CHALMERS

GÖTEBORGS UNIVERSITET

Potentialspel och iterativt spelande

Potential games and repeated play dynamics

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet

Jonas Högberg
Victor Fingal

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

Potentialspel och iterativt spelande

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet

Jonas Högberg Victor Fingal

Handledare: Jeffrey Steif

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

Förord

Vi vill tacka vår handledare Jeffrey Steif för visat förtroende och för de mycket värdefulla samtalen under projektets gång. Vi vill också rikta ett stort tack till Hanna för hennes engagemang i de tidigare skedena av projektet.

Ansvarsområden

Skrivandeprocessen har varit sådan att en gruppmedlem har producerat ett första utkast, därefter har texten bearbetats av båda. Utöver detta har Jonas Högberg ansvarat för programkod, och Victor Fingal för informationssökning och referenser. Nedan följer en redogörelse för vilka avsnitt respektive gruppmedlem skrivit det första utkastet av:

- **Jonas Högberg:** Populärvetenskaplig presentation, Avsnitt 1.3, 2.1-3, 3.1, 4.2-3, 4.4.2, 4.5 och Appendix A.1, B.
- **Victor Fingal:** Avsnitt 1.1-2, 2.4-6, 3.2-3, 4.1, 4.4.1 och Appendix A.2.

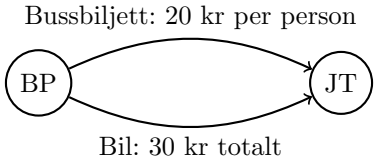
Vi har även fört individuell tidslogg och ansvarat för den gemensamma dagboken varannan vecka.

Populärvetenskaplig presentation

Med spelteori kan vi förklara situationer där enskilda beslut leder till ett sämre kollektivt utfall än vad som kan nås genom samarbete. I vissa fall leder även större valfrihet till ett sämre utfall för samtliga inblandade. För att förstå hur sådana paradoxala situationer kan uppstå studerar vi konkreta exempel på trängselspel, samt en algoritm som söker efter stabila utfall.

Antag att du och en vän ska färdas från Brunnsparken till Järntorget. En bussbiljett kostar 20 kr, medan den totala kostnaden för att ta bilen är 30 kr. Om båda väljer bilen delar ni på kostnaden. Vilket färdmedel väljer du?

Om målet är att minska den totala kostnaden är det självklara valet att åka bil tillsammans. Men om din kompis väljer bussen blir det billigare för dig att också ta bussen. Figur 1 illustrerar situationen.



Figur 1: Två personer väljer mellan att åka buss eller bil från Brunnsparken till Järntorget.

Vi har just beskrivit ett trängselspel. Trängselspel kan användas för att modellera vägnätverk, marknader och andra situationer där en eller flera spelare utnyttjar gemensamma resurser. De individuella valen spelarna gör kallas för strategier, och vi antar alltid att varje spelare *väljer den strategi som minimerar den egna kostnaden utifrån de andra spelarnas nuvarande val av strategi.*

Antag nu att ni gör resan varje dag, och att ni inte får diskutera valet av färdmedel. Chansar du på att din vän tar bilen, eller garderar du dig med den fasta kostnaden för en bussbiljett? Varje dag ges *endast en* utav er möjligheten att ändra sitt val baserat på vad den andre valde igår, förutsatt att det minskar den egna kostnaden.

När ingen spelare längre tjänar på ett enskilt byte av strategi har man nått en Nashjämvikt. I exemplet är t.ex. "båda väljer bilen" en Nashjämvikt. En Nashjämvikt kan betraktas som ett dödläge, eftersom ingen spelare har något incitament att byta strategi. Därför är det naturligt att betrakta Nashjämvikter som lösningar till spel.

Om båda valde bussen igår tjänar ingen utav er på att enskilt byta till bilen. Därför är även strategiparet "båda väljer bussen" en Nashjämvikt. Men totalkostnaden för resan blev $\frac{40 \text{ kr}}{30 \text{ kr}} = \frac{4}{3}$ gånger större än för strategiparet "båda väljer bilen".

Exemplet kan verka tillrättalagt för att ha just denna egenskap, men fenomenet är vanligt förekommande och har fått namnet anarkipriset (eng. the price of anarchy). Anarkipriset är kvoten mellan den dyraste Nashjämvikten och den minsta möjliga totalkostnaden, och ger ett mått på hur kostsamt det kan bli när spelare inte kan, eller vill samarbeta.

Anarkipriset beror inte bara på hur spelarna agerar, utan även på själva strukturen i spelet. Vi kan eliminera den dyra Nashjämvikten genom att t.ex. införa en trängselavgift på 15 kr för bilresor eller subventionera halva bussbiljetten. I båda fallen skulle det alltid löna sig att byta från bilen till bussen. Anarkipriset blir då 1, eftersom "båda väljer bussen" blir den enda Nashjämvikten såväl som den minsta möjliga totalkostnaden.

I exemplet kan vi hitta alla Nashjämvikter genom att studera alla fyra möjliga utfall. Men ett spel med 10 spelare och 10 möjliga strategier vardera ger upphov till tio miljarder möjliga utfall. Det är då inte effektivt att undersöka alla utfall.

Ett sätt att hitta en Nashjämvikt i stora spel är att tillämpa en algoritm. Vi utgår från någon kombination av strategier, och låter sedan en spelare i taget byta strategi tills ingen längre kan minska sin kostnad. På så vis kan vi hitta Nashjämvikter även i stora spel.

Sammanfattning

Vi sammanfattar delar av teorin kring potentialspel och visar dess koppling till trängselspel, samt illustrerar dess relevans med exempel. Dessutom introducerar vi iterativt spelande som metod för att hitta rena Nashjämvikter i potentialspel, och anarkipriset som ett kvantitativt mått på konsekvenserna av att varje spelare strävar efter att minimera den egna kostnaden i ett spel.

Därefter använder vi trängselspel för att modellera ett vägnätverk inspirerat av Braess paradox, där den genomsnittliga restiden i ren Nashjämvikt förvånansvärt nog försämras när en ny väg läggs till i nätverket. Vi definierar också en algoritm baserad på iterativt spelande och ger empiriskt stöd för att det genomsnittliga antalet iterationer innan den når en ren Nashjämvikt i slumpmässiga potentialspel varken beror på antalet spelare eller strategier.

