Universitatea
Transilvania
din Brașov

Universitatea
Transilvania
din Brașov
FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

ȘCOALA DOCTORALĂ INTERDISCIPLINARĂ

Facultatea: Matematică și Informatică

Luciana CĂRĂBĂNEANU
(căsătorită MAJERCSIK)

METODE INCREMENTALE ÎN REȚELE DINAMICE/ INCREMENTAL METHODS IN DYNAMIC NETWORKS

REZUMAT

Conducător ştiințific

Prof. dr. Marius-Sabin TĂBÎRCĂ

Brașov, 2024

Mulțumiri

Aș dori să-mi exprim cea mai profundă recunoștință față de toți cei care m-au susținut pe parcursul cercetării și finalizării acestei teze.

În primul rând, sunt profund recunoscătoare conducatorului meu de doctorat, Prof. Dr. Sabin Tabirca, pentru îndrumarea, încurajarea și sprijinul său neprețuit. Cunoștințele sale, feedback-ul și ideile sale au fost fundamentale în conturarea acestei teze, iar răbdarea și înțelegerea sa au fost o sursă de motivație pe parcursul întregii mele perioade de doctorat.

De asemenea, aș dori să-mi exprim recunoștința sinceră față de membrii comisiei mele de îndrumare a tezei, Dr. Laura Ciupala, Dr. Eleonor Ciurea și Dr. Dorin Bocu, pentru criticile constructive și sugestiile lor, care au îmbunătățit considerabil calitatea acestei teze.

Sunt, de asemenea, recunoscătoare colegilor mei din departament și din grupul de cercetare, în special domnului Conf. Dr. Adrian Deaconu, care au oferit un mediu stimulant și de susținere. Discuțiile, colaborările și experiențele împărtășite au fost o parte integrantă a călătoriei mele academice.

Nu în cele din urmă, dedic această teză familiei și prietenilor mei, a căror credință neclintită în abilitățile mele mi-a oferit forța și motivația necesare pentru a finaliza această lucrare. Sprijinul lor constant, dragostea și încurajarea au fost neprețuite și le sunt etern recunoscătoare pentru prezența lor în viața mea.

Mi-e teamă că nu am cuvinte pentru a exprima recunoștința mea față de cei care au avut încredere în mine și m-au încurajat. Sprijinul vostru neclintit, răbdarea și dragostea voastră, înseamnă pentru mine mai mult decât pot exprima.

Vă mulțumesc tuturor pentru tot.

"Every act of conscious learning requires the willingness to suffer an injury to one's self-esteem. That is why young children, before they are aware of their own self-importance, learn so easily." — Thomas Szasz

Capitolul 1

Introducere

O serie de industrii au fost transformate de creșterea rapidă a datelor interconectate în ultimele decenii. Multe dintre cele mai mari companii, inclusiv Amazon, Google, Meta, Instagram, LinkedIn, Pinterest și Airbnb, au plasat datele și gestionarea acestora în centrul strategiei lor operaționale. Aceste date, care sunt adesea reprezentate ca un graf sau o rețea, sunt vaste și dinamice și necesită strategii eficiente de gestionare.

Rețelele dinamice sunt supuse modificărilor care apar în diverse moduri, inclusiv modificări ale arcelor sau nodurilor individuale sau modificări mai extinse care afectează mai multe părți ale rețelei simultan. De exemplu, ponderea unui arc individual poate fi crește sau descrește, un nod poate fi adăugat sau eliminat, sau mai multe conexiuni pot fi modificate din cauza unor circumstanțe. Gestionarea eficientă a acestor schimbări este importantă pentru a permite rețelei să se adapteze și să mențină performanța optimă.

Pentru a face față provocărilor ridicate de datele care se modifică în mod dinamic, este esențial să avem algoritmi care să poată gestiona eficient schimbările în structura rețelei. Acești algoritmi sunt proiectați să actualizeze doar acele părți ale rețelei care sunt afectate, evitând astfel povara computațională de a recalcula soluțiile de la început. Concentrându-se pe schimbările incrementale, acești algoritmi îmbunătățesc semnificativ capacitatea de a gestiona eficient rețelele dinamice. Acest lucru asigură că sistemele rămân atât performante, cât și adaptabile. Astfel de algoritmi sunt esențiali într-o gamă largă de aplicații, inclusiv optimizarea fluxurilor prin rețea, menținerea celor mai scurte drumuri, ajustarea dinamică a configurațiilor nodurilor prin algoritmi de clustering și identificarea modelelor neobișnuite în timp real prin algoritmi de detectare a anomaliilor. Ei permit o adaptare rapidă la schimbări, asigurându-se că sistemele rămân eficiente și adaptabile la schimbările din mediu.

1.1 Motivație

În lumea interconectată de astăzi, rețelele la scară largă sunt peste tot, de la platformele de social media, la sistemele de transport, infrastructurile de telecomunicații și multe altele. Dimensiunea și natura lor dinamică necesită algoritmi scalabili care să poată gestiona eficient și optimiza funcționarea lor. Pe măsură ce aceste rețele continuă să crească și să evolueze, abilitatea de a se adapta rapid la schimbări devine din ce în ce mai importantă pentru a menține performanța optimă.

Două tipuri de algoritmi apar adesea interconectați în diverse aspecte ale gestionării rețelelor dinamice, și anume algoritmii pentru drumuri minime și algoritmii pentru flux maxim. Relația dintre aceștia este remarcabilă, deși cele doua categorii de algoritmi se concentrează pe obiective diferite de optimizare. Algoritmii pentru determinarea fluxului maxim optimizează utilizarea generală a rețelei, asigurând fluxul maxim posibil de la o sursă la un destinație, în timp ce algoritmii pentru drumurile minime vizează minimizarea costului total sau a distanței dintre noduri. Având în vedere aplicabilitatea largă și natura interconectată a acestor probleme, dezvoltarea algoritmilor eficienți care pot gestiona schimbare dinamică a condițiilor din rețea este esențială.

Schimbările incrementale în rețea, cum ar fi ajustările ponderilor arcelor sau adăugările și eliminările de noduri, au un efect considerabil asupra calculului drumurilor minime și a fluxurilor maxime. Abordarea acestor schimbări incremental, în loc de recalcularea soluțiilor de la început, este esențială pentru menținerea performanței optime. Alegerea de a ne concentra pe algoritmii pentru fluxul maxim și drumurile minime de la o singură sursă (SSSP) este motivată de importanța lor fundamentală în diverse aplicații din lumea reală și capacitatea lor de a aborda provocările critice în optimizarea și gestionarea rețelelor. Interacțiunea dintre acești algoritmi evidențiază rolurile lor complementare. Explorând și avansând acești algoritmi, putem îmbunătăți capacitatea rețelelor dinamice de mare amploare de a funcționa eficient și de a rezista sau de a se recupera rapid după schimbări sau perturbări. Aceasta implică asigurarea faptului că rețelele pot gestiona fluctuațiile cererii, se pot adapta la noile condiții și pot continua să funcționeze fără degradări semnificative ale performanței.

1.2 Obiective

Principalele obiective ale acestei teze sunt:

- **Analiza algoritmilor existenți**. Primul obiectiv este realizarea unei investigații detaliate a celor mai performanți algoritmi actuali pentru problemele de flux incremental și drumuri minime de la o singură sursă (SSSP) în contexte dinamice. Aceasta va include o analiză detaliată a fundamentelor lor teoretice, a complexității computaționale și a strategiilor utilizate pentru a face față schimbărilor dinamice ale rețelei. Identificarea punctelor forte și slabe ale acestor algoritmi va evidenția limitările și posibilele îmbunătățiri. În plus, înțelegerea aplicațiilor lor practice va oferi perspective asupra performanței algoritmilor în diverse scenarii din viața reală și va sublinia cazurile în care aceștia sunt deosebit de eficienți.

- **Dezvoltarea algoritmilor dinamici eficienți**. Al doilea obiectiv este propunerea unor noi algoritmi capabili să gestioneze eficient schimbările dinamice ale condițiilor sau cerințelor rețelei, bazându-se pe informațiile obținute din studiul algoritmilor existenți. Aceste noi metode vor fi axate pe minimizarea complexității computaționale prin actualizarea doar a părților afectate ale rețelei, asigurându-se astfel performanța optimă. Scopul este de a dezvolta algoritmi care să fie nu doar performanți din punct de vedere teoretic, ci și practic, capabili să gestioneze rețele mari menținând în același timp performanța și scalabilitatea ridicate.

- **Aplicabilitatea la rezolvarea unor probleme reale**. Un alt obiectiv al acestei teze este examinarea potențialului de integrare a datelor de teledetecție cu rețele dinamice în scenarii în care se dorește studierea fenomenelor care produc schimbări vizibile pe suprafața Pământului în timp. Acest obiectiv subliniază relevanța și aplicabilitatea cercetării la probleme din viața reală, evidențiind beneficiile practice și impactul potențial al soluțiilor propuse.

1.3 Structura Tezei

Prezenta teză este structurată în următoarele capitole:

Capitolul 2: Fundamente Teoretice

Acest capitol oferă conceptele de bază și fundamentele teoretice necesare pentru a înțelege algoritmii dinamici și incrementali legați de problema drumurilor minime și problema fluxului maxim. Acesta acoperă bazele problemelor de flux în rețea, algoritmii pentru drumurile minime și tehnicile computaționale relevante.

Capitolul 3: Revizuirea Literaturii

Acest capitol revizuiește algoritmii actuali pentru problemele dinamice de tip SSSP și flux incremental. Include o discuție detaliată a celor mai cunoscute soluții, metodologiile pe care le folosesc și aplicațiile în care sunt utilizate.

Capitolul 4: Ajustarea Dinamică a Drumurilor Minime de la o Singură Sursă

Acest capitol introduce un nou algoritm SSSP dinamic. Descrie modul în care acesta gestionează schimbările în ponderile arcelor și menține eficient arborele drumurilor minime. Folosește o metodologie care minimizează complexitatea computațională și asigură faptul că actualizările sunt efectuate rapid. Corectitudinea și complexitatea algoritmului sunt investigate amănunțit, oferind o analiză detaliată a fundamentelor sale teoretice. În plus, sunt prezentate mai multe exemple pentru a ilustra aplicarea practică și performanța algoritmului, arătând modul în care acesta se adaptează la schimbările din rețea, efectuând în același timp calculul precis al drumului minim.

Capitolul 5: Fluxul Maxim prin Extinderea Rețelei

Acest capitol examinează aspectele practice și teoretice ale problemei fluxului maxim, cu un accent pe extinderea rețelei. Adesea, extinderea rețelei este necesară pentru a crește cantitatea de flux transportată printr-o rețea. Pornind de la o rețea dată G, obiectivul problemei de extindere a rețelei cu cost minim (MCNEP) studiată în acest capitol este obținerea unei noi rețele G' prin creșterea capacităților arcelor în limite date sau prin adăugarea de noi arce. Scopul este de a minimiza costul schimbărilor, permițând în același timp rețelei G' să transporte w unități de flux de la sursă la destinație.

Capitolul 6: Integrarea Datelor de Teledetecție cu Procesarea Rețelelor Dinamice

Obiectivul acestui capitol este introducerea unei metode de integrare a datelor de teledetecție cu rețelele dinamice și utilizarea lor în combinație într-un algoritm dinamic de monitorizare a vegetației. Abordarea propusă poate fi utilizată în agricultură sau monitorizarea mediului, potențial în combinație cu măsurători sau dispozitive in situ , pentru a permite intervenții la timp în situații nefavorabile, datorate fenomenelor naturale sau intervențiilor umane.

Capitolul 7: Concluzii

Capitolul final rezumă constatările tezei și reflectă asupra progresului realizat. Discută posibile direcții viitoare de cercetare, subliniind importanța dezvoltării unor algoritmi robuști, care să fie performanți într-o varietate de aplicații și scenarii dinamice și care să fie scalabili pentru utilizarea în rețele mari.

Capitolul 2

Fundamente Teoretice

2.1 Introducere

Domeniul optimizării rețelelor reprezintă o arie importantă în cercetarea operațională și informatică, concentrându-se pe analiza și optimizarea fluxului de resurse, date sau entități prin rețele reprezentate ca grafuri. Acest domeniu de studiu abordează o serie de probleme importante, inclusiv determinarea cantității maxime de flux care poate fi transmisă printr-o rețea (problema fluxului maxim) și identificarea drumurilor minime între noduri (problema drumurilor minime). Semnificația acestor probleme constă în aplicațiile lor practice extinse în diverse situații reale.

Relația dintre algoritmii pentru fluxul maxim și cei pentru cele mai scurte căi este profundă, în ciuda faptului că se concentrează pe obiective distincte de optimizare. Algoritmii pentru fluxul maxim prioritizează optimizarea utilizării capacității totale a rețelei, în timp ce algoritmii pentru cele mai scurte căi urmăresc identificarea celui mai eficient traseu sau a celei mai eficiente căi între două noduri, cu obiectivul de a minimiza costul total sau distanța parcursă. Similarități computaționale există între cele două în sensul că algoritmii pentru cele mai scurte căi pot fi relaționați cu algoritmii pentru fluxul maxim. Aceasta include metodele de găsire a căii de augmentare, utilizate în algoritmii Ford-Fulkerson și Edmonds-Karp. Mai mult, în aplicațiile practice, soluțiile acestor probleme se completează adesea reciproc. Următoarele exemple ilustrează acest lucru.

În secțiunea următoare, ne vom concentra asupra definițiilor specifice, terminologiei și tipurilor de probleme asociate cu problema drumurilor minime.

2.2 Problema drumului minim

Definițiile și terminologiile prezentate aici se bazează pe [1].

Definiție 1. *Un graf orientat $G = (V, A)$ constă într-un set V de vârfuri (noduri) și un set $A = \{(i, j) \mid i \in V, j \in V\}$ de perechi ordonate de noduri distincte. O rețea orientată sau un graf ponderat orientat este un graf orientat în care fiecare arc are o valoare numerică asociată (cost sau capacitate) determinată de funcția*

$$w : A \to \mathbb{R} \tag{2.1}$$

cunoscută sub denumirea de funcție pondere. În acest caz, graful $G = (V, A, w)$ mai este denumit și graf ponderat.

Problema drumului minim

Într-un graf orientat ponderat $G = (V, A, w)$, problema drumului minim se referă la identificarea drumului minim între noduri pe baza ponderilor arcelor. În particular, problema drumului minim urmărește determinarea unei căi în care suma ponderilor arcurilor (sau costurilor) este minimizată, dat fiind un nod sursă s și un nod țintă t. În abordarea problemei drumului minim, se fac următoarele presupuneri:

1. Graful este orientat.
2. Graful nu conține cicluri negative, adică nu există cicluri orientate cu greutăți totale negative.
3. Există o cale orientată de la un nod sursă specificat, desemnat s, la fiecare alt nod din graf.

De-a lungul anilor, cercetătorii au explorat o serie de tipuri distincte de probleme ale drumului minim în teoria grafurilor, inclusiv:

- **Problema drumului minim de la o singură sursă (SSSP)**, care urmărește identificarea drumurilor minime de la un singur nod sursă la toate celelalte noduri din rețea.
- **Problema drumului minim între două noduri specifice (SPSP)**, unde obiectivul este determinarea drumului minim între două noduri specifice din rețea.

- **Problema drumului minim între toate perechile de noduri (APSP)**, care se referă la calcularea drumului minim între fiecare pereche de noduri din graf.

Există, de asemenea, o serie de versiuni specializate ale problemei drumului minim: problema drumului cu capacitate maximă, problema drumului cu fiabilitate maximă, celui mai scurt drum cu constrângeri suplimentare sau problema drumului minim cu resurse limitate. Fiecare dintre aceste variații abordează o provocare sau o aplicație specifică.

2.3 Problema Fluxului Maxim

Această secțiune va introduce conceptele fundamentale de flux maxim, tăietură minimă și flux cu cost minim, care vor fi importante pentru înțelegerea tehnicilor și algoritmilor discutați în capitolele următoare. Introduce, de asemenea, conceptele cheie necesare pentru a înțelege cum funcționează algoritmii de flux în rețea. Definițiile și teoremele prezentate aici se bazează pe [1].

Fie $G = (V, A, s, t, u)$ o rețea orientată $s - t$, unde V este setul de $n > 0$ vârfuri (noduri) și $A \subseteq V \times V$ este setul de $m \geq 0$ arcuri (muchii orientate). Fiecare arc $a = (i, j) \in A$ leagă două noduri i și j din V, s este un nod special numit sursă, iar t este un nod numit scurgere. În G, definim funcția capacitate $u : A \to \mathbb{R}_+^*$. Valoarea $u(a)$ este fluxul maxim care poate fi transportat de la nodul i la nodul j pe arcul $a = (i, j) \in A$.

