Jocuri de Paritate: Teorie, Complexitate și Algoritmi

Constantin Cristiana Georgiana

Universitatea Bucuresti

2025

Cuprins

Ce sunt Jocurile de Paritate?

- Jocurile de paritate sunt jocuri cu două persoane, cu durată infinită, jucate pe un graf dirijat finit
- Reprezintă un model fundamental pentru verificarea formală și sinteza sistemelor reactive
- Sunt echivalente cu problema de verificare a modelelor pentru μ-calculul modal
- Ocupă o poziție unică în teoria complexității: sunt în NP $\cap$ co-NP

Definiția Formală

Un joc de paritate este un tuplu $G = (V, V_0, V_1, E, \Omega)$ unde:

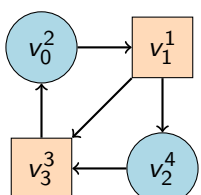
- V este o mulțime finită de noduri (sau poziții)
- V_0 și V_1 formează o partiție a lui V ($V_0 \cap V_1 = \emptyset$ și $V_0 \cup V_1 = V$)
- $E \subseteq V \times V$ este relația de adiacență, unde fiecare nod are cel puțin o muchie ieșitoare
- $\Omega : V \to \mathbb{N}$ este funcția de prioritate care atribuie fiecărui nod un număr natural (prioritate)

Concepte Cheie

- **Jucători**: Doi jucători, de obicei numiți Jucătorul 0 (Par) și Jucătorul 1 (Impar)
- **Poziții**: Nodurile din V_0 sunt pozițiile unde Jucătorul 0 face o mutare, iar nodurile din V_1 sunt pozițiile unde Jucătorul 1 face o mutare
- **Mutări**: O mutare constă în deplasarea unui jeton de la nodul curent la un nod adiacent de-a lungul unei muchii din E
- **Joc**: O secvență infinită de noduri $\pi = v_0, v_1, v_2, \ldots$ astfel încât $(v_i, v_{i+1}) \in E$ pentru toți $i \geq 0$

Condiția de Câștig

- Jucătorul 0 câștigă un joc π dacă cea mai mare prioritate care apare infinit de multe ori în π este pară
- Jucătorul 1 câștigă dacă cea mai mare prioritate care apare infinit de multe ori este impară
- De aici și numele de "jocuri de paritate" - paritatea (par/impar) a celei mai mari priorități recurente determină câștigătorul



Strategii și Regiuni de Câștig

- **Strategie**: O funcție care determină următoarea mutare pe baza istoricului jocului
- **Strategie pozițională**: O strategie care depinde doar de poziția curentă, nu de istoricul jocului
- **Strategie câștigătoare**: O strategie este câștigătoare pentru un jucător dacă toate jocurile conforme cu acea strategie sunt câștigătoare pentru acel jucător
- **Regiune de câștig**: Mulțimea nodurilor din care un jucător are o strategie câștigătoare

Determinare Poziţională

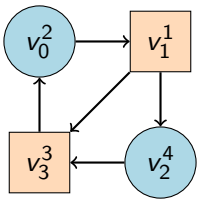
Teorema de Determinare Poziţională

În orice joc de paritate:

1. Din fiecare nod, exact un jucător are o strategie câștigătoare.
2. Dacă un jucător are o strategie câștigătoare dintr-un nod, atunci are o strategie poziţională câștigătoare care depinde doar de nodul curent, nu de istoricul jocului.

Această teoremă, demonstrată de Emerson și Jutla, este crucială pentru soluțiile algoritmice ale jocurilor de paritate, deoarece reduce semnificativ spațiul de căutare pentru strategii câștigătoare.

Exemplu de Joc de Paritate



- Cercuri: nodurile Jucătorului 0
- Dreptunghiuri: nodurile Jucătorului 1
- Numerele în exponent: prioritățile
- Joc optim: $v_0, v_1, v_3, v_0, v_1, v_3, \ldots$
- Prioritatea maximă care apare infinit: 3 (impară)
- Câștigător: Jucătorul 1

