Teoria grafurilor în optimizarea rețelelor de infrastructură urbană într-un oraș inteligent

Florentina PANĂ-MICU,
National University of Political Studies and Public Administration, Bucharest, Romania
florentina.micu@snspa.ro

Abstract
Crearea unor orașe inteligente reprezintă un deziderat din ce în ce mai larg dezbătut în societatea în care trăim, întrucât, beneficiind de tehnologii avansate, aceste orașe contribuie la creșterea calității vieții locuitorilor. În acest context, teoria grafurilor se dovedește a fi o unealtă deosebit de importantă în procesele de optimizare a rețelelor de infrastructură urbană. Obiectivul principal al cercetării este identificarea principalelor concepte din teoria grafurilor care pot fi aplicate în crearea de orașe inteligente cu scopul optimizării rețelelor de infrastructură urbană. Plecând de la acest obiectiv, cercetarea se va axa pe definirea principalelor concepte din teoria grafurilor precum: algoritmul lui Kruskal de determinare a unui arbore parțial de cost minim și algoritmul lui Dijkstra ce stabilește drumul de cost minim de la un nod de start la oricare altul dintr-un graf. Ulterior, articolul va cuprinde exemple prin care cu ajutorul teoriei grafurilor putem modela rețeaua de infrastructură urbană. Algoritmul lui Dijkstra are aplicabilitate în identificarea celor mai eficiente rute de transport și reducerea congestiei și a timpului de călătorie, precum și la optimizarea rețelei de conducte și a distribuției a apei. Pe de altă parte, algoritmul lui Kruskal vizează optimizarea unor arii ale orașului inteligent precum: gestionarea eficientă a resurselor, planificarea spațiilor verzi și conectivitatea durabilă. Așadar, în cadrul acestui articol vom explora potențialul pe care teoria grafurilor îl poate avea în optimizarea infrastructurii urbane a unui oraș inteligent și în creșterea eficienței și a sustenabilității acestui tip de oraș, întrucât alegerea optimă a conexiunilor și rețelelor poate contribui la crearea unui mediu urban care să răspundă nevoilor locuitorilor într-un mod durabil și eficent.

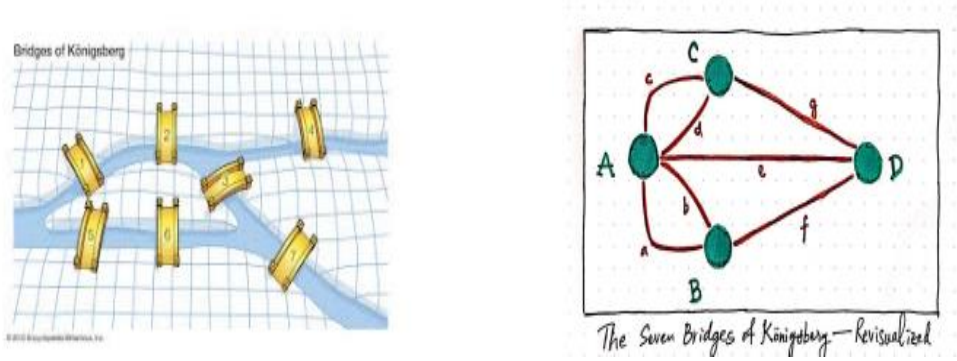
Cuvinte cheie: algoritmul lui Dijkstra, algoritmul lui Kruskal, arbore, graf, oraș inteligent.

1. Introducere
Aplicarea teoriei grafurilor în procesul de optimizare nu este una nouă, prima lucrare de teoria grafurilor fiind scrisă de Euler în anul 1736, cu scopul rezolvării "problemei celor 7 poduri" din Königsberg, prin identificarea unui traseu care să treacă o singură dată peste fiecare pod. [1]

Königsberg, oraş din Prusia Orientală (astăzi cunoscut sub numele de Kaliningrad în Rusia), este aşezat pe râul Pregel care înconjoară insula Kneiphof, după care se desparte în două braţe. Apa delimitează terenul în patru zone distincte. Malurile râului şi insula sunt legate prin şapte poduri, ca în fig.1. [2]

Problema pe care Leonard Euler încearcă să o rezolve constă în: "Este posibil să facem o plimbare prin oraş trecând pe fiecare din cele şapte poduri o singură dată şi să ne întoarcem de unde am plecat?".