Definiție 2. *Un (flux admisibil) într-o rețea orientată $s - t$ G este o funcție $f : A \to \mathbb{R}_+$ care satisface restricțiile de capacitate (2.2) și condițiile de conservare din (2.3).*

$$0 \leq f(a) \leq u(a), \forall a = (i, j) \in A \tag{2.2}$$

$$\sum_{\substack{j \in V, \\ (i,j) \in A}} f(i, j) - \sum_{\substack{j \in V, \\ (j,i) \in A}} f(j, i) = \begin{cases} 0, \textit{dacă } i \in V - \{s, t\} \\ v_f, \textit{dacă } i = s \\ -v_f, \textit{dacă } i = t \end{cases} \tag{2.3}$$

unde $v_f = v(f) \geq 0$ este numită valoarea fluxului f.

Definiție 3. *Un flux admisibil f^* este un flux maxim dacă are valoarea maximă dintre toate fluxurile admisibile din rețeaua G, adică:*

$$v(f^*) = max\{v(f) \mid f \textit{ este un flux admisibil în } G\} \tag{2.4}$$

În (2.4), $v(f^*)$ reprezintă cantitatea maximă de flux care poate fi transportată în rețeaua G, de la s la t.

Conceptul de rețea reziduală a jucat un rol cheie în dezvoltarea tuturor algoritmilor pentru fluxul maxim pe care îi vom discuta. Prin urmare, vom continua prin a defini rețeaua reziduală astfel:

Definiție 4. *Dat un flux admisibil f, capacitatea reziduală a oricărui arc $(i,j) \in A$ reprezintă fluxul suplimentar maxim care poate fi trimis de la nodul i la nodul j folosind arcurile (i,j) și (j,i). Capacitatea reziduală a arcului (i,j) are două componente:*

1. *$u(i,j) - f(i,j)$ care reprezintă capacitatea neutilizată a arcului $a = (i,j) \in A$;*
2. *$f(j,i)$ care reprezintă fluxul pe arcul (j,i) care poate fi redus pentru a crește fluxul de la nodul i la nodul j.*

Prin urmare, capacitatea reziduală se calculează astfel:

$$r(i,j) = u(i,j) - f(i,j) + f(j,i). \tag{2.5}$$

și rețeaua reziduală în raport cu fluxul f, $\widetilde{G}(f) = (V, \widetilde{A}, s, t, r)$ constă doar din arcuri cu capacități reziduale pozitive, adică $\widetilde{A} = \{(i,j) \in A \mid r(i,j) > 0\}$.

Definiție 5. *Într-o rețea reziduală $\widetilde{G}(f) = (V, \widetilde{A}, s, t, r)$, o funcție $d : V \to \mathbb{Z}_+$ este numită funcție de distanță. Funcția de distanță d este considerată validă dacă satisface următoarele două condiții:*

$$d(t) = 0 \tag{2.6}$$

$$d(i) \leq d(j) + 1, \text{ pentru orice } (i,j) \in \widetilde{A}. \tag{2.7}$$

Valoarea $d(i)$ este numită *eticheta de distanță* a nodului i, iar condițiile 2.6 și 2.7 sunt numite *condiții de validitate*.

Este cunoscut faptul că dacă etichetele de distanță sunt valide, atunci eticheta de distanță $d(i)$ reprezintă o limită inferioară pentru lungimea drumului minim de la nodul i la nodul t în rețeaua reziduală $\widetilde{G}(f)$.

Definiție 6. *Etichetele de distanță sunt considerate exacte dacă, pentru fiecare nod i, $d(i)$ este egală cu lungimea drumului minim de la nodul i la nodul t în rețeaua reziduală $\widetilde{G}(f)$.*

Definiție 7. *Un arc* (i, j) *în rețeaua reziduală* $\widetilde{G}(f)$ *este numit admisibil dacă satisface condiția*

$$d(i) = d(j) + 1 \tag{2.8}$$

altfel, arcul (i, j) *este numit inadmisibil.*

O cale de la nodul sursă la nodul de scurgere în rețeaua reziduală $\widetilde{G}(f)$ este numită *admisibilă* dacă conține doar arcuri admisibile; altfel este numită *inadmisibilă*.

Definiție 8. *Un preflux într-o rețea orientată* $s - t$ G *este o funcție* $f : A \to \mathbb{R}_+$ *care satisface restricțiile de limită (2.2) și o relaxare a condițiilor de conservare din (2.3).*

$$e(i) = \sum_{\substack{j \in V, \\ (j,i) \in A}} f(j, i) - \sum_{\substack{j \in V, \\ (i,j) \in A}} f(i, j) \geq 0, pentru orice i \in V - \{s, t\} \tag{2.9}$$

Pentru o funcție de preflux dată f, diferența dintre fluxul de intrare și fluxul de ieșire, $e(i)$ este numită *excesul* nodului $i \in V - \{s, t\}$. Dacă un nod are un exces pozitiv, atunci este numit nod *activ*.

Definiție 9. *Un pseudoflux într-o rețea orientată* $s - t$ G *este o funcție* $f : A \to \mathbb{R}_+$ *care satisface doar restricțiile de limită (2.2).*

Pentru orice pseudoflux, f definim *desechilibrul* nodului i astfel:

$$e(i) = \sum_{\substack{j \in V, \\ (j,i) \in A}} f(j, i) - \sum_{\substack{j \in V, \\ (i,j) \in A}} f(i, j), pentru orice i \in V \tag{2.10}$$

Spunem că i este un *nod de exces* dacă $e(i) > 0$ și un *nod de deficit* dacă $e(i) < 0$. Dacă $e(i) = 0$, nodul i este *echilibrat*. În consecință, un preflux este un caz particular de pseudoflux.

Pentru a înțelege mai bine conceptele de flux în rețea, introducem conceptul de tăietură s–t și problema fluxului cu cost minim.

Definiție 10. *O tăietură s–t într-o rețea* $G = (V, A, s, t, u)$ *este o partiție a* V *în două subseturi disjuncte* S *și* T *astfel încât* $s \in S$ *și* $t \in T$*. Capacitatea tăierii s–t, denumită* $c(S, T)$*, este suma capacităților arcurilor de la* S *la* T*:*

$$c(S, T) = \sum_{(i,j) \in S \times T} u(i, j). \tag{2.11}$$

Teorema Fluxului Maxim și Tăierii Minime afirmă că valoarea fluxului maxim într-o rețea este egală cu capacitatea tăierii minime s–t din rețea. Această teoremă stabilește o relație fundamentală între fluxuri și tăieri în teoria rețelelor.

Teorema 1 (Teorema Fluxului Maxim și Tăierii Minime [1])**.** *În orice rețea de flux, valoarea maximă a fluxului este egală cu capacitatea tăierii minime care separă sursa de scurgere.*

În timp ce problema fluxului maxim se referă la identificarea fluxului admisibil maxim într-o rețea, problema fluxului cu cost minim se referă la stabilirea modalității celei mai puțin costisitoare pentru transmiterea unui volum dat de flux printr-o rețea. Problema fluxului cu cost minim poate fi considerată o extensie a problemei fluxului maxim, unde fiecare arc este asociat cu un factor de cost care reflectă costul trimiterii fluxului prin acesta și este definită astfel:

Definiție 11. *Problema Fluxului cu Cost Minim într-o rețea orientată* $G = (V, A)$ *cu o funcție de capacitate* $u : A \to \mathbb{R}_+^*$ *și o funcție de cost* $c : A \to \mathbb{R}_+$ *caută să găsească un flux* $u : A \to \mathbb{R}_+$ *care să minimizeze costul total:*

$$\min \sum_{(i,j)\in A} c(i,j)f(i,j) \tag{2.12}$$

subiect la:

$$0 \leq f(i,j) \leq u(i,j), pentruorice(i,j) \in A \tag{2.13}$$

și unde:

$$\sum_{j\in V,(i,j)\in A} f(i,j) - \sum_{j\in V,(j,i)\in A} f(j,i) = b(i), pentruoricei \in V \tag{2.14}$$

cu $b(i)$ *fiind oferta/cererea la nodul* i*. Dacă* $b(i) > 0$*, nodul* i *este un nod de ofertă, iar dacă* $b(i) < 0$*, nodul* i *este un nod de cerere.*

În general, se fac următoarele presupuneri:

1. Toate datele sunt întregi.
2. Rețeaua este orientată.
3. Suma tuturor exceselor (ofertele) și deficitelor (cererile) este zero:

$$\sum_{i\in V} b(i) = 0 \tag{2.15}$$

4. Toate costurile arcelor sunt pozitive.

În concluzie, acest capitol a prezentat principalele definiții și terminologii asociate cu problema drumului minim și problema fluxului maxim. Acest lucru a stabilit o bază pentru înțelegerea metodologiilor computaționale utilizate pentru a aborda aceste probleme.

Algoritmii statici eficienți pentru problemele fluxului maxim și drumurilor minime au fost un subiect de cercetare de ani de zile. Dezvoltarea algoritmilor incrementali și dinamici pentru rețele provine din recunoașterea limitărilor algoritmilor statici în abordarea scenariilor reale unde caracteristicile rețelei sau cerințele fluxului se schimbă în timp. Deși algoritmii statici oferă tehnici computaționale valoroase pentru problemele fluxului maxim și drumurilor minime, ei operează sub presupunerea structurilor și capacităților rețelei fixe. Cu toate acestea, în medii dinamice, cum ar fi sistemele de transport, rețelele de comunicații sau lanțurile de aprovizionare, aceste presupuneri frecvent nu sunt valabile. Prin urmare, algoritmii incrementali și dinamici sunt necesari pentru a acomoda natura dinamică a acestor sisteme. Algoritmii incrementali și dinamici sunt două tipuri de algoritmi utilizați pentru a actualiza soluțiile în răspuns la schimbările din rețea. Algoritmii incrementali actualizează eficient soluțiile în răspuns la schimbări mici în rețea, cum ar fi adăugarea de muchii sau ajustările capacităților/ponderilor. Algoritmii dinamici, pe de altă parte, se pot adapta la variații cum ar fi inserțiile de muchii, eliminările sau ajustările capacităților, asigurându-se că soluțiile rămân eficiente pe măsură ce rețeaua evoluează. Algoritmii dinamici urmăresc să mențină soluții optime sau aproape optime în ciuda schimbărilor în topologia rețelei. Prin incorporarea dinamicilor temporale în problema de optimizare, algoritmii incrementali și dinamici oferă flexibilitate și adaptabilitate, permițându-le să răspundă eficient la condițiile de rețea în schimbare. Această adaptabilitate este crucială pentru aplicații cum ar fi gestionarea traficului în timp real și alocarea dinamică a resurselor în medii în evoluție. Prin urmare, este esențial să utilizăm algoritmi incrementali și dinamici pentru a aborda complexitățile scenariilor de rețea dinamice. Capitolul 3 al tezei va discuta algoritmii incrementali (dinamici) de ultimă generație.

Capitolul 3

Revizuirea Literaturii

3.1 Introducere

Algoritmii statici pentru problemele drumului minim de la o singură sursă (SSSP) presupun, de obicei, că rețeaua rămâne constantă în timp. Cu toate acestea, această abordare standard nu este adecvată pentru situațiile reale în care condițiile se schimbă probabil ca rezultat al unui mediu în evoluție. În astfel de cazuri, algoritmii SSSP dinamici sunt special concepuți pentru a gestiona schimbările, cum ar fi actualizările greutăților arcelor, ajustând incremental arborele drumului minim (SPT) în loc să-l recalculăm de la zero. Această metodă este deosebit de importantă în scenariile în care rețeaua experimentează schimbări incrementale și necesită strategii adaptive pentru a menține drumurile optime fără a necesita o recalculare completă.

Algoritmii SSSP dinamici actualizează eficient arborele drumului minim ca răspuns la schimbările în ponderile arcelor. Acești algoritmi ajustează incremental doar părțile afectate ale rețelei, îmbunătățind semnificativ complexitatea temporală și permițând adaptarea rapidă la condițiile în schimbare. Această actualizare selectivă permite răspunsuri rapide la condițiile de rețea în schimbare, făcând algoritmii SSSP dinamici cruciali pentru aplicațiile în care rețeaua este frecventă în schimbare.

În recenzia noastră, examinăm fundamentele algoritmilor SSSP dinamici, baza lor teoretică și importanța lor în aplicațiile practice în care schimbările rețelei sunt un fenomen frecvent. Explorăm modul în care acești algoritmi minimizează complexitatea computațională prin actualizarea selectivă doar a părților rețelei direct afectate de schimbări, asigurând astfel recalculări eficiente ale arborilor drumurilor minime.

3.2 Algoritmi Dinamici pentru Problema Drumului Minim de la o Singură Sursă

Deși există o cantitate considerabilă de literatură despre problemele drumurilor minime, în acest capitol al tezei, ne vom concentra pe cele mai cunoscute soluții pentru problema drumului minim de la o singură sursă în contexte dinamice. Menținerea dinamică a drumurilor minime de la o singură sursă, sau echivalent a arborelui drumului minim (SPT), implică menținerea unui arbore optim al drumului minim într-un graf pe măsură ce ponderile arcelor se schimbă în timp. Această problemă este importantă într-o serie de aplicații, cum ar fi rutarea în rețea și gestionarea traficului urban. Un set de algoritmi au fost propuși pentru a aborda această problemă într-un mod eficient, utilizând metodologii diverse pentru a acomoda actualizările dinamice. O serie dintre aceștia vor fi discutați în această secțiune.

Algoritmul bazat pe consistența nodurilor al lui Ramalingam și Reps

În 1996, Ramalingam și Reps [19] au introdus un algoritm iterativ, denumit Dynamic SWSF-FP. Acest algoritm a fost conceput pentru a gestiona inserțiile și ștergerile de arce într-un graf dinamic, dar poate fi adaptat și pentru a gestiona creșteri și scăderi ale greutăților. Conceptele fundamentale ale abordării lor implică definirea nodurilor și arcelor în termeni de **consistență**. Ideea principală a acestui algoritm este de a menține un arbore al drumului minim consistent utilizând coada de prioritate Q pentru a gestiona actualizările eficient. Un nod j este numit consistent dacă distanța sa $d[j]$ corespunde distanței minime obținute prin traversarea oricărui arc de intrare (i, j). Formal, această valoare consistentă $con(j)$ este dată de:

$$\text{con}(j) = \begin{cases} \min\limits_{(i,j)\in A} \{d[i] + w(i,j)\}, & \text{dacă } j \neq s \\ 0, & \text{dacă } j = s \end{cases} \tag{3.1}$$

Un nod este definit ca supra-consistent dacă $d[j] > con(j)$. Un arc (i, j) este definit ca consistent dacă $d[j] = w(i, j) + d[i]$, și subconsistent dacă $d[j] > w(i, j) + d[i]$.

Algoritmul menține date auxiliare pentru a urmări dacă un arc se află pe subarborele drumurilor minime (SPT), care este rădăcinat în nodul sursă s. În plus, numărul de arce de intrare pentru fiecare nod în SPT este de asemenea înregistrat. Pașii algoritmului pot fi rezumați pe scurt astfel:

- **Procesarea scăderilor de greutate**: Algoritmul actualizează lungimea arcului, relaxează arcul și procesează nodurile pentru a menține consistența în arborele drumurilor minime. Când greutatea unui arc scade, algoritmul actualizează mai întâi lungimea arcului și îl relaxează. Dacă arcul este subconsistent, actualizează distanța nodului și îl inserează în coada de prioritate Q.

- **Procesarea creșterilor de greutate**: Algoritmul actualizează lungimea arcului și elimină arcul din SPT dacă acesta se află în el. La ștergerea arcului, un subarbore B al SPT nu mai este conectat la s. Fiecare arc din B este șters din SPT. Pentru fiecare nod i din B care este acum afectat de schimbare, se efectuează o actualizare a distanțelor. Aceasta se face folosind doar nodurile neafectate adiacente cu i, și inserându-le în Q după cum este necesar.

- **Faza principală**: Faza principală a algoritmului implică procesarea nodurilor din Q prin extragerea nodului minim, ajustarea distanțelor și actualizarea sau inserarea arcelor în SPT până când Q este goală. Apoi, faza principală a algoritmului continuă ca înainte.

Algoritmul lui Frigioni et al.

În 2000, Frigioni și al. [8] au propus un algoritm iterativ similar ca natură cu algoritmul lui Ramalingam și Reps, cu excepția faptului că sunt utilizate date auxiliare mai complexe, oferind limite teoretice mai bune în cazurile cele mai nefavorabile. Abordarea lor utilizează o **funcție de contabilitate k-limită pe** G, care este o funcție $K : A \to V$ astfel încât pentru fiecare arc (i, j), nodul $K(i, j)$ este fie i, fie j și nu mai mult de k arce sunt asociate cu orice nod i.