Motivația Studierii Jocurilor de Paritate

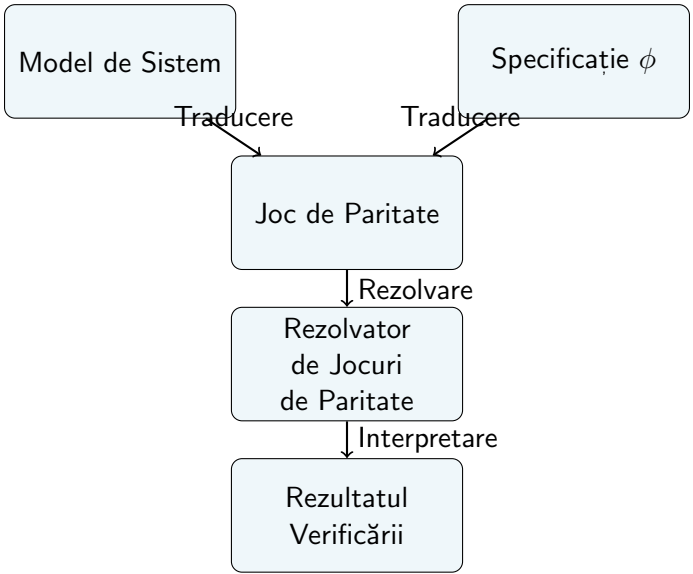
- **Semnificație Teoretică**: Ocupă o poziție unică în teoria complexității (NP $\cap$ co-NP)
- **Conexiunea cu μ-calculul Modal**: Echivalente cu problema verificării modelelor pentru μ-calculul modal
- **Teoria Automatelor**: Strâns legate de automate pe obiecte infinite și teoria jocurilor infinite
- **Aplicații Practice**: Soluțiile se traduc direct în soluții pentru diverse probleme de verificare și sinteză
- **Provocări Algoritmice**: Căutarea de algoritmi eficienți a condus la dezvoltarea de tehnici algoritmice noi

Verificarea Modelelor

Jocurile de paritate sunt fundamentale pentru verificarea modelelor, în special pentru μ-calculul modal:

- **μ-calculul Modal**: Problema verificării modelelor pentru μ-calculul modal poate fi redusă la rezolvarea unui joc de paritate
- **Logici Temporale**: Problemele de verificare a modelelor pentru diverse logici temporale (CTL*, LTL) pot fi traduse în jocuri de paritate
- **Logica Temporală Alternantă (ATL)**: Verificarea modelelor pentru ATL poate fi, de asemenea, redusă la rezolvarea jocurilor de paritate

Fluxul de Verificare a Modelelor



Sinteza Programelor

Jocurile de paritate oferă un cadru pentru sinteza automată a programelor:

- **Sisteme Reactive**: Sinteza controlerelor pentru sisteme reactive care satisfac specificațiile date
- **Corect-prin-Construcție**: Generarea programelor care sunt corecte prin construcție în raport cu specificațiile lor
- **Realizabilitate**: Determinarea dacă o specificație este realizabilă (dacă există un program care o satisface)
- **Sinteza cu Resurse Limitate**: Tehnici pentru sinteza sistemelor cu resurse limitate (de exemplu, memorie)

Exemplu: Sinteza unui Controler pentru Semafoare

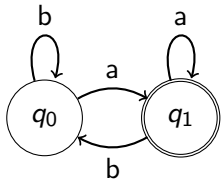
- Problema: Sinteza unui controler pentru un semafor la o intersecție între un drum principal și unul secundar
- Specificații:
  - Proprietăți de siguranță: luminile pentru drumul principal și cel secundar nu pot fi verzi simultan
  - Proprietăți de vivacitate: dacă o mașină așteaptă pe drumul secundar, lumina pentru drumul secundar va deveni eventual verde
- Formulare ca joc de paritate:
  - Jucătorul 0 (controlerul) decide culorile semafoarelor
  - Jucătorul 1 (mediul) decide dacă sosesc mașini pe fiecare drum
  - Prioritățile nodurilor codifică proprietățile de siguranță și vivacitate

Teoria Automatelor

Jocurile de paritate sunt strâns legate de automate pe obiecte infinite:

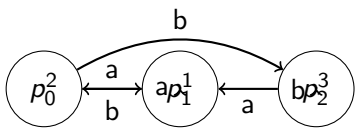
- **Determinizare**: Determinizarea automatelor pe cuvinte și arbori infiniți poate fi abordată folosind jocuri de paritate
- **Complementare**: Construirea automatelor complement pentru automate pe obiecte infinite
- **Verificarea Vidului**: Determinarea dacă un automat de paritate acceptă vreo intrare corespunde rezolvării unui joc de paritate derivat din automat
- **Transformări între Automate**: Diverse transformări între diferite tipuri de automate pe obiecte infinite