Abstract

We summarize parts of the theory surrounding potential games and show its connection to congestion games, and illustrate its relevance with examples. Furthermore, we introduce repeated play dynamics as a method for finding pure Nash equilibria in potential games, and the price of anarchy as a quantitative measure of the consequences of that each player's goal is to to minimize the individual cost in a game.

Thereafter, we use congestion games to model a road network inspired by Braess' paradox, where the average travel time in pure Nash Equilibrium surprisingly increases when a new road is added to the network. We also define an algorithm based on repeated play dynamics and provide empirical support that the average number of iterations until it reaches a pure Nash equilibrium in random potential games depends neither on the number of players or strategies.

Innehåll

1 Inledning.

Inför Earth Day 1990 i New York beslutades det att 42d street, en väl trafikerad gata, skulle stängas. Det förutspåddes trafikkaos, men istället förbättrades den genomsnittliga restiden [4]. Hur kunde New Yorks bilförare, trots att de den dagen hade färre vägval, förbättra den genomsnittliga restiden?

I den här uppsatsen visar vi hur man med spelteori delvis kan besvara den här frågan. Vi använder oss av trängselspel för att visa att rena Nashjämvikter inte alltid innebär det bästa möjliga utfallet för alla spelare. En ren Nashjämvikt är en situation där ingen enskild spelare kan nå ett bättre utfall genom att byta strategi, givet att de andra spelarna inte byter strategi.

Att ett spel är i en ren Nashjämvikt säger dock ingenting om den totala vinsten eller kostnaden för alla spelare. Braess [1] presenterade 1968 en paradoxal situation som varit central för att illustrera den här skillnaden. Han beskrev ett vägnätverk där den genomsnittliga restiden för alla förare försämrades när en ny väg lades till i nätverket.

Antagandet bakom Braess paradox är att varje förare strävar efter att minimera sin egna restid. I spel kan konsekvenserna av det antagandet kvantifieras med anarkipriset, (eng. price of anarchy) som introducerades av Papadimtrou [9]. Anarkipriset är kvoten mellan den totala kostnaden i den rena Nashjämvikt som ger störst total kostnad, och den minsta möjliga totala kostnaden för alla spelare.

1.1 Varför potentialspel?

Ett centralt problem i spelteorin är: *Vilka är spelets rena Nashjämvikter?*. Bortom de enklaste spelen där varje utfall kan undersökas är problemet att hitta en ren Nashjämvikt svårt, och det är inte ens säkert att en sådan existerar. Däremot har hoppfulla resultat visats för potentialspel.

Potentialspel, som introducerades av Monderer & Shapley [6], är en klass av spel där en ren Nashämvikt alltid existerar. Det första exemplet på ett potentialspel presenterades 1973 av Rosenthal [10]. Rosenthals exempel kallas idag för trängselspel och kännetecknas av att spelarnas strategier involverar nyttjandet av gemensamma resurser. Trängselspel är användbara för att beskriva trafiksituationer, eftersom vägstycken i ett vägnätverk kan betraktas som resurser.

Inspirerade av en enkel idé, iterativt spelande (eng. repeated play dynamics), kan vi definiera en algoritm som alltid hittar en ren Nashjämvikt i ett givet potentialspel.

1.2 Uppsatsen

I arbetet med uppsatsen har vi inspirerats av klassiska modeller som leder till oväntade resultat. Med exempel och tolkningar inspirerade av dessa modeller hoppas vi kunna visa läsaren varför teorin om potentialspel, och i synnerhet trängselspel är relevant. Vi vill också visa hur man på ett enkelt sätt kan tillämpa iterativt spelande för att definiera en algoritm som hittar en ren Nashjämikt i potentialspel. Uppsatsen utgörs således både av presentation och tillämpningar av den relevanta teorin kring potentialspel.

Teorin, som presenteras i avsnitt 2, inleds med grundläggande begrepp och resultat från spelteorin, och följs av de för uppsatsen centrala koncepten och resultaten rörande potentialspel och trängselspel. I avsnitt 3 och 4 tillämpar vi teorin från avsnitt 2.

I avsnitt 3 studerar vi trängselspel med linjära kostnadsfunktioner. Vi analyserar först en diskret modell inspirerad av Braess paradox för att illustrera hur strukturen i ett vägnätverk kan påverka förarnas restid och beslut. Därefter formulerar och bevisar vi en övre begränsning på anarkipriset, samt ger exempel på ett spel som uppfyller den övre begränsningen.

I avsnitt 4 implementerar vi en algoritm baserad på iterativt spelande på slumpmässigt genererade potentialspel. Först visar vi en metod för att konstruera potentialspel med godtyckliga värden på potentialfunktionen. Sedan definierar algoritmen och tillämpar den på ett stort antal slumpgenererade potentialspel, samt ger empiriskt stöd för att antalet iterationer som krävs för att nå en ren Nashjämvikt varken beror på antalet spelare eller strategier.

1.3 Terminologi och notation

Innan vi lägger fram teorin vill vi förtydliga vad vi menar när vi använder följande begrepp och symboler.

(i) Vi använder operatorn ':=' när vi definierar vänsterledet med det som står i högerledet.

(ii) En **tupel** är en ordnad lista med godtyckliga element. En *k-tupel* är en tupel av längd k. En k-dimensionell vektor är i vår terminologi en k-tupel där varje element är ett numeriskt värde. När vi beskriver spel använder vi "spetsiga" parenteser, som i $\langle N, \mathbf{S}, (u_i)\rangle$.

(iii) Givet en tupel $\mathbf{s} = (s_1, \ldots, s_k)$ använder vi ofta skrivsättet $\mathbf{s} = (s_i, \mathbf{s}_{-i})$, när vi undersöker en enskild komponent s_i i $\mathbf{s}$. Då innehåller $(k-1)$-tupeln $\mathbf{s}_{-i}$ alla element i $\mathbf{s}$ förutom s_i.