Pentru a rezolva această problemă, Euler a dezvoltat concepte și notații care au stat la baza teoriei grafurilor. A introdus ideea de graf, reprezentând orașul și podurile sub formă de noduri și muchii, respectiv, și a demonstrat că soluția problemei depinde de conexiunile dintre aceste elemente marcând începutul teoriei grafurilor.



a) b)

Fig.1. Reprezentare grafică "problema celor șapte poduri din Königsberg"
Sursa: Alnassar, I. [3]

În prezent, teoria grafurilor are multiple aplicații practice în domenii precum: chimie, sociologie, tehnologia comunicațiilor, rețele de calculatoare, etc. De exemplu, un graf neorientat este compus din două mulțimi: o mulțime de vârfuri și o mulțime de muchii care pot reprezenta legăturile rutiere, legăturile dintre nodurile unei rețele de calculatoare sau ierarhiile dintr-o structură administrativă.

Plecând de la aceste considerente, cercetarea de față va pleca de la definirea conceptelor principale din teoria grafurilor cu rol în optimizarea infrastructurii urbane și va oferi un model de aplicare al algoritmului lui Kruskal si al algoritmului lui Dijkstra într-un oraș inteligent.

Importanța acestei cercetări derivă inclusiv de la definiția conceptului de "oraș inteligent" ca fiind acel tip de oraș care înglobează utilizarea infrastructurii tehnologice cu scopul îmbunătățirii eficienței serviciilor oferite și calității vieții locuitorilor. Un oraș inteligent înseamnă interconectivitate și acces facil la reurse, de aceea optimizarea joacă un rol foarte important în planificarea rețelelor de infrastructură urbană.

În acest context, optimizarea rețelelor de infrastructură urbană devine un element esențial în dezvoltarea unui oraș inteligent, iar teoria grafurilor o unealtă indispensabilă în proiectarea eficientă și durabilă a rețelelor de infrastructură urbană.

În acest articol, vom explora felul în care acești algoritmi matematici din teoria grafurilor (algoritmul lui Kruskal și algoritmul lui Dijkstra) pot fi integrați în procesul de planificare a rețelelor de infrastructură urbană.

Din punct de vedere metodologic, formulăm următoarea întrebare de cercetare: "Cum pot contribui algoritmii de optimizare din teoria grafurilor la o planificare și dezvoltare mai bună a infrastructurii urbane?".

Obiectivul general al cercetării este să identifice modalități în care algoritmii de optimizare din teoria grafurilor, cum ar fi algoritmul lui Kruskal și algoritmul lui Dijkstra, pot

îmbunătăți semnificativ dezvoltarea infrastructurii urbane și interconectivitatea într-un oraș inteligent.

2. Definirea conceptelor principale

Vom defini noțiunile elementare despre grafuri referindu-ne la grafurile neorientate pe care le vom numi pe scurt, grafuri.

Definiție: Un graf este o pereche (V(G), E(G)), unde V(G) este o mulțime finită și nevidă de elemente numite vârfuri**,** iar E(G) este o mulțime de perechi neordonate de elemente distincte din V(G), numite muchii. [1]

O muchie se notează {x, y} și vom spune că ea unește vârfurile x şi y.

Orice graf G poate fi desenat în plan reprezentând vârfurile sale prin puncte şi muchiile prin linii care unesc anumite perechi de vârfuri. Într-o astfel de reprezentare nu contează distanţele dintre vârfuri sau unghiurile dintre muchii.