Determinizarea Automatelor



Automat Büchi Nedeterminist

$\Rightarrow$



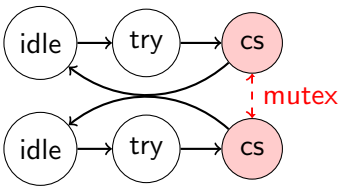
Automat de Paritate Determinist

Verificare Formală

Jocurile de paritate sunt utilizate în instrumente de verificare formală:

- **Verificarea Hardware**: Verificarea că proiectele hardware îndeplinesc specificațiile lor
- **Verificarea Software**: Verificarea că sistemele software satisfac proprietăți de siguranță și vivacitate
- **Verificarea Protocoalelor**: Asigurarea că protocoalele de comunicare funcționează corect și satisfac specificațiile lor
- **Proprietăți de Securitate**: Verificarea proprietăților de securitate precum non-interferența și fluxul de informații

Exemplu: Verificarea unui Protocol de Excludere Mutuală



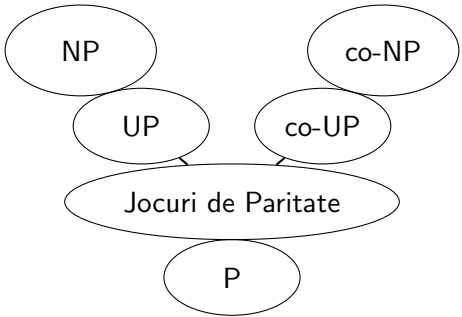
- Problema: Verificarea că un protocol de excludere mutuală asigură că niciun proces nu poate fi în secțiunea sa critică simultan cu altul
- Proprietatea poate fi exprimată în μ-calcul modal: $\nu X.(\neg(cs_1 \wedge cs_2) \wedge \Box X)$
- Se construiește un joc de paritate din model și proprietate
- Dacă Jucătorul 0 (verificatorul) are o strategie câștigătoare, atunci protocolul satisface proprietatea de excludere mutuală

Clasificarea în Teoria Complexității

Jocurile de paritate prezintă mai multe proprietăți interesante din punct de vedere al teoriei complexității:

- **NP ∩ co-NP**: Jocurile de paritate sunt atât în NP, cât și în co-NP. Aceasta înseamnă că atât un răspuns pozitiv ("Jucătorul 0 câștigă din nodul v"), cât și un răspuns negativ ("Jucătorul 1 câștigă din nodul v") pot fi verificate în timp polinomial.
- **UP ∩ co-UP**: Mai restrictiv, jocurile de paritate sunt în UP ∩ co-UP (Timp Polinomial Unic), ceea ce înseamnă că pentru orice instanță, există cel mult un certificat de acceptare care poate fi verificat în timp polinomial.
- **PLS (Căutare Locală Polinomială)**: Jocurile de paritate aparțin și clasei de complexitate PLS, care caracterizează problemele unde o soluție optimă local poate fi găsită prin îmbunătățire iterativă.

Clasele de Complexitate



Linia punctată: Apartenență conjecturată

Jocurile de Paritate și P vs. NP

Statutul jocurilor de paritate are implicații pentru întrebarea P vs. NP:

- Dacă jocurile de paritate sunt în P, acest lucru nu ar rezolva întrebarea P vs. NP, dar ar oferi informații despre structura problemelor din NP $\cap$ co-NP.
- Dacă jocurile de paritate s-ar dovedi a fi NP-hard, aceasta ar implica NP = co-NP, ceea ce este considerat improbabil. Acest lucru face improbabil ca jocurile de paritate să fie NP-complete.

Reduceri și Echivalențe

Jocurile de paritate sunt echivalente polinomial cu mai multe probleme importante:

- **Verificarea Modelelor pentru μ-calculul Modal**: Determinarea dacă o structură dată satisface o formulă în μ-calculul modal.
- **Evaluarea μ-calculului Boolean**: Evaluarea unei formule booleene cu operatori de punct fix minim și maxim.
- **Determinizarea Automatelor**: Convertirea automatelor de paritate nedeterministe în automate deterministe.
- **Jocuri cu Câștig Mediu**: Jocuri unde obiectivul este maximizarea câștigului mediu pe termen lung.
- **Jocuri cu Câștig Discontat**: Jocuri unde câștigurile viitoare sunt discontate cu un factor mai mic decât 1.