(iv) En n-dimensionell **sannolikhetsvektor** är en kolumnvektor $\mathbf{x} = (x_1, \ldots, x_n)^T$ sådan att

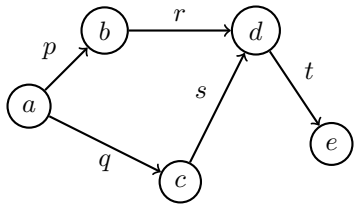
$$x_i \geq 0, \;\; i = 1, \ldots, n \quad \text{och} \quad \sum_{i=1}^{n} x_i = 1. \tag{1}$$

(v) En **graf** $G = (V, E)$ utgörs av en ändlig mängd V, och en mängd E vars element sammankopplar par av element i V. Elementen i V kallas *noder* och elementen i E kallas *kanter*. Om en kant $e \in E$ sammankopplar två noder endast i den ena riktningen sägs e vara en *riktad* kant. Om t.ex. $e = (v_1, v_2)$, där $v_1, v_2 \in V$ är en riktad kant gäller att v_2 går att nå från v_1 via e, men inte vice versa.

En **väg** av längd n är en sekvens av noder $(v_1, \ldots, v_n)$ sådan att

$$(v_i, v_{i+1}) \in E, \;\; i = 1, \ldots, n.$$

Figur 2 illustrerar ett exempel på en riktad graf.



Figur 2: En riktad graf $G = (V, E)$ med $V = \{a, b, c, d, e\}$ och $E = \{p, q, r, s, t\}$. Följden (a, c, d, e) är ett exempel på en väg i G.

2 Teori

2.1 Grundläggande spelteori

Med ett **spel** menar vi en ändlig mängd **spelare**, som väljer bland en mängd **strategier** för att maximera sin **utdelning**.

Vad man i litteraturen kallar för kombinatoriska spel spelas ofta över ett antal rundor, och med en viss turording bland spelarna. De spel vi kommer att studera avgörs omedelbart efter att varje spelare valt en strategi. I mer dynamiska situationer talar vi istället om iterativt spelande, vilket behandlas i avsnitt 2.3.

Exempel 1. I ett spel med en spelare där spelarens mål är att på kortast möjliga tid hitta en väg genom en labyrint är en av de tillgängliga strategierna att hålla vänster vid varje vägskäl. Eftersom labyrintens struktur är fix avgör valet av strategi den fullständiga vägen spelaren går genom labyrinten. Spelet är därmed avgjort omedelbart efter att spelaren valt sin strategi.

Definition 1. Vi betecknar mängden av alla **spelare** med $N := \{1, \ldots, k\}$, och låter således varje spelare representeras av ett heltal. En spelares individuella val bland tillgängliga alternativ i ett spel kallas för spelarens val av **strategi** s_i ur **strategirummet** S_i, där $i \in N$. En **strategiprofil** för ett spel med k spelare är en k-tupel $\mathbf{s} := (s_1, \ldots, s_k) \in S_1 \times \ldots \times S_k$, där s_i är spelare i:s val av strategi i strategirummet S_i.

Definition 2. Sätt $\mathbf{S} := S_1 \times \ldots \times S_k$. Till varje spelare $i \in N$ hör en **nyttofunktion** $u_i : \mathbf{S} \to \mathbb{R}$.

Nyttofunktionen kvantifierar en spelares individuella utdelning i spelet. Värdet på $u_i(\mathbf{s})$ kan representera spelare i:s vinst, förlust, kostnad, tidsåtgång, etc., med avseende på given strategiprofil $\mathbf{s} \in \mathbf{S}$. Vi sammanfattar informationen om ett spel Γ i en tupel $\Gamma := \langle N, \mathbf{S}, (u_i)_{i \in N} \rangle$.

Anmärkning 1. I avsnitt 2.6 och 3 studerar vi kostnadsminimeringsspel. I sådana spel är varje spelare ute efter att *minimera* sin nyttofunktion. Fram tills dess låter vi det vara underförstått att det ligger i varje spelares intresse att *maximera* sin nyttofunktion.

Exempel 2. I ett 2-spelar-spel där varje spelare har två strategier att välja bland kan nyttofunktionen representeras av en 2×2-matris. Om s_i, t_j, är strategier för spelare 1 resp. 2 och matrisen $A = (a_{ij})$ representerar nyttofunktionen för spelare 1 får matrisen utseendet

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} u_1(s_1, t_1) & u_1(s_1, t_2) \\ u_1(s_2, t_1) & u_1(s_2, t_2) \end{bmatrix}. \tag{2}$$

Exempel 3. I spel med 2 spelare kan vi manipulera framställningen ytterligare genom att sätta $c_{ij} = (a_{ij}, b_{ij}) = (u_1(s_i, t_j), u_2(s_i, t_j))$. Matrisen $C = (c_{ij})$ innehåller då all relevant information om utfallet i spelet. Sätter vi in numeriska värden på nyttofunktionerna u_i får C utseendet

$$C = \begin{bmatrix} (4,2) & (1,3) \\ (3,3) & (4,2) \\ (3,2) & (2,5) \end{bmatrix}. \tag{3}$$

Om strategieparet (s_1, t_1) spelas blir utdelningen $(4, 2)$, alltså 4 för spelare 1 och 2 för spelare 2. Om paret (s_2, t_1) spelas blir båda spelarnas utdelning 3.

2.2 Nashjämvikter

Definition 3. En **ren Nashjämvikt** är en strategiprofil $\mathbf{s}^* = (s_1^*, \ldots, s_k^*) \in S_1 \times \ldots \times S_k$ sådan att för varje spelare i och val strategi $s_i \in S_i$, $i = 1, \ldots, k$ gäller

$$u_i(s_i^*, \mathbf{s}_{-i}^*) \geq u_i(s_i, \mathbf{s}_{-i}^*). \tag{4}$$

En ren Nashjämvikt karakteriseras av egenskapen att ingen spelare kan förbättra sin position genom ett enskilt byte av strategi (dvs. om ingen av de övriga spelarna byter strategi).

Exempel 4. Betrakta följande 2×2-matris där rader och kolumner representerar strategier för spelare 1 respektive 2.

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} (1,3) & (3,5) \\ (2,1) & (0,2) \end{bmatrix} \tag{5}$$

Strategiparet rad 1, kolumn 2 är en ren Nashjämvikt, eftersom den första och andra koordinaten i c_{12} är kolumn- resp. radmaximum för respektive spelare. Ingen spelare kan därmed förbättra sin utdelning om den andre står fast vid sin strategi.