Două vârfuri unite printr-o muchie se numesc adiacente.

De exemplu:

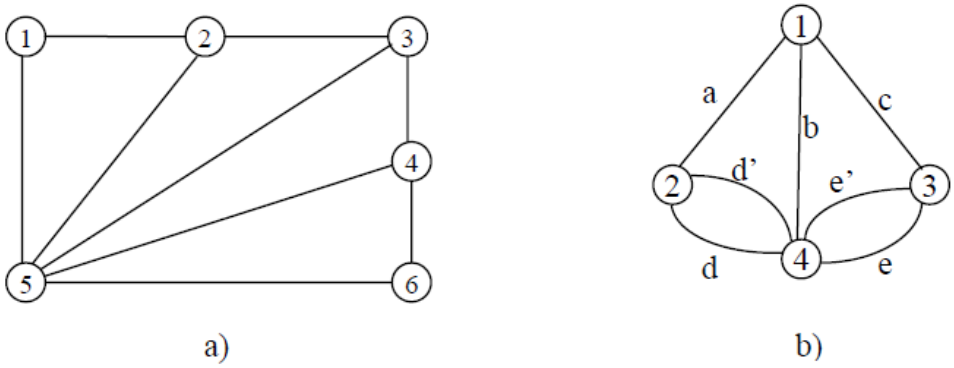


Fig. 2. Reprezentarea grafică a unui graf
Sursa: Nica, V.T. [4]

Cu alte cuvinte, punctele se numesc vârfuri (în imaginea a) vârfurile sunt reprezentate de numerele 1,2,3,3,5,6, iar în b) 1,2,3,4) , iar muchiile sunt liniile care unesc aceste vârfuri (în imaginea b) muchiile sunt reprezentate de a,b,c, d`, d, e, e`). Exprimare incoerentă!

Există situații cand nu este suficientă definirea sau reprezentarea printr-un graf neorientat, fiind necesar să fie dată o direcție muchiilor.

Un graf orientat G este o pereche (V, E) unde V este o mulţime finită şi nevidă de elemente numite vârfuri şi E este o mulţime de perechi ordonate de elemente distincte din V, numite arce.

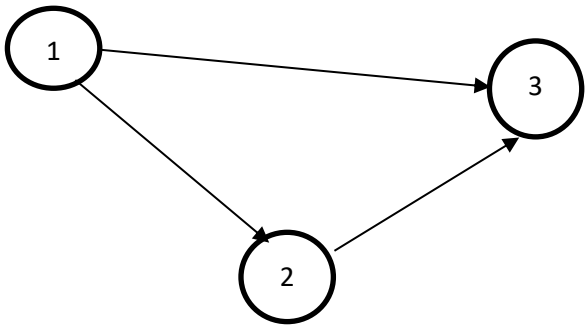


Fig.3. Reprezentarea grafică a unui graf orientat
Sursa: autorul

Graful pune în evidență conexiunile care există între anumite entități precum:

- Reţele de transport în care entităţile sunt localităţi, fabrici, depozite, iar legăturile sunt drumuri auto, feroviare, aeriene, navale;
- Reţele de comunicaţie audio, video, calculator prin care se conectează localităţi, locuinţe, instituţii, birouri;
- Reţele electrice;
- Rețele genealogice etc.