Cazuri Speciale cu Soluții în Timp Polinomial

În timp ce problema generală a jocurilor de paritate rămâne deschisă, mai multe cazuri speciale s-au dovedit a fi rezolvabile în timp polinomial:

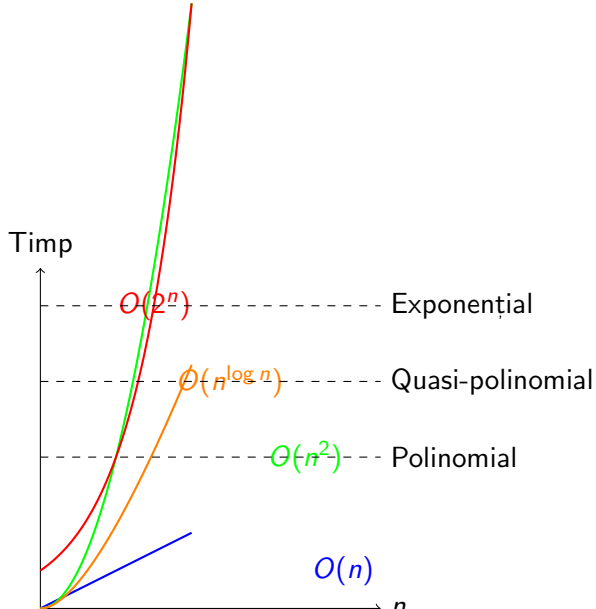
- **Jocuri cu un Număr Limitat de Priorități**: Jocuri de paritate unde numărul de priorități diferite este limitat de o constantă.
- **Jocuri cu Structuri Grafice Speciale**:
  - Arbori și păduri
  - Grafuri cu lățime arborescentă limitată
  - Grafuri cu lățime DAG limitată
  - Grafuri cu încâlcire limitată
- **Jocuri cu Structuri de Priorități Speciale**:
  - Jocuri unde prioritățile sunt restricționate la un interval mic
  - Jocuri cu un număr limitat de cicluri

Descoperirea Quasi-Polinomială

În 2017, a fost realizat un progres semnificativ în complexitatea jocurilor de paritate când Calude et al. au prezentat primul algoritm în timp quasi-polinomial pentru rezolvarea jocurilor de paritate.

- **Complexitate Quasi-Polinomială**: Un algoritm în timp quasi-polinomial are un timp de execuție de $n^{O(\log n)}$, unde n este dimensiunea intrării.
- Algoritmul lui Calude et al. are o complexitate de timp de aproximativ $n^{O(\log d)}$, unde n este numărul de noduri și d este numărul de priorități diferite în joc.
- Aceasta a fost o îmbunătățire majoră față de cei mai buni algoritmi anteriori, care aveau complexitate sub-exponențială.

Comparație de Complexități



Idei Cheie ale Algoritmului Quasi-Polinomial

Algoritmul de referință se bazează pe mai multe idei cheie:

1. **Automate Separatoare**: Algoritmul utilizează conceptul de automate separatoare, care pot distinge între jocurile câștigătoare pentru Jucătorul 0 și cele câștigătoare pentru Jucătorul 1.
2. **Martori Succinți**: Demonstrează că strategiile câștigătoare pot fi reprezentate prin martori succinți de dimensiune quasi-polinomială.
3. **Măsuri de Progres**: Algoritmul folosește o măsură de progres nouă care urmărește evoluția jocului într-un mod mai eficient decât abordările anterioare.
4. **Arbori Universali**: Conceptul de arbori universali oferă un cadru pentru organizarea informațiilor strategiei într-un mod compact.

Îmbunătățiri Ulterioare

După descoperirea inițială, mai mulți cercetători au propus rafinări și algoritmi quasi-polinomiali alternativi:

- **Jurdziński și Lazić (2017)**: Au dezvoltat un algoritm quasi-polinomial bazat pe măsuri de progres succinte, care oferă o perspectivă diferită asupra soluției quasi-polinomiale.
- **Lehtinen (2018)**: A prezentat o abordare folosind jocuri de registru, care reduce jocurile de paritate la jocuri de paritate echivalente cu un număr logaritmic de priorități.
- **Parys (2019)**: A simplificat algoritmul original și a îmbunătățit performanța sa practică combinând idei din algoritmul recursiv cu abordarea quasi-polinomială.