Definition 4. Låt $n_i = |S_i|$. En **mixad strategi** för spelare i är en n_i-dimensionell sannolikhetsvektor $\mathbf{x}_i = (x_1^i, \ldots, x_{n_i}^i)^T$, där x_j^i är sannolikheten att spelare i väljer strategin $s_j \in S_i$.

Vi utvidgar nu Definition 1 till att omfatta mixade strategier.

Definition 5. En **mixad strategiprofil** i ett spel med k spelare är en tupel $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ där $\mathbf{x}_i$ är en mixad strategi för spelare i. I spel med mixade stategier definierar vi nyttofunktionen enligt

$$u_i(\mathbf{X}) := \sum_{\mathbf{s} \in \mathbf{S}} \prod_{p \in N} x_p(s_p) u_i(\mathbf{s}), \tag{6}$$

där $x_p(s_p)$ är sannolikheten att spelare p väljer den rena strategin s_p. Vi talar då om nyttofunktionens *väntevärde* för spelare i. Följande proposition förtydligar vad nyttofunktionen i ekvation 6 innebär för varje enskild spelare.

Proposition 1. *Om $(x_1^i, \ldots, x_{n_i}^i)^T$ är en mixad strategi för spelare i kan nyttofunktionen skrivas*

$$u_i(\mathbf{X}) = \sum_{j=1}^{n_i} x_j^i u_i(s_j^i, \mathbf{X}_{-i}).$$

Bevis. Sätt u_i som i ekvation 6 och skriv om uttrycket enligt

$$\begin{aligned} u_i(\mathbf{X}) = \sum_{\mathbf{s} \in \mathbf{S}} \prod_{p \in N} x_p(s_p) u_i(\mathbf{s}) &= \sum_{\mathbf{s} \in \mathbf{S}} x_i(s_i) \prod_{p \neq i} x_p(s_p) u_i(s_i, \mathbf{s}_{-i}) \\ &= \sum_{j=1}^{n_i} x_j^i \sum_{\mathbf{s}_{-i}} \prod_{p \neq i} x_p(s_p) u_i(s_j^i, \mathbf{s}_{-i}) \\ &= \sum_{j=1}^{n_i} x_j^i u_i(s_j^i, \mathbf{X}_{-i}). \end{aligned} \tag{7}$$

□

Exempel 5. Låt $A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$ representera spelare 1:s nyttofunktion i ett tvåspelarspel. Om $\mathbf{x} = (x, 1-x)^T$, $\mathbf{y} = (y, 1-y)^T$ är sannolikhetsvektorer för spelare 1 resp. 2 har vi

$$u_1(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \mathbf{y} = 3 + 3xy - 2(x+y). \tag{8}$$

Vi undersöker vad som händer då spelare 2 spelar den rena strategin $\mathbf{y} = (\frac{2}{3}, \frac{1}{3})$. Vi får

$$u_1\left(\mathbf{x}, (\tfrac{2}{3}, \tfrac{1}{3})\right) = 3 + 3x \cdot \tfrac{2}{3} - 2(x + \tfrac{2}{3}) = 2 - \tfrac{1}{3},$$

dvs. $u_i(\mathbf{x}, (\frac{2}{3}, \frac{1}{3}))$ är oberoende av $\mathbf{x}$. Spelare 1 har med andra ord ingen möjlighet att förbättra sin utdelning genom att byta till en annan mixad strategi.

Utan att explicit ange spelare 2:s utdelningsmatris kan vi föreställa oss en situation där ingen av spelarna kan förbättra sin utdelning genom att enskilt byta till en annan mixad strategi. Sådana situationer motiverar en tolking av jämviktsbegreppet i spel med mixade strategier.

Definition 6. En **mixad Nashjämvikt** är en mixad strategiprofil $\mathbf{X}^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_k^*)$ sådan att för varje spelare $i = 1, \ldots, k$, och för varje sannolikhetsvektor $\mathbf{x}_i$ gäller

$$u_i(\mathbf{x}_i^*, \mathbf{X}_{-i}^*) \geq u_i(\mathbf{x}_i, \mathbf{X}_{-i}^*). \tag{9}$$

En ren Nashjämvikt kan således ses som ett specialfall av en mixad Nashjämvikt med restriktionen att varje sannolikhetsvektor är en standardbasvektor.

Definition 7. I ett **ändligt spel** har varje spelare ett ändligt antal tillgängliga strategier.

En central person inom spelteorin är John F. Nash, som år 1950 visade att det alltid existerar mixade Nashjämvikter i ändliga spel [7]. Resultatet är en utav hörnstenarna i spelteorin.

Beviset till *Nashs sats* utnyttjar idéer på en förhållandevis hög nivå av abstraktion. Vi ger här ett annat bevis ur Mendelson [5] som är mer konkret, och förhoppningsvis lättare att följa. Beviset är dock omfattande och av föga relevans för de spel vi studerar längre ned i texten. Därför är det formella argumentet förlagt till Appendix A.1, och vi nöjer oss här med att ge sammanfattning.

Nashs sats ([5])**.** *Varje ändligt spel har en mixad Nashjämvikt.*

Bevis av Nashs Sats (sammanfattning). Låt $\boldsymbol{\Delta}$ beteckna mängden av alla mixade strategiprofiler $\mathbf{X}$. I beviset anger vi explicit en kontinuerlig funktion $\varphi : \boldsymbol{\Delta} \to \boldsymbol{\Delta}$ med egenskapen att $\mathbf{X}$ är en Nashjämvikt om och endast om $\varphi(\mathbf{X}) = \mathbf{X}$. Eftersom $\boldsymbol{\Delta}$ är sluten, begränsad och konvex utnyttjar vi sedan *Brouwers fixpunktssats*, som säger att varje kontinuerlig funktion från en sådan mängd till sig själv har en fixpunkt. Med det drar vi slutsatsen att en Nashjämvikt alltid existerar i ett ändligt spel. □

2.3 Iterativt spelande

I det här avsnittet definierar vi iterativt spelande, som vi senare i avsnitt 4.2 baserar en algoritm på, och motiverar när den är användbar.