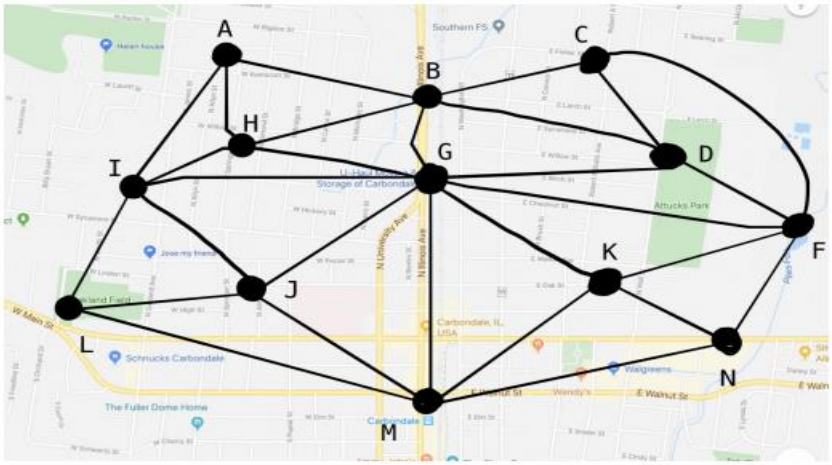


Fig. 4. Reprezentarea grafică a unui graf al stațiilor de transport public
Sursa: Alnassar, I. [3]

Definiții:

1. Un lanț într-un graf G este o succesiune finită de vârfuri ale lui G care se notează $\left[x_0, x_1, ..., x_r \right]$, cu proprietatea că oricare două vârfuri vecine sunt adiacente, adică $x_i x_{i+1} \in E(G)$, pentru orice $0 \le i \le r-1$.
2. Vârfurile x_0, x_r se numesc extremități, iar r reprezintă numărul de muchii al lanțului sau lungimea sa.

3. Graful G'=(V',E') se va numi subgraf al grafului G=(V,E) dacă V' ⊆ V , E' ⊆ E şi orice muchie din E'are aceleaşi extremităţi în G şi G'. [5]
4. Un ciclu într-un graf G este o succesiune finită de vârfuri ale lui G, notată $\left[x_0, x_1, ..., x_r \right]$, cu următoarele proprietăți:

a) $x_i x_{i+1} \in E(G)$ pentru orice $0 \le i \le r-1$;

b) $x_0 = x_r$;

c) Toate muchiile $x_0x_1, ..., x_{r-1}x_r$ sunt distincte.

5. Un graf G se numeşte conex dacă pentru orice pereche de vârfuri distincte x, y ale lui G, există un lanţ de extremităţi x şi y în G.
6. Un graf conex şi fără cicluri se numeşte arbore.

2.1. Problema arborelui parţial minim (algoritmul lui Kruskal)

Fie G un graf conex şi c o funcţie $c : E(G) \to (0, \infty)$ care asociază fiecărei muchii a grafului G un număr real pozitiv numit costul acelei muchii.

Costul unui graf parţial al lui G este egal prin definiţie cu suma costurilor asociate muchiilor lui H, ceea ce vom scrie: $c(H) = \sum_{u \in E(H)} c(u)$.

Un graf G pentru care a fost definită o funcție de cost se numește graf ponderat.
În aplicații, costul unei muchii poate fi:

- lungimea drumului dintre două localităţi;
- costul parcurgerii rutei reprezentate prin arcul corespunzător;
- durata parcurgerii rutei respective;
- cantitatea transportată pe ruta respectivă;
- capacitatea maximă a rutei respective.

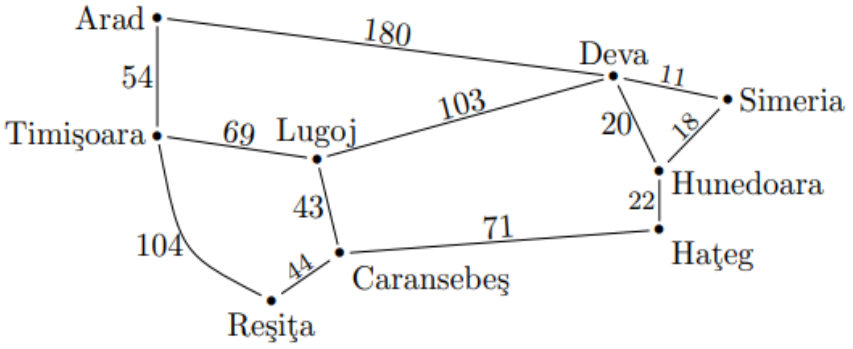


Fig. 5. Graful ponderat cu distanțele drumurilor dintre unele localități din România
Sursa: Marin, M. [6]

Algoritmul lui Joseph Kruskal, elaborat în 1956, permite determinarea unui arbore parțial de cost minim (APM) într-un graf ponderat.