Comparație a Algoritmilor Quasi-Polinomiali

Algoritm	**Complexitate**	**Tehnică Cheie**
Calude et al. (2017)	$O(n^{\log d+6})$	Automate separatoare
Jurdziński și Lazić (2017)	$O((nd)^{\log d+1})$	Măsuri de progres succinte
Lehtinen (2018)	$O(n^{\log d+3})$	Jocuri de registru
Parys (2019)	$O(n^{\log d+1} \cdot d^3)$	Descompunere universală de atractori

Implicațiile Descoperirii

Algoritmul quasi-polinomial pentru jocurile de paritate are mai multe implicații importante:

1. Reduce semnificativ decalajul dintre limitele superioare și inferioare pentru complexitatea jocurilor de paritate.
2. Sugerează că un algoritm în timp polinomial ar putea fi realizabil, deoarece algoritmii quasi-polinomiali adesea preced pe cei polinomiali în istoria dezvoltării algoritmilor.
3. Oferă noi tehnici și perspective care au fost aplicate problemelor conexe din verificarea formală și teoria automatelor.
4. A stârnit un interes reînnoit în studiul jocurilor de paritate și al problemelor conexe, conducând la progrese suplimentare în domeniu.

Algoritmul Recursiv (Zielonka)

Algoritmul recursiv pentru rezolvarea jocurilor de paritate, dezvoltat de Wiesław Zielonka, este una dintre abordările clasice ale acestei probleme. În ciuda complexității sale exponențiale în cel mai rău caz, adesea funcționează bine în practică.

- **Idee Principală**: Algoritmul se bazează pe conceptul de atractori și pe proprietățile condiției de câștig de paritate.
- **Complexitate**: $O(n^d)$, unde n este numărul de noduri și d este numărul de priorități diferite.
- **Avantaj**: Simplu de implementat și eficient pentru jocuri cu un număr mic de priorități.

Algoritmul Recursiv al lui Zielonka

function REZOLVĂ($G = (V, V_0, V_1, E, \Omega)$)
  if $V = \emptyset$ **then return** $(\emptyset, \emptyset)$ ▷ Returnează regiuni de câștig vide
  end if
  $p \leftarrow \max\{\Omega(v) \mid v \in V\}$ ▷ Cea mai mare prioritate din joc
  $i \leftarrow p \bmod 2$ ▷ Jucătorul care pierde dacă prioritatea p apare infinit de multe ori
  $U \leftarrow \{v \in V \mid \Omega(v) = p\}$ ▷ Noduri cu prioritatea maximă
  $A \leftarrow \text{Atractor}_i(G, U)$ ▷ Atractorul lui U pentru Jucătorul i
  $G' \leftarrow G \setminus A$ ▷ Elimină atractorul din joc
  $(W'_0, W'_1) \leftarrow \text{Rezolvă}(G')$ ▷ Rezolvă recursiv jocul mai mic
  if $W'_{1-i} = \emptyset$ **then**
    $W_i \leftarrow V$ ▷ Jucătorul i câștigă peste tot
    $W_{1-i} \leftarrow \emptyset$
  else
    $B \leftarrow \text{Atractor}_{1-i}(G, W'_{1-i})$ ▷ Atractorul regiunii de câștig a adversarului
    $G'' \leftarrow G \setminus B$ ▷ Elimină acest atractor
    $(W''_0, W''_1) \leftarrow \text{Rezolvă}(G'')$ ▷ Rezolvă recursiv din nou
    $W_i \leftarrow W''_i$ ▷ Actualizează regiunile de câștig
    $W_{1-i} \leftarrow W''_{1-i} \cup B$
  end if
  return (W_0, W_1)
end function

Calculul Atracterilor

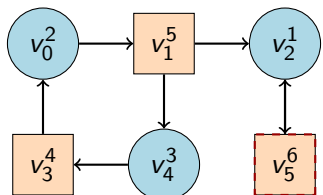
Algorithm 1 Calculul Atracterilor

function ATRACTOR($G = (V, V_0, V_1, E, \Omega), i, U$)
    $A_0 \leftarrow U$
    $k \leftarrow 0$
    repeat
        $A_{k+1} \leftarrow A_k \cup \{v \in V_i \mid \exists (v, w) \in E : w \in A_k\}$
        $A_{k+1} \leftarrow A_{k+1} \cup \{v \in V_{1-i} \mid \forall (v, w) \in E : w \in A_k\}$
        $k \leftarrow k + 1$
    until $A_k = A_{k-1}$ **return** A_k
end function

Atractorul Atractor$_i(G, U)$ este mulțimea nodurilor din care jucătorul i poate forța o vizită în U. Este calculat iterativ, adăugând:

- Nodurile jucătorului i care au cel puțin o muchie către atractorul curent
- Nodurile jucătorului $1 - i$ care au toate muchiile conducând către atractorul curent

Ilustrarea Algoritmului Recursiv



Pasul 1: Calculează atractorul nodului cu prioritatea maximă v_5^6

Măsuri de Progres Mici (Jurdziński)

Algoritmul măsurilor de progres mici, dezvoltat de Marcin Jurdziński, este o altă abordare clasică pentru rezolvarea jocurilor de paritate. Utilizează o paradigmă diferită față de algoritmul recursiv, concentrându-se pe calculul unui punct fix al unei funcții de clasificare.