Algoritmul stochează funcția de contabilitate k-limită K a grafului. Pentru un nod dat i, setul de arce (i, j) cu $K(i, j) = i$ este denumit *proprietatea* (i). Setul de alte arce adiacente cu i este denumit *nu proprietatea* (i). Nivelul posterior al unui arc (k, l) și al nodului l relativ la nodul k este definit ca $b\_level_k(l) := d[l] - w(k, l)$. Nivelul anterior al unui arc (k, l) și al nodului l relativ la nodul k este definit ca $f\_level_k(l) := d[l] + w(k, l)$.

Pentru fiecare nod i, algoritmul menține două cozi de prioritate, fiecare dintre ele conținând arcele din *nu proprietatea* (i). Coada B_i este bazată pe maxim, cu prioritatea unui arc (i, j) dată de $b\_level_i(j)$, în timp ce coada F_i este bazată pe minim, cu prioritatea unui arc (i, j) dată de $f\_level_i(j)$. În plus, se menține un arbore al drumurilor minime stocând un nod părinte $p[i]$ pentru fiecare nod $i \neq s$. Pașii algoritmului pot fi rezumați pe scurt astfel:

- **Procesarea scăderilor de greutate**: Dat un arc (i, j) cu o scădere a greutății, algoritmul

actualizează $w(i,j)$ și cozile B_i, F_i, B_j, și F_j. Dacă (i,j) este subconsistent, setează $d[j] = d[i] + w(i,j)$ și inserează j cu prioritatea $d[j]$ în Q.

- **Procesarea creșterilor de greutate**: Dat un arc (i,j) cu o creștere a greutății, algoritmul actualizează $w(i,j)$ și cozile B_i, F_i, B_j, și F_j. Subgraful tentativ al drumurilor minime SPT este dat implicit de toate arcele consistente. La ștergerea arcului cu greutate crescută din SPT, un subarbore B al SPT nu mai este conectat la s. Fiecare arc din B este șters din SPT. Pentru fiecare nod j din B care este acum afectat de schimbare, se efectuează o actualizare a distanțelor $d[j] := \min\{d[i] + w(i,j) \mid (i,j) \in A, i \notin B\}$, urmată de inserarea lui j în Q.

- **Faza principală**: Faza principală continuă astfel până când Q este goală:

  - Extrage și șterge nodul minim j din Q.
  - Actualizează cozile $B(j)$ și $F(j)$.
  - Verifică fiecare arc (j,k) din *nu proprietatea* (j) și fiecare arc (j,k) din *proprietatea* (j) cu $b\_level_j(k) > d[j]$. Dacă (v,w) este subconsistent, setează $d[k] = d[j] + w(j,k)$ și inserează k cu prioritatea $d[k]$ în Q.

Acest algoritm gestionează eficient schimbările dinamice utilizând date auxiliare complexe și funcții de contabilitate k-limită pentru a menține structura arborelui drumurilor minime cu limite de performanță mai bune în cazurile cele mai nefavorabile.

Algoritmul lui Narvaez et al.

În 2000, Narvaez et al. [12] au propus un algoritm de procesare în loturi flexibile, care încorporează diferite structuri de date și variante de procesare pentru a optimiza gestionarea schimbărilor dinamice. Acest algoritm oferă două grade de libertate: alegerea tipului de coadă de prioritate (FIFO, heap sau D'Esopo-Pape) și două variante principale de fază. Ideea de bază a acestui algoritm este de a oferi flexibilitate și eficiență în procesarea actualizărilor dinamice utilizând diferite structuri de date și metode de procesare. Algoritmul menține un arbore al drumurilor minime SPT al grafului prin înregistrarea nodului părinte $p[j]$ pentru fiecare nod j. Setul $B(i,j)$ denotă colecția de noduri din ramura SPT care începe cu arcul (i,j), excluzând i. În timpul execuției algoritmului, un părinte tentativ $p'[i]$ este stocat pentru fiecare nod i în coada de prioritate Q. Pașii algoritmului pot fi rezumați pe scurt astfel:

- **Faza de inițializare**: Faza de inițializare gestionează creșterile și scăderile de greutate iterativ.

- Pentru fiecare arc (i, j) cu creșteri de greutate Δ, algoritmul actualizează lungimea fiecărui arc afectat $w(i, j) = w(i, j) + \Delta$ și ajustează distanțele nodurilor din ramura $B(i, j)$, $d[k] := d[k] + \Delta$ pentru fiecare $k \in B(i, j)$, și pune în coadă nodurile afectate. Setul tuturor vârfurilor cu greutăți incrementate anterior este denumit N_{inc}. Ulterior, fiecare arc cu o țintă în N_{inc} este relaxat, iar fiecare nod supra-consistent $k \in N_{\text{inc}}$ este introdus în Q cu prioritatea $con(k)$, iar $p'[k]$ este actualizat în consecință.
- Pentru scăderile de greutate, algoritmul actualizează în mod similar lungimile, ajustează distanțele și pune în coadă nodurile pentru actualizări ulterioare. În mod specific, actualizează lungimea fiecărui arc afectat, setează $d[k] := d[k] - \Delta$ pentru fiecare $k \in B(i, j)$, și denumește setul tuturor vârfurilor cu greutăți decrementate anterior N_{dec}. Fiecare arc cu o sursă în N_{dec} este apoi relaxat, și fiecare nod k pentru care $d_l := \min\{d[k] + w(k, l) \mid k \in N_{\text{dec}}\} < d[l]$ este introdus în Q cu prioritatea d_l, și $p'[l]$ este actualizat în consecință.

Faza principală:

- **Prima variantă**: Nodurile sunt procesate din coadă prin extragerea nodului următor j, setând distanța sa $d[j]$ la prioritatea lui j în Q, și actualizând părinte său $p[j]$ la $p'[j]$. Algoritmul relaxează apoi fiecare arc de ieșire (j, k). Dacă (j, k) este subconsistent, inserează k cu prioritatea $d[j] + w(j, k)$ în Q și setează $p'[k] = j$. Dacă k este deja în Q, algoritmul actualizează doar prioritatea și $p'[k]$.
- **A doua variantă**: După extragerea nodului următor j, algoritmul identifică și procesează descendenții care necesită actualizare. Extrage nodul următor j din Q, unde $key(j)$ denotă prioritatea lui j în Q. Setează $\lambda := key(j) - d[j]$ și $p[j] := p'[j]$. Algoritmul identifică apoi setul N de toți descendenții lui j în SPT. Subramurile care încep cu noduri i care deja au $key(i) < d[i] + \lambda$ în Q sunt excluse din N. Fiecare nod din N cu $key(i) \geq d[i] + \lambda$ în Q este eliminat din Q. Algoritmul relaxează apoi fiecare arc (j, k) ieșind dintr-un nod $j \in N$. Dacă (j, k) este subconsistent, inserează k cu prioritatea $d[j] + w(j, k)$ în Q și setează $p'[k] = j$. Dacă k este deja în Q, algoritmul actualizează doar prioritatea și $p'[k]$.

Această abordare permite o flexibilitate și eficiență semnificativă, adaptându-se la diferite scenarii și optimizând propagarea actualizărilor distanțelor prin arborele drumurilor minime.

Modelul bilei și sforii

În 2001, Narvaez [13] a propus un model intuitiv de bilă și sfoară pentru a explica formularea problemei SPT dinamice prin programare liniară. Ideea de bază este de a interpreta fizic problema drumurilor minime folosind un model vizual și intuitiv. În acest model, fiecare nod din graf este reprezentat de o bilă, iar fiecare arc cu o greutate este reprezentat de o sfoară inelastică cu o lungime egală cu greutatea arcului. Pentru a asigura fezabilitatea, distanța euclidiană între bile nu trebuie să depășească lungimea sforii. O sfoară devine **strânsă** atunci când distanța între nodurile sale este egală cu lungimea sa și reprezintă un arc consistent în arborele drumurilor minime.

Pe măsură ce ponderile se schimbă, modelul ajustează prin mișcarea bilelor pentru a menține arborele drumurilor minime. Creșterile de greutate determină bilele să coboare, reprezentând distanțe crescute, în timp ce scăderile de greutate determină bilele să se ridice, reprezentând căi mai scurte. Acest model ajută la vizualizarea propagării schimbărilor în graf și la menținerea arborilor valizi ai drumurilor minime.

În execuția algoritmului, nodul rădăcină este ancorat într-o poziție fixă, în timp ce alte noduri cad sub gravitație. Pe măsură ce gravitația trage aceste noduri în jos, distanța euclidiană rezultată a unui nod față de nodul rădăcină ancorat este egală cu distanța cea mai scurtă. Pe măsură ce lungimea unei sfori crește, bila atașată la capătul inferior va cădea până când una dintre sforile sale atașate devine strânsă. În mod similar, pe măsură ce lungimea unei sfori scade, bila se ridică în consecință. Eficiența acestei abordări constă în ordinea naturală a propagării distanțelor, care minimizează efortul computațional.

Execuția algoritmului poate fi rezumată pe scurt astfel:

- **Procesarea creșterilor de greutate**: Actualizați lungimea fiecărui arc cu o creștere de greutate. Dacă arcul face parte din SPT-ul existent, marcați nodul final și descendenții săi accesibili ca plutitori. Pentru nodurile ancorate conectate la nodurile plutitoare, calculați distanțele potențiale și puneți aceste noduri plutitoare în coadă pentru actualizări ulterioare.

- **Procesarea scăderilor de greutate**: Calculați distanțele potențiale noi pentru nodurile finale pentru arcele cu greutate scăzută. Dacă noua distanță este mai mică decât cea veche, atunci nodul este pus în coadă cu noul său părinte potențial și noua sa distanță.

- **Faza principală**: Extrageți nodul cu cea mai mică schimbare din coadă și actualizați părintele său. Ajustați structura arborelui pentru a reflecta noile relații părinte-copil și actuali-

zați distanțele. Marcați nodurile ca ancorate și eliminați-le din coadă, dacă este necesar. Repetați până când coada este goală, asigurându-vă că toate nodurile afectate sunt actualizate eficient.

3.3 Concluzii

Din algoritmii existenți în literatură, se poate concluziona că cel mai provocator aspect al algoritmilor dinamici pentru problema drumului minim de la o singură sursă (SSSP) este gestionarea creșterii sau scăderii greutăților arcelor. Aceste operații necesită actualizarea rapidă a arborelui drumurilor minime (SPT) fără a-l recalcula de la zero. Eficiența unui algoritm dinamic SSSP este influențată semnificativ de capacitatea sa de a gestiona aceste schimbări eficient.

Tehnicile care ajustează selectiv doar părțile afectate ale SPT sunt esențiale pentru menținerea eficienței acestor algoritmi. Utilizarea structurilor de date competitive, cum ar fi heap-urile Fibonacci sau arborii dinamici, joacă un rol crucial în asigurarea actualizărilor rapide. Aceste ajustări selective și structurile de date eficiente ajută la minimizarea complexității computaționale, permițând astfel algoritmului să se adapteze rapid la schimbările din rețea.

Deși s-au făcut progrese semnificative în domeniul algoritmilor dinamici SSSP, există încă un potențial considerabil pentru obținerea unor timpuri de execuție mai rapide în medii practice. O întrebare pertinentă este dacă este posibil să se dezvolte algoritmi robuști care să performeze eficient într-o gamă largă de aplicații și pentru diferite tipuri de rețele, atât în medii statice, cât și dinamice, garantând în același timp o complexitate polinomială puternică în cel mai rău caz.

Capitolul 4

Ajustarea Dinamică a Drumului Minim de la o Sursă Unică

4.1 Introducere

Problema calculării drumurilor minime a constituit un domeniu important de cercetare în teoria grafurilor pentru o perioadă considerabilă de timp și rămâne o subrutină fundamentală pentru numeroase sarcini avansate în diverse domenii. Rezolvarea acestei probleme este de mare importanță în domenii precum transportul, navigația, logistica, gestionarea lanțului de aprovizionare, telecomunicațiile, precum și planificarea urbană. În plus, calculele drumurilor minime sunt de mare semnificație în rețelele sociale, unde măsurarea gradului de separare între indivizi este importantă în contextul analizei influenței sociale și detectării comunității. În rețelele biologice, calculele drumurilor minime ajută la analiza rețelelor de interacțiune proteică, permițând identificarea căilor critice în procesele biologice.

O abordare tradițională a problemei drumului minim presupune un graf static, unde ponderile arcelor și structurile rămân neschimbate după calcularea unei căi scurte. Cu toate acestea, în medii dinamice, schimbările, cum ar fi creșterile sau scăderile ponderilor arcelor, necesită recalcularea drumurilor minime. Acest proces poate fi costisitor din punct de vedere computațional și ineficient, în special în cazul grafurilor mari sau al actualizărilor frecvente.

În acest capitol, introducem o abordare dinamică pentru recalcularea drumurilor minime de la o sursă unică (SSSP) atunci când greutatea unui arc crește sau scade. Spre deosebire de abordarea convențională de recalculare a întregii soluții de la zero, algoritmii dinamici actuali-

zează informațiile existente despre cele mai scurte căi în funcție de schimbările din graf. Dacă greutatea unui arc crește, de exemplu, algoritmul trebuie doar să verifice și să actualizeze căile afectate. În schimb, dacă greutatea unui arc scade, algoritmul poate disemina eficient această îmbunătățire în întreaga rețea, actualizând căile cele mai scurte după cum este necesar. Pentru a minimiza timpul de recalculare, actualizarea dinamică a drumurilor minime de la o sursă unică (SSSP) utilizează căile cele mai scurte calculate anterior. Această abordare este deosebit de utilă în medii în care grafurile se schimbă frecvent, permițând actualizări în timp real, menținând în același timp căile optime. Acest algoritm este o extindere a ideilor noastre prezentate în [3].

4.2 Preliminarii

Algoritmii secvențiali pentru cele mai scurte căi utilizează de obicei metode iterative de etichetare, care mențin o valoare temporară a distanței $d[v]$ pentru toate nodurile. Valoarea atribuită distanței temporare poate fi fie ∞, fie greutatea unei căi de la sursa s la v. Această valoare a distanței temporare servește drept limită superioară pentru distanța reală între sursa s și v. Distanțele temporare sunt actualizate printr-o operațiune numită "relaxarea arcului": pentru un arc $(u, v) \in A$, $d[v] = min\{d[v], d[u] + w(u, v)\}$.

Există două categorii principale de metode de etichetare: etichetarea stabilită și etichetarea corectată. Algoritmii care efectuează etichetarea stabilită, cum ar fi algoritmul lui Dijkstra, stabilesc distanța unui nod ca optimă în fiecare iterație. Aceasta necesită cel mult n iterații, unde n este numărul de noduri. În schimb, algoritmii care corectează etichetele, pot relaxa arcele nodurilor non-optime și numărul de iterații necesare poate varia. Toate etichetele sunt considerate temporare până la pasul final, când devin permanente. În ciuda superiorității limitelor în cazuri nefavorabile pentru algoritmii de etichetare stabilită față de cei de etichetare corectată, abordările de etichetare corectată prezintă adesea performanțe practice bune, în special pentru clase mari de grafuri cu greutăți aleatorii ale arcelor. Pentru a exploata performanța practică avantajoasă a abordărilor de etichetare corectată, [11] introduce un algoritm de etichetare corectată, cunoscut sub numele de algoritmul Delta-stepping. Acest algoritm poate fi implementat într-un mod extrem de eficient atât în setări secvențiale, cât și paralele pentru diverse clase de grafuri. În plus, este capabil să atingă un timp optim linear cu o probabilitate mare.

Algoritmul Delta-stepping menține nodurile eligibile cu distanțe temporare, organizate într-un array de bucket-uri. Fiecare bucket corespunde unui interval specific de distanțe, cu dimensiunea Δ a acestui interval determinată de utilizator. În timpul fiecărei faze, algoritmul elimină și procesează toate nodurile din prima bucket ne-goală și relaxează toate arcele de ieșire care au greutăți de cel mult Δ. Arcele cu greutăți mai mari sunt relaxate numai ulterior. Alegerea valorii Δ este crucială pentru a echilibra între reconsiderarea excesivă a nodurilor și traversarea excesivă a găleților.

Deoarece o versiune modificată a algoritmului Delta-stepping este utilizată în algoritmul nostru de Ajustare Dinamică a drumurilor minime de la o Sursă Unică (DynAdjSSSPA), se oferă mai jos o prezentare succintă a acestuia. Informații detaliate despre algoritmul Delta-stepping și funcțiile sale auxiliare pot fi găsite în [11].

Algoritmul Delta-stepping procesează un graf $G = (V, A, w)$ și un nod sursă s pentru a returna array-ul $d[v]$ care conține costul minim pentru a ajunge la orice nod v pornind de la s. Algoritmul păstrează nodurile încă de procesat și care au distanțe cele mai scurte non-optime într-un array B de bucket-uri, fiecare bucket reprezentând un interval de distanțe egal cu Δ. Parametrul Δ este un număr real pozitiv, cunoscut și sub numele de "dimensiunea pasului" sau "lățimea găleții" și care determină distribuția nodurilor în interiorul găleților. Fiecare bucket $B[i]$ conține nodurile de procesat într-o iterație, și dacă pentru un nod $v \in V$, $d[v]$ reprezintă cel mai bun cost cunoscut pentru a ajunge la v de la s, atunci v aparține găleții $B[i]$, unde $i = \left\lfloor \frac{d[v]}{\Delta} \right\rfloor$.