Fie G un graf conex cu n≥2 vârfuri şi o funcţie de cost $c: E(G) \to (0, \infty)$.
Muchiile unui arbore parţial de cost minim sunt selectate una câte una, după cum urmează:

Pasul 1: Dintre muchiile nealese ale lui E(G) se selectează o muchie de cost minim cu condiţia să nu formeze cicluri cu muchiile deja alese.

Pasul 2: Au fost selectate n-1 muchii? Dacă da, ne oprim. Muchiile selectate sunt muchiile unui arbore parţial minim al lui G. Dacă nu, se repetă pasul 1.

2.2. Drumul de cost minim (algoritmul lui Dijkstra)

Algoritmul lui Edsger Dijkstra a fost inventat în anul 1959 pentru a determina drumul de cost minim de la un nod de start al unui graf la oricare altul.

Fie G un graf orientat cu n vârfuri 1, …, n, pentru care lungimea fiecărui arc (i,j) se notează cu $d_{ij} > 0$. Inițial, se fixează un vârf $t \in G$.

La sfârșitul algoritmului, l(i) va reprezenta distanța minima de la vârful t la vârful i, pentru orice $1 \le i \le n$.

Prin definiție, l(t)=0.

Dacă după aplicarea algoritmului pentru un vârf k se obține l(k)= ∞, înseamnă că nu există drumul de la vârful t la vârful k în graf.

Pașii algoritmului Dijkstra:

1. $T \leftarrow \varnothing$, $\overline{T} \leftarrow V(G)$,
2. $l(t) \leftarrow 0$ și $l(i) \leftarrow \infty,\ i \neq t$
3. Cât timp $|T| < n$ se execută:

a) Se selectează un vârf $i \in \overline{T}$ pentru care $l(i) = \min\{l(j) / \ j \in \overline{T}\}$

b) $T \leftarrow T \cup \{i\}$, $\overline{T} \leftarrow \overline{T} \setminus \{j\}$

c) Pentru fiecare arc (i,j) care are extremitatea inițială în i și extremitatea finală în $j \in \overline{T}$, $l(j) > l(i) + d_{ij}$, atunci $l(j) \leftarrow l(i) + d_{ij}$ și $pred(j) \leftarrow i$.

Unde:

- d este funcția distanță $d: E(G) \to (0, \infty)$ care asociază fiecărui arc din G lungimea sa $d(i, j) > 0$, notată cu d_{ij};
- pred reprezintă funcția predecesor, care pentru $(i, j) \in E, pred(j) = i$.

Algoritmul etichetează fiecare vârf i al grafului cu o etichetă notată l(i) care este un majorant al distanţei minime de la vârful t la vârful i , iar în final va fi egală chiar cu distanţa minimă de la t la i .

La fiecare pas intermediar algoritmul partiţionează vârfurile în două mulţimi: mulţimea T a vârfurilor cu etichete permanente și complementara sa $\overline{T}$, a vârfurilor cu etichete temporare. Etichetele permanente ale vârfurilor din T reprezintă distanţele minime de la t la vârfurile respective, în timp ce etichetele etichetele temporare ale vârfurilor din T reprezintă nişte majorări ale distanţelor minime.

Algoritmul poate fi definit și pentru grafuri neorientate înlocuind arcul (i,j) cu muchia ij, cu o extremitate în vârful i selectat la acel pas și cealaltă extremitatea într-un vârf $j \in \overline{T}$.

Se specifică în text semnificația parametrilor și a variabilelor, însă nu este furnizată o legendă în format text 9 pct separată pentru fiecare dintre acestea (se regăsește într-un singur loc).

