei orientări tranzitive a grafului $G$ în cazul dat se reduce la construirea unei orientări tranzitive a factorului intern $I_F$. Conform observației 3.1, mulțimea de vârfuri a subgrafului B-stabil $F$, ce formează un factor intern, poate fi sau subgraf stabil minimal sau subgraf stabil complet maximal.

Dacă $F$ este un subgraf stabil minimal, atunci pentru construirea orientării tranzitive având arcul $[x, y]$, putem aplica procedura $OrientareSSMin$ din paragraful 2.3, dacă alegem în calitate de arc inițial $[x, y]$.

În cazul construirii unei orientări tranzitive pentru un subgraf stabil complet maximal este posibil ca direcția arcului $[x, y]$ să aparțină mai multor astfel de orientări. Deoarece nu sunt puse alte restricții asupra muchiilor din graful $G$ și respectiv din subgraful B-stabil $F$, vom modifica algoritmul de construire a unei orientări tranzitive a grafului complet, descris în paragraful 2.3 prin reordonarea vârfurilor în mulțimea $X_F = \{x_{i_1}, x_{i_2}, \dots, x_{i_p}\}$, astfel încât în șirul vârfurilor ordonate vârful $x$ să fie primul, iar $y$ al doilea.

Având metoda de orientare a factorului intern, orientarea întregului graf se face aplicând aceeași operație de factorizare ca și pentru construirea unei orientări tranzitive arbitrare.

2. Vom analiza cazul când arcul inițial $[x, y]$ aparține unui factor extern $E_F$.

Conform teoremei 3.1, este cunoscut faptul că factorul extern în șirul de grafuri factor este și factor intern pentru un graf factor. Totuși, la fiecare pas al operației de factorizare este necesar de luat în considerare faptul că muchiile factorilor externi se comasează în una singură. Astfel, orientarea arcului din factorul extern factorizat trebuie să corespundă orientării arcului în graful factor obținut.

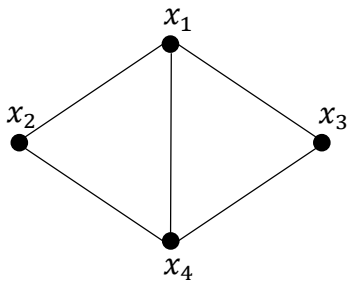



Fig. 3.18. *Factor extern al grafului G determinat de mulțimea de muchii* $\{[x_2, x_4], [x_3, x_4]\}$.

De exemplu, în figura 3.18 muchia $[x_2, x_4]$ aparține factorului extern determinat de mulțimea $E_F = \{[x_2, x_4], [x_3, x_4]\}$. Astfel, după aplicarea operației de factorizare, factorul extern $E_F$ va corespunde muchiei $[x_{23}, x_4]$ din graful factor din figura 3.19.

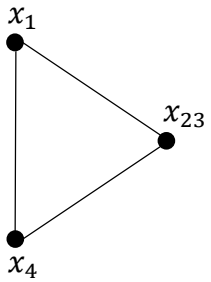



Fig. 3.19. *Graf factor ce conține un factor extern.*

Dacă muchia obținută în graful factor rezultant la fel aparține unui factor extern, atunci se aplică operația descrisă mai sus. Dacă însă muchia aparține unui factor intern atunci se aplică operația descrisă în punctul 1.

3. Arcul nu aparține nici factorului intern și nici factorului extern.

Este evident faptul că sunt posibile cazurile când arcul inițial dat nu aparține nici unui factor intern și nici extern. De exemplu, în graful din figura 3.18, arcul $[x_1, x_2]$ nu aparține factorului intern determinat de subgraful B-stabil $F$, unde $X_F = \{x_4, x_5\}$. Dar acest arc nu aparține nici factorilor externi determinați de mulțimea $E_{F_1} = \{[x_3, x_4], [x_3, x_5]\}$ și $E_{F_2} = \{[x_2, x_4], [x_2, x_5]\}$.

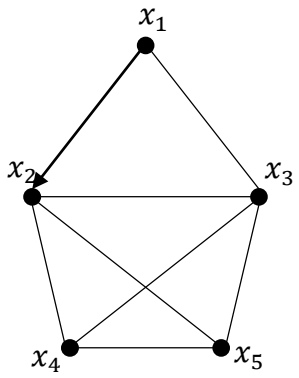



Fig. 3.20. *Graf tranzitiv orientabil.*

Din teorema 3.1 rezultă că situația 3 descrisă mai sus se reduce la una din situațiile 1 sau 2.

Din rezultatele descrise anterior putem elabora un algoritm cu ajutorul căruia vom construi o orientare tranzitivă având un arc inițial dat. Algoritmul dat se împarte în două etape:

I. Determinarea șirului de grafuri factor;

II. Construirea orientării tranzitive.

**Algoritm de construire a unei orientări tranzitive având un arc dat**

**Date de intrare:** Graful $G$, arcul $[x, y]$

**Date de ieșire:** Orientarea tranzitivă $\vec{G}$

**Etapa I**: Determinarea șirului de grafuri factor

Pas 1. $i := 1$

Pas 2. Se determină subgraful B-stabil $F_i$

Pas 3. Dacă muchia $[x, y] \in U_{F_i}$: Se construiește orientarea tranzitivă $\overrightarrow{U_{F_i}}$

Pas 4. Se construiește $G^i/F_i$

Pas 5. Se construiește $E_{F_i}$

Pas 6. Dacă muchia $[x, y] \in E_{F_i}$: Se construiește arcul corespunzător în $G^i/F_i$

Pas 7. Dacă $G^i/F_i = G^{i-1}/F_{i-1}$: **STOP**

Pas 8. $i := i + 1$. Se trece la Pas 2.

**Etapa a II-a:** Construirea orientării tranzitive

Pas 1. $i := k$

Pas 2. Dacă există arc în $U_{G/F_i}$: Se construiește orientarea corespunzătoare arcului $[x, y]$

Pas 3. Se construiește o orientare tranzitivă $\overrightarrow{G^i/F_i}$

Pas 4. $i := i - 1$

Pas 5. Dacă $i = 0$: **STOP**

Pas 6. Se trece la Pas 2.

**Teorema 3.10.** *Construirea unei orientări tranzitive a unui graf neorientat $G$ poate fi efectuată în timpul $O(n\Delta)$, unde $n$ este numărul de vârfuri al grafului $G$, iar $\Delta$ este gradul maximal al grafului.*

**Demonstrație:** La pasul 2 din etapa I se determină subgraful B-stabil $F_i$. Această operație poate fi executată utilizând procedura $SBS$ descrisă în secțiunea 2.2 și necesită timpul $O(\Delta)$. Construirea orientării tranzitive a subgrafului B-stabil $F_i$ necesită resursele $O(\Delta)$. Celelalte operații se realizează în timp constant, fapt ce nu influențează valoarea teoretică. Deoarece algoritmul rulează de $n$ ori rezultă că prima etapă necesită timpul de $O(n\Delta)$. A doua etapă reprezintă parcurgerea inversă a șirului grafului factor și orientarea consecutivă a fiecărui factor. Astfel, timpul de lucru este la fel de $O(n\Delta)$. Deci timpul de lucru total al algoritmului de construire a unei orientări tranzitive având un arc dat este de $O(n\Delta)$.■

În exemplul ce urmează vom reprezenta lucrul schematic al algoritmului descris mai sus. Fie graful $G = (X; U)$, unde mulțimea $X_G = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ este dată în figura 3.19 și arcul inițial este $[x_4, x_3]$.

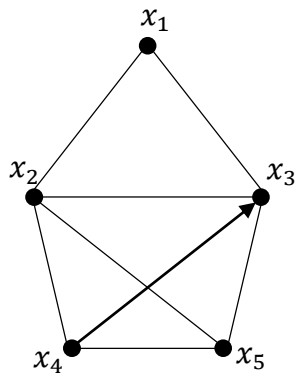



Fig. 3.21. *Graf tranzitiv orientabil împreună cu arcul dat.*

Observăm că arcul $[x_4, x_3]$ aparține factorului extern $E_F = \{[x_4, x_3], [x_5, x_3]\}$. În asemenea caz factorizăm graful $G$ și construim arcul corespunzător $[x_{45}, x_3]$ din $G^1/F_1$ dat în figura 3.18.

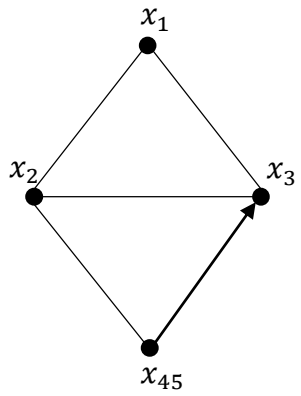



Fig. 3.22. *Graf factor* $G^1/F_1$.

În noul graf factor obținut, arcul $[x_{45}, x_3]$ la fel aparține unui factor extern $E_{F_2} = \{[x_1, x_3], [x_{45}, x_3]\}$. Vom aplica iarăși operația de factorizare și vom construi arcul $[x_{451}, x_3]$ în graful factor $G^2/F_2$ dat în figura 3.23.

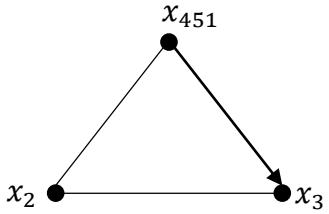



Fig. 3.23. *Graf factor* $G^2/F_2$

Observăm că graful din figura 3.21 este complet. Putem construi o orientare tranzitivă având arcul deja dat. În figura 3.23 estre construită o astfel de orientare.

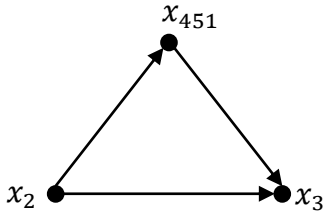



Fig.3.24. *Orientarea tranzitivă a grafului* $G^2/F_2$

Vom trece la etapa a II-a a algoritmului. Revenim la graful factor $G^2/F_1$ cu orientarea tranzitivă corespunzătoare.