Algoritmul procesează gălețile $B[i]$ una câte una, începând de la $i = 0$. La fiecare iterație i, algoritmul se concentrează pe prima bucket care nu este goală (bucket-ul curentă) și elimină toate nodurile din ea. Pentru fiecare nod u eliminat, relaxează toate arcele ușoare (cu $w(u, v) \leq \Delta$) care ies din acest nod. Acest proces de relaxare poate duce la adăugarea de noduri noi în bucket-ul curentă. Aceste noduri vor fi procesate în faza următoare. În plus, nodurile eliminate anterior pot fi reinserate dacă distanța lor temporară se îmbunătățește în timpul fazei curente.

Procesul de relaxare a arcelor ușoare se desfășoară într-o buclă pentru a lua în considerare posibilele re-inserții în aceeași bucket. Se utilizează două seturi auxiliare în timpul acestui proces: setul de cereri Req, care stochează perechi de noduri care pot fi atinse de arcele ușoare, împreună cu costurile lor corespunzătoare, și setul R (noduri eliminate), care colectează toate nodurile din bucket-ul $B[i]$ care au fost explorate pentru a evita re-procesarea arcelor lor ușoare în iterațiile ulterioare ale buclei interioare. La sfârșitul fiecărei iterații a buclei interioare, funcția de relaxare este apelată pe perechile din setul de cereri.

Relaxarea arcelor grele (cu $w(u, v) > \Delta$) este amânată în timpul procesării găleții curente, deoarece aceste arce pot afecta doar distanțele temporare din afara intervalului de distanță al găleții curente. Ca urmare, acestea nu vor cauza reinserarea nodurilor înapoi în bucket-ul curentă. Dacă bucket-ul curentă rămâne goală după o fază, toate nodurile din intervalul său de distanță au primit valorile lor permanente de distanță în timpul fazelor precedente. Prin urmare, toate arcele grele care provin din aceste noduri sunt relaxate. După aceasta, algoritmul caută secvențial următoarea bucket care nu este goală și repetă procesul. Algoritmul continuă până când toate gălețile au fost procesate și nu mai există noduri de explorat. În acest moment se oprește și returnează array-ul d, unde $d[v]$ conține costul minim pentru a ajunge la v de la s.

Următoarea teoremă oferă o estimare a complexității pentru algoritmul secvențial Delta-stepping, care servește ca o componentă fundamentală în dezvoltarea algoritmului nostru dinamic.

Teorema 2. *([11]) Algoritmul secvențial delta-stepping are o complexitate temporală de* $O(n + m + L/\Delta + n_\Delta + m_\Delta)$.

4.3 Algoritmul de Ajustare Dinamică a drumului minim de la o Sursă Unică

Fie $G = (V, A, w)$ graful orientat ponderat inițial, unde cele mai scurte căi de la nodul sursă s la celelalte noduri au fost deja determinate. Fie d array-ul care conține ponderile drumurilor minime și p array-ul care conține predecesorii nodurilor din aceste căi. Rezultatele rulării unui algoritm precum Dijkstra pe G pentru a găsi cele mai scurte căi de la o sursă unică pot fi reprezentate ca un arbore al drumurilor minime, care este un subgraf $G' = (V, A', w)$ al lui G. Acest arbore este rădăcinat în nodul sursă și se extinde la toate nodurile accesibile din graf. Setul de arce din arbore este denumit A' și este definit astfel: $A' = \{(p[x], x) | p[x] \neq null\}$. Fiecare cale de la rădăcină la orice alt nod din arbore reprezintă cea mai scurtă cale de la sursă la acel nod în graful original.

În cazul unei modificări a greutății unui arc în graful G, fie că este o creștere sau o scădere, este posibil ca unele dintre cele mai scurte căi să fie modificate, în timp ce altele să rămână neschimbate, dar să experimenteze o modificare a distanței totale. În plus, pot exista unele căi cele mai scurte care nu sunt afectate. Nodurile din graf care pot experimenta o modificare a drumurilor minime și, implicit, a distanțelor lor de la sursă, sunt denumite **Noduri Afectate**.

Setul de noduri care pot avea doar etichetele de distanță modificate, fără nicio schimbare în structura reală a drumului minim, sunt denumite **Noduri Doar de Reetichetat**. În final, există noduri care nu suferă nicio schimbare în căile lor cele mai scurte sau în etichetele lor de distanță și acestea pot fi identificate ca **Noduri Neafectate**.

Algoritmul pentru actualizarea drumurilor minime ca răspuns la schimbările ponderilor arcelor este împărțit în trei faze.

Faza 1: Identificare
În această fază, algoritmul identifică nodurile afectate de schimbarea greutății arcului și pe cele care trebuie doar să fie reetichetate. Acest pas este important, deoarece identifică care noduri necesită procesare ulterioară.

Faza 2: Reetichetare
În această fază, algoritmul efectuează unele actualizări necesare ale etichetelor de distanță, specifice Nodurilor Doar de Reetichetat sau Nodurilor Afectate, în funcție de caz. Aceste actualizări constituie un pas preliminar înainte de a începe cea de-a treia fază.

Faza 3: Algoritmul Delta-Stepping
În această fază, algoritmul completează calculul etichetelor de distanță pentru nodurile afectate folosind un algoritm Delta-stepping modificat. Această subrutină se concentrează pe actualizarea etichetelor de distanță ale Nodurilor Afectate. Menține o listă de bucket-uri și o listă de noduri actualizate pentru procesare, și asigură evaluarea tuturor căilor potențial mai scurte, luând în considerare atât arcele ușoare, cât și cele grele. Această abordare țintită garantează actualizarea eficientă a drumurilor minime.

Următoarele două subsecțiuni discută specificul creșterii și scăderii ponderilor arcelor. Acestea sunt urmate de o subsecțiune în care este prezentat în detaliu Algoritmul de Ajustare Dinamică a drumurilor minime de la o Sursă Unică (DynAdjSSSPA).

4.3.1 Creșterea Capacității Arcului

Fie $\widehat{G} = (N, A, \widehat{w})$ definit ca un graf orientat ponderat care diferă de G doar în ceea ce privește greutatea unui arc individual, în special greutatea arcului (k, l). În acest caz, greutatea arcului (k, l) este mai mare decât greutatea inițială a arcului (k, l). Prin urmare, $\widehat{w}(x, y) = w(x, y)$ pentru fiecare $(x, y) \in A \setminus \{(k, l)\}$ și $\hat{w}(k, l) = w(k, l) + \Delta w$, unde Δw este o cantitate pozitivă dată. Există două cazuri posibile:

Cazul 1: Când $p[l] \neq k$ căile cele mai scurte din G vor fi identice cu cele din $\widehat{G}$, cu aceleași distanțe ca în G. Aceasta înseamnă că toate nodurile din $\widehat{G}$ sunt Noduri Neafectate.

Cazul 2: Când $p[l] = k$ este posibil ca căile cele mai scurte din graful modificat $\widehat{G}$ să difere de cele din graful inițial G. Aceste căi cele mai scurte modificate pot fi doar de la nodul sursă la nodul l, precum și la descendenții lui l în subgraful de predecesori $G' = (V, A', w)$ al lui G. Prin urmare, aceste noduri vor fi Nodurile Afectate și pot fi identificate folosind căutarea în lățime (BFS) în cadrul arborelui drumului minim $G' = (V, A', w)$. Toate celelalte noduri vor fi Noduri Neafectate.

4.3.2 Scăderea Capacității Arcului

În timp ce o creștere a ponderilor arcelor are ca rezultat o actualizare localizată, concentrată pe căile direct afectate de creștere, o scădere a ponderilor arcelor necesită un proces de actualizare mai global. Acest lucru se datorează faptului că greutatea scăzută poate crea noi căi mai scurte care se propagă prin rețea. Prin urmare, scăderea greutății unui arc necesită adesea o actualizare mai cuprinzătoare.

Fie $\widehat{G} = (N, A, \widehat{w})$ graful orientat ponderat care diferă de G prin greutatea arcului (k, l), care este mai mică decât greutatea inițială a arcului (k, l). Prin urmare, $\widehat{w}(x, y) = w(x, y)$ pentru fiecare $(x, y) \in A \setminus \{(k, l)\}$ și $\hat{w}(k, l) = w(k, l) - \Delta w$, unde Δw este o cantitate pozitivă dată. Există din nou, două cazuri posibile:

Cazul 1: Când $p[l] \neq k$ căile cele mai scurte pot fi modificate pentru nodurile care sunt accesibile de la sursă prin nodul l. Prin urmare, aceste noduri vor fi Nodurile Afectate și pot fi identificate folosind căutarea în lățime (BFS) în graful $\widehat{G} = (V, A, \widehat{w})$. Toate celelalte noduri vor fi Noduri Neafectate.

Cazul 2: Când $p[l] = k$ căile cele mai scurte din graful modificat $\widehat{G}$ pot diferi de cele din graful inițial G. Aceste căi cele mai scurte modificate pot fi doar de la nodul sursă la nodurile care sunt accesibile de la sursă prin nodul l în graful $\widehat{G} = (V, A, \widehat{w})$, prin urmare, aceste noduri vor fi Nodurile Afectate. Pentru nodul l și descendenții săi în arborele drumurilor minime $G' = (V, A', w)$, căile cele mai scurte vor rămâne aceleași, și doar etichetele de distanță vor scădea cu Δw. Prin urmare, aceste noduri vor fi Nodurile Doar de Reetichetat și pot fi identificate folosind căutarea în lățime (BFS) în cadrul arborelui drumurilor minime $G' = (V, A', w)$. Toate celelalte noduri, cu excepția celor două seturi anterioare, vor fi Noduri Neafectate.

4.3.3 Descrierea Algoritmului

Algoritmul de Ajustare Dinamică a drumurilor minime de la o Sursă Unică (Algoritmul 7) este conceput pentru a actualiza eficient cele mai scurte căi într-un graf orientat atunci când greutatea unui arc se modifică. Acest algoritm folosește informațiile despre cele mai scurte căi calculate anterior, în special vectorul de etichete de distanță d și vectorul de predecesori p, pentru a implementa ajustările necesare ca urmare a modificării greutății unui arc $w(k, l)$ cu o valoare egală cu Δw, care poate fi pozitivă sau negativă.

Algoritmul începe prin apelarea subrutinei 'Identificarea Nodurilor Afectate și Doar de Reetichetat' (Algoritmul 2) pentru a identifica acele noduri care pot fi afectate de schimbarea greutății arcului. Această subrutină efectuează o căutare în lățime (Algoritmul 1) fie în graful G, fie în subgraful de predecesori G', și returnează două seturi: Nodurile Afectate și Nodurile Doar de Reetichetat. Primul este format din acele noduri ale căror cele mai scurte căi pot fi afectate ca o consecință a modificării greutății arcului, în timp ce cel de-al doilea este format din noduri care își păstrează căile cele mai scurte, dar necesită actualizarea etichetelor de distanță.

Algoritm 1 Căutare în Lățime

Input: $G = (V, A)$, nod de start r;

Output: Setul de Noduri Accesibile

```
1:  Queue = {r}
2:  Reachable = {r}
3:  while Queue ≠ ∅ do
4:      elimină un nod x din Queue
5:      for fiecare (x, y) ∈ A do
6:          if y ∉ Reachable then
7:              Reachable = Reachable ∪{y}
8:              Queue = Queue ∪{y}
9:          end if
10:     end for
11: end while
12: return Setul de Noduri Accesibile
```

Dacă Δw este negativă, Nodurile Doar de Reetichetat pot fi un set ne-gol. Prin urmare, algoritmul actualizează etichetele de distanță pentru nodurile din setul Nodurilor Doar de Re-

etichetat pentru a reflecta scăderea greutății arcului (k, l). Dacă Δw este pozitivă, algoritmul actualizează etichetele de distanță pentru nodurile din Nodurile Afectate pentru a reflecta creșterea greutății arcului (k, l) din căile curente (vezi liniile 4-12 din Algoritmul 7).

După actualizarea nodurilor reetichetate, algoritmul continuă cu subrutina sa principală, care este Algoritmul Delta-Stepping Modificat (Algoritmul 3). Această subrutină folosește o versiune modificată a algoritmului Delta-Stepping care se concentrează doar pe nodurile care nu sunt Afectate și pe Nodurile Afectate care au fost actualizate. Procesează fiecare nod, luând în considerare atât arcele ușoare, cât și cele grele care duc către Nodurile Afectate, pentru a se asigura că toate căile potențial mai scurte sunt evaluate.

Algoritm 2 Identificarea Nodurilor Afectate și Doar de Reetichetat

Input: $G = (V, A)$, $G' = (V, A')$, s, k, l, Δw, p

Output: Noduri Afectate, Noduri Doar de Reetichetat

```
1: Noduri Afectate = ∅
2: Noduri Doar de Reetichetat = ∅
3: if Δw > 0 then
4:     if p[l] = k then
5:         Noduri Afectate = BFS(G', l)
6:     end if
7: else if Δw < 0 then
8:     if p[l] ≠ k then
9:         Noduri Afectate = BFS(G, l)
10:    else
11:        Noduri Doar de Reetichetat = BFS(G', l)
12:        Toate Nodurile Afectate = BFS(G, l)
13:        Noduri Afectate = Toate Nodurile Afectate − Noduri Doar de Reetichetat − {s}
14:    end if
15: end if
16: return (Noduri Afectate, Noduri Doar de Reetichetat)
```

Componenta principală a algoritmului este procesul iterativ, care continuă atâta timp cât există bucket-uri ne-goale sau seturi ne-goale în $UpdatedNodesList$. În fiecare iterație, se identifică cel mai mic index de bucket ne-goală i. Două seturi, R și $R\_updated$, sunt inițializate pentru a păstra înregistrările nodurilor care sunt procesate. Inițial, algoritmul procesează Nodu-

rile Neafectate din bucket-ul curentă $B[i]$. Nodurile extrase din bucket sunt adăugate una câte una în setul R, după care arcele ușoare (arcele cu greutăți $\leq \Delta$) care duc către Nodurile Afectate sunt relaxate. Acest lucru este realizat folosind funcția $process\_light\_arcs\_to\_affected(u)$ (Algoritmul 4), prin care etichetele de distanță ale nodurilor țintă sunt actualizate dacă se găsește o cale mai scurtă. Odată ce toate Nodurile Neafectate din $B[i]$ au fost procesate, nodurile actualizate din $UpdatedNodesList[i]$ din bucket-ul curentă sunt abordate. Fiecare nod actualizat este șters din listă, apoi adăugat în setul $R\_updated$, iar arcele lor ușoare către Nodurile Afectate sunt relaxate în mod similar.

Odată ce arcele ușoare pentru nodurile din bucket-ul i au fost complet procesate, algoritmul intră în următoarea etapă, care este procesarea arcelor grele (arcele cu greutăți > Δ). Mai întâi, funcția $process\_heavy\_arcs\_to\_affected(R)$ (Algoritmul 5) este invocată pentru a gestiona nodurile din R; apoi, $process\_heavy\_arcs\_to\_affected(R\_updated)$ este invocată pentru a aborda nodurile din $R\_updated$. Acest proces garantează evaluarea tuturor căilor potențial mai scurte, inclusiv cele care implică arce cu greutăți mai mari.

Funcția de relaxare utilizată atât în Algoritmul 4, cât și în Algoritmul 5 este o funcție de relaxare modificată (Algoritmul 6). Această versiune a funcției de relaxare extinde operațiunea clasică de relaxare prin incorporarea gestionării găleților pentru a optimiza procesarea nodurilor. Funcția este responsabilă pentru calcularea unei noi distanțe potențiale pentru un nod țintă v printr-un arc (u, v). Dacă noua distanță calculată este mai scurtă decât eticheta de distanță actuală, funcția actualizează eticheta de distanță și marchează nodul ca actualizat, introducându-l în lista corespunzătoare a nodurilor actualizate, conform noii sale distanțe.

Algoritmul de Ajustare Dinamică a drumurilor minime de la o Sursă Unică se încheie prin returnarea drumurilor minime optime de la nodul sursă la toate celelalte noduri din graful modificat. Aceste căi pot fi reconstruite din vectorul actualizat de etichete de distanță d și vectorul actualizat de predecesori p, care reprezintă ieșirea algoritmului. Acest mecanism de ajustare dinamică asigură că recalcularea este eficientă, fără a fi necesară recomputarea întregului arbore al drumurilor minime de la zero.