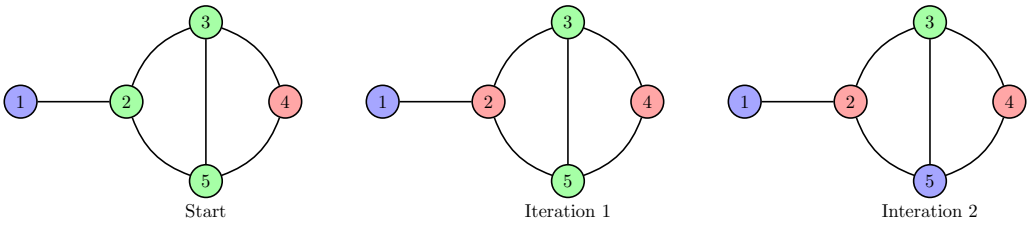
Definition 8. Givet ett ändligt spel och en godtycklig strategiprofil till spelet, låt en spelare i taget byta strategi om denne kan strikt förbättra sin utdelning. Processen avbryts när ingen spelare längre har denna möjlighet. Vi kallar denna process för **iterativt spelande**.

Anmärkning 2. Iterativt spelande avbryts om och endast om den slutliga strategiprofilen är en ren Nashjämvikt.

De 2 nedanstående exemplen visar hur metoden fungerar i olika spel.

Exempel 6 (Graffärgning)**.** Vi har ett spel med 5 spelare. Vissa spelare är vänner med andra spelare. Spelarnas relation till varandra sammanfattar vi med en oriktad graf. Spelet går ut på att varje spelare väljer mellan tre färger; blå, grön eller röd, på ett sådant sätt så att de har ett minimalt antal vänner som har valt samma färg.

Vi väljer en godtycklig färg för varje spelare och låter sedan en i taget byta färg fram till att ingen spelare längre kan minska antalet vänner som de delar färg med. Figur 3 beskriver ett möjligt utfall av den här processen.



Figur 3: Det sociala nätverket i spelet från Exempel 6. Noderna representerar spelarna och de kanter som sammankopplar noder visar vilka spelare som är vänner.

Notera att iterativt spelande inte når en unik ren Nashjämvikt. I iteration 1 skulle spelare 3:s byte från grön till blå också slutat i en ren Nashjämvikt.

Exempel 7 (Matchande mynt)**.** Matchande mynt är ett välkänt spel med 2 spelare. Spelarna har varsitt mynt och skall välja en sida; krona eller klave. De avslöjar sedan sina val samtidigt. Spelare 1 vinner om båda spelarna väljer samma sida, om de väljer motsatt sida vinner spelare 2. En vinst ger 1 poäng och en förlust 0 poäng.

Man kan enkelt observera att iterativt spelande inte når någon ren Nashjämvikt, då ingen sådan existerar. För en godtycklig strategiprofil finns det alltid en vinnare och en förlorare. Därav kan alltid förloraren förbättra sin utdelning från 0 till 1 genom att byta sida på sitt mynt i nästa iteration av spelet.

Vi har nu visat vad som kan förväntas av iterativt spelande i två olika spel. Om ingen ren Nashjämvikt existerar uppstår cykel som i Exempel 7. Däremot fungerade metoden bra i Exempel 6 och man kan enkelt att verfiera att metoden leder till en ren Nashjämvikt oavsett initial strategiprofil. I nästa avsnitt introducerar vi en klass av spel där minst en ren Nashjämvikt alltid existerar: *potentialspel*. Vi visar också att iterativt spelande oavsett initial strategiprofil alltid når en ren Nashjämvikt i potentialspel.

2.4 Potentialspel

Definition 9. Låt $\Gamma = \langle N, \mathbf{S}, (u_i)_{i\in N}\rangle$ vara ett ändligt spel. Spelet Γ är ett *potentialspel* om det existerar en funktion $\Psi : \mathbf{S} \to \mathbb{R}$ sådan att

$$\Psi(\tilde{s}_i, \mathbf{s}_{-i}) - \Psi(s_i, \mathbf{s}_{-i}) = u_i(\tilde{s}_i, \mathbf{s}_{-i}) - u_i(s_i, \mathbf{s}_{-i}) \tag{10}$$

för alla $i \in N$, $s_i, \tilde{s}_i \in S_i$ och $\mathbf{s}_{-i} \in \mathbf{S}_{-i}$. Vi kallar Ψ en *potentialfunktion* till Γ.

Från ekvation 10 utläser vi att Ψ förändras exakt lika mycket som en enskild spelares nyttofunktion då spelaren byter strategi. Vi understryker att potentialfunktionen inte ger någon information om förändringar i de övriga spelarnas nyttofunktioner.

Exempel 6, forts (Graffärgning)**.** Sätt $N = \{1, 2, 3, 4, 5\}$, $S_i = \{$blå, grön, röd$\}, i \in N$, och låt u_i beteckna antalet vänner till spelare i som valt samma färg som spelare i. Då är $\langle N, \mathbf{S}, (u_i)_{i\in N}\rangle$ ett potentialspel med en potientialfunktion $\Psi = \frac{1}{2}\sum_{i\in N} u_i$.

Sats 1 ([3])**.** *Varje potentialspel har en ren Nashjämvikt.*

Bevis. Låt $\Gamma = \langle N, \mathbf{S}, (u_i)_{i\in N}\rangle$ vara ett potentialspel med en potentialfunktion Ψ. Tag en godtycklig strategiprofil $\mathbf{s}^0 \in \mathbf{S}$ och tillämpa iterativt spelande på Γ med start i $\mathbf{s}^0$. I varje iteration t väljer om möjligt någon spelare $i \in N$ en ny strategiprofil $\mathbf{s}^{t+1} := (s_i, \mathbf{s}^t_{-i})$ sådan att $u_i(\mathbf{s}^{t+1}) > u_i(\mathbf{s}^t)$.