- **Idee Principală**: Algoritmul se bazează pe noțiunea de măsuri de progres, care sunt funcții ce asociază nodurilor tupluri de numere naturale.
- **Complexitate**: $O(d \cdot m \cdot (n/d)^{d/2})$, unde n este numărul de noduri, m este numărul de muchii și d este numărul de priorități diferite.
- **Avantaj**: Mai bun decât algoritmul recursiv pentru jocuri cu un număr mic de priorități, dar tot exponențial în cel mai rău caz.

Măsuri de Progres

O măsură de progres este o funcție $\mu : V \to M_\Omega$, unde M_Ω este o mulțime de tupluri $(m_1, m_2, \ldots, m_d)$ cu constrângeri specifice:

- d este numărul de priorități impare din joc.
- Pentru fiecare prioritate impară p, componenta corespunzătoare m_p este limitată de numărul de noduri cu prioritatea p.
- Tuplurile sunt ordonate lexicografic.
- Există o valoare specială $\top$ (top) care este mai mare decât toate celelalte tupluri.

Măsura de progres satisface anumite condiții locale care asigură că identifică corect regiunile de câștig.

Algoritmul Măsurilor de Progres Mici

Algorithm 2 Algoritmul Măsurilor de Progres Mici

function RezolvăCuMăsuriDeProgresMici($G = (V, V_0, V_1, E, \Omega)$)
    Inițializează $\mu(v) = (0, 0, \ldots, 0)$ pentru toți $v \in V$
    while μ nu este un punct fix **do**
        for fiecare $v \in V$ **do**
            if $v \in V_0$ **then**
                $\mu(v) \leftarrow \min\{\text{Lift}(v, w, \mu) \mid (v, w) \in E\}$
            else
                $\mu(v) \leftarrow \max\{\text{Lift}(v, w, \mu) \mid (v, w) \in E\}$
            end if
        end for
    end while
    $W_0 \leftarrow \{v \in V \mid \mu(v) \neq \top\}$ ▷ Noduri cu măsură finită
    $W_1 \leftarrow V \setminus W_0$ ▷ Noduri cu măsură infinită
    return (W_0, W_1)
end function

Algoritmi de Îmbunătățire a Strategiei

Algoritmii de îmbunătățire a strategiei reprezintă o abordare diferită pentru rezolvarea jocurilor de paritate, bazată pe principiul îmbunătățirii iterative a unei strategii până când se găsește una optimă. Această abordare a fost pionierată de Jürgen Vöge și Marcin Jurdziński.

- **Idee Principală**: Algoritmul începe cu o strategie arbitrară pentru Jucătorul 0 și o îmbunătățește iterativ până când se găsește o strategie câștigătoare sau se determină că nu există nicio strategie câștigătoare.
- **Complexitate**: În cel mai rău caz, poate necesita un număr exponențial de iterații, dar în practică, algoritmul adesea converge rapid.
- **Avantaj**: Oferă nu doar regiunile de câștig, ci și strategii câștigătoare explicite.

Componentele Cheie ale Îmbunătățirii Strategiei

1. **Evaluarea Strategiei**: Dată o strategie σ pentru Jucătorul 0, se calculează o funcție de evaluare care atribuie o valoare fiecărui nod, reprezentând cât de bună este strategia din acel nod.
2. **Îmbunătățirea Strategiei**: Se identifică nodurile unde strategia poate fi îmbunătățită (adică unde o mutare diferită ar duce la o evaluare mai bună) și se actualizează strategia în consecință.
3. **Terminare**: Algoritmul se termină când nu mai sunt posibile îmbunătățiri, moment în care strategia curentă este optimă.