4.4 Concluzii

Acest capitol prezintă Algoritmul de Ajustare Dinamică a drumului minim de la o Sursă Unică (DynAdjSSSPA), care folosește o versiune modificată a algoritmului Delta-stepping pen-

Algoritm 3 Algoritmul Delta-Stepping Modificat

Input: $G = (V, A, w)$, etichetele de distanță d, predecesorii p, Nodurile Afectate, dimensiunea pasului Δ

Output: Vectorul de etichete de distanță actualizat d, vectorul de predecesori actualizat p

1: Initializează gălețile B ca o listă de seturi
2: Initializează UpdatedNodesList ca o listă de seturi, câte una corespunzând fiecărei bucket-uri
3: $num_buckets = \lfloor$distanța maximă$/\Delta \rfloor + 1$
4: **for** fiecare nod u în V $\setminus$ Nodurile Afectate **do**
5: $\quad bucket\_index = \left\lfloor \frac{d[u]}{\Delta} \right\rfloor$
6: $\quad B[bucket\_index] = B[bucket\_index] \cup \{u\}$
7: **end for**
8: **while** există un index i astfel încât $B[i] \neq \emptyset$ sau $UpdatedNodesList[i] \neq \emptyset$ **do**
9: $\quad i =$ min$\{j : B[j] \neq \emptyset$ sau $UpdatedNodesList[j] \neq \emptyset\}$
10: $\quad R = \emptyset$
11: $\quad R\_updated = \emptyset$
12: $\quad$ **while** $B[i] \neq \emptyset$ **do**
13: $\quad\quad u = B[i]$.pop$()$
14: $\quad\quad R = R \cup \{u\}$
15: $\quad\quad$ process_light_arcs_to_affected(u)
16: $\quad$ **end while**
17: $\quad$ **while** $UpdatedNodesList[i] \neq \emptyset$ **do**
18: $\quad\quad u = UpdatedNodesList[i]$.pop$()$
19: $\quad\quad R\_updated = R\_updated \cup \{u\}$
20: $\quad\quad$ process_light_arcs_to_affected(u)
21: $\quad$ **end while**
22: $\quad$ **if** $R \neq \emptyset$ **then**
23: $\quad\quad$ process_heavy_arcs_to_affected(R)
24: $\quad$ **end if**
25: $\quad$ **if** $R\_updated \neq \emptyset$ **then**
26: $\quad\quad$ process_heavy_arcs_to_affected($R\_updated$)
27: $\quad$ **end if**
28: **end while**

Algoritm 4 Procesarea Arcelor Ușoare către Nodurile Afectate

1: **Funcția process_light_arcs_to_affected(**u**)**
2: **for** fiecare arc de ieșire (u, v) astfel încât $w(u, v) \leq \Delta$ **do**
3: **if** $v \in$ Nodurile Afectate **then**
4: relax($u, v, w(u, v)$)
5: **end if**
6: **end for**

Algoritm 5 Procesarea Arcelor Grele către Nodurile Afectate

1: **Funcția process_heavy_arcs_to_affected(**R**)**
2: **for** fiecare nod u în R **do**
3: **for** fiecare arc de ieșire (u, v) astfel încât $w(u, v) > \Delta$ **do**
4: **if** $v \in$ Nodurile Afectate **then**
5: relax($u, v, w(u, v)$)
6: **end if**
7: **end for**
8: **end for**

tru a gestiona eficient calculele drumurilor minime în medii grafice dinamice. Investigația noastră asupra corectitudinii și complexității algoritmului evidențiază câteva constatări potențiale și domenii pentru cercetări și îmbunătățiri viitoare.

Pentru a asigura corectitudinea DynAdjSSSPA, este important să se identifice și să se proceseze toate nodurile afectate ale căror cele mai scurte căi pot fi modificate datorită modificărilor greutății arcelor. Acest lucru este realizat printr-o abordare structurată care implică identificarea nodurilor afectate (faza 1), reetichetarea nodurilor (faza 2) și aplicarea algoritmului Delta-stepping pentru ajustările finale (faza 3). Corectitudinea algoritmului este susținută de capacitatea sa de a diferenția corect între nodurile afectate, nodurile doar de reetichetat și nodurile neafectate, asigurând astfel actualizări precise și eficiente ale drumurilor minime.

Utilizarea găleților în algoritmul Delta-stepping permite procesarea organizată a nodurilor pe baza intervalelor lor de distanță. În ciuda limitelor nefavorabile pentru algoritmii de etichetare stabilită, precum Dijkstra, performanța practică a algoritmilor de etichetare corectată, cum ar fi Delta-stepping, este adesea superioară pentru grafuri mari cu greutăți aleatorii ale arcelor.

Algoritm 6 Funcția de Relaxare cu Verificarea Găleții

Funcția relax(u, v, w**)**
$new\_distance = d[u] + w$
if $new\_distance < d[v]$ **then**
    $current\_bucket\_index = \left\lfloor \frac{d[v]}{\Delta} \right\rfloor$
    $new\_bucket\_index = \lfloor \frac{new\_distance}{\Delta} \rfloor$
    $found =$ **false**
    for fiecare bucket j în UpdatedNodesList **do**
        if $v \in UpdatedNodesList[j]$ **then**
            $found =$ **true**
            if $current\_bucket\_index \neq new\_bucket\_index$ **then**
                Elimină v din $UpdatedNodesList[j]$
                $UpdatedNodesList[new\_bucket\_index] = UpdatedNodesList[new\_bucket\_index] \cup \{v\}$
            end if
            break
        end if
    end for
    if not $found$ **then**
        $UpdatedNodesList[new\_bucket\_index] = UpdatedNodesList[new\_bucket\_index] \cup \{v\}$
    end if
    $d[v] = new\_distance$
    $p[v] = u$
end if

Algoritm 7 Algoritmul de Ajustare Dinamică a drumului minim de la o Sursă Unică

Input: Graf orientat $G = (V, A, w)$, vectorul de etichete de distanță d, vectorul de predecesori p, arc (k, l), modificarea greutății Δw

Output: Vectorul de etichete de distanță actualizat d, vectorul de predecesori actualizat p

Faza 1: Găsirea Nodurilor Afectate și Doar de Reetichetat
(Noduri Afectate, Noduri Doar de Reetichetat) =
    = Identificarea Nodurilor Afectate și Doar de Reetichetat$(G, G', (k, l), \Delta w, p)$
Faza 2: Reetichetarea Nodurilor
if $\Delta w < 0$ **then**
    for fiecare nod u în Nodurile Doar de Reetichetat **do**
        $d[u] = d[u] + \Delta w$
    end for
end if
if $\Delta w > 0$ **then**
    for fiecare nod u în Nodurile Afectate **do**
        $d[u] = d[u] + \Delta w$
    end for
end if
Faza 3: Actualizarea distanțelor. Apelarea Algoritmului Delta-Stepping Modificat
$(d, p) =$ Delta-Stepping Modified Algorithm$(\widehat{G}, d, p,$ Nodurile Afectate$)$
return (d, p)

Capitolul 5

Creșterea fluxului prin expansiunea rețelei

5.1 Introducere

În gestionarea infrastructurii moderne, extinderea rețelei este o sarcină critică pentru a satisface cererea tot mai mare de servicii esențiale, cum ar fi electricitatea, gazul, apa și datele. Această expansiune poate fi realizată prin creșterea capacității de transport a firelor, conductelor sau lățimii de bandă existente. Alternativ, pot fi adăugate noi conexiuni la sistem. Totuși, fiecare dintre aceste strategii este constrânsă de limitările fizice sau tehnologice și implică un cost semnificativ. Având în vedere aceste constrângeri, provocarea este să optimizăm proiectarea rețelei pentru a obține eficiență maximă la cost minim.

Domeniul optimizării fluxului în rețea oferă o serie de formulări ale problemelor care sunt direct legate de aceste provocări. Două astfel de probleme, care sunt centrale în discuțiile din acest capitol, sunt **problema fluxului maxim invers** și **problema fluxului maxim revers**. Aceste probleme utilizează modele matematice sofisticate pentru a găsi soluții optime și implică modificări strategice ale capacității arcelor rețelei pentru a atinge obiective specifice.

Problema fluxului maxim invers oferă un cadru matematic pentru a face ajustări minime ale capacităților existente ale unei rețele pentru a atinge fluxul maxim dorit, minimizând în același timp modificările capacităților originale. Problema fluxului maxim invers a apărut din necesitatea de a optimiza infrastructurile de rețea existente fără costurile extinse asociate cu construirea de noi conexiuni sau reproiectarea completă a rețelei. Această problemă este deosebit de relevantă în contexte în care constrângerile fizice, financiare sau de reglementare limitează posibilitatea de a extinde capacitatea rețelei prin mijloace tradiționale. Optimizarea rețelei este cuantificată folosind diferite norme pentru a măsura ajustările capacității. În special, problema

sub norma L_∞ poate fi rezolvată eficient folosind o abordare de căutare binară, așa cum este arătat în [4]. Când se consideră norma L_k, a fost dezvoltat un algoritm puternic polinomial care implică în principal calculul unui tăieturii minime într-o rețea specială [5]. În plus, s-a demonstrat că problema este NP-hard [21], iar complexitatea acesteia crește în scenariile care implică pierderi sau câștiguri pe arce.

Problema fluxului maxim revers [22] urmărește să identifice noi vectori de capacitate care să asigure menținerea fluxului maxim într-o rețea modificată peste un prag specificat, v_0, reducând în același timp distanța, măsurată prin norma Chebyshev, între vectorii de capacitate inițiali și cei noi. Un algoritm polinomial, care funcționează în două faze, a fost dezvoltat pentru a rezolva această problemă. În faza inițială, o căutare binară este utilizată pentru a identifica un interval care conține valoarea optimă a fluxului, v_0. Ulterior, în a doua fază, metoda Newton, așa cum este descrisă în [24], este utilizată pentru a determina vectorul optim de capacitate. Problema fluxului maxim revers este deosebit de relevantă în contexte în care este critic să se controleze sau să se reducă capacitatea rețelei fără a o reproiecta complet, cum ar fi în timpul operațiunilor de reducere, sau pentru a respecta noile reglementări de mediu sau de siguranță. Aplicațiile includ managementul mediului pentru a preveni suprautilizarea resurselor, asigurarea conformității cu siguranța în industriile chimice, petroliere și de gaze, și gestionarea capacității rețelei în timpul încetinirilor economice sau reorganizărilor corporative.

Problema extinderii rețelei în condiții de buget a fost studiată în [7]. Obiectivul este de a utiliza bugetul într-un mod optim, maximizând în același timp fluxul în rețea, dat fiind un buget, o expansiune maximă potențială a capacității arcului și o funcție de cost de expansiune a capacității arcului. Aceasta este denumită problema extinderii fluxului constrâns de buget (BFEP). Abordarea fundamentală pentru rezolvarea acestei probleme este de a crește incremental fluxul în rețea cu o unitate, folosind cel mai rentabil drum de la sursă la destinație. La fiecare iterație, bugetul rămas este recalculat, iar procesul se încheie atunci când întregul buget a fost epuizat. Soluția acestei probleme se bazează pe două algoritmi polinomiali. Un algoritm identifică arcele care necesită extindere pentru a obține o creștere de o unitate a fluxului la un cost minim. A doua iterație apelează la prima și verifică dacă limita impusă a bugetului a fost atinsă. Problema extinderii fluxului constrâns de buget (BFEP) încorporează efectiv strategii din atât problema fluxului maxim invers, cât și problema fluxului maxim revers pentru a-și îmbunătăți abordarea de optimizare a extinderii rețelei în limitele bugetului fix. Conexiunea cu problema fluxului maxim invers este evidentă, deoarece ambele se concentrează pe maximizarea fluxului în rețea - BFEP face acest lucru în limitele bugetului, în timp ce problema inversă caută să

realizeze acest lucru prin ajustări minime ale capacității. Acest obiectiv comun permite BFEP să folosească strategii din problema inversă pentru a determina care modificări vor îmbunătăți cel mai eficient fluxul. În mod similar, relația cu problema fluxului maxim revers este esențială în ghidarea managementului capacității. Problema reversă, care de obicei urmărește reducerea capacității pentru a evita depășirea anumitor praguri de flux, oferă perspective valoroase în gestionarea expansiunilor pentru a se asigura că acestea sunt necesare și durabile. În același mod în care problema reversă folosește ajustări ale capacității pentru a menține eficiența operațională și siguranța, BFEP aplică aceste principii pentru a se asigura că orice expansiune este atât viabilă economic, cât și esențială pentru a satisface cerințele rețelei, asigurând astfel creșterea strategică și rentabilă a rețelei.

O gamă largă de aplicații ale problemei fluxului maxim pot fi găsite în literatură, cum ar fi problemele de transport [20], [2] și problemele de transport prin conducte [9], [10], [23], iar în toate aceste tipuri de probleme, expansiunea rețelei poate fi necesară pentru a crește cantitatea de flux transportată prin rețea. Această necesitate oferă un context convingător pentru analiza care este realizată în acest capitol, în care tratăm problema extinderii rețelei la cost minim (MCNEP).

Obiectivul MCNEP, începând cu o rețea dată G, este de a obține o nouă rețea G', prin creșterea capacităților arcelor din G în limite definite sau prin adăugarea de noi arce, pentru a obține cel mai mic cost posibil de modificare, permițând în același timp transportul a w unități de flux de la sursă la destinație. Costul asociat cu creșterea capacității și inserția arcelor este o funcție liniară în raport cu modificarea capacităților respective. Cercetarea noastră este ghidată de cerințele practice de aplicare a acestei probleme în domeniul infrastructurii, asigurând că modificările, fie pentru creșterea capacității, fie pentru adăugarea de noi arce, sunt atât rentabile cât și suficient de substanțiale pentru a satisface cerințele specifice de flux. Această focalizare subliniază relevanța studiului nostru și evidențiază implicațiile mai largi ale soluțiilor MCNEP în scenarii reale de gestionare a rețelelor.

Rezultatele din acest capitol au fost publicate în [6] și prezentarea lor în acest capitol al tezei este organizată după cum urmează. Problema Extinderii Rețelei cu Cost Minim (MCNEP) este descrisă formal în Secțiunea 5.2 și este introdus un algoritm puternic polinomial pentru rezolvarea acestei probleme. Secțiunea 5.2 încheie această lucrare și identifică direcțiile viitoare potențiale de cercetare.

5.2 Creșterea fluxului prin extinderea rețelei

După cum s-a menționat anterior, Problema Extinderii Rețelei cu Cost Minim (MCNEP) se referă la extinderea strategică a unei rețele date G într-o nouă rețea G prin creșterea capacităților arcelor existente sau prin adăugarea de noi arce în limite definite, pentru a asigura că rețeaua modificată G poate permite trecerea a w unități de flux de la sursă la destinație, minimizând în același timp costurile de modificare. Costul modificării rețelei este o funcție liniară în raport cu schimbările de capacitate. Aspectele economice ale acestei extinderi de rețea sunt abordate prin introducerea unei funcții de cost $c \to \mathbb{R}+^*$, unde $c(a)$ este costul modificării capacității pe arcul $a \in A$ pe unitate. Astfel, dacă capacitatea arcului a crește cu d unități, costul total al schimbării capacității lui a va fi $c(a) \cdot d$. Mai mult, pentru a evita supra-extinderea capacității rețelei, se impune o funcție de limitare superioară $\alpha \to \mathbb{R}+^*$, stabilind $u(a) + \alpha(a)$ ca fiind capacitatea maximă permisă pentru fiecare arc a.

Fie Q setul de arce care pot fi adăugate la rețea. Desigur, $Q \subseteq V \times V$, și $Q \cap A = \phi$, adică $Q \subseteq V \times V - A$. Se introduce și o funcție de cost $c_Q \to \mathbb{R}+^*$, unde $c_Q(a)$ este costul pe unitate de capacitate dacă arcul $a \in Q$ este adăugat la rețea. Prin urmare, dacă arcul $a \in Q$ având capacitatea $u_Q(a) > 0$ este introdus în rețea, atunci costul adăugării lui a la rețea este $c_Q(a) \cdot u_Q(a)$. Se introduce și o funcție de limitare superioară $\beta \to \mathbb{R}+^*$ pentru capacitățile arcelor din Q, unde $\beta(a)$ este capacitatea maximă permisă pentru arcul $a \in Q$ dacă este introdus în rețea, adică $u_Q(a) \leq \beta(a)$, unde $u_Q(a) > 0$ este capacitatea lui a.