Enligt Definition 9 får vi att $\Psi(\mathbf{s}^{t+1}) > \Psi(\mathbf{s}^t)$, dvs. iterativt spelande ger upphov till en strikt växande sekvens av värden på potentialfunktionen. Eftersom $\mathbf{S}$ är ändlig når vi till slut ett index τ sådant att $\Psi(\mathbf{s}^\tau) \geq \Psi(s_i, \mathbf{s}^\tau_{-i})$ för alla i och s_i, och iterativt spelande avbryts. Enligt Definition 9 gäller då att $u_i(\mathbf{s}^\tau) \geq u_i(s_i, \mathbf{s}^\tau_{-i})$ för alla i och s_i. Alltså är $\mathbf{s}^\tau$ en ren Nashjämvikt. □

Korollarium 1. *Iterativt spelande i ett potentialspel når en ren Nashjämvikt.*

Potentialspel kan konstrueras från två enklare spel, *koordinations-* och *dummyspel*, genom att definiera potentialspelets nyttofunktion som summan av nyttofunktionerna från koordination- och dummyspelet. Vi drar nytta av det här resultatet i avsnitt 4.1 när vi konstruerar potentialspel.

Definition 10. Ett spel $\Gamma = \langle N, \mathbf{S}, (u_i)_{i\in N}\rangle$ är ett

- **koordinationsspel**, om det existerar en funktion $u : \mathbf{S} \to \mathbb{R}$ sådan att $u_i = u$ för alla $i \in N$.
- **dummy-spel**, om det för alla $i \in N$ och $\mathbf{s}_{-i} \in S_{-i}$ existerar ett $x \in \mathbb{R}$ s.a $u_i(s_i, \mathbf{s}_{-i}) = x$ för alla $s_i \in S_i$, dvs. en spelares nytta är oberoende av spelarens val av strategi, givet att alla andra spelare redan har valt en strategi.

Sats 2 ([12])**.** *Ett spel $\Gamma = \langle N, \mathbf{S}, (u_i)_{i\in N}\rangle$ är ett potentialspel om och endast om det existerar funktioner $(c_i)_{i\in N}$ och $(d_i)_{i\in N}$ sådana att*

- *$\langle N, \mathbf{S}, (c_i)_{i\in N}\rangle$ är ett koordinationsspel,*
- *$\langle N, \mathbf{S}, (d_i)_{i\in N}\rangle$ är ett dummy-spel,*
- *$u_i = c_i + d_i$ för alla $i \in N$.*

Bevis. Antag först att $\Gamma = \langle N, \mathbf{S}, (u_i)_{i\in N}\rangle$ är ett potentialspel med en potentialfunktion Ψ. Observera att

$$u_i = \Psi + (u_i - \Psi), \text{ för alla } i \in N. \tag{11}$$

Sätt $c_i := \Psi$, då är $\langle N, \mathbf{S}, (c_i)_{i\in N}\rangle$ ett koordinationsspel. Tag godtyckligt $i \in N$, $s_i, \tilde{s}_i \in S_i$ och $\mathbf{s}_{-i} \in \mathbf{S}_{-i}$. Definiera $d_i := u_i - \Psi$. Då gäller enligt ekvation 10 att

$$\begin{aligned} & u_i(\tilde{s}_i, \mathbf{s}_{-i}) - u_i(s_i, \mathbf{s}_{-i}) = \Psi(\tilde{s}_i, \mathbf{s}_{-i}) - \Psi(s_i, \mathbf{s}_{-i}) \\ \iff & u_i(\tilde{s}_i, \mathbf{s}_{-i}) - \Psi(\tilde{s}_i, \mathbf{s}_{-i}) = u_i(s_i, \mathbf{s}_{-i}) - \Psi(s_i, \mathbf{s}_{-i}) \\ \iff & d_i(\tilde{s}_i, \mathbf{s}_{-i}) = d_i(s_i, \mathbf{s}_{-i}) \end{aligned} \tag{12}$$

så $\langle N, \mathbf{S}, (d_i)_{i\in N}\rangle$ är ett dummy spel.

Antag det omvända, att $\langle N, \mathbf{S}, (c_i)_{i\in N}\rangle$ är ett koordinationsspel och $\langle N, \mathbf{S}, (d_i)_{i\in N}\rangle$ är ett dummy-spel. Låt $\Gamma = \langle N, \mathbf{S}, (u_i)_{i\in N}\rangle$ vara ett spel där $u_i = c_i + d_i$ för alla $i \in N$. Funktionen $\Psi := c_i$ för godtyckligt $i \in N$ är en potentialfunktion. Därav är spelet Γ ett potentialspel. □

2.5 Trängselspel.

Trängselspel är en välstuderad klass av potentialspel som är användbar för att beskriva spel över nätverk av olika slag, t.ex. dataflöden eller trafiknätverk. Det som kännetecknar trängselspel är att spelarna utnyttjar gemensamma resurser som t.ex. bandbredd eller vägstycken.

Exempel 8 (Vägnätverksspel)**.** Låt grafen $G = (V, E)$ representera ett vägnätverk, där noderna $v \in V$ är distinkta platser, och kanterna $e \in E$ är vägstycken mellan två olika platser. Vi har k personbilsförare som skall färdas från en plats till en annan. Målet för varje förare är att minimera sin egna restid.

Varje förare $i \in \{1, \ldots, k\}$ har ett ändligt antal vägar $s_i \in S_i$ att välja mellan, och varje väg är en sekvens av vägstycken $e \in E$ som sammankopplar förarens start- och slutpunkt. Restiden w_e längs varje vägstycke e beror på hur många spelare som valt en väg som innehåller det vägstycket. Den totala restiden för en spelare i är $\sum_{e \in s_i} w_e$, dvs. summan av restiderna längs varje vägstycke som ingår den valda vägen.

Definition 11. Ett *trängselspel* är en tupel $\Gamma = \langle N, F, \mathbf{S}, (w_f)_{f \in F} \rangle$ där:

- N är en ändlig mängd av spelare,
- F är en ändlig mängd av resurser,
- $\mathbf{S} = \prod_i S_i$, där S_i är en familj av delmängder till F för varje spelare $i \in N$,
- $w_f : \mathbb{N} \to \mathbb{R}_+$ är en icke-negativ funktion för varje resurs $f \in F$.

Låt $n_f(\mathbf{s})$ beteckna antalet spelare som använder resurs f givet strategiprofilen $\mathbf{s}$. I ett trängselspel ges nyttofunktionen för spelare i av

$$u_i(\mathbf{s}) := \sum_{f \in s_i} w_f(n_f(\mathbf{s})). \tag{13}$$

Till skillnad från allmänna potentialspel bestäms varje spelares nyttofunktion i ett trängselspel av vilka resurser spelaren använder, samt hur många andra som använder samma resurser.