De asemenea, în cazul ecuațiilor, nu s-a numerotat consecutiv cu cifre arabe între paranteze, în partea dreaptă a paginii.

3. Aplicații ale teoriei grafurilor în optimizarea și planificarea infrastructurii urbane
Teoria grafurilor oferă un cadru matematic puternic pentru analiza și proiectarea infrastructurii urbane. Aplicațiile sale în planificarea infrastructurii urbane sunt diverse și acoperă o gamă largă de domenii, contribuind la eficiența, sustenabilitatea și funcționarea optimă a orașelor inteligente.

Așa cum am putut vedea în capitolul anterior, algoritmul lui Kruskal este un algoritm utilizat pentru a găsi arborele parțial de cost minim pentru un graf dat. Algoritmul funcționează începând cu toate vârfurile din graf și le conectează pentru a forma un arbore. Arborele este apoi redus prin eliminarea muchiilor care nu fac parte din arborele de cost minim.

Dintre aplicațiile practice ale algoritmului lui Kruskal enumerăm [5]:

- proiectarea rețelelor feroviare și rutiere (algoritmul lui Kruskal este folosit în proiectarea rețelelor feroviare și rutiere pentru a conecta mai multe orașe. Scopul este să se stabilească rutele de transport cele mai eficiente și economice între orașe, iar algoritmul lui Kruskal ajută la identificarea arborelui partial de cost minim, asigurând o conectivitate optimă și minimizând costul total);
- proiectarea canalelor de irigații în agricultură (în ingineria agricolă, algoritmul lui Kruskal poate fi folosit pentru proiectarea canalelor de irigații. Algoritmul ajută la identificarea rețelei de canale care acoperă eficient zona agricolă, asigurând o distribuție optimă a apei pentru irigarea culturilor. Această aplicație contribuie la optimizarea resurselor și la eficiența costurilor);
- proiectarea rețelelor de fibră optică (ajută la stabilirea conexiunilor cele mai eficiente între diferite puncte din rețea, asigurând că datele pot fi transmise eficient

prin rețeaua de fibră optică. Acest aspect este crucial pentru optimizarea performanței sistemelor de comunicații);
- optimizarea rețelelor de distribuție a apei.

În lucrarea "Applied Graph Theory to Real Smart City Logistic Problems", Gutierrez et al ne relevă faptul că „ una dintre cele mai convenabile modalități de lucru cu trasee rutiere și rutare este tratarea rețelei de drumuri ca pe un graf" și de asemenea că „ teoria grafurilor este unul dintre pilonii pentru soluții inovatoare în Logistica Inteligentă…". [7]

Autorii aplică algoritmul lui Kruskal în calcularea rutelor de conducere în regiunea de Nord Jutland, Danemarca, acoperind 100% din drumurile din zonă.

Algoritmul descris de autori în lucrarea menționată cuprinde următorii pași:
1. Calculul arborelui de parțial de cost minim tradițional, având ca rădăcină cel mai apropiat vârf față de un garaj unde sunt parcate mașinile
2. Modificarea arborelui de parțial de cost minim pentru a include și muchiile care nu aparțin arborelui de parțial de cost minim inițial, numite muchii fantomă și vârfuri
3. Aplicarea algoritmului de navigare în graf peste arborele de parțial de cost minim modificat pentru a sorta vârfurile în ordinea vizitării și a crea astfel ruta (ca o secvență de puncte de interes).

Pe de altă parte, algoritmul lui Dijkstra de determinare a drumului de cost minim într-un graf are mai multe aplicații practice, dintre care [8]:
- Google Maps: găsirea distanței în Google Maps de la un oraș la altul sau de la locația curentă la cea mai apropiată locație dorită. Algoritmul lui Dijkstra este folosit pentru a găsi distanța minimă între două locații de-a lungul traseului.
- Aplicații în rețele de socializare: în mai multe rețele de socializare, aplicația sugerează lista de prieteni pe care un anumit utilizator i-ar putea cunoaște. Algoritmul standard Dijkstra poate fi aplicat folosind calea cea mai scurtă între utilizatori măsurată prin conexiunile între ei.
- Open Shortest Path First (OSPF): Open Shortest Path First (OSPF) este un protocol de rutare bazat pe stare de legătură folosit pentru a găsi cea mai bună cale între routerul sursă și destinație. Algoritmul oferă calea cu cel mai mic cost de la routerul sursă la alte routere din rețea.