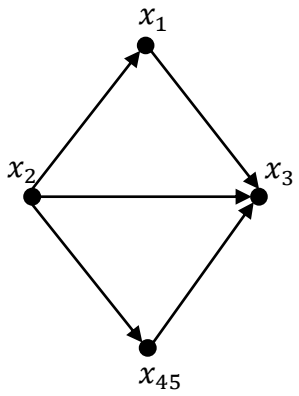



Fig. 3.25. *Orientarea tranzitivă a grafului* $G^1/F_1$.

În figura 2.26 este construită orientarea tranzitivă a grafului $G$ ce este determinată de orientarea arcului $[x_4, x_3]$.

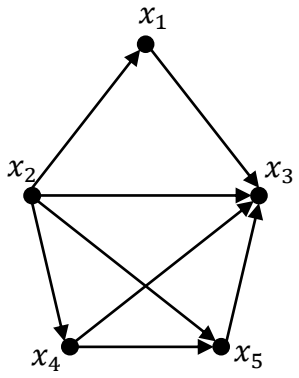



Fig. 3.26. *Orientarea tranzitivă a grafului* $G$.

Din cele expuse mai sus putem face concluzia că, elaborarea unei orientări tranzitive specifice unui arc dat necesită aceleași resurse ca și construirea unei orientări tranzitive arbitrare. Același rezultat poate fi obținut prin utilizarea algoritmului descris în [40] cu ajutorul claselor de implicații. Totuși algoritmul descris în acest paragraf oferă posibilitatea de a fi generalizat. Prin generalizare se are în vedere construirea unei orientări tranzitive dată de o mulțime de arce.

În paragraful următor vom analiza cazul când se cere construirea unei orientări tranzitive având o mulțime de arce.

## 3.5. Construirea unei orientări tranzitive având o mulțime de arce date

În paragraful precedent am analizat cazul când în calitate de condiții inițiale pentru construirea unei orientări tranzitive a fost dat un arc. În cele ce urmează vom analiza cazul când condițiile inițiale vor fi date de două sau mai multe arce. Vom nota prin $\overrightarrow{A_G}$ mulțimea de arce, care

induc o orientare tranzitivă. În mod formal mulțimea $\overrightarrow{A_G}$ poate fi reprezentată în felul următor $\overrightarrow{A_G} = \{[x,y] \in \overrightarrow{U_G} | \vec{G} = (X_G; \overrightarrow{U_G})\}$, iar $\vec{G}$ este o orientare tranzitivă. Ușor se observă că $\overrightarrow{A_G} \subseteq \overrightarrow{U_G}$.

Este evident că nu orice mulțime de arce $\overrightarrow{A_G}$ este submulțime a $\overrightarrow{U_G}$, unde $\vec{G} = (X_G; \overrightarrow{U_G})$ este o orientare tranzitivă. În continuare vom stabili unele condiții impuse mulțimii $\overrightarrow{A_G}$, astfel încât relația $\overrightarrow{A_G} \subseteq \overrightarrow{U_G}$ să fie adevărată.

**Lema 3.5.** *Dacă arcele* $\{[x,y],[y,z],[z,x]\} \subseteq \overrightarrow{A_G}$ *atunci această mulțime nu generează o orientare tranzitivă.*

Fie $F$ un subgraf B-stabil al grafului tranzitiv orientabil $G$, iar $I_F$ este un factor intern determinat de subgraful $F$, atunci este adevărată următoarea observație.

**Lema 3.6.** *Dacă* $[x,y]$ *și* $[s,t]$ *sunt două arce care aparțin factorului intern* $I_F$ *determinat de subgraful B-stabil* $F$*, atunci orientarea tranzitivă generată de arcul* $[x,y]$ *coincide cu orientarea tranzitivă generată de arcul* $[s,t]$.

Lema de mai sus poate fi ușor demonstrată aplicând teorema 2.5.

**Lema 3.7.** *Dacă* $F$ *este un subgraf B-stabil al grafului tranzitiv orientabil* $G$*, iar* $E_F$ *este un factor extern determinat de subgraful* $F$*, atunci toate arcele din* $E_F$ *au aceeași direcție.*

Deoarece conform observației 3.1 în calitate de factor intern pot fi doar subgrafuri stabile minimale și subgrafuri stabile complete maximale, rezultă că pentru a construi o orientare tranzitivă având mulțimea $\overrightarrow{A_G}$ vom verifica dacă nu sunt încălcate condițiile menționate în lemele 3.5 - 3.7.

Din cele menționate putem formaliza un algoritm de construire a unei orientări tranzitive, având o mulțime de arce date $\overrightarrow{A_G}$.

Ca și algoritmul din paragraful precedent, acest algoritm îl vom diviza în două etape. În prima etapă vom construi șirul de grafuri factor. Această operație va fi aplicată și asupra mulțimii $\overrightarrow{A_G}$, după cum este descris în secțiunea 3.3. În a doua etapă va fi construită orientarea tranzitivă prin parcurgerea în direcție inversă a șirului de grafuri factor.

**Algoritm de construire a orientării tranzitive având o mulțime de arce date**

**Date de intrare:** Graful tranzitiv orientabil $G$, mulțimea de arce $\overrightarrow{A_G}$

**Date de ieșire:** Orientarea tranzitivă $\vec{G}$

**Etapa I**

Pas 1. $i \coloneqq 1$

Pas 2. Dacă $\overrightarrow{A_{G^i}}$ nu respectă condiția de tranzitivitate: **STOP** – *Nu poate fi construită o orientare tranzitivă cu condițiile date*

Pas 3. $\overrightarrow{A_{G^0/F_0}} = \overrightarrow{A_G}$

Pas 4. Se determină subgraful B-stabil $F_i$

Pas 5. Se determină graful factor $G^i/F_i$

Pas 6. Se determină mulțimea $\overrightarrow{A_{G^i/F_i}}$

Pas 7. Dacă $G^i/F_i = G^{i-1}/F_{i-1}$: **STOP** – *Trec la Etapa a II-a*

Pas 8. $i \coloneqq i + 1$. Trec la Pas 2.

**Etapa a II-a**

Pas 1. $i \coloneqq k$

Pas 2. $p_i = |A_{G/F_i}|, j \coloneqq 1$

Pas 3. Dacă arcul $u_{i_j} \in \overrightarrow{A_{G^i/F_i}}$ generează o altă orientare tranzitivă: **STOP** – *Nu poate fi construită o orientare tranzitivă cu condițiile date*

Pas 4. Se construiește orientarea tranzitivă $\overrightarrow{G^i/F_i}$ indusă de arcul $u_{i_j}$

Pas 5. Dacă $j \geq p_i$: Trecem la Pas 7

Pas 6. $j \coloneqq j + 1$: Trecem la Pas 3

Pas 7. Se orientează muchiile care nu sunt dependente de $\overrightarrow{A_{G^i/F_i}}$

Pas 8. Dacă $i = 0$: **STOP** – Se r*eturnează* $\overrightarrow{G^0/F_0}$

Pas 9. $i \coloneqq i - 1$: Trec la Pas 2

Deoarece algoritmul curent este o modificare a algoritmului prezentat în paragraful 2.4, timpul necesar pentru construirea orientării tranzitive, având o mulțime de arce dată, este dependent doar de mărimea mulțimii asociate. Ușor poate fi observat că timpul de lucru al algoritmul este de $O(tn\Delta)$, unde t este numărul de arce din $\overrightarrow{A_G}$, $n$ este numărul de vârfuri al grafului $G$, iar $\Delta$ este gradul maxim al grafului $G$.

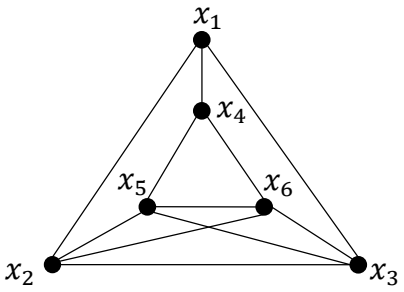



Fig. 3.27. *Graf tranzitiv orientabil G*

În continuare vom analiza un exemplu în baza algoritmului prezentat mai sus. Fie $G$ un graf tranzitiv orientabil prezentat în figura 3.27. Să se construiască o orientare tranzitivă $\vec{G}$, astfel încât mulțimea de arce $\overrightarrow{A_G} = \{[x_3, x_1], [x_2, x_5], [x_5, x_6]\}$ să respecte următoarea condiție: $\overrightarrow{A_G} \subseteq \overrightarrow{U_G}$.

După cum este menționat în etapa I a algoritmului se construiește șirul de grafuri factor, pornind cu graful $G$ până când poate fi aplicată operația de factorizare. De asemenea etapa dată presupune crearea șirului de mulțimi $\overrightarrow{A_{G^i/F_i}}$ care corespund fiecărui graf factor obținut. În continuare vom prezenta graful factor obținut la ultimul pas asupra căruia nu mai poate fi aplicată operația de factorizare prezentată în figura 3.28, precum și mulțimea $\overrightarrow{A_{G^3/F_3}} = \{[x_{234}, x_1], [x_{234}, x_{56}], x_{56}\}$.

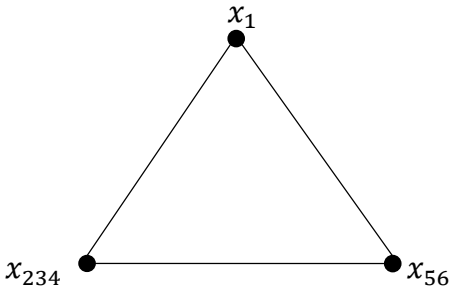



Fig. 3.28. *Graf factor* $G^3/F_3$.

Având graful $G^3/F_3$ și mulțimea $\overrightarrow{A_{G^3/F_3}}$ putem iniția etapa a II-a a algoritmului. Astfel, prima iterație oferă orientarea tranzitivă $\overrightarrow{G^3/F_3}$.