Algoritmul de Îmbunătățire a Strategiei

1: **function** REZOLVĂCUÎMBUNĂTĂȚIREDESTRATEGII($G = (V, V_0, V_1, E, \Omega)$)
2: Inițializează o strategie σ pentru Jucătorul 0
3: îmbunătățit $\leftarrow$ adevărat
4: **while** îmbunătățit **do**
5: Calculează evaluarea val_σ pentru strategia curentă σ
6: îmbunătățit $\leftarrow$ fals
7: **for** fiecare $v \in V_0$ **do**
8: **for** fiecare $(v, w) \in E$ **do**
9: **if** $\text{val}_\sigma(w) > \text{val}_\sigma(\sigma(v))$ **then**
10: $\sigma(v) \leftarrow w$ ▷ Îmbunătățește strategia
11: îmbunătățit $\leftarrow$ adevărat
12: **end if**
13: **end for**
14: **end for**
15: **end while**
16: Calculează regiunile de câștig (W_0, W_1) pe baza strategiei finale σ **return** (W_0, W_1)
17: **end function**

Algoritmi Quasi-Polinomiali

Dezvoltarea algoritmilor quasi-polinomiali pentru jocurile de paritate reprezintă un progres major în domeniu. Acești algoritmi au îmbunătățit semnificativ algoritmii anteriori, care aveau complexitate exponențială sau sub-exponențială.

- **Algoritmul Calude et al. (2017)**: Primul algoritm quasi-polinomial, bazat pe automate separatoare și martori succinți.
- **Măsuri de Progres Succinte (Jurdziński și Lazić, 2017)**: O reprezentare mai compactă a măsurilor de progres, conducând la un algoritm quasi-polinomial.
- **Jocuri de Registru (Lehtinen, 2018)**: Reduce jocurile de paritate la jocuri de paritate echivalente cu un număr logaritmic de priorități.
- **Descompunere Universală de Atractori (Parys, 2019)**: Simplifică abordarea originală și îmbunătățește performanța practică.

Algoritmul Quasi-Polinomial (Calude et al.)

Algorithm 3 Algoritmul Quasi-Polinomial (Calude et al.)

1: **function** REZOLVĂQUASIPOLINOMIAL($G = (V, V_0, V_1, E, \Omega)$)
2: Construiește un joc de separare G' bazat pe G
3: Rezolvă G' folosind o versiune modificată a calculului de atractori
4: Mapează soluția lui G' înapoi la o soluție a lui G **return** Regiunile de câștig (W_0, W_1)
5: **end function**

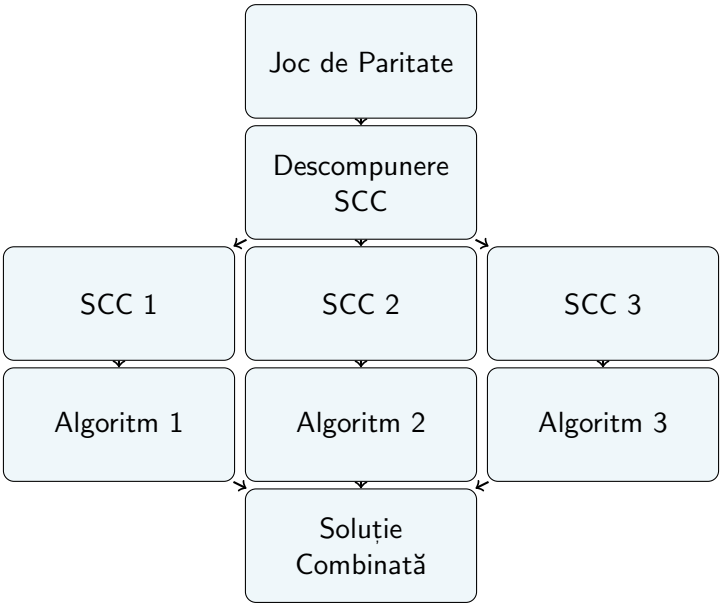
Jocul de separare G' este construit astfel încât strategiile câștigătoare în G' corespund strategiilor câștigătoare în jocul original G, dar cu avantajul că G' are o structură care permite metode de soluție mai eficiente.

Algoritm Hibrid Bazat pe Descompunere

Propun o abordare nouă pentru rezolvarea jocurilor de paritate care combină elemente din algoritmii existenți cu un accent pe exploatarea proprietăților structurale ale grafului de joc. Ideea cheie este că jocurile de paritate din lumea reală au adesea o structură care poate fi exploatată pentru soluții mai eficiente.

- **Descompunere SCC**: Jocurile de paritate pot fi descompuse în componente tare conexe (SCC), care pot fi rezolvate independent și apoi combinate.
- **Compresie de Priorități**: Multe jocuri de paritate au priorități care pot fi comprimate fără a schimba regiunile de câștig.
- **Selecția Algoritmului**: Diferiți algoritmi funcționează mai bine pe diferite tipuri de structuri de joc.
- **Rezolvare Parțială**: Adesea, porțiuni mari ale unui joc de paritate pot fi rezolvate rapid folosind reguli simple.