În contextul optimizării planificării rețelelor de infrastructură urbană într-un oraș inteligent, algoritmul lui Dijkstra poate fi utilizat pentru a rezolva probleme precum:
- Gestionarea traficului prin optimizarea semafoarelor și a rutelor pentru a minimiza congestia și a îmbunătăți fluxul de trafic în timp real;
- Planificarea transportului public prin identificarea traseelor optime pentru autobuze, tramvaie sau metrouri pentru a asigura acoperirea eficientă a orașului și pentru a reduce timpii de așteptare pentru călători;
- Optimizarea rețelelor de apă și canalizare prin identificarea căilor optime pentru conductele de apă și canalizare pentru a asigura distribuția eficientă a resurselor de apă;

- Rutarea inteligentă a vehiculelor electrice prin identificarea optimă a locurilor de încărcare pentru a contribui astfel la încurajarea utilizării transportului electric.

Algoritmul lui Dijkstra este folosit de mai mulți autori în identificarea în găsirea rutei optime în traficul urban.

În lucrarea, "Pathfinding through urban traffic using Dijkstra's Algorithm", autoarea Tubagus Andhika Nugraha descoperă rezultate plauzibile în identificarea unei rute optime de transport pentru locuitorii din Jakarta folosind algoritmul lui Dijkstra. [9]

Autoarea pleacă în cercetarea sa de la harta orașului și transformă elementele hărții în obiecte discrete astfel încât harta să poată fi reprezentată sub forma unui graf. Ulterior adaugă ponderi grafului reprezentat de harta orașului, prin colectarea de date cu privire la distanța drumurilor, precum și datele despre trafic, măsurate prin viteza medie a mașinilor. Concluzia la care ajunge autoarea la finalul cercetării este că Dijkstra este cel mai eficient algoritm pentru găsirea celor mai scurte drumuri în grafuri cu ponderi non-negative.

O altă lucrare, "Implementation and Analysis of Dijkstra's Shortest Path Algorithm in the Context of Romanian Cities", ne prezintă o problemă de identificare a celui mai scurt si eficient drum dintre Arad și București. Algoritmul examinează orașele și căile lor de conectare, actualizând distanțele cele mai scurte către orașele învecinate pe măsură ce avansează.

Astfel, de la început, "Arad" are o distanță de 0, iar toate celelalte orașe au o distanță setată la infinit (inf), simbolizând că încă nu știm distanța cea mai scurtă pentru a ajunge la ele.

Algoritmul lui Dijkstra acționează sistematic astfel:
1. Alege un oraș.
2. Verifică orașele învecinate.
3. Actualizează distanțele lor dacă găsește o cale mai scurtă.
4. Se mută la următorul oraș și repetă procesul. [10]

Algoritmul lui Kruskal și algoritmul lui Dijkstra sunt complementare, în timp ce algoritmul lui Kruskal se concentrează pe găsirea unui arbore de parțial de cost minim într-un graf neorientat ponderat, algoritmul lui Dijkstra se concentrează pe găsirea celei mai scurte căi între un nod de start și toate celelalte noduri într-un graf orientat ponderat.

Integrarea ambilor algoritmi în optimizarea rețelelor de infrastructură urbană poate aduce contribuții semnificative în ceea ce privește gestionarea resurselor și îmbunătățirea conectivității într-un oraș inteligent.