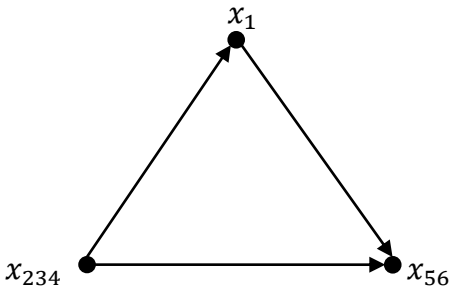



Fig. 3.29. *Orientarea tranzitivă* $\overrightarrow{G^3/F_3}$

Urmând operația inversă operației de factorizare și aplicând orientarea arcelor în conformitate cu mulțimea $\overrightarrow{A_G}$, obținem orientarea tranzitivă $\vec{G}$.

În acest paragraf am analizat procedura de construire a unei orientări tranzitive pornind de la unele restricții impuse arcelor care trebuie să facă parte din această orientare. În urma analizei eficacității algoritmului, am văzut că acesta reduce din timpul de prelucrare, însă în multe situații concrete este necesar de a construi o orientare tranzitivă anume având condiții stricte impuse arcelor din orientarea tranzitivă dorită.

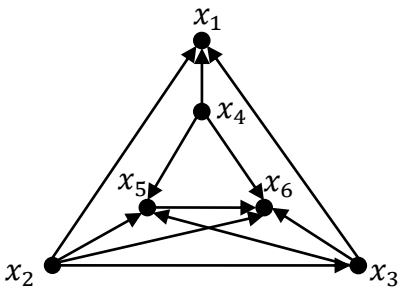



Fig. 3.30. *Orientarea tranzitivă* $\vec{G}$.

## 3.6. Concluzii la capitolul 3

În capitolul 3 au fost obținute rezultate ce țin de extinderea problemei examinate în capitolul anterior prin studierea grafurilor mixte cu construirea ulterioară a orientării tranzitive a acestora, precum și examinarea condițiilor de reorientare parțială a unui graf tranzitiv orientat, astfel încât graful nou obținut să posede aceeași proprietate. Rezultatele ce țin de studierea grafurilor mixte sunt importante prin faptul că anume astfel de structuri matematice se folosesc în calitate de model la soluționarea unor probleme practice. Rezultatele obținute se încadrează în schema generală a cercetărilor efectuate în capitolul doi și se finalizează cu descrierea unor metode eficiente pentru construirea orientărilor tranzitive în grafurile mixte.

În același timp, au fost obținute rezultate cu privire la problema reorientării arcelor unui graf tranzitiv orientat. Aceste rezultate pot conduce la elaborarea unei alte metode de generare a tuturor orientărilor tranzitive a unui graf neorientat, chestiune ce poate fi studiată separat în afara acestei lucrări. Și în acest caz rezultatele teoretice obținute au stat la baza elaborării unor metode de reorientare a arcelor grafului tranzitiv orientat. Printre rezultatele importante ale capitolului trei putem menționa:

- a fost descrisă dependența dintre factorul intern și factorul extern determinate de un subgraf B-stabil al grafului neorientat;
- a fost descrisă metoda de reorientare tranzitivă a grafului orientat;
- au fost determinate condițiile în care schimbarea orientării unui singur arc păstrează proprietatea grafului de a fi graf tranzitiv orientat;
- a fost elaborat algoritmul pentru determinarea mulțimii satelit a unui arc dat cu examinarea complexității acestuia;
- a fost elaborat algoritmul de construire a mulțimii satelit a unei mulțimi apriori date.

Ținând cont de rezultatele obținute în capitolul 3 deducem următoarele concluzii:

1. rezultatele teoretice obținute cu privire la proprietățile factorului intern și a factorului extern determinat de un subgraf B-stabil, au servit drept imbold pentru soluționarea problemei determinării mulțimii minimale de arce într-o orientare tranzitivă a grafului, schimbarea orientării cărora conduce la obținerea unui graf nou cu aceeași proprietate. Rezultatele teoretice obținute au permis elaborarea unui algoritm de o complexitate polinomială;
2. rezultatele cunoscute cu privire la orientarea tranzitivă a grafurilor au fost extinse asupra grafurilor mixte, chestiune importantă prin faptul că astfel de grafuri au multiple aplicații la rezolvarea problemelor practice, cum ar fi gestionarea fișierelor de înregistrare pentru procesoarele stream;
3. rezultatele teoretice cu privire la soluționarea problemei de construire a orientării tranzitive a grafului au permis elaborarea unor algoritmi eficienți pentru reorientarea arcelor grafului: algoritmul reorientării tranzitive a grafului impusă de schimbarea orientării unei submulțimi de arce, algoritmul de determinare a mulțimii minimale de arce reorientarea cărora păstrează proprietatea de a fi tranzitiv orientat.

## CONCLUZII GENERALE ȘI RECOMANDĂRI

Studiul realizat în prezenta lucrare conduce la următoarele concluzii și recomandări.

**Concluzii generale asupra rezultatelor obținute:** Problema examinată în teza de doctor „*Grafuri tranzitiv orientabile*” face parte de direcția de cercetare din matematică ce ține de elaborarea modelelor și metodelor corespunzătoare pentru soluționarea problemelor teoretico-aplicative din diverse domenii. Rezultatele teoretice obținute în legătură cu studierea orientărilor tranzitive ale grafurilor neorientate, precum și aplicațiile acestora la studierea unor probleme de ordin aplicativ, conduc la următoarele concluzii:

1. au fost formulate problemele de caracterizare structurală a grafurilor tranzitiv orientabile, ce au permis obținerea soluției de construire a tuturor orientărilor tranzitive într-un graf. Acest rezultat este important din punct de vedere teoretic prin faptul că permite o vizualizare integrală a orientărilor tranzitive a unui graf;
2. caracterizarea grafurilor tranzitiv orientabile a devenit posibilă datorită unor proprietăți noi demonstrate ale lanțurilor netriangulate și ale subgrafurilor stabile. În mod special, cunoașterea proprietăților subgrafurilor B-stabile și a șirului complet de grafuri factor a permis caracterizarea completă a tuturor orientărilor tranzitive, precum și obținerea unei formule recurente de calculare a numărului de orientări tranzitive;
3. elaborarea metodelor și algoritmilor eficienți ce vizează problema orientării tranzitive a grafurilor neorientate a devenit posibilă datorită rezultatelor teoretice ce țin de studierea proprietăților lanțurilor netriangulate și a subgrafurilor stabile;
4. cunoașterea mecanismului de obținere a tuturor orientărilor tranzitive ale unui graf neorientat a permis soluționarea unor cazuri speciale de orientare sau reorientare tranzitivă a grafului determinată de o mulțime de arce apriori dată;
5. rezultatele teoretice obținute și metodele generate de acestea cu privire la orientarea tranzitivă a grafurilor pot servi drept bază pentru soluționarea unor probleme importante cu caracter teoretico-aplicativ, cum ar fi decompoziția rețelelor Petri, analiza codului sursă a programelor;
6. rezultatele obținute pot servi drept punct de reper pentru studierea problemei orientării tranzitive mixte, ceea ce constituie extinderea cercetării în domeniul vizat în lucrare.

Teza conține și o componentă practică, algoritmii propuși au fost realizați sub formă de bibliotecă, scrisă în limbajul javascript care a fost implementată într-un modul, realizat în limbajul PHP al platformei Drupal.

Rezultatele prezentate în lucrare pot servi ca suport pentru continuarea cercetărilor din domeniul teoriei grafurilor, cum ar fi colorarea grafurilor, problema determinării mulțimii stabile interior, precum și soluționarea cazurilor din diverse probleme ale lumii reale, cum ar fi: gestionarea memoriei cache în procesoarele de tip stream, decompoziția rețelelor Petri, analiza codului sursă a programelor, etc.

**Avantajele și valoarea elaborărilor propuse:** Rezultatele prezentate constau în lărgirea ariei de cercetare și aplicare a grafurilor perfecte, în special a grafurilor tranzitiv orientabile prin definirea a noi clase de subgrafuri stabile, precum și prin formalizarea metodelor de construire a orientărilor tranzitive și reconstruire a lor în baza unor condiții inițiale date. Elaborările propuse au o valoare științifică importantă datorită gradelor lor de noutate și originalitate. Rezultatele obținute pot fi utilizate în diverse domenii și pot avea aplicații practice în sortarea mulțimilor parțial ordonate, gestionarea memoriei cache în procesoarele de tip stream, decompoziția rețelelor Petri.

**Recomandări:** În calitate de recomandări am putea menționa necesitatea dezvoltării următoarelor direcții de cercetare:

- în contextul rezultatelor obținute merită interes studierea cazului grafurilor infinite;
- rezultatele prezentate în lucrare pot servi ca suport pentru continuarea cercetărilor cu privire la orientarea tranzitivă mixtă a grafurilor, chestiune ce nu se regăsește la soluționarea unor probleme practice;
- rezultatele obținute pot fi aplicate pentru soluționarea problemelor actuale ce pot fi modelate cu ajutorul grafurilor tranzitiv orientabile.
- rezultatele tezei pot servi drept suport pentru un curs opțional la masterat.

## BIBLIOGRAFIE


[1] Allaire F., *Another proof of the four colour theorem. Part I,* Proceedings of the 7th Manitoba Conference on Numerical Mathematics and Computing, no. 20, p. 3-72, 1978.

[2] Banakar R., Steinke S., Lee B.-S., Balakrishnan P., Marwedel P., *A Design Alternative for Cache on-chip memory in Embedded Systems*, CODES '02: Proceedings of the tenth international symposium on Hardware/software codesign, New York, 2002, p. 73-78.

[3] Bang-Jensen J., Gutin G., *Digraphs: Theory, Algorithms and Applications*. Berlin: Springer-Verlag, 2007, 772 p.

[4] Basak S. C., Restrepo G., Villaveces L J., *Advances in Mathematical Chemistry and Applications*. vol. 1, Bentham Books, Bogota, Columbia, 2014, 337 p.

[5] Bazydlo G., *Graphic specification of programs for reconfigurable logic controllers using Unified Modeling Language*, Lecture Notes in Control and Computer Science vol. 19 ser. LNCCS, University of Zielona Góra Press, Poland, Zielona Góra, , 2012, 117p.

[6] Bazydlo G., Adamski M., *Specification of UML 2.4 HSM and Electrical Review,* Lecture Notes in Control and Computer Science vol. 11, 2011, p. 145-149.