Exempel 8, forts (Vägnätverksspel)**.** Om vi definierar en reellvärd funktion $w_e(n_e)$ för varje $e \in E$, där n_e är antalet förare som färdas på $e \in E$ kan vi definiera vägnätverket som ett trängselspel $\Gamma = \langle N, E, \mathbf{S}, (w_e)_{e \in E} \rangle$, med N och $\mathbf{S}$ som i definition 11.

Exempel 8 utgör basen för de exempel som vi i senare avsnitt presenterar. Vi visar nu att trängselspel är ett speciellt exempel av ett potentialspel.

Sats 3 ([3])**.** *Låt $\Gamma = \langle N, F, \mathbf{S}, (w_f)_{f \in F} \rangle$ vara ett godtyckligt trängselspel med $k = |N|$ spelare. Då är Γ ett potentialspel med potentialfunktion*

$$\Psi(\mathbf{s}) = \sum_{f \in F} \sum_{j=1}^{n_f(\mathbf{s})} w_f(j). \tag{14}$$

Bevis. Tag godtyckligt $i \in N$ och $\mathbf{s} \in \mathbf{S}$ och definiera

$$\Psi_{-i}(\mathbf{s}_{-i}) := \sum_{f \in F} \sum_{j=1}^{n_f(\mathbf{s}_{-i})} w_f(j). \tag{15}$$

där $n_f(\mathbf{s}_{-i}) = |\{j \in N \setminus \{i\} \mid f \in s_j\}|$. Vi delar upp resurserna i två olika fall. Om $f \in s_i$ så är $n_f(\mathbf{s}) = n_f(\mathbf{s}_{-i}) + 1$ och därav

$$\left(\sum_{j=1}^{n_f(\mathbf{s}_{-i})} w_f(j) \right) + w_f(n_f(\mathbf{s})) = \sum_{j=1}^{n_f(\mathbf{s}_{-i})+1} w_f(j) = \sum_{j=1}^{n_f(\mathbf{s})} w_f(j). \tag{16}$$

Om $f \in F \setminus s_i$ så är $n_f(\mathbf{s}) = n_f(\mathbf{s}_{-i})$ och därav

$$\sum_{j=1}^{n_f(\mathbf{s}_{-i})} w_f(j) = \sum_{j=1}^{n_f(\mathbf{s})} w_f(j). \tag{17}$$

Med dessa observationer ser vi att

$$\begin{aligned}
\Psi_{-i}(\mathbf{s}_{-i}) + u_i(\mathbf{s}) &= \sum_{f \in F} \sum_{j=1}^{n_f(\mathbf{s}_{-i})} w_f(j) + \sum_{f \in s_i} w_f(n_f(\mathbf{s})) \\
&= \sum_{f \in F \setminus s_i} \sum_{j=1}^{n_f(\mathbf{s}_{-i})} w_f(j) + \sum_{f \in s_i} \left[\sum_{j=1}^{n_f(\mathbf{s}_{-i})} w_f(j) + w_f(n_f(\mathbf{s})) \right] \\
&= \sum_{f \in F} \sum_{j=1}^{n_f(\mathbf{s})} w_f(j) \\
&= \Psi(\mathbf{s}), \text{ för alla } i \in N.
\end{aligned} \tag{18}$$

Det är enkelt att med hjälp av ekvation 18 verifiera att Ψ är en potentialfunktion. Låt $\tilde{s}_i \in S_i$ vara en godtycklig strategi. Då gäller

$$\begin{aligned}
\Psi(\tilde{s}_i, \mathbf{s}_{-i}) - \Psi(s_i, \mathbf{s}_{-i}) &= (\Psi_{-i}(\mathbf{s}_{-i}) + u_i(\tilde{s}_i, \mathbf{s}_{-i})) - (\Psi_{-i}(\mathbf{s}_{-i}) + u_i(s_i, \mathbf{s}_{-i})) \\
&= u_i(\tilde{s}_i, \mathbf{s}_{-i}) - u_i(s_i, \mathbf{s}_{-i}).
\end{aligned} \tag{19}$$

□

Även det omvända påståendet; att potentialspel är trängselspel har en viss sanning bakom sig. Monderer & Shapley [6] visade att det för varje ändligt potentialspel går att definiera ett trängselspel som är *isomorft* med potentialspelet. Med en isomorfi så menas här att det existerar bijektioner, g_i, för varje spelare mellan respektive strategirum sådana att

$$u_i(s_1, ..., s_k) = v_i(g_1(s_1), ..., g_n(s_k)) \tag{20}$$

där u_i är nyttofunktion i potentialspelet och v_i i trängselspelet. Därmed kan vi betrakta potentialspel och trängselspel som ekvivalenta även om de beskriver situationer på olika sätt. I Appendix A.2 ger vi ett bevis av det här påståendet från Voorneveld, Borm, Van Megen, Tijs & Facchini [12].

2.6 Anarkipriset

Eftersom potentialspel kan ha flera rena Nashjämvikter är det relevant att undersöka vilken av dessa som är att föredra. I detta avsnittet introducerar vi anarkipriset som ett kvantitativt mått på rena Nashjämvikter givet en målfunktion. Men först ger vi en definition av potentialspel där målet för varje spelare är att minimera sin nyttofunktion.

Definition 12. Ett **kostnadsminimeringsspel** är ett potentialspel $\Gamma = \langle N, \mathbf{S}, (c_i)_{i \in N} \rangle$, där $c_i : \mathbf{S} \to \mathbb{R}$ är **kostnadsfunktionen** för alla spelare i.

Anmärkning 3. När vi refererar till ett trängselspel låter vi det vara underförstått att det är ett kostnadsminimeringsspel som avses.

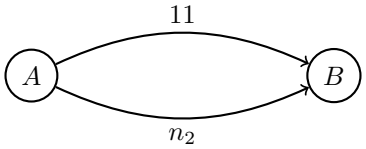
Definition 12 är endast en konvention, alla resultat som följer nedan går att översätta till spel där varje spelare istället vill maximera sin utdelning. Anledningen till att vi inför konventionen är främst historisk, då de resulat vi presenterar ursprungligen visades för kostnadsminimeringsspel.