Obiectivul este de a identifica **extinderea cu cost minim a rețelei**, G, prin creșterea capacităților arcelor și prin adăugarea de noi arce, astfel încât în rețeaua rezultată G', w unități de flux să poată fi transportate de la s la t. Adică, există un flux admisibil de valoare w în G'. Prin urmare, trebuie rezolvată următoarea problemă:

$$\begin{cases} min\{\sum_{a\in A}(c(a)\cdot(v(a)-u(a))+\sum_{a\in Q}(c_Q(a)\cdot u_Q(a))\} \\ u(a)\leq v(a)\leq u(a)+\alpha(a), \forall a\in A \\ 0\leq u_Q(a)\leq \beta(a), \forall a\in Q \\ \text{exista un flux admisibil cu valoare } w \text{ in } G'=(V,A',s,t,u') \\ A'=A\cup\{a\in Q \mid u_Q(a)>0\} \\ u'(a)=\begin{cases} v(a), a\in A \\ u_Q(a), a\in A'-A \end{cases}, \forall a\in A' \end{cases} \tag{5.1}$$

Vom denumi problema din Eq. (5.1) ca *problema extinderii rețelei cu cost minim* și o vom nota cu MCNEP. Putem formula următorul rezultat:

Teorema 3. *Dacă $v(f^*)\geq w$, unde f^* este fluxul maxim în rețeaua G, atunci G este soluția MCNEP.*

Demonstrație. Fie f^* fluxul maxim în rețeaua G. Presupunem că $v(f^*)\geq w$. Rezultă că există un flux admisibil f în G astfel încât $v(f)=w$. Deci, G este o soluție fezabilă a MCNEP. Deoarece costul funcției obiectiv din Eq. (5.1) pentru G este 0, este evident că G reprezintă soluția optimă pentru MCNEP. □

Vom investiga acum fezabilitatea MCNEP. Se poate arăta că valoarea maximă a fluxului care poate fi transportat de la s la t apare atunci când capacitățile tuturor arcelor din A sunt crescute la valoarea lor maximă și toate arcele din Q sunt incorporate în rețea, folosind capacitățile maxime permise. Rezultă astfel că valoarea maximă a fluxului care poate fi transportat de la s la t este obținută în rețeaua $G''=(V,A\cup Q,s,t,u'')$, unde:

$$u''(a)=\begin{cases} u(a)+\alpha(a), a\in A \\ \beta(a), a\in Q \end{cases} \tag{5.2}$$

Vom numi G'' *rețeaua extinsă maximă* deoarece toate capacitățile arcelor din $A\cup Q$ sunt setate la valoarea lor maximă.

Avem următoarea teoremă de fezabilitate pentru MCNEP:

Teorema 4. *MCNEP este admisibil dacă și numai dacă $v(g^*)\geq w$, unde g^* este un flux maxim în G''.*

Demonstrație. Să presupunem că MCNEP este admisibil și $v(g^*) < w$, unde g^* este un flux maxim în G''. Deoarece MCNEP este admisibil, rezultă că există o rețea $G' = (V, A', s, t, u')$ astfel încât $u(a) \leq u'(a) \leq u(a) + \alpha(a), \forall a \in A$, $0 \leq u'(a) \leq \beta(a), \forall a \in Q$, și există un flux admisibil f' în G' astfel încât $v(f') = w$. Din (5.2) putem concluziona că f' este un flux admisibil în G'', și deoarece f^* este un flux maxim în G'' rezultă că $v(g^*) \geq v(f') = w$, în contradicție cu presupunerea inițială că $v(g^*) < w$.

Acum, pentru implicația inversă, presupunem că pentru fluxul maxim g^* în G'' avem $v(g^*) \geq w$. Rezultă că există un flux admisibil f'' în G'' astfel încât $v(f'') = w$. Deci, G'' este o soluție fezabilă pentru Eq. (5.1). □

Dacă $v(f^*) < w$, cu f^* fiind flux maxim în rețeaua G (G nu este soluția MCNEP, vezi Teorema 3) și MCNEP trece testul de fezabilitate dat de Teorema 4, aceasta înseamnă că există o soluție pentru MCNEP și trebuie găsită. Pentru a face acest lucru, creăm o nouă rețea notată $G^e = (V^e, A^e, s, t, u^e, c^e)$. Setul V^e conține toți nodurile din V, și pentru fiecare arc $a \in A$, un nou nod notat i_a este introdus în V^e, adică,

$$V^e = V \cup V_A, V_A = i_a \mid a \in A \tag{5.3}$$

Setul A^e conține toate arcele din $A \cup Q$, și pentru fiecare arc $a = (i, j) \in A$ se adaugă două noi arce, (i, i_a), și, respectiv, (i_a, j), adică,

$$A^e = A \cup Q \cup A_1^e \cup A_2^e, \tag{5.4}$$

unde:

$$A_1^e = (i, i_a) \mid a = (i, j) \in A, A_2^e = (i_a, j) \mid a = (i, j) \in A \tag{5.5}$$

Funcția de capacitate u^e este definită astfel:

$$u^e(i,j) = \begin{cases} u(i,j), \text{dacă } (i,j) \in A \\ \beta(i,j), \text{dacă } (i,j) \in Q \\ \alpha(a), \text{dacă } j = i_a \text{ sau } i = i_a \text{ unde } i_a \in V_A \end{cases} \tag{5.6}$$

Funcția de cost c^e este definită astfel:

$$c^e(i,j) = \begin{cases} 0, \text{dacă } (i,j) \in A \\ c_Q(i,j), \text{dacă } (i,j) \in Q \\ c(a)/2, \text{dacă } j = i_a \text{ sau } i = i_a \text{ unde } i_a \in V_A \end{cases} \tag{5.7}$$

Fie f^e un flux de cost minim de valoare w în G^e, ceea ce înseamnă că, f^e este un flux admisibil de valoare w în G^e care are cel mai mic cost dintre toate fluxurile admisibile de valoare w în G^e, unde costul unui flux admisibil f în G^e notat $c^e(f)$ este definit astfel:

$$c^e(f) = \sum_{a \in A^e} c^e(a) \cdot f(a) \tag{5.8}$$

Considerăm rețeaua notată $G^{=}(V, A, s, t, u)$ având:

$$A^* = A \cup A', A' = a \in Q \mid f^e(a) > 0 \tag{5.9}$$

și

$$u^{(}a) = \begin{cases} u(a) + f^e(i, i_a), \text{dacă } a \in A \\ f^e(a), \text{dacă } a \in A^{-}A \end{cases} \tag{5.10}$$

Teorema 5. *Rețeaua $G^{=}(V, A, s, t, u^*)$ definită folosind Eq. (5.9) și Eq. (5.10) este soluția optimă a MCNEP.*

Demonstrație. Amintindu-ne că G^* este construit folosind un flux de cost minim f^e de valoare w calculat în rețeaua $G^e = (V^e, A^e, s, t, u^e, c^e)$, înseamnă că f^e este soluția problemei:

$$\begin{cases} \min\{\sum_{a \in A^e} c^e(a) \cdot f(a)\} \\ f \text{ este un flux admisibil de valoare } w \text{ în } G^e \end{cases} \tag{5.11}$$

Prin urmare:

$$\begin{aligned} &\sum_{a \in A^e} c^e(a) f^e(a) = \\ &= \sum_{a \in A} c^e(a) f^e(a) + \sum_{a \in Q} c^e(a) f^e(a) + \sum_{a \in A_1^e} c^e(a) f^e(a) + \sum_{a \in A_2^e} c^e(a) f^e(a) = \\ &= \sum_{a \in Q} c_Q(a) f^e(a) + \sum_{a=(i,j) \in A} c^e(i, i_a) f^e(i, i_a) + \sum_{a=(i,j) \in A} c^e(i_a, j) f^e(i_a, j) = \\ &= \sum_{a \in Q} c_Q(a) f^e(a) + \sum_{a=(i,j) \in A} c(a)/2(f^e(i, i_a) + f^e(i_a, j)) = \\ &= \sum_{a \in Q} c_Q(a) f^e(a) + \sum_{a=(i,j) \in A} c(a) f^e(i, i_a) = \\ &= \sum_{a \in Q} c_Q(a) u^*(a) + \sum_{a \in A} c(a)(u^*(a) - u(a)). \end{aligned} \tag{5.12}$$

Folosind Ecuația (5.11) și Ecuația (5.12) obținem următoarea problemă de optimizare:

$$\min\{\sum_{a \in A} c(a) \cdot (u^*(a) - u(a)) + \sum_{a \in Q} c_Q(a) \cdot u^*(a)\} \tag{5.13}$$

Folosind Ecuația (5.10), avem:

$$0 \leq u^*(a) - u(a) = f^e(i, i_a) \leq \alpha(a), \forall a \in A \tag{5.14}$$

$$0 < u^*(a) = f^e(a) \leq \beta(a), \forall a \in Q \text{ și } f^e(a) > 0 \tag{5.15}$$

$$0 = u^*(a) \leq \beta(a), \forall a \in Q, \text{ și, suntem de acord că, } f^e(a) = 0. \tag{5.16}$$

Considerăm următorul flux notat f^* în G^*:

$$f^*(a) = \begin{cases} f^e(a) + f^e(i, i_a), \text{dacă } a \in A \\ f^e(a), \text{dacă } a \in A^* - A \end{cases} \tag{5.17}$$

Deoarece f^e este un flux admisibil în G^e, putem concluziona că f^* satisface condițiile de conservare în G^*, și din Ecuația (5.10) rezultă că f^* respectă condițiile de limită în G^*. Aceasta înseamnă că f^* este un flux admisibil în G^*. Folosind Ecuațiile (5.13-5.16) putem concluziona că u^* este soluția optimă a Ecuației (5.1).

□

Corolar 5.1. *Costul expansiunii rețelei de la G la soluția optimă G^* a MCNEP este $v(f^e)$, unde f^e este fluxul de cost minim în G^e.*

Demonstrație. Rezultatul este imediat din Ecuația (5.12). □

Folosind Teorema 3, Teorema 4, și Teorema 5, următorul algoritm (Algoritmul 8) pentru rezolvarea MCNEP este dezvoltat.

Teorema 6. *Complexitatea algoritmului 8 (AMCNEP) este*

$$O((m+q)^2 \cdot \log n + (m+q) \cdot m \cdot \log^2 n)$$

unde n este numărul de noduri din G, m este numărul de arce ale rețelei G, și q numărul de arce din Q (care pot fi adăugate în rețeaua G).

Demonstrație. Algoritmul 8 trebuie să calculeze o dată fluxul maxim în G și o dată fluxul maxim în G''. Cel mai eficient algoritm de astăzi pentru problema fluxului maxim este cel propus de Orlin [15]. Acesta are o complexitate de timp de $O(m \cdot n)$ când este aplicat în rețeaua originală G. În G'' sunt tot n noduri, dar $m + q$ arce. Prin urmare, în acest caz, fluxul maxim poate fi calculat în $O((m+q) \cdot n)$.

În G^e trebuie să calculăm fluxul de cost minim. Fluxul de cost minim este considerat a funcționa pe rețele cu capacități de valori întregi [1]. Deoarece în G^e costul arcelor din $A_1^e \cup A_2^e$ sunt valori întregi împărțite la 2 (vezi Ecuația (5.7), înainte de a proceda la calculul fluxului de cost minim, toate costurile arcelor din A^e sunt înmulțite cu 2, pentru a asigura că au valori întregi, și, la final, costul total al fluxului obținut f^e va fi împărțit la 2. Cel mai eficient algoritm cunoscut astăzi pentru calcularea fluxului de cost minim este tot datorat lui Orlin [14]. Deoarece rețeaua

Algoritm 8 Algoritm pentru rezolvarea MCNEP (AMCNEP)

Input: $G = (V, A, s, t, u)$, c, α, β, Q, c_Q, w

Output: $G^* = (V, A^*, s, t, u^*)$

1: Găsiți un flux maxim f^* în G
2: **if** $v(f^*) \geq w$ **then**
3: Nu este necesar să modificați G, deoarece poate acomoda un flux admisibil de valoare w.
4: **return**
5: **end if**
6: Construiți rețeaua G'' folosind Ecuația (5.2)
7: Găsiți un flux maxim g^* în G''
8: **if** $v(g^*) < w$ **then**
9: MCNEP nu este admisibil
10: **return**
11: **end if**
12: Construiți rețeaua $G^e = (V^e, A^e, s, t, u^e, c^e)$ folosind Ecuațiile (5.3) - (5.7)
13: Găsiți un flux de cost minim f^e în G^e
14: Construiți rețeaua $G^* = (V, A^*, s, t, u^*)$ folosind f^e, Ecuațiile (5.9) și (5.10)
15: Rețeaua G^* este soluția optimă pentru MCNEP

G^e are $m+n$ noduri și $3m+q$ arce, dacă algoritmul este aplicat în G^e, atunci are o complexitate de timp de $O((m+q)^2 \cdot \log n + (m+q) \cdot m \cdot \log^2 n)$.

Prin urmare, complexitatea totală a algoritmului propus este $O((m+q)^2 \cdot \log n + (m+q) \cdot m \cdot \log^2 n)$. □

5.3 Concluzii

Acest capitol a investigat în detaliu Problema Expansiunii Rețelei de Cost Minim (MCNEP), o problemă importantă care apare atunci când rețelele existente trebuie să fie adaptate pentru a susține un flux crescut de la nodurile sursă la cele de destinație. Problema este complexă și implică fie creșterea capacității de transport a arcelor existente, fie integrarea de arce noi în rețea. Un aspect cheie al MCNEP este optimizarea costurilor asociate acestor modificări. Pentru a aborda această provocare complexă, a fost dezvoltat și prezentat un algoritm puternic polinomial (Algoritmul 8), care oferă o abordare sistematică pentru obținerea unei expansiuni cost-eficiente a rețelei.

O direcție interesantă pentru cercetările viitoare ar putea fi generalizarea MCNEP. Studiile viitoare ar putea explora scenarii în care costurile asociate schimbării capacităților arcelor și introducerea de arce noi nu urmează un model liniar. Această extindere a cadrului problemei ar putea include funcții de cost care sunt poate polinomiale, exponențiale sau bazate pe alte modele neliniare care ar putea reflecta mai precis costurile reale ale schimbărilor în rețea. Această generalizare ar putea îmbunătăți semnificativ aplicabilitatea și relevanța cadrului MCNEP pentru o gamă mai largă de situații din lumea reală, unde dinamica costurilor este mai complicată și variată.

Capitolul 6

Integrarea Datelor de Teledetecție cu Procesarea Dinamică a Rețelelor

6.1 Introducere

Teledetecția este achiziționarea de informații despre un obiect sau fenomen fără contact fizic direct. Tehnologii precum imaginile multispectrale (MS) și hiperspectrale (HS) au transformat semnificativ metodele utilizate pentru observarea și analizarea suprafeței Pământului. Combinarea acestor tehnici de imagistică cu algoritmii grafurilor dinamice îmbină două tehnologii avansate, oferind soluții inovatoare pentru probleme complexe din lumea reală.

Imaginile MS și HS capturează informații prin detectarea radiațiilor electromagnetice. Principala diferență între ele este rezoluția spectrală: imaginile HS capturează date în mai multe benzi înguste și contigue, în timp ce imaginile MS capturează date în mai puține benzi discrete. Aceste date detaliate permit identificarea și analiza precisă a materialelor, îmbunătățind capacitatea noastră de a monitoriza și gestiona mediile naturale și artificiale. Aplicațiile variază de la monitorizarea mediului și agriculturii la diagnosticarea medicală și planificarea urbană.

Rețelele dinamice și algoritmii asociați sunt esențiali pentru problemele în care structura rețelei subiacente se schimbă în timp. Ele pot fi un instrument important în numeroase aplicații de mediu sau scenarii de planificare urbană.

Imaginile MS și HS oferă date de mediu de mare acuratețe, care pot fi utilizate pentru construirea rețelelor dinamice. Această integrare permite identificarea precisă a nodurilor și muchiilor și atribuirea greutăților și capacităților corespunzătoare. Prin urmare, combinarea aces-

tor tehnici de imagistică cu algoritmii rețelelor dinamice deschide noi posibilități pentru abordarea problemelor complexe cu dimensiuni temporale și spațiale. De exemplu, datele spectrale în monitorizarea mediului pot ajuta la maparea regiunilor și a condițiilor acestora într-o rețea dinamică, permițând ajustări în timp real bazate pe schimbările actuale ale mediului.

Capitolul prezent se bazează pe cercetări anterioare privind procesarea imaginilor MS și HS [17], [16], [18]. Obiectivul este de a combina aceste tehnici sofisticate de imagistică cu teoria grafurilor dinamice pentru a spori capabilitățile și aplicațiile ambelor tehnologii, abordând eficient provocările din lumea reală.

6.2 Fundamentele Imaginilor MS și HS

Percepția obiectelor de către ochiul uman este rezultatul reflexiei luminii de la obiectele în cauză. Această lumină reflectată ajunge la ochi, unde este procesată de creier pentru a crea o imagine vizuală. Lumina este o formă de radiație electromagnetică, care se propagă atât ca unde electrice, cât și magnetice, în pachete de energie cunoscute sub numele de fotoni.