[7] Berge C., *Perfect graphs,* Studies in graph theory, Part I (D. R. Fulkerson, ed.), Studies in Math., Vol. 11, Math. Assoc. Amer., Washington, D. C., 1975, p. 1-22.

[8] Berge C., *Some classes of perfect graphs,* Combinatorial Mathematics and its Applications (Proc. Conf., Univ. North Carolina, Chapel Hill, N.C., 1967; R.C. Bose, T.A.Dowling, eds.) Univ. North Carolina Press, Chapel Hill, New York City, 1969, p. 539-552.

[9] Berge C., *Théorie des graphes et ses applications*, Dunod, Ed. Paris, 1958, 275 p..

[10] Berge C., Ghouila-Houri A., *Programmes, jeux et réseaux de transport,* Dunod, Ed. Paris, 1962.

[11] Bonomo F., Mattia S., Oriolo G., *Bounded coloring of co-comparability graphs and the pickup and delivery tour combination problem*, Theoretical Computer Science, vol. 412, no. 45, 2011, p. 6261-6268.

[12] Brandstädt A., Van Bang L., Spinrad J. P., *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999, 295 p.

[13] Cameron K, Eschen E. M., Hoang C. T., Sritha R., *Recognition of Perfect Circular-arc Graphs*, Graph Theory in Paris, Proceedings of a Conference in Memory of Claude Berge, Paris, 2006, p. 97-108.

[14] Cataranciuc S., *Teoria grafurilor în probleme și aplicații*.Chișinău:CEP USM, 2004,102 p.

[15] Cataranciuc S., Grigoriu N., *Algorithmic Approach in Reorientation of Comparability Graphs*, Studia Universitatis Babeș-Bolyai, Informatica, vol. LX, no. 2, 2015, p. 37-46.

[16] Cataranciuc S., Grigoriu N. *Clase de subgrafuri stabile în orientarea tranzitivă a grafurilor*, Studia Universitatis Moldaviae, Seria „Științe exacte și economice". 2015, No.2(82), p. 21-30.

[17] Cataranciuc S., Grigoriu N., *Lanțuri netriangulate și mulțimi stabile într-un graf neorientat*, Conferința interuniversitară. Educație prin cercetare – garant al performanței învățământului superior., Chișinău, 2012, p. 88-90.

[18] Cataranciuc S., Grigoriu N., *Transitive orientations on undirected graphs*, Proceedings of the „Doctoral Intensive Summer School on Evolutionary Computing in Optimisation and Data Mining (ECODAM)", România, Iași, 2012, p. 165-169.

[19] Cataranciuc S., Niculiță A., *Aspecte algoritmice ale teoriei grafurilor. Partea I*. Chișinău: CEP USM, 2006, 142 p.

[20] Cataranciuc S., Scripnic M., Soltan P., *About the median does not depend on the space*, The 30-th Annual Congres of the American Romanian Academy of Arts and Sciences (ARA), Proceedings., Chişinău, 2005, p. 58-61.

[21] Cataranciuc S., Soltan P., Cepoi V., *Convexitatea generalizată şi aplicaţiile ei*., Lucrările conferinței pregătitoare pentru Congresul matematicienilor români, Bucureşti, 1990, p. 145-154.

[22] Chartrand G., Lesniak L., Zhang P., *Graphs & Digraphs*, 5th ed.: Chapman and Hall/CRC, 2011, 598 p..

[23] Christofides N., *Graph Theory. An Algorithmic Approach*. Orlando: Academic Press Inc, Orlando, 1975, 415 p.

[24] Chudnovsky M., Cornuéjols G., Liu X., Seymour P., Vušković K., *Recognizing Berge graphs*, Combinatorica, vol. 25, no. 2, 2005, p. 143–186.

[25] Chudnovsky M., Robertson N., Seymour P., *The strong perfect graph theorem*, Annals of Mathematics, vol. 164, no. 1, 2006, p. 51–229.

[26] Chvátal V., Berge C., *Topics on Perfect Graphs*, Annals of discrete mathematics, vol. 21. Amsterdam, Elsevier, 1984.

[27] Clifford M. W., *Applied Graph Theory*. Michigan: John Wiley & Sons Inc, 1972, 322 p.

[28] Cornelsen S., Di Stefano G., *Treelike comparability graphs: Characterization, Recognition, and Applications*, Elsevier, vol. 157, no. 8, 2009, p. 1711–1722.

[29] Curtis A. R., Izurieta C., Joeris B., Lundberg S., *An implicit representation of chordal comparability graphs in linear time*, Discrete Applied Mathematic, vol. 158, 2010, p. 869-875.

[30] Del Cuvillo J., Zhu W., Hu Z., Gao G.R., *FAST: A Functionally Accurate Simulation Toolset for the Cyclops64 Cellular Architecture*, Workshop on Modeling, Benchmarking, and Simulation (MoBS2005), in conjuction with the 32nd Annual International Symposium on Computer Architecture, Newark, Delaware, 2005, p. 11-20.

[31] Dally W. J. ș.a., *Merrimac: Supercomputing with Streams*, SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing. IEEE Computer Society, New York, 2003, p. 35-44.

[32] Daskalakis C., Karp R. M, Mossel E., Riesenfeld S. J., Verbin E., *Sorting and selection in posets*, SIAM Journal on Computing, vol. 40, no. 3, Society for Industrial and Applied Mathematics, 2011, p. 597-622.

[33] Dell'Olmo P., Grazia M., Speranzab Z., *Comparability graph augmentation for some multiprocessor scheduling problems*, Discrete Applied Mathematics, vol. 7, 1997, p.71-84.

[34] Dickinson S., Pelillo R. Z., *Introduction to the special section on graph algorithms in computer vision*, IEEE on pattern analysis, Pattern Analysis and Machine Intelligence, vol. 23, no. 10, 2001.

[35] Euler L., *Demonstratio nonnullarum insignium proprietatum, quibus solida hedris planis*, Novi Commentarii Academiae Scientiarum Petropolitanae, no. 4, 1758, p. 140-160.

[36] Floyd R. W., *Algorithm 97, shortest path*, Communications of the Association for Computing Machinery, vol. 5, no.6, 1962, p. 345.

[37] Fulkerson D. R., *The perfect graph conjecture and pluperfect graph theorem*, The 2nd Chapel Hill Conf. on Combinatorics, Mathematics and its Applications, 1969, p. 171-175.

[38] Ghouila-Houri A., *Caracterisation des graphes non orientes dont on peut orienter les arretes de maniere*, C.R. Acad. Sci. Paris, no. 254, Paris, 1962, p. 1370-1371.

[39] Gilmore P. C., Hoffman A. J., *A characterization of comparability graphs and of interval graphs*, Canadian Journal of Math, no. 16, 1964, p. 539-548.

[40] Golumbic M.C., *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics)*. Amsterdam: North-Holland Publishing Co., 2004, vol. 57, 340 p..

[41] Golumbic M.C., *Comparability graphs and intersection graphs*, Discrete Mathematics, vol. 43 no. 1, 1983, p. 37-46.

[42] Golumbic M.C., *Comparability graphs and new martroid*, J. Combinatorial theory, vol. 55, no. 12575, 1977, p. 68-90.

[43] Golumbic M.C., *The complexity of comparability graph recognition and coloring*, vol. 18, no. 3, 1977, p. 199-208.

[44] Golumbic M.C., Trenk A. N, *Tolerance Graphs*. Cambridge: Cambridge University Press , 2004.

[45] Grigoriu N., *Asupra numărului de orientări tranzitive într-un graf*, Conferinţa Ştiinţifică, „Integrare prin Cercetare şi Inovare”, Chișinău, 2015, p.180-182 .

[46] Grigoriu N., *B-stable subgraphs in undirected graphs*, The third conference of Mathematical Society of the Republic of Moldova, Chișinău, 2014, p. 354-357.

[47] Grigoriu N., *Construction of a transitive orientation using B-stable subgraphs*, Computer Science Journal of Moldova, vol. 23, no. 67, 2015, p. 11-23.

[48] Grigoriu N., *Minimal stable subgraphs in undirected graphs*, International conference, Mathematics & Information Technologies: Research and Education(MITRE-2013), Chișinău, 2013, p. 48-49.

[49] Grigoriu N., *Number of transitive orientations in an undirected graph*, International conference, Mathematics & Information Technologies: Research and Education(MITRE-2015), Chișinău, 2015, p. 42-43.

[50] Grigoriu N., *Orientarea tranzitivă a grafurilor ce nu conține subgrafuri stabile maximale*, Математическое моделирование, оптимизация и информационные технологии, Chișinău, 2014, p. 87-94.

[51] Grigoriu N., *Stable subgraphs and non-triangulated chains*, The 20th conference on applied and industrial mathematics dedicated to academician Mitrofan M. Ciobanu, Chișinău, 2012, p. 118-119.

[52] Grigoriu N., *Subgrafuri stabile într-un graf neorientat*, Modelare matematică, optimizare și tehnologii informaționale. Conferința internațională, Chișinău, 2012, p. 19-23.

[53] Grigoriu N., *Subgrafurile B-stabile în orientarea tranzitivă a grafurilor*, Conferința Științifică Internațională a doctoranzilor Tendințe Contermporane ale Dezvoltării Științei: Viziuni ale Tinerilor Cercetători, Chișinău, 2015, p. 19.

[54] Grigoriu N., *Transitive Orientation of Graph using B-stable subgraphs*, The 23rd Conference on Applied and Industrial Mathematics (CAIM-2015), Suceava, 2015, p. 65.

[55] Grobelna I., *Formal verification of embedded logic controller specifcation with computer deduction in temporal logic*, Electrical Review, no. 12a, 2011, p. 40-43.

[56] Gross J. L., Yellen J., *Graph Theory and Its Applications*, 2nd ed., Chapman and Hall/CRC, 2005, 800 p.

[57] Gross J.L:, Yellen J., Zhang P., *Handbook of Graph Theory*, 2nd ed., Chapman and Hall/CRC, 2013, 1630 p.