Exempel 9. Låt $G = (V, E)$ vara en graf med två noder, A och B, och två kanter, e_1 och e_2, mellan dem. Likt Exempel 8 definierar vi ett trängselspel med 10 spelare som skall åka från plats

A till plats B. Alla spelare väljer mellan e_1 och e_2, dvs. $S_i = \{e_1, e_2\}$ för alla $i = 1, ..., 10$. Restiden längs vägstyckena definieras av

$$\begin{cases} w_{e_1}(n) := 11, \\ w_{e_2}(n) := n. \end{cases} \tag{21}$$

Restiden för varje spelare är samma som restiden längs vägstycket, eftersom varje strategi bara innehåller ett vägstycke. Låt n_1 och n_2 beteckna antalet spelare som väljer e_1 resp. e_2. Figur 4 illustrerar vägnätverket och restiderna längs varje vägstycke.



Figur 4: Vägnätverket i Exempel 9 med restider längs de båda vägstyckena utmarkerade.

Spelet har en ren Nashjämvikt då alla spelare väljer e_2. En naturlig frågeställning är huruvida Nashjämvikten är den strategiprofil som minimerar den genomsnittliga restiden. Vi undersöker detta genom att studera den totala restiden, eftersom denna är minimal då den genomsnittliga restiden är minimal. Den totala restiden för en godtycklig strategiprofil $\mathbf{s}$ ges av

$$\sum_{i \in N} c_i(\mathbf{s}) = 11 \cdot w_{e_1}(n_1) + n_2 \cdot w_{e_2}(n_2) = 11n_1 + (n_2)^2 = 11n_1 + (10 - n_1)^2. \tag{22}$$

Den totala restiden blir $\sum_i c_i = 100$ om alla spelare väljer e_2. Vi kan dock observera att om exakt en spelare, säg spelare i, istället väljer e_1 minskar den totala restiden till 92. Då befinner vi oss dock ej längre i en ren Nashjämvikt, eftersom spelare i kan förbättra sin egna restid genom att byta tillbaka till e_2. Den totala restiden minimeras med $\sum c_i = 80$ om 4 eller 5 spelare väljer e_1 och övriga spelare väljer e_2. Dock är ingen av dessa två strategiprofiler en ren Nashjämvikt.

Exempel 9 illustrerar en ren Nashjämvikt där den totala kostnaden inte är den minsta möjliga. En liknande analys kan göras med en godtycklig målfunktion. Hädanefter kommer vi dock att begränsa oss till funktionen

$$C(\mathbf{s}) = \sum_{i \in N} c_i(\mathbf{s}), \tag{23}$$

dvs. summan av kostnaden för varje spelare, vilket också är målfunktionen som användes i Exempel 9. Funktionen kallas ibland för "utilitaristfunktionen" och är en av de vanligaste målfunktionerna att betrakta i sådana här sammanhang [8, s. 443].

Definition 13. Låt Γ vara ett kostnadsminimeringsspel, sätt $C^* = \min_{\mathbf{s} \in \mathbf{S}} C(\mathbf{s})$ och antag att $C^* > 0$.[1] **Anarkipriset** $\rho(\Gamma)$ för Γ definieras som

$$\rho(\Gamma) = \frac{C(\tilde{\mathbf{s}})}{C^*}, \tag{24}$$

där $\tilde{\mathbf{s}}$ är den rena Nashjämvikt som ger störst värde på C. Om det inte råder tvivel om vilket spel vi hänvisar till skriver vi endast ρ.

Anarkipriset introducerades först av Papadimitriou [9] och kallades ursprungligen för "koordinationsratio" då det kvantifierar spelarnas brist på koordinering i ett spel. I spelet från Exempel 9 är anarkipriset 5/4.

[1] Anarkipriset definieras i [8] för godtyckliga spel och jämvikter.

3 Trängselspel med linjära kostnadsfunktioner

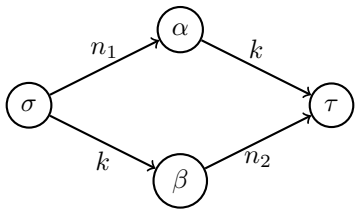
I det föregående avsnittet presenterade vi iterativt spelande, etablerade en användbar koppling mellan potentialspel och trängselspel och avslutade sedan med att introducera anarkipriset. Vi tar nu ett steg från det generella och studerar en speciell typ av trängselspel; spel över vägnätverk där restiderna bestäms av linjära funktioner.

3.1 Diskret tolkning av Braess paradox

Tidigare har Braess paradox [1] studerats som ett spel med oändligt antal spelare, se till exempel [3] s. 148 - 149. Vi analyserar här samma nätverk men definierar istället spelet med ett ändligt antal spelare. Analysen delas upp i två delar. I 3.1.1 introducerar vi ett enkelt trängselspel och bestämmer dess unika rena Nashjämvikt och anarkipris. I 3.1.2 lägger vi till ett vägstycke som ger upphov till en ny strategi och undersöker dess effekt med avseende på rena Nashjämvikter och anarkipris.

3.1.1 Det ursprungliga vägnätverket

Låt Γ_1 beteckna ett vägnätverksspel med $k \geq 4$ spelare och 4 platser, σ, α, β och τ. Vägstyckena i Γ_1 är riktade och utgörs av $\{(\sigma,\alpha),(\alpha,\tau),(\sigma,\beta),(\beta,\tau)\}$. Varje spelare vill hitta den snabbaste vägen mellan σ och τ, och det finns två strategier att välja bland, $s_1 := (\sigma,\alpha,\tau)$ och $s_2 := (\sigma,\beta,\tau)$. Längs (σ,β) och (α,τ) är restiden konstant lika med antalet spelare k, och längs (σ,α) och (β,τ) är restiden lika med antalet spelare n_1, n_2 som använder respektive vägstycke. Se Figur 5 för en illustration av spelet.



Figur 5: Vägnätverksspelet Γ_1 med k spelare. Längs kanterna (σ,α) och (β,τ) är restiden lika med antalet spelare som använder respektive vägstycke och för de två andra vägstycken är restiden konstant lika med antalet spelare.

Observera att n_1 och n_2 sammanfaller med antalet spelare som väljer s_1 resp. s_2. Den totala restiden $C(\mathbf{s})$ i Γ_1 givet en strategiprofil