De asemenea, în contextul Internet of Things (IoT), algoritmii precum Kruskal și Dijkstra pot avea diverse aplicații în optimizare, precum:
- rutare eficientă a datelor în rețele IoT: algoritmul lui Dijkstra poate fi utilizat pentru a găsi cea mai scurtă cale între dispozitivele IoT, facilitând transmiterea eficientă a datelor între acestea;

- ghidarea dispozitivelor mobile: algoritmul lui Dijkstra poate ghida aceste dispozitive pe rutele optime pentru a îndeplini sarcinile sau pentru a evita obstacole;
- gestionarea consumului de energie: algoritmul Dijkstra poate fi utilizat pentru a găsi rute care să minimizeze consumul de energie, prelungind durata de viață a dispozitivelor;
- conectivitatea senzorilor în rețele IoT: algoritmul lui Kruskal poate fi aplicat pentru a conecta senzorii sau dispozitivele IoT într-o rețea eficientă, asigurându-se că acestea sunt conectate într-un mod care optimizează comunicarea și acoperirea zonelor;
- optimizarea structurii rețelei: algoritmul Kruskal poate contribui la optimizarea fizică a rețelelor IoT, conectând dispozitivele într-un mod care minimizează costurile de instalare și întreținere;

4. Concluzii

Așa cum am putut vedea, teoria grafurilor a evoluat de-a lungul timpului de la încercarea de a soluționa probleme precum "problema podurilor din Königsberg" către oferirea de soluții de optimizare pentru diferite probleme de infrastructură urbană în proiectarea arhitecturii unui oraș inteligent.

Un oraș inteligent nu este doar un oraș în care tehnologia este încorporată și folosită la cel mai înalt nivel ci și un oraș în care interoconectivitatea contează în gestionarea resurselor și accesul facil la toate serviciiile oferite.

Pentru a transforma o rețea urbană într-o rețea "inteligentă" înseamnă gestionarea unei sinergii complexe a mai multor rețele, dispozitive, semafoare și contoare inteligente de apă și energie, împreună cu oameni.

Utilizarea algoritmilor de optimizare din teoria grafurilor poate aduce contribuții semnificative în ceea ce privește gestionarea resurselor și îmbunătățirea conectivității într-un oraș inteligent.

References

[1] I. Tomescu, "Combinatorică și teoria grafurilor," *Tipografia Universității din București,* p. 83, 1978.

[2] R. Scînteie, "Baze de date și algoritmi pentru căi de comunicație," *Editura Societății Academice "Mateiu-Teiu Botez",* p. 168, 2003.

[3] I. Alnassar, "Applications of Graph Theory for Controlling City Infrastructure," *https://opensiuc.lib.siu.edu/cgi/viewcontent.cgi?article=2276&context=gs_rp,* 13.11.2023.

[4] V. T. Nica, "Introducere în programarea în numere întregi. Introducere în optimizarea combinatorială," *Editura ASE,* p. 50, 2011.

[5] "Curs Mircea Marin," https://staff.fmi.uvt.ro/~mircea.marin/lectures/TGC/Curs-08-extra.pdf.

[6] "https://www.mygreatlearning.com/blog/kruskals-algorithm/".

[7] J. M. Gutierrez, M. Jensen and T. Riaz, "Applied Graph Theory to Real Smart City Logistic Problems," *https://www.sciencedirect.com/science/article/pii/S1877050916324632* .

[8] “https://www.analyticssteps.com/blogs/how-dijkstras-algorithm-used-real-world”.

[9] T. A. Nugraha, “Pathfinding through urban traffic using Dijkstra’s Algorithm,” *https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2011-2012/Makalah2011/Makalah-IF2091-2011-040.pdf.*

[10] M. Kanchan, “Implementation and Analysis of Dijkstra's Shortest Path Algorithm in the Context of Romanian Cities,” *https://www.researchgate.net/publication/374384461_Implementation_and_Analysis_of_Dijkstra%27s_Shortest_Path_Algorithm_in_the_Context_of_Romanian_Cities.*