Structura Algoritmului Hibrid



Algoritmul Hibrid Bazat pe Descompunere

Algorithm 4 Algoritm Hibrid Bazat pe Descompunere

```
1:  function RezolvăHibrid(G = (V, V_0, V_1, E, Ω))
2:      Descompune G în componente tare conexe (SCC)
3:      Sortează SCC-urile în ordine topologică
4:      for fiecare SCC S în ordine topologică do
5:          if EsteSCCTrivial(S) then
6:              RezolvăJocTrivial(S)
7:          else
8:              S' ← ComprimăPriorități(S)
9:              (Rezolvat, Rămas) ← DescompunereAtractori(S')
10:             if nu Gol(Rămas) then
11:                 if EsteMic(Rămas) then
12:                     RezolvăCuAlgoritmRecursiv(Rămas)
13:                 else if AreLățimeArborescentăMică(Rămas) then
14:                     RezolvăCuAlgoritmDeLățimeArborescentă(Rămas)
15:                 else
16:                     RezolvăCuAlgoritmQuasiPolinomial(Rămas)
17:                 end if
18:             end if
19:             CombinăSoluții(Rezolvat, Rămas)
20:         end if
21:     end for
22:     PropagăSoluții(SCC) return RegiuniDeCâștig
23: end function
```

Avantajele Abordării Hibride

Algoritmul hibrid propus oferă mai multe avantaje:

1. **Adaptabilitate**: Se adaptează la structura jocului de intrare, folosind algoritmul cel mai potrivit pentru fiecare componentă.
2. **Eficiență**: Prin exploatarea structurii și utilizarea tehnicilor de rezolvare parțială, poate rezolva multe instanțe mai eficient decât algoritmii generali.
3. **Scalabilitate**: Abordarea se scalează bine cu dimensiunea jocului, deoarece descompunerea permite procesarea paralelă a componentelor independente.
4. **Robustețe**: Chiar și în cel mai rău caz, algoritmul revine la cel mai bun algoritm general cunoscut, asigurând o performanță competitivă.

Rezumat

- Jocurile de paritate sunt un model fundamental pentru verificarea formală și sinteza sistemelor reactive.
- Ele ocupă o poziție unică în teoria complexității, fiind în NP $\cap$ co-NP, dar fără un algoritm polinomial cunoscut.
- Mai mulți algoritmi au fost dezvoltați pentru rezolvarea jocurilor de paritate, inclusiv algoritmul recursiv, măsurile de progres mici, îmbunătățirea strategiei și, mai recent, algoritmii quasi-polinomiali.
- Abordarea hibridă propusă combină tehnici din algoritmi existenți și exploatează structura jocurilor pentru a obține o performanță îmbunătățită în practică.
- Aplicațiile jocurilor de paritate includ verificarea modelelor, sinteza controlerelor și verificarea proprietăților temporale.

Contribuții

- **Analiză Comprehensivă**: O analiză detaliată a jocurilor de paritate, a complexității lor și a algoritmilor pentru rezolvarea lor.
- **Algoritm Hibrid**: Propunerea unui algoritm hibrid care exploatează structura jocurilor pentru a îmbunătăți performanța.
- **Evaluare Comparativă**: O comparație a diferitelor abordări pentru rezolvarea jocurilor de paritate.
- **Direcții de Cercetare**: Identificarea direcțiilor promițătoare pentru cercetarea viitoare în domeniul jocurilor de paritate.

Referințe

[1] Calude, C. S., Jain, S., Khoussainov, B., Li, W., & Stephan, F. (2017). Deciding parity games in quasipolynomial time. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (pp. 252-263).

[2] Jurdziński, M. (2000). Small progress measures for solving parity games. In Annual Symposium on Theoretical Aspects of Computer Science (pp. 290-301).

[3] Zielonka, W. (1998). Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theoretical Computer Science, 200(1-2), 135-183.

[4] Jurdziński, M., & Lazić, R. (2017). Succinct progress measures for solving parity games. In 2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (pp. 1-9).

[5] Lehtinen, K. (2018). A modal μ perspective on solving parity games in quasi-polynomial time. In Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (pp. 639-648).

Mulțumiri

Vă mulțumesc pentru atenție!

Întrebări?