Spectrul electromagnetic este o gamă continuă de radiații electromagnetice, clasificată după lungimea de undă. Este împărțit în benzi, fiecare desemnată pentru tipuri specifice de radiații în intervale de lungime de undă. Spectrul electromagnetic cuprinde o gamă vastă de radiații electromagnetice, cu lungimi de undă variind de la raze gamma foarte scurte până la unde radio foarte lungi (vezi Figura 6.1). Fiecare bandă a spectrului servește unor aplicații științifice, tehnologice și practice diferite.

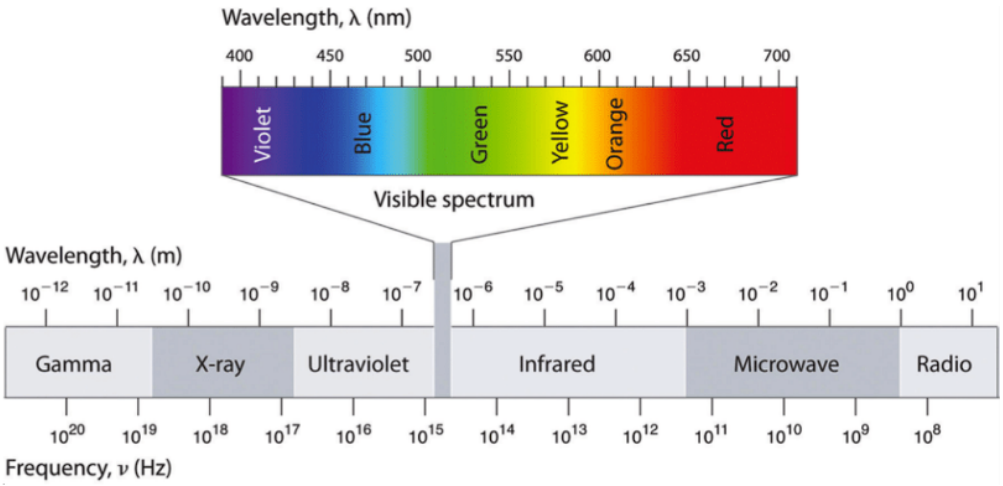


Figura 6.1: Ilustrarea spectrului electromagnetic. Imaginea este oferită de [25]

Semnificația benzilor spectrale constă în faptul că materialele reflectă și absorb lumina

în moduri unice pe tot parcursul spectrului, permițând identificarea lor pe baza semnăturilor spectrale. Reflexia luminii în diferite benzi spectrale permite diferențierea stării materialului, sănătății vegetative și compoziției suprafeței.

Principiile spectrului electromagnetic stau la baza tehnologiilor de imagistică MS și HS. Aceste tehnologii utilizează benzi de lungime de undă specifice pentru a captura informații detaliate dincolo de imagistica convențională în lumină vizibilă, oferind perspective semnificative într-o gamă largă de aplicații.

Imagistica MS este definită ca o tehnologie de imagistică care utilizează caracteristicile și proprietățile spectrale ale luminii pentru a captura date în mai multe benzi discrete de lungime de undă, de obicei până la 20, pe întreg spectrul electromagnetic. Spre deosebire de abordarea tradițională RGB, care utilizează trei benzi largi, imagistica MS folosește benzi mai înguste și mai numeroase, extinzându-se de la ultraviolet la regiuni apropiate de infraroșu. Această abordare permite analiza detaliată a materialelor, oferind date de reflectanță calibrate, esențiale pentru aplicații precum agricultura, monitorizarea mediului și diagnosticul medical.

Imagistica HS implică achiziționarea de date pe un spectru continuu, înregistrând sute de benzi înguste și contigue pentru fiecare pixel din imagine. Această rezoluție spectrală înaltă permite detectarea chiar și a diferențelor subtile în semnăturile spectrale, făcând din imagistica HS o alegere optimă pentru o gamă de aplicații care necesită informații spectrale detaliate, inclusiv teledetecția, diagnosticul medical și analiza calității alimentelor. Deși mai complexă și consumatoare de timp decât imaginile MS, imaginile HS oferă profiluri spectrale complete pentru fiecare pixel, oferind o înțelegere mai profundă a materialelor studiate.

Aplicațiile imagisticii MS și HS sunt extinse și diverse, având numeroase aplicații practice în diverse domenii. Acestea includ agricultura, cercetarea de mediu, gestionarea dezastrelor și multe altele. În timp ce imagistica HS este extrem de sensibilă la condițiile de mediu și necesită o calibrare riguroasă, fiind cea mai potrivită pentru condiții controlate sau studii științifice specifice, este deosebit de valoroasă în geologie și dezvoltarea mineralelor pentru colectarea de informații detaliate despre materiale. Prin contrast, imagistica MS este relativ mai puțin constrânsă de vicisitudinile mediului și interferențele atmosferice; este astfel mai potrivită pentru un spectru mai larg de medii și aplicații. Utilizarea imaginilor MS în agricultură și silvicultură este extinsă. Aplicațiile sale includ colectarea de date despre suprafața și acoperirea Pământului, precum și modelele de schimbare observate în acestea. Imagistica MS este o opțiune mai accesibilă și mai practică pentru cerințele generale de teledetecție, majoritatea sateliților

și constelațiilor oferind date MS. Prin urmare, deși imagistica HS are un potențial considerabil neexploatat, imagistica MS este în prezent suficientă pentru nevoile utilizatorilor ocazionali în diverse aplicații de teledetecție.

6.3 Monitorizarea Agricolă în Timp Real Utilizând Rețele Dinamice

Integrarea rețelelor dinamice cu datele de teledetecție reprezintă un avans semnificativ în seria de activități legate de monitorizarea Pământului și gestionarea mediului. Pe baza capacităților tehnologiilor de teledetecție, cum ar fi imaginile MS și HS, în combinație cu procesarea graficelor dinamice, este posibil să se obțină perspective mai precise și mai oportune asupra diverselor fenomene naturale și create de om. Această teză se concentrează pe un studiu de caz privind utilizarea tehnicilor de teledetecție în sectorul agricol. Această alegere este motivată de caracteristicile particulare ale setului de date disponibil, care cuprinde un set de imagini MS predominant de teren agricol cu o componentă urbană.

Având în vedere cererea globală în creștere pentru resurse agricole, agricultura reprezintă un domeniu important de studiu și cercetare. Această cerere se extinde la producția de alimente, biocombustibili și alte produse agricole esențiale care susțin umanitatea. Odată cu creșterea populației, necesitatea unor practici agricole mai eficiente și sustenabile se intensifică. În plus, semnificația economică a agriculturii în numeroase regiuni subliniază necesitatea unor strategii inovatoare care pot îmbunătăți productivitatea și sustenabilitatea.

Este de o importanță semnificativă să recunoaștem variabilitatea considerabilă a proprietăților solului care caracterizează câmpurile agricole. O astfel de variabilitate poate exercita o influență considerabilă asupra sănătății și randamentului culturilor. Caracteristicile solului, inclusiv conținutul de nutrienți, pH-ul, nivelurile de umiditate și activitatea microbiană, pot varia în diferite părți ale unui câmp. Prin urmare, aplicarea uniformă a îngrășămintelor și pesticidelor poate duce la o serie de efecte adverse:

- S-a demonstrat că utilizarea utilajelor grele pe terenurile agricole poate duce la compactarea solului, ceea ce îi perturbă structura. Acest proces reduce porozitatea solului până la un nivel care limitează circulația aerului și infiltrarea apei. Acești factori sunt esențiali pentru o creștere sănătoasă a rădăcinilor și absorbția nutrienților de către culturi.

- Solul găzduiește diverse microorganisme care joacă un rol activ în ciclul nutrienților și în descompunerea materiei organice. Utilizarea excesivă a chimicalelor și perturbarea mecanică pot dăuna acestor comunități microbiene, reducând fertilitatea și sănătatea solului.

Având în vedere acești factori, se poate concluziona că intervențiile localizate sunt mai benefice decât aplicările uniforme de îngrășăminte sau pesticide. Aplicarea țintită a îngrășămintelor și pesticidelor minimizează perturbarea solului și păstrează activitatea microbiană naturală esențială pentru agricultura sustenabilă. Mai mult, prin aplicarea îngrășămintelor și pesticidelor doar acolo unde este necesar, fermierii pot reduce semnificativ utilizarea generală a chimicalelor, reducând riscul de contaminare a mediului și păstrând biodiversitatea locală. Această abordare ajută la protejarea speciilor din lanțul trofic de chimicalele nocive. Minimizarea utilizării utilajelor grele și a agenților chimici servește, de asemenea, la reducerea amprentei de carbon asociate operațiunilor agricole.

Modelul rețelei dinamice prezentat în acest capitol al tezei, integrat cu procesarea în timp real a imaginilor multispectrale, permite monitorizarea și gestionarea precisă a câmpurilor agricole. Pentru implementarea acestui model, folosim un Cadru de Monitorizare Dinamică a Vegetației (DVMF). Acest cadru implică mai mulți pași cheie:

1. **Achiziționarea de date în timp real**. Achiziționarea imaginilor multispectrale în timp real permite identificarea și monitorizarea la timp a oricăror schimbări în starea vegetației și condițiile solului. Datele în timp real sunt extrem de importante pentru construirea precisă și actualizată a rețelei dinamice.

2. **Construirea graficului dinamic**. Construirea unui grafic dinamic este un alt pas cheie în acest proces. În acest grafic, nodurile reprezintă zonele vegetale și muchiile denotă relațiile, cum ar fi proximitatea sau sistemele de irigare comune. Această etapă cuprinde construirea preliminară a graficului și actualizarea sa continuă ulterioară, după cum va fi detaliat în algoritm.

3. **Intervenție localizată**. Procesarea dinamică a rețelei permite fermierilor să identifice zonele specifice care necesită intervenții, fie pentru irigare, fertilizare sau controlul dăunătorilor. Acest lucru promovează utilizarea eficientă a resurselor și minimizează impactul asupra mediului. Mecanismele de detectare a schimbărilor și alertele cadrului asigură că intervențiile sunt la timp și țintite.

Secțiunile următoare vor prezenta un cont detaliat al setului de date și al cadrului, însoțit de ilustrații pas cu pas ale exemplelor generate utilizând setul de date disponibil și un set de indici de vegetație pentru măsurarea sănătății vegetației, împreună cu o metrică specifică pentru măsurarea schimbărilor nedorite în starea vegetației. Aceste secțiuni vor sublinia aplicarea practică a algoritmului, demonstrând avantajele tangibile și implementarea acestei abordări.

În plus, măsurătorile disponibile in-situ integrate cu rețeaua dinamică ar crește precizia proceselor de monitorizare și de luare a deciziilor, conducând în final la rezultate agricole mai favorabile. Această abordare susține obiectivul agriculturii sustenabile prin asigurarea că intervențiile sunt țintite și bazate pe date în timp real, reducând astfel risipa și minimizând efectele adverse asupra mediului.

6.3.1 Cadrul de Monitorizare Dinamică a Vegetației

Cadrul de Monitorizare Dinamică a Vegetației este conceput pentru a procesa imagini multispectrale în timp real și pentru a menține graficul dinamic necesar pentru monitorizarea eficientă și intervențiile localizate. Acest cadru poate fi utilizat pentru orice tip de monitorizare a schimbărilor în starea vegetației. În funcție de caracteristicile vegetației, alte măsurători (indici de vegetație) pot necesita să fie calculate și utilizate, iar intervențiile țintite trebuie personalizate. Mai jos sunt pașii detaliați ai cadrului:

Pașii și procesele detaliate implicate în Cadrul de Monitorizare Dinamică a Vegetației (DVMF) sunt ilustrate în Figura 6.2. Această figură oferă o reprezentare vizuală a cadrului, evidențiind secvența operațiunilor de la achiziția inițială de date până la bucla de monitorizare în timp real, inclusiv segmentarea, analiza și răspunsul la alerte.

Algoritm 9 Cadrul de Monitorizare Dinamică a Vegetației (DVMF) - Partea 1

Input: Imagini multispectrale în timp real, un set de indici de vegetație utilizați ca măsură a sănătății vegetației
Output: Graficul dinamic inițial al monitorizării vegetației
Pasul 1: Achiziția Inițială de Date
imagine ← `AcquireAndPreprocessImage`()
indici ← `ComputeVegetationIndices`(imagine)
Pasul 2: Inițializarea Graficului
zone ← `IdentifyRelevantZones`(indici)
for fiecare zonă în zone **do**
    Creează un nod cu atribute: {set de indici de vegetație corespunzători, informații spațiale (coordonate pixel sau dimensiune segment)}
end for
grafic ← `ConstructInitialGraph`(zone)
`EstablishEdges`(grafic)

Algoritm 10 IdentifyRelevantZones

Input: Indici de vegetație
Output: Zone cu vegetație relevantă
Identifică și delimitează zonele din zona mare de interes care au vegetație relevantă pentru monitorizare utilizând un prag pe indexul de vegetație relevant. Aceste zone pot reprezenta pixeli individuali sau zone agregate (segmente) cu caracteristici vegetale similare.
Return zone

Algoritm 11 ConstructInitialGraph

Input: Zone cu vegetație relevantă
Output: Graficul dinamic inițial
Construiește graficul inițial cu noduri reprezentând zone vegetate (fie pixeli, fie zone agregate) care urmează să fie monitorizate
Return grafic

Algoritm 12 EstablishEdges

Input: Grafic
Stabilește muchii între nodurile adiacente (pixeli sau zone) pe baza informațiilor disponibile, cum ar fi relațiile spațiale, relațiile de similaritate, sistemele de irigare comune, tipurile și calitatea solului etc.

Algoritm 13 Cadrul de Monitorizare Dinamică a Vegetației (DVMF)

Input: Imagini multispectrale în timp real, indici de vegetație și criterii pentru schimbări
Output: Grafic dinamic actualizat cu alerte
Pasul 3: Bucla de Monitorizare în Timp Real
while o nouă imagine multispectrală este disponibilă **do**
    imagine ← `AcquireAndPreprocessImage`()
    indici ← `ComputeVegetationIndices`(imagine)
    grafic ← `CopyOldGraph`()
    `UpdateNodes`(grafic, indici)
    `UpdateGraph`(grafic, indici)
    `SegmentAndAnalyze`(grafic)
    `VisualizeAndReport`(grafic)
    `RespondToAlerts`()
    `LogChanges`()
end while

Algoritm 14 AcquireAndPreprocessImage

Output: Imagine multispectrală preprocesată
Achiziționează imaginea multispectrală
Efectuează georeferențiere, mascare nori și corecție atmosferică
Return imagine preprocesată

Algoritm 15 ComputeVegetationIndices

Input: Imagine multispectrală preprocesată
Output: Indici de vegetație
Calculează indicii de vegetație din imagine (de exemplu, NDVI, NDWI)
Return indici

Algoritm 16 CopyOldGraph

Output: Copie a vechiului grafic
Copiază starea anterioară a graficului dinamic
Return grafic copiat

Algoritm 17 UpdateNodes

Input: Grafic, Indici de vegetație
for fiecare nod în grafic **do**
    Actualizează indicii de vegetație
    if schimbarea indică stres vegetativ **then**
        Marchează nodul pentru analiza alertelor
    else
        Elimină nodul
    end if
end for

Algoritm 18 UpdateGraph

Input: Grafic, Indici de vegetație
for fiecare punct din zona de interes și care nu se află în vechiul grafic **do**
    Calculează indicii de vegetație
    if indexul este peste prag **then**
        Adaugă punctul ca nod nou
    end if
end for

Algoritm 19 SegmentAndAnalyze

Input: Grafic
Segmentează nodurile marcate pentru analiza alertelor pe baza caracteristicilor comune (de exemplu, indici similari, proximitate) pentru a forma zone mai mari
for fiecare zonă **do**
    if zona prezintă o schimbare semnificativă nedorită **then**
        Declanșează o alertă
    end if
end for

Algoritm 20 VisualizeAndReport

Input: Grafic
Generează vizualizări ale graficului dinamic, evidențiind nodurile și zonele cu alerte
Creează un raport al nodurilor și zonelor cu alerte și acțiuni recomandate