[58] Harary F., *Graph Theory*, Addison-Wesley Publishing Co., Reading, Mass.-Menlo Park, Calif.-London, 1969, 274p.

[59] Harzheim E., *Ordered Sets*. New york: Springer Science & Business Media, 2005, 386 p.

[60] Heggernes P., Mancini F., Papadopoulos C., *Making arbitrary graphs transitively orientable: Minimal comparability completions*, Algorithms and Computation. Kolkata: Springer Berlin Heidelberg, vol. 4288, 2006, p. 419-428.

[61] Heggernes P., Mancini F., Papadopoulos C., *Minimal comparability completions of arbitrary graphs*, Discrete Applied Mathematics, vol. 156, no. 5, 2008, p. 705-718.

[62] Hell P., Huang J., *Lexicographic orientation and representation algorithms for comparability graphs, proper circular arc graphs, and proper interval graphs*, Journal of Graph Theory, vol. 20, no. 3, 2006, p. 361-374.

[63] Hoàng C. T., *Efficient algorithms for minimum weighted colouring of some classes of perfect graphs*," Discrete Applied Mathematics, vol. 55, no. 2, 1994, p. 133-143.

[64] Pal A. J., Sarma S. S., Biman R., *CCTP, Graph Coloring Algorithms - Soft Computing Solutions*, Proceedings of the 6th IEEE International Conference on Cognitive Informatics, 2007, p. 364-372.

[65] Jean M., *An Interval Graph is a Comparability Graph*, *Journal of combinatorial theory*, vol. 7, 1969, p. 189-190.

[66] Jung H. A., *On a Class of Posets and the Corresponding Comparability Graphs*, Journal of combinatorial theory, vol. 24, 1978, p. 125-133.

[67] Kasyanov V.N., Evstigneev V. A., *Graph Theory for Programmers: Algorithms for Processing Trees*., Kluwer Academic Publisher, Norwell, 2000.

[68] Klavík P., Kratochvíl J., Krawczyk T., Walczak B., *Extending Partial Representations of Function Graphs and Permutation Graphs*, Algorithms –- ESA 2012 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings, vol. 7501, 2012, p. 671-682.

[69] Kovalyov A., Esparza J., *A polynomial algorithm to compute the concurrency relation of free-choice signal transition graphs*, Proceeding of the International Workshop on Discrete Event Systems - WODES, 1995, p. 1-6.

[70] Lancia G., Bafna V., Istrail S., Lippert R., Schwartz R. , *SNPs Problems, Complexity, and Algorithms*, Algorithms — ESA 2001, Friedhelm Meyer auf der Heide, Ed.: Springer Berlin Heidelberg, ch. 5, 2001, p. 182-193.

[71] Lovász L., *Normal hypergraphs and the perfect graph conjecture*, Discrete Mathematics, 1972, p. 253–267.

[72] Makino J., Hiraki K., Inaba M., *GRAPE-DR: 2-Pflops massively-parallel computer with 512-core, 512-Gflops processor chips for scientific computing*, SC '07. Proceedings of the 2007 ACM/IEEE Conference on Supercomputing. ACM, 1C11, 2007, p. 1-11.

[73] Marcialis G. L., Roli F., Serrau A., *Graph Based and Structural Methods for Fingerprint Classification*, Heidelberg, Ed. Berlin : Springer verlag, 2007, p. 205-226..

[74] McConnell R. M., Spinrad J.P., *Linear-time transitive orientation*, Proceeding SODA '97 Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms, 1997, p. 19-25.

[75] Milik A., Hrynkiewicz E., *Synthesis and Implementation of Reconfigurable PLC on FPGA Platform*, International Journal of Electronics, vol. 58, no. 1, 2012, p. 85-94.

[76] Möhring R.H., *Algorithmic aspects of comparability graphs and interval graphs*, Graphs and Order, Springer Netherlands, 1985, vol. 147, p. 41-101.

[77] Möhring R.H., *Almost all comparability graphs are UPO*, Discrete Mathematics, vol. 50, p. 63-70, 1984.

[78] Aigner AP. M., Prins G., *Uniquely partially orderable graphs*, Journal London Math. Society. Vol. 2, 1970, p. 260–266

[79] Möhring R.H., *Folding, Graph Problems Related to Gate Matrix Layout and PLA*, Computational Graph Theory, vol. 7, 1990, p. 17-51.

[80] Moldovan G., *Bazele informaticii*. Cluj-Napoca: Litografia. Univiversității Babeș-Bolyai, 1985, vol. II., 286 p.

[81] Morvan M., Viennot L., *Parallel comparability graph recognition and modular decomposition*, STACS 96, vol. 1046, 1996, p. 169-180.

[82] Muller J. H., Spinrad J.P., *Incremental modular decomposition*, Journal of the ACM, vol. 36, no. 1, 1989, p. 1-19.

[83] Narasingh D., *Graph theory with applications to engineering and computer science*, Prentice Hall of Indiana, 1990, 496 p.

[84] Olariu S., *On sources in comparability graphs, with applications*, Discrete Mathematics, vol. 110, 1992, p. 289-292.

[85] Ore O., *Theory of graphs*, American Mathematical Society Colloquium Publications, Vol. XXXVIII, American Mathematical Society, 1962, 270 p.

[86] Penrice S. G., *On-line algorithms for ordered sets and comparability graphs*, Discrete Applied Mathematic, vol. 60, 1995, p. 319-329.

[87] Pirzada S., Dharwadker A., *Applications of gaph theory*, Journal of the Korean Society for Industrial and applied Mathematics, vol. 11, no. 4, 2007, p. 19-38.

[88] Roberts F.S., *Graph Theory and Its Applications to Problems of Society*, (CBMS-NSF Regional Conference Series in Applied Mathematics). Philadelphia: Society for Industrial and Applied Mathematics, 1987, vol. 29, 128 p..

[89] Roșu A., *Teoria grafelor, aloritmi, aplicații*. București: Ed. Militară, 1974, 271 p.

[90] Sands B., *Unsolved problems*, Order, vol. 1, no. 3, 1985, p. 311-313.

[91] Santarelli E., *Directed Graph Theory And The Economic*, Metroeconomic, vol. 46, no. 2, 1995, p. 111-126.

[92] Schenker A., Last M., Horst B., Ande A., *Clustering of Web documents using a graph model*, Springer werlog, 2007.

[93] Schroeder B., *Ordered Sets: An Introduction*. Basel: Springer Science & Business Media, 2003, 377 p.

[94] Shamir R., *Advanced Topics in Graph Algorithms*, Sariel Har-Peled, Ed. Tel-Aviv: Tel-Aviv University, 1994, 153 p.

[95] Simon K., Trunz P., *A cleanup on transitive orientation*, Lecture Notes in Computer Science, vol. 831, 1994, p. 59-85.

[96] Spinrad J. P., *Efficient Graph Representations*, American Mathematical Society, Fields Institute, 2003, 342 p.

[97] Spinrad J. P., *On Comparability and Permutation Graphs*, Siam Journal on Computing - SIAMCOMP, vol. 14, no. 3, 1985, p. 658-670.

[98] Stefanowicz L., Adamski M., Wiśniewski R., Lipiński J., *Application of Hypergraphs to SMCs Selection*, Technological Innovation for Collective Awareness Systems, vol. 423, 2014, p. 249-256.

[99] Toadere T., *Grafe: teorie, algoritmi și aplicații*, Smaranda Derveșeanu, Ed. Cluj-Napoca: Editura Albastră, 2009, 199 p.

[100] Toadere T., Cataranciuc S., Iacob E-M., *Probleme de teoria grafelor*. Cluj-Napoca: Lit. Univ. Babeș-Bolyai, 1994.

[101] Tomescu I., *Grafuri și Programare Liniară (Introducere elementară)*, Ed. Tehnică, București, 1975, 132 p.

[102] Trotter W.T, *Combinatorics and Partially Ordered Sets, Dimension Theory (Johns Hopkins Studies in the Mathematical Sciences)*. Baltimore: The Johns Hopkins University Press, 1992, 328 p.

[103] Trotter W.T., Moore J. I. Jr., Sumner D. P., *The dimension of a comparability graph*, Procedings of the American Math Society, vol. 60, 1976, p. 35-38.

[104] Wallis W D, *A Beginner's Guide to Graph Theory*, 2nd ed., Basel: Birkhäuser Basel, 2007, 224 p..

[105] Wen-Lian H., Tze-Heng M., *Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs*, SIAM J. Comput., vol. 28, 1999, p. 1004-1020.

[106] Wilson R. J., *Introduction to Graph Theory*, 4th ed. Edinburgh, Longman, 1996, 192 p.

[107] Wisniewska M., *Application of Hypergraphs in Decomposition*, University of Zielona Góra, Zielona Góra, Lecture Notes in Control and Computer Science Vol. 23 ser. LNCCS, 2012, 147 p.

[108] Wisniewski R., Karatkevich A., Adamski M., Kur D., *Application of comparability graphs in decomposition of Petri nets*, Human System Interactions (HSI), 2014 7th International Conference, 2014, p. 216-220.

[109] Wolk E. S., *The comparability graph of a tree*, Proceedings of the American Mathematical Society, 13, 1962, p. 789-795.

[110] Xuejun Y., Li W., Jingling X., Yu D., Ying Z., *Comparability Graph Coloring for Optimizing Utilization of Stream Register Files in Stream Processors*, Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming, vol. 44, no. 4, 2009, p. 111-120.

[111] Yap I. V. et al., *A Graph-Theoretic Approach to Comparing and Integrating Genetic, Physical and Sequence-Based Maps*, Genetics, vol. 165, no. 4, 2003, p. 2235-2247.

[112] Белов В. В., *Теория графов : Учебное пособие для втузов*. Москва: Высшая школа, 1976, 392 с..

[113] Евстигнеев В. А., *Применение теории графов в программировании.* Москва: Наука, 1985, 352 с.

[114] Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И., *Лекции по теории графов*. Москва: Наука, 1990, 384 с.

[115] Зыков А. А., *Основы теории графов*. Москва: Вузовская книга, 2004, 664 с.

[116] Зыков А. А., *Теория конечных графов*. Новосибирск: Наука, 1969, vol. I, 543 с.

[117] Солтан П., Болтянский В. , *Комбинаторная геометрия различных классов выпуклых*. Кишинев: Штиинца, 1978, 282 с.

[118] Солтан П.С., Замбицкий Д.К., Присакару К.Ф., *Экстремальные задачи на графах и алгоритмы их решения*, Кишинев: Штиинца, 1973, 92 с.

[119] Шеврин Л.Н., Филипов Н. Д., *Частично упорядоченные множества и их графы сравнимости*, Сибирский Математический Журнал, vol. 11, no. 3, 1970, p. 648-667.

# ANEXE

## Anexa 1. Implementarea algoritmilor elaborați în limbajul Javascript

```
/**
 * Declare global variables for recursive usage.
 */
var processed = [],
    emptyGraph,
    potential = [],
    stableGraph,
    factorVertices = [];

function convertToAdjList(links) {
  var list = [];
  links.forEach(function (link) {
    var s = link.source.index;
    var t = link.target.index;
    if (!list[s]) {
      list[s] = [];
    }
    list[s].push(t);

    if (!list[t]) {
      list[t] = [];
    }
    list[t].push(s);
  });
  return list;
}

/**
 * Get B-Stable Subgraph
 */
function SBS(graph) {
  if (!isKn(graph)) {
    potential = sortDesc(graph);
    potential.forEach(function(node) {
      emptyGraph = [];
      Potential(node, node);
      if (stableGraph.length != potential.length) {
        potential = SBS(stableGraph);
      }
     if (!list[s]) {
      list[s] = [];
     }
      list[s].push(t);

      if (!list[t]) {
        list[t] = [];
      }
      list[t].push(s);
    });
  return list;
    });
  }
  return stableGraph;
}

/**
```

```
 * Get potential B-stable subgraph.
 */
function Potential(x, y, graph) {
  var notGamma = notGamma(x, graph);
  notGamma.forEach(function (z) {
    if (Gamma(x, graph).length == Gamma(z, graph).length) {
      emptyGraph.push(z);
    }
  });
  if (emptyGraph.length == 0) {
    emptyGraph.push(y);
  }
  var gamma = Gamma(y, graph);
  gamma.forEach(function(z) {
    if(processed.indexOf(z) == -1 && Gamma(z, graph).length <= Gamma(x,
graph).length) {
      processed.push(z);
      stableGraph.push(z);
      Potential(x, z, stableGraph);
    }
  });
}

/**
 * Check if graph is complete
 * @param graph as adjacency list.
 *
 * @returns boolean
 */
function isKn(graph) {
  var isKn = true;
  graph.forEach(function(adjacency){
    if (adjacency.length != graph.length -1) {
      isKn = false;
    }
  });
  return isKn;
}

/**
 * Check if graph is empty
 * @param graph as adjacency list.
 *
 * @returns boolean
 */
function isEmpty(graph) {
  graph.forEach(function(adjacency){
    if (adjacency.length != graph.length -1) {
      isKn = false;
    }
  });
  return isKn;
  }
  var isEmpty = true;
  graph.forEach(function(adjacenty) {
    if (adjacenty.length == 0) {
      isEmpty = false;
    }
  });
  return isEmpty;
}
```

```
/**
 * Sort graph based on adjacency of vertexes in descending order.
 *
 * @param graph
 * @returns sorted graph
 */
function sortDesc(graph) {
  var swapped;
  do {
    swapped = false;
    for (var i = 0; i < graph.length - 1; i++) {
      if (graph[i].length < graph[i + 1].length) {
        var temp = graph[i];
        graph[i] = graph[i + 1];
        graph[i + 1] = temp;
        swapped = true;
      }
    }
  } while (swapped);
  return graph;
}

/**
 * Get adjacency list of a vertex.
 * @param x
 * @param graph
 * @returns array of adjacent vertices.
 */
function Gamma(x, graph) {
  var gamma = [];
  if(graph[x]) {
    gamma = graph[x];
  }
  return gamma;
}

/**
 * Get vertices that are not adjacent to x.
 *
 * @param x
 * @param graph
 * @returns array of not adjacent vertices.
 */
function notGamma(x, graph) {
  var notGamma = [];
  graph.forEach(function(node){
    if (graph[x].indexOf(node) == -1) {
      notGamma.push(node);
    }
  });
  return notGamma;
}

function factorize(F, G) {
  factorVertices.push({id: G.length + 1, label: F.join()});
  var factorVertex;

  F.forEach(function (vertex) {
    G.splice(G.indexOf(vertex), 1);
    G.forEach(function (node) {
      node.splice(node.indexOf(vertex), 1);
```

```
      factorVertex = node;
    });
    node.push(G.length + 1);
  });
  factorVertex.push(G.length + 1);
  G.push(factorVertex);

  return G;
}

/**
 * Get type of the B-stable subgraph
 *
 * @param F B-stable subgraph
 *
 * @returns graph type as numbers
 *   0 - Empty subgraph
 *   1 - Complete subgraph
 *   2 - Minimal stable subgraph
 */
function TypeOfSubGraph(F) {
  if (isEmpty(F)) {
    return 0;
  }
  else if (isKn(F)) {
    return 1;
  }
  else {
    return 2;
  }
}

function KnTRO(F) {

}

(function ($) {
  Drupal.behaviors.d3_graph = {
    attach: function () {
      // set up SVG for D3
      var width  = 960,
          height = 500,
          colors = d3.scale.category10();

      var svg = d3.select('.graph-canvas')
        .attr('width', width)
        .attr('height', height);

        var gid = Drupal.settings.graph.gid;
        //var vertexes = Drupal.settings.graph.nodes;
        //var edges = Drupal.settings.graph.links;

      // set up initial nodes and links
      //  - nodes are known by 'id', not by index in array.
      //  - reflexive edges are indicated on the node (as a bold black
circle).
      //  - links are always source < target; edge directions are set by
'left' and 'right'.
      var nodes = [
          {id: 1, reflexive: false},
          {id: 2, reflexive: false},
```

```
    {id: 3, reflexive: false},
    {id: 4, reflexive: false},
    {id: 5, reflexive: false}
  ],
  lastNodeId = 5,
  links = [
    {source: nodes[0], target: nodes[1], left: false, right: false },
    {source: nodes[0], target: nodes[2], left: false, right: false },
    {source: nodes[0], target: nodes[3], left: false, right: false },
    {source: nodes[0], target: nodes[4], left: false, right: false },
    {source: nodes[4], target: nodes[1], left: false, right: false },
    {source: nodes[4], target: nodes[2], left: false, right: false },
    {source: nodes[4], target: nodes[3], left: false, right: false },
    {source: nodes[1], target: nodes[2], left: false, right: false }
  ];

// init D3 force layout
var force = d3.layout.force()
    .nodes(nodes)
    .links(links)
    .size([width, height])
    .linkDistance(150)
    .friction(0)
    .gravity(0)
    .charge(-100)
    .on('tick', tick);

// define arrow markers for graph links
svg.append('svg:defs').append('svg:marker')
    .attr('id', 'end-arrow')
    .attr('viewBox', '0 -5 10 10')
    .attr('refX', 6)
    .attr('markerWidth', 3)
    .attr('markerHeight', 3)
    .attr('orient', 'auto')
  .append('svg:path')
    .attr('d', 'M0,-5L10,0L0,5')
    .attr('fill', '#000');

svg.append('svg:defs').append('svg:marker')
    .attr('id', 'start-arrow')
    .attr('viewBox', '0 -5 10 10')
    .attr('refX', 4)
    .attr('markerWidth', 3)
    .attr('markerHeight', 3)
    .attr('orient', 'auto')
  .append('svg:path')
    .attr('d', 'M10,-5L0,0L10,5')
    .attr('fill', '#000');


// line displayed when dragging new nodes
var drag_line = svg.append('svg:path')
  .attr('class', 'link dragline hidden')
  .attr('d', 'M0,0L0,0');

// handles to link and node element groups
var path = svg.append('svg:g').selectAll('path'),
    circle = svg.append('svg:g').selectAll('g');

// mouse event vars
```

```
      var selected_node = null,
          selected_link = null,
          mousedown_link = null,
          mousedown_node = null,
          mouseup_node = null;

      function resetMouseVars() {
        mousedown_node = null;
        mouseup_node = null;
        mousedown_link = null;
      }

      // update force layout (called automatically each iteration)
      function tick() {
        // draw directed edges with proper padding from node centers
        path.attr('d', function(d) {
          var deltaX = d.target.x - d.source.x,
              deltaY = d.target.y - d.source.y,
              dist = Math.sqrt(deltaX * deltaX + deltaY * deltaY),
              normX = deltaX / dist,
              normY = deltaY / dist,
              sourcePadding = d.left ? 17 : 12,
              targetPadding = d.right ? 17 : 12,
              sourceX = d.source.x + (sourcePadding * normX),
              sourceY = d.source.y + (sourcePadding * normY),
              targetX = d.target.x - (targetPadding * normX),
              targetY = d.target.y - (targetPadding * normY);
          return 'M' + sourceX + ',' + sourceY + 'L' + targetX + ',' +
targetY;
        });

        circle.attr('transform', function(d) {
          return 'translate(' + d.x + ',' + d.y + ')';
        });
      }

      // update graph (called when needed)
      function restart() {
        // path (link) group
        path = path.data(links);

        // update existing links
        path.classed('selected', function(d) { return d === selected_link; })
          .style('marker-start', function(d) { return d.left ? 'url(#start-
arrow)' : ''; })
          .style('marker-end', function(d) { return d.right ? 'url(#end-
arrow)' : ''; });


        // add new links
        path.enter().append('svg:path')
          .attr('class', 'link')
          .classed('selected', function(d) { return d === selected_link; })
          .style('marker-start', function(d) { return d.left ? 'url(#start-
arrow)' : ''; })
          .style('marker-end', function(d) { return d.right ? 'url(#end-
arrow)' : ''; })
          .on('mousedown', function(d) {
            if(d3.event.ctrlKey) return;

            // select link
```

```
            mousedown_link = d;
            if(mousedown_link === selected_link) selected_link = null;
            else selected_link = mousedown_link;
            selected_node = null;
            restart();
          });

        // remove old links
        path.exit().remove();


        // circle (node) group
        // NB: the function arg is crucial here! nodes are known by id, not
by index!
        circle = circle.data(nodes, function(d) { return d.id; });

        // update existing nodes (reflexive & selected visual states)
        circle.selectAll('circle')
          .style('fill', function(d) { return (d === selected_node) ?
d3.rgb(colors(d.id)).brighter().toString() : colors(d.id); })
          .classed('reflexive', function(d) { return d.reflexive; });

        // add new nodes
        var g = circle.enter().append('svg:g');

        g.append('svg:circle')
          .attr('class', 'node')
          .attr('r', 12)
          .style('fill', function(d) { return (d === selected_node) ?
d3.rgb(colors(d.id)).brighter().toString() : colors(d.id); })
          .style('stroke', function(d) { return
d3.rgb(colors(d.id)).darker().toString(); })
          .classed('reflexive', function(d) { return d.reflexive; })
          .on('mouseover', function(d) {
            if(!mousedown_node || d === mousedown_node) return;
            // enlarge target node
            d3.select(this).attr('transform', 'scale(1.1)');
          })
          .on('mouseout', function(d) {
            if(!mousedown_node || d === mousedown_node) return;
            // unenlarge target node
            d3.select(this).attr('transform', '');
          })
          .on('mousedown', function(d) {
            if(d3.event.ctrlKey) return;

            // select node
            mousedown_node = d;
            if(mousedown_node === selected_node) selected_node = null;
            else selected_node = mousedown_node;
            selected_link = null;

            // reposition drag line
            drag_line
              .style('marker-end', 'url(#end-arrow)')
              .classed('hidden', false)
              .attr('d', 'M' + mousedown_node.x + ',' + mousedown_node.y +
'L' + mousedown_node.x + ',' + mousedown_node.y);

            restart();
          })
```

```
          .on('mouseup', function(d) {
            if(!mousedown_node) return;

            // needed by FF
            drag_line
              .classed('hidden', true)
              .style('marker-end', '');

            // check for drag-to-self
            mouseup_node = d;
            if(mouseup_node === mousedown_node) { resetMouseVars(); return; }

            // unenlarge target node
            d3.select(this).attr('transform', '');

            // add link to graph (update if exists)
            // NB: links are strictly source < target; arrows separately
specified by booleans
            var source, target, direction;
            if(mousedown_node.id < mouseup_node.id) {
              source = mousedown_node;
              target = mouseup_node;
              direction = 'right';
            } else {
              source = mouseup_node;
              target = mousedown_node;
              direction = 'left';
            }

            var link;
            link = links.filter(function(l) {
              return (l.source === source && l.target === target);
            })[0];

            if(link) {
              link[direction] = true;
            } else {
              link = {source: source, target: target, left: false, right:
false};
              link[direction] = true;
              links.push(link);
            }

            // select new link
            selected_link = link;
            selected_node = null;
            restart();
          });

        // show node IDs
        g.append('svg:text')
            .attr('x', 0)
            .attr('y', 4)
            .attr('class', 'id')
            .text(function(d) { return d.id; });

        // remove old nodes
        circle.exit().remove();

        // set the graph in motion
        force.start();
```

```
  }

  function mousedown() {
    // prevent I-bar on drag
    //d3.event.preventDefault();

    // because :active only works in WebKit?
    svg.classed('active', true);

    if(d3.event.ctrlKey || mousedown_node || mousedown_link) return;

    // insert new node at point
    var point = d3.mouse(this),
        node = {id: ++lastNodeId, reflexive: false};
    node.x = point[0];
    node.y = point[1];
    nodes.push(node);

    restart();
  }

  function mousemove() {
    if(!mousedown_node) return;

    // update drag line
    drag_line.attr('d', 'M' + mousedown_node.x + ',' + mousedown_node.y +
'L' + d3.mouse(this)[0] + ',' + d3.mouse(this)[1]);

    restart();
  }

  function mouseup() {
    if(mousedown_node) {
      // hide drag line
      drag_line
        .classed('hidden', true)
        .style('marker-end', '');
    }

    // because :active only works in WebKit?
    svg.classed('active', false);

    // clear mouse event vars
    resetMouseVars();
  }

  function spliceLinksForNode(node) {
    var toSplice = links.filter(function(l) {
      return (l.source === node || l.target === node);
    });
    toSplice.map(function(l) {
      links.splice(links.indexOf(l), 1);
    });
  }

  // only respond once per keydown
  var lastKeyDown = -1;

  function keydown() {
    d3.event.preventDefault();
```

```
  if(lastKeyDown !== -1) return;
  lastKeyDown = d3.event.keyCode;

  // ctrl
  if(d3.event.keyCode === 17) {
    circle.call(force.drag);
    svg.classed('ctrl', true);
  }

  if(!selected_node && !selected_link) return;
  switch(d3.event.keyCode) {
    case 8: // backspace
    case 46: // delete
      if(selected_node) {
        nodes.splice(nodes.indexOf(selected_node), 1);
        spliceLinksForNode(selected_node);
      } else if(selected_link) {
        links.splice(links.indexOf(selected_link), 1);
      }
      selected_link = null;
      selected_node = null;
      restart();
      break;
    case 66: // B
      if(selected_link) {
        // set link direction to both left and right
        selected_link.left = true;
        selected_link.right = true;
      }
      restart();
      break;
    case 76: // L
      if(selected_link) {
        // set link direction to left only
        selected_link.left = true;
        selected_link.right = false;
      }
      restart();
      break;
    case 82: // R
      if(selected_node) {
        // toggle node reflexivity
        selected_node.reflexive = !selected_node.reflexive;
      } else if(selected_link) {
        // set link direction to right only
        selected_link.left = false;
        selected_link.right = true;
      }
      restart();
      break;
  }
}

function keyup() {
  lastKeyDown = -1;

  // ctrl
  if(d3.event.keyCode === 17) {
    circle
      .on('mousedown.drag', null)
      .on('touchstart.drag', null);
```

```
          svg.classed('ctrl', false);
        }
      }

      // app starts here
      svg.on('mousedown', mousedown)
        .on('mousemove', mousemove)
        .on('mouseup', mouseup);
      d3.select(window)
        .on('keydown', keydown)
        .on('keyup', keyup);
      restart();

      //Actions on save button submit event
     jQuery('#edit-save').on('click', function() {
        var graph_title = jQuery('#edit-title').val();
        //Pass graph to the $_POST variable for MySql insert
        jQuery.ajax({
          type: 'POST',
          url: 'graph-ajax-operations',
          data: {'nodes': nodes, 'links': links, 'gid': gid, 'graph':
graph_title, 'operation': 'save'},
          success: function(data) {
            console.log(data);
            alert(Drupal.t('Save operation executed.'));
          },
          error: function(XMLHttpRequest, textStatus, errorThrown) {
            console.log(errorThrown);
          }
        });
        return false;
      });

      // Perform tro calculations on form calculate button click.
      $('#edit-calculate').on('click', function() {
        var list = convertToAdjList(links);
        var BSS = SBS(list);
        return false;
      });
    }
  };
})(jQuery);
```

**Anexa 2. Codul sursă pentru reprezentarea grafică a algoritmilor propuși**

```
/**
 * @file graph.module
 * Implements functions for the connection of the server with the js
library
 */

/**
 * Implements hook menu()
 * @return array menu items
 */
function graph_menu() {
  $items['graph'] = array(
    'title' => 'Transitively orientable graph',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('graph_form'),
    'access arguments' => array('access content'),
    'type' => MENU_CALLBACK,
  );
  $items['graph/%'] = array(
    'title' => 'Transitively orientable graph',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('graph_form', 1),
    'access arguments' => array('access content'),
    'type' => MENU_CALLBACK,
  );
   //Basic ajax callback
  $items['graph-ajax-operations'] = array(
    'title' => 'AJAX processing of graph operations',
    'page callback' => 'graph_ajax_operations',
    'access callback' => TRUE,
    'type' => MENU_CALLBACK,
  );
  $items['graphs'] = array(
    'title' => 'Graphs',
    'page callback' => 'graph_gallery',
    'access arguments' => array('access content'),
    'type' => MENU_CALLBACK,
  );
  return $items;
}

/**
 * Implements hook library()
 * @return array libraries
 */
function graph_library() {
  $libraries['d3'] = array(
    'title' => 'D3',
    'website' => 'http://d3js.org/',
    'version' => '3.5.3',
    'js' => array(
      drupal_get_path('module', 'graph') . '/js/d3.min.js' => array(),
    ),
```

```
  );

  return $libraries;
}

/**
 * Implements hook_theme
 *
 * Theme of the form using the graph-canvas template
 *
 * @return array rendered array of the theme function
 */
function graph_theme() {
  return array(
    'graph_form' => array(
      'arguments' => array('form' => NULL),
      'template' => 'graph-canvas',
      'render element' => 'form',
    ),
  );
}

/**
 * Form construction
 *
 * @param  array $form         form variables
 * @param  array &$form_state state of the form variables
 * @return array               custom form elements
 */
function graph_form($form, &$form_state, $gid = NULL) {
  $vertexes = array();
  $links = array();
  if ($gid != NULL) {
      while ($row = $result->fetchAssoc()) {
        $row['left'] = ($row['to_left'] == 1) ? TRUE : FALSE;
        unset($row['to_left']);
        $row['right'] = ($row['to_right'] == 1) ? TRUE : FALSE;
        unset($row['to_right']);
        $row['source'] = $vertexes[$row['source']];
        $row['target'] = $vertexes[$row['target']];
        $links[] = $row;
      }
    //Select graph by its ID
    $result_graph = db_select('graph', 'g')
      ->fields('g', array('title'))
      ->condition('gid', $gid, '=')
      ->execute()
      ->fetchAssoc();

      //Select all vertexes by graph ID
      $query = db_select('vertex', 'v');
      $query->addField('v', 'vid', 'id');
      $query->fields('v', array('reflexive', 'label', 'weight', 'x',
'y', 'px', 'py'))
        ->condition('gid', $gid, '=');
      $result = $query->execute();
```

```
      while ($row = $result->fetchAssoc()) {
        $vertexes[] = array(
          'id' => (int)$row['id'],
          'reflexive' => ($row['reflexive'] == 1) ? TRUE : FALSE,
          'index' => (int)$row['label'],
          'weight' => (int)$row['weight'],
          'x' => (float)$row['x'],
          'y' => (float)$row['y'],
          'px' => (float)$row['px'],
          'py' => (float)$row['py'],
        );
      }

      //Select all links/edges by the graph ID
      $query = db_select('links', 'l');
      $query->fields('l', array('source', 'target', 'to_right',
'to_left'));
      $query->condition('gid', $gid, '=');
      $result = $query->execute();
      while ($row = $result->fetchAssoc()) {
        $row['left'] = ($row['to_left'] == 1) ? TRUE : FALSE;
        unset($row['to_left']);
        $row['right'] = ($row['to_right'] == 1) ? TRUE : FALSE;
        unset($row['to_right']);
        $row['source'] = $vertexes[$row['source']];
        $row['target'] = $vertexes[$row['target']];
        $links[] = $row;
      }
      $result_graph = db_select('graph', 'g')
      ->fields('g', array('title'))
      ->condition('gid', $gid, '=')
      ->execute()
      ->fetchAssoc();
  }

  $form['title'] = array(
    '#type' => 'textfield',
    '#title' => t('Graph name'),
    '#required' => TRUE,
    '#size' => 12,
    '#value' => ($gid != NULL) ? $result_graph['title'] : t('Graph'),
  );
  $form['save'] = array(
    '#type' => 'submit',
    '#value' => t('Save'),
  );

  //attach d3 libraries and custom functionality to the page
  $form['#attached']['library'][] = array('graph', 'd3');

  $graph = array(
    'gid' => $gid,
    'nodes' => $vertexes,
    'links' => $links,
  );
  //Attach graph operations and pass variable from php to js
```

```
  $form['#attached']['js'] = array(
    drupal_get_path('module', 'graph') . '/js/d3.graph_op.js' =>
array(),
    array(
      // Pass PHP variables to Drupal.settings.
      'data' => array('graph' => $graph),
      'type' => 'setting',
     ),
  );
    $vertexes[] = array(
      'gid' => $gid,
      'vid' => $node['id'],
      'reflexive' => ($node['reflexive'] === 'true') ? 1 : 0,
      'label' => $node['index'],
      'weight' => $node['weight'],
      'x' => $node['x'],
      'y' => $node['y'],
      'px' => $node['px'],
      'py' => $node['py'],
    );

  $form['#attached']['css'] = array(
    drupal_get_path('module', 'graph') . '/css/graph.css',
  );
  return $form;
}

/**
 * Graph AJAX operations.
 */
function graph_ajax_operations(&$form, &$form_state) {
  if ($_POST) {
    $nodes = $_POST['nodes'];
    $links = $_POST['links'];
    $graph = $_POST['graph'];
    $gid = $_POST['gid'];

    if ($gid == NULL) {
      //Save graph title and image
      db_insert('graph')->fields(array(
        'title' => $graph,
//        'image' => ($values['image']) ? $image . $values['image'] :
$image . 'graph.png'
      ))->execute();

      //Extract graph id for vertexes saving
      $result = db_select('graph', 'g')
        ->fields('g', array('gid'))
        ->orderBy('gid', 'DESC')
        ->range(0, 1)
        ->execute()
        ->fetchAssoc();
      $gid = $result['gid'];
    }
    else {
      //There could be added new links and nodes in a graph, so it is
```

```
neccessary to
      //delete all of them in order to save them again after graph
altering
      $deleted_vertexes = db_delete('vertex')
        ->condition('gid', $gid)
        ->execute();
      $delteed_links = db_delete('links')
        ->condition('gid', $gid)
        ->execute();
    }

    //Insert vertexes into the 'vertex' table
    $vertexes = array();
    foreach ($nodes as $node) {
      $vertexes[] = array(
        'gid' => $gid,
        'vid' => $node['id'],
        'reflexive' => ($node['reflexive'] === 'true') ? 1 : 0,
        'label' => $node['index'],
        'weight' => $node['weight'],
        'x' => $node['x'],
        'y' => $node['y'],
        'px' => $node['px'],
        'py' => $node['py'],
      );
    }
    $query = db_insert('vertex')->fields(array('gid', 'vid',
'reflexive', 'label', 'weight', 'x', 'y', 'px', 'py'));
    foreach ($vertexes as $record) {
      $query->values($record);
    }
    $query->execute();

    //Insert links into the 'links' table
    $edges = array();
    foreach ($links as $link) {
      $edges[] = array(
        'gid' => $gid,
        'source' => $link['source']['id'],
        'target' => $link['target']['id'],
        'to_left' =>  ($link['left'] === 'true') ? 1 : 0,
        'to_right' => ($link['right'] === 'true') ? 1 : 0,
      );
    }
    $query = db_insert('links')->fields(array('gid', 'source',
'target', 'to_left', 'to_right'));
    foreach ($edges as $edge) {
      $query->values($edge);
    }
    $query->execute();

    print json_encode(array('nodes' => $nodes, 'links' => $links,
'graph' => $graph));
  }
  exit();
}
```

```
function graph_gallery() {
  //Select all graphs from database
  $result = db_select('graph', 'g')
    ->fields('g')
    ->execute();
  $output = '';
  while ($row = $result->fetchAssoc()) {
    $output .= '<div class="graph-data"><div class="graph-
image"></div>'
            . '<a href="graph/' . $row['gid'] . '">' . $row['title'] .
'</a></div>';
  }

  return $output;
}
```

## DECLARAȚIA PRIVIND ASUMAREA RĂSPUNDERII

Subsemnatul, declar pe răspundere personală că materialele prezentate în teza de doctorat sunt rezultatul propriilor cercetări și realizări științifice. Conștientizez, că în caz contrar, urmează să suport consecințele în conformitate cu legislația în vigoare.

Grigoriu Nicolae

Semnătura:

Data:

**CURRICULUM VITAE**



***Numele de familie și prenumele:*** Grigoriu Nicolae

***Data nașterii:*** 19.07.1988

***Locul nașterii:*** Republica Moldova,
or. Cahul

***Cetățenia:*** Republica Moldova

***Studii:***

***licență:*** Universitatea de Stat din Moldova, 2007 – 2010, specialitatea Matematică Aplicată, calificarea Licențiat în Științe Exacte.

***masterat:*** Universitatea de Stat din Moldova 2010 – 2012, specialitatea Analiza Statistică a Datelor și Optimizarea Proceselor Economico-Financiare, calificarea Master în științe Exacte.

***doctorat:*** Universitatea de Stat din Moldova 2012 – 2015 , specialitatea 112.03 – Cibernetică Matematică și Cercetări Operaționale.

***Domenii de interes științific:***

Teoria grafurilor, teoria algoritmilor, metode de optimizare.

***Activitatea profesională:***

Universitatea de Stat din Moldova, Laboratorul de Informatică și Optimizare Discretă, laborant de categoria I, 2008 – 20010.

Universitatea de Stat din Moldova, Departamentul de Matematici Aplicate, lector, 2010 – 2014

***Participări la proiecte științifice naționale și internaționale:***

1. Proiect 06.411.030F (Proiect instituțional)

   Tema: „Complexe de relații multi-are și aplicații” (anul 2010).

2. Proiect 11.817.08.47A (proiect instituțional)

   Tema: „Modele matematice și metode de soluționare a problemelor de optimizare și analiză numerică cu aplicații.” (anii 2011-2014).

3. Proiect pentru tineri cercetători 15.819.02.02

   Tema: „Metode numerice și algoritmi de aflare a strategiilor optime în problemele de dirijare a sistemelor dinamice deterministe și stocastice” (2015-2016).

***Participări la foruri științifice:***

1. Conferința Internațională „Modelare Matematică, Optimizare și Tehnologii Informaționale”, ATIC, Ediția a III-a, Chișinău, 19−23 martie 2012.
2. Conferința interuniversitară. Educație prin cercetare – garant al performanței învățământului superior., Chișinău, 3-4 mai 2012.
3. Doctoral Intensive Summer School on Evolutionary Computing and Optimisation and Data Mining, Iași, 17-24 iunie 2012.
4. International Conference „The 20th Edition of the Annual Conference on Applied and Industrial Mathematics – CAIM”, UST, Chișinău, august 22–25, 2012.
5. Stagiu de cercetare/documentare la Facultatea de Matematică și Informatică, Universitatea Babeș-Bolyai, Cluj-Napoca, România, 5 noiembrie – 5 decembrie 2012, în cadrul proiectului Center of Excellence for Applications in Mathematics.
6. The Conference “Mathematics & Information Technologies: Research and Education” (MITRE – 2013), August 18-22, 2013, Chișinău, Republic of Moldova.
7. Conferința științifică internațională “Modelare Matematică, Optimizare și Tehnologii Informaționale”, Ediția a IV-a, Academia de Transporturi, Informatică și Comunicații, Chișinău, 25-28 martie 2014.
8. The third conference of mathematical society of the Republic of Moldova: dedicated to the 50th anniversary of the foundation of the Institute of Mathematics and Computer Science - IMCS-50, Chisinau, August 19-23, 2014.
9. Conferința științifică internațională a doctoranzilor „Tendințe contemporane ale dezvoltării științei: viziuni ale tinerilor cercetători.”, 10 martie 2015 Chișinău.
10. *The Conference “Mathematics & Information Technologies: Research and Education” (MITRE – 2015), July 2-5, 2015, Chișinău, Republic of Moldova.*
11. The 23rd conference on applied and industrial mathematics (CAIM-2015), September 17-20, 2015, Suceava, Romania.

***Lucrări științifice publicate:***

Articole – 7, teze ale comunicațiilor științifice – 7.

***Cunoașterea limbilor:*** română – maternă, rusă – bine, engleză – bine, franceză – începător.

***Date de contact:*** grigoriunicolae@gmail.com.