Algoritm 21 RespondToAlerts

1: **for** fiecare alertă din raport **do**

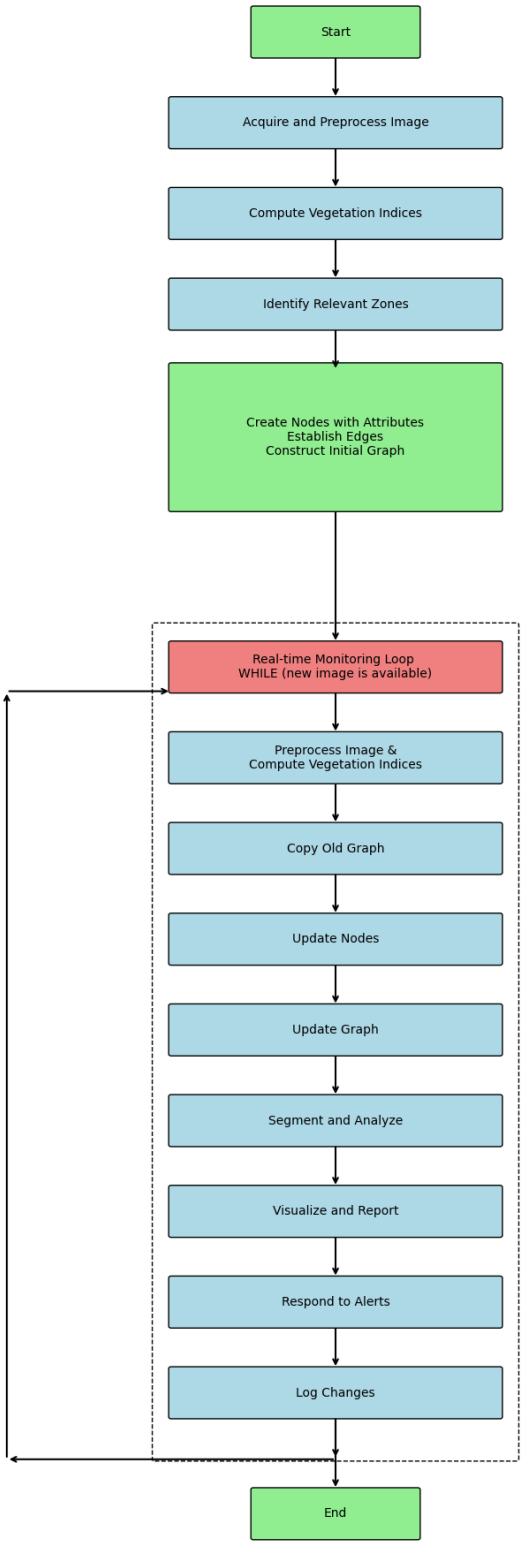
2: Răspunde cu acțiuni specifice (de exemplu, ajustarea irigației, aplicarea îngrășămintelor)

3: Monitorizează eficacitatea intervenției și înregistrează schimbările pentru analiza tendințelor

4: **end for**

Algoritm 22 LogChanges

1: Înregistrează schimbările în timp pentru analiză și predicție



Descrierea indicilor de vegetație

Pentru a oferi o perspectivă suplimentară asupra stării vegetației și pentru a facilita monitorizarea acestui parametru în timp, au fost calculați mai mulți indici de vegetație din setul de date spectral pentru exemplificarea noastră. Acești indici simplifică datele spectrale complexe în metrici interpretabile, permițând identificarea unor atribute specifice ale vegetației, cum ar fi sănătatea, conținutul de apă și nivelurile de stres. Următoarele paragrafe prezintă o prezentare generală a indicilor de vegetație utilizați în acest studiu, însoțită de o discuție a relevanței și semnificației lor.

Indicele de Vegetație Normalizat Diferențiat (NDVI), Indicele de Umiditate Normalizat Diferențiat (NDWI) și Indicele de Stres al Umidității (MSI) se numără printre cei mai importanți indici de vegetație utilizați în cercetările agricole și de mediu. Cu toate acestea, există o serie de alți indici care pot fi utilizați, în funcție de circumstanțele specifice.

Indicele de Vegetație Normalizat Diferențiat (NDVI), un indice de vegetație teledetecție utilizat pe scară largă, cuantifică verdele vegetației și servește ca un indicator important al sănătății și biomasei plantelor. Este utilizat în monitorizarea practicilor agricole, monitorizarea mediului și clasificarea acoperirii terenului. NDVI măsoară sănătatea plantelor pe baza reflexiei diferențiale a luminii roșii și infraroșu apropiat, vegetația sănătoasă reflectând mai mult IR apropiat și mai puțină lumină roșie, rezultând valori NDVI mai mari. NDVI este calculat folosind următoarea formulă:

$$NDVI = \frac{NIR - Rou}{NIR + Rou} \tag{6.1}$$

Acest index are valori cuprinse între -1 și 1, valorile mai mari, peste 0,3, indicând zone acoperite cu vegetație verde sănătoasă. Valorile apropiate de zero indică zone cu vegetație rară sau nesănătoasă, în timp ce valorile mai mici de 0 corespund de obicei zonelor fără vegetație, cum ar fi suprafețele de apă, solul gol sau zonele construite.

Un alt indice de vegetație important utilizat în teledetecție este Indicele de Umiditate Normalizat Diferențiat (NDWI). NDWI măsoară conținutul de apă din plante și are valori tot de la -1 la 1. Valorile mai mari indică un conținut mai mare de apă. Este un instrument util pentru evaluarea stării apei în plante și pentru detectarea condițiilor de secetă. NDWI este calculat folosind următoarea formulă:

$$NDWI = \frac{NIR - SWIR}{NIR + SWIR} \tag{6.2}$$

În timp ce NDVI este un indice al verdeții vegetației, NDWI este un indice al conținutului de apă

pentru vegetație. Cei doi indici se completează reciproc pentru a oferi o evaluare cuprinzătoare a sănătății vegetației și a stării apei.

Indicele de Stres al Umidității (MSI) este, de asemenea, un indicator al stresului hidric al plantelor și este calculat folosind următoarea formulă:

$$MSI = \frac{SWIR}{NIR} \tag{6.3}$$

Valorile mai mari indică niveluri mai ridicate de stres hidric. Pentru majoritatea vegetației sănătoase, valorile MSI pot fi mai mici de 1,0, în timp ce valorile între 1,0 și 1,5 corespund vegetației moderat stresate, iar valorile mai mari de 1,5 indică de obicei stres hidric semnificativ.

Atât indicele de apă normalizat diferențiat (NDWI), cât și indicii MSI oferă informații despre conținutul de apă și stresul din vegetație. Cu toate acestea, ei fac acest lucru din perspective ușor diferite și se pot completa reciproc în mod eficient. Secțiunea următoare va explica de ce utilizarea ambilor indici poate fi benefică. MSI este deosebit de sensibil la schimbările în conținutul de umiditate al vegetației și este utilizat pentru a detecta condițiile de stres hidric. Acesta facilitează identificarea regiunilor în care plantele pot fi supuse secetei sau unui aport de apă inadecvat. NDWI se concentrează pe conținutul de apă din coronamentele vegetației, luând în considerare faptul că plantele pot suferi și din cauza excesului de apă, o condiție cunoscută sub numele de supraudare. Acest lucru poate fi dăunător pentru sănătatea plantelor și poate duce la complicații. Prin urmare, NDWI oferă o măsură a cantității de apă prezente în vegetație, ceea ce poate fi esențial pentru evaluarea sănătății generale a plantelor.

6.4 Concluzii

Acest capitol demonstrează potențialul semnificativ al integrării rețelelor dinamice cu datele de teledetecție pentru avansarea practicilor agricole către metode mai sustenabile și eficiente. Prin utilizarea datelor în timp real și a intervențiilor țintite, fermierii pot îmbunătăți sănătatea culturilor, proteja mediul și reduce amprenta de carbon, contribuind în final la un viitor mai sustenabil. Această integrare susține nu numai productivitatea agricolă, ci și preocupările ecologice mai largi, asigurând că practicile agricole sunt în armonie cu sustenabilitatea ecologică.

Capitolul 7

Concluzii

Această teză se bazează pe obiectivele detaliate prezentate în introducere prin investigarea algoritmilor existenți și dezvoltarea unora noi. Contribuțiile principale sunt rezumate mai jos.

7.1 Contribuții ale cercetării

Contribuțiile acestei cercetări sunt diverse și abordează unele lacune din domeniul algoritmilor de rețea dinamică.

1. Capitolele de **Fundamentare Teoretică** și **Revizuire a Literaturii de Specialitate** au oferit o analiză detaliată a algoritmilor existenți pentru problemele SSSP și flux maxim, în situații statice și dinamice, din punct de vedere al metodologiei și eficienței, identificând punctele lor forte și limitările.

2. În capitolul **Ajustarea Dinamică a Drumului Cel Mai Scurt cu O Singură Sursă** am dezvoltat un algoritm eficient care abordează problema SSSP și se concentrează doar pe părțile afectate ale rețelei, asigurând performanțe optime în scenarii dinamice.

3. În capitolul **Creșterea Fluxului prin Expansiunea Rețelei** am introdus un algoritm pentru rezolvarea MCNEP, îmbunătățind capacitățile de flux ale rețelei cu costuri minime, aplicabil în diverse scenarii practice care necesită augmentarea dinamică a fluxului.

4. În capitolul **Integrarea Datelor de Teledetecție cu Procesarea Dinamică a Rețelelor** am propus o metodă de combinare a datelor de teledetecție cu rețele dinamice, oferind o

soluție practică pentru monitorizarea în timp real a mediului și agriculturii.

7.2 Limitări

În ciuda acestor contribuții, cercetarea prezintă câteva limitări:

1. **Validarea Algoritmului**: Deși algoritmul dinamic SSSP este teoretic solid, necesită validare suplimentară prin testări ample în situații reale, diverse scenarii de rețea.

2. **Generalizarea MCNEP**: Algoritmul propus pentru MCNEP trebuie extins pentru a reflecta mai bine situațiile reale și costurile modificărilor de rețea. Ar trebui să investigheze unele scenarii de expansiune în care funcțiile de cost sunt neliniare, cum ar fi cele polinomiale sau exponențiale.

3. **Provocări în Integrare**: Integrarea datelor de teledetecție cu rețelele dinamice, deși o direcție de cercetare promițătoare, nu este lipsită de provocări. La momentul redactării, în afară de imaginile multispectrale, nu am avut acces la date reale și in situ. Incorporarea unor astfel de date ar fi avantajoasă pentru validarea cazurilor de utilizare în aceste scenarii, precum și pentru creșterea acurateței și aplicabilității soluțiilor propuse. Mai mult, inputul experților ar fi avantajos, deoarece considerațiile specifice pot varia în funcție de situația particulară, cum ar fi monitorizarea agricolă sau de mediu. Astfel de perspective ale experților ar putea ajuta la abordarea aspectelor specifice care necesită atenție pentru o aplicare mai eficientă.

7.3 Lucrări viitoare

În lumina constatărilor și limitărilor acestei cercetări, sunt prezentate câteva propuneri pentru lucrări viitoare.

1. **Validare Extinsă**: Testarea amplă în condiții reale a algoritmilor propuși ar fi benefică pentru a asigura robustețea și scalabilitatea acestora în diverse condiții și dimensiuni ale rețelei.

2. **Îmbunătățirea Algoritmilor**: Perfecționarea suplimentară a algoritmilor pentru a gestiona dinamici de rețea mai complexe și pentru a îmbunătăți eficiența computațională. În

special, testarea empirică a parametrului delta și investigarea celor mai bune structuri de date care pot fi utilizate în algoritmul dinamic SSSP ar putea oferi informații valoroase despre setările și performanța sa optimă.

3. **Generalizarea MCNEP**: O direcție interesantă pentru cercetările viitoare ar putea fi generalizarea MCNEP.

4. **Paralelizare**: Ar fi benefic să se investigheze potențialul de paralelizare a algoritmilor dinamici, în vederea creșterii eficienței computaționale și scalabilității acestora. Acest lucru ar putea facilita adecvarea lor pentru aplicații în timp real în rețele de mari dimensiuni.

5. **Aplicații Mai Largi**: Extinderea integrării datelor de teledetecție cu rețelele dinamice în alte domenii, cum ar fi planificarea urbană și gestionarea dezastrelor, unde datele în timp real pot avea un impact semnificativ asupra proceselor decizionale.

În concluzie, această teză a adus unele contribuții la avansarea înțelegerii și dezvoltării algoritmilor dinamici pentru probleme de rețea. Deși există provocări și limitări, soluțiile propuse oferă o bază solidă pentru cercetările viitoare și aplicațiile practice, contribuind la evoluția continuă a algoritmilor de rețea și integrarea acestora cu datele din lumea reală.

Bibliografie

[1] Ravindra K Ahuja, Thomas L Magnanti și James B Orlin, *Network flows: theory, algorithms and applications*, Prentice hall, 1995.

[2] Avishai Ceder, *Public transit planning and operation: Modeling, practice and behavior*, CRC press, 2016.

[3] Laura Ciupala, Adrian Deaconu și Luciana Majercsik, „Shortest paths in a diagraph with an underestimated arc weight", în *Bulletin of the Transilvania University of Brasov. Series III: Mathematics and Computer Science* (2022), pp. 193–196.

[4] Adrian Deaconu, „The inverse maximum flow problem considering l∞ norm", în *RAIRO-Operations Research* 42.3 (2008), pp. 401–414.

[5] Adrian Deaconu și Eleonor Ciurea, „The inverse maximum flow problem under norms", în *Carpathian Journal of Mathematics* (2012), pp. 59–66.

[6] Adrian Marius Deaconu și Luciana Majercsik, „Flow Increment through Network Expansion", în *Mathematics* 9.18 (2021), p. 2308.

[7] Amir Elalouf, Ron Adany și Avishai Avi Ceder, „Flow expansion on transportation networks with budget constraints", în *Procedia-Social and Behavioral Sciences* 54 (2012), pp. 1168–1175.

[8] Daniele Frigioni, Alberto Marchetti-Spaccamela și Umberto Nanni, „Fully dynamic algorithms for maintaining shortest paths trees", în *Journal of Algorithms* 34.2 (2000), pp. 251–281.

[9] T Karpagam et al., „Flow Based Algorithm", în *American Journal of Applied Sciences* 9.2 (2012), p. 238.

[10] Myint Than Kyi și Lin Lin Naing, „Application of Ford-Fulkerson algorithm to maximum flow in water distribution pipeline network", în *International Journal of Scientific and Research Publications* 8.12 (2018), pp. 306–310.

[11] Ulrich Meyer și Peter Sanders, „Δ-stepping: a parallelizable shortest path algorithm", în *Journal of Algorithms* 49.1 (2003), pp. 114–152.

[12] Paolo Narvaez, Kai-Yeung Siu și Hong-Yi Tzeng, „New dynamic algorithms for shortest path tree computation", în *IEEE/ACM Transactions On Networking* 8.6 (2000), pp. 734–746.

[13] Paolo Narvaez, Kai-Yeung Siu și Hong-Yi Tzeng, „New dynamic SPT algorithm based on a ball-and-string model", în *IEEE/ACM transactions on networking* 9.6 (2001), pp. 706–718.

[14] James Orlin, „A faster strongly polynomial minimum cost flow algorithm", în *Proceedings of the Twentieth annual ACM symposium on Theory of Computing*, 1988, pp. 377–387.

[15] James B. Orlin, „A faster strongly polynomial time algorithm for submodular function minimization", în *Mathematical Programming* 118.2 (2013), pp. 237–251, DOI: 10.1007/s10107-006-0079-7.

[16] Ioana Cristina Plajer, Alexandra Baicoianu și Luciana Majercsik, „AI-based visualization of remotely-sensed spectral images", în *2023 International Symposium on Signals, Circuits and Systems (ISSCS)*, IEEE, 2023, pp. 1–4.

[17] Ioana Cristina Plajer et al., „NDVI Computation from Hyperspectral Images", în *2023 13th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, IEEE, 2023, pp. 1–5.

[18] Ioana Cristina Plajer et al., „Multisource Remote Sensing Data Visualization Using Machine Learning", în *IEEE Transactions on Geoscience and Remote Sensing* (2024).

[19] Ganesan Ramalingam și Thomas Reps, „An incremental algorithm for a generalization of the shortest-path problem", în *Journal of Algorithms* 21.2 (1996), pp. 267–305.

[20] Alexander Schrijver, „On the history of the transportation and maximum flow problems", în *Mathematical programming* 91 (2002), pp. 437–445.

[21] Javad Tayyebi și Adrian Deaconu, „Inverse generalized maximum flow problems", în *Mathematics* 7.10 (2019), p. 899.

[22] Javad Tayyebi, Abumoslem Mohammadi și Seyyed Mohammad Reza Kazemi, „Reverse maximum flow problem under the weighted Chebyshev distance", în *RAIRO-Operations Research-Recherche Opérationnelle* 52.4-5 (2018), pp. 1107–1121.

[23] Claudemir Duca Vasconcelos et al., „Network flows modeling applied to the natural gas pipeline in Brazil", în *Journal of natural gas science and engineering* 14 (2013), pp. 211–224.

[24] Zhao Zhang și Xiaohui Huang, „Discrete Newton Method", în *Nonlinear Combinatorial Optimization*, ed. de Ding-Zhu Du, Panos M. Pardalos și Zhao Zhang, Cham: Springer International Publishing, 2019, pp. 37–56, ISBN: 978-3-030-16194-1, DOI: 10.1007/978-3-030-16194-1_2, URL: https://doi.org/10.1007/978-3-030-16194-1_2.

[25] Amanda Ziemann, „A manifold learning approach to target detection in high-resolution hyperspectral imagery", Teză de doct., Apr. 2015, DOI: 10.13140/RG.2.1.2503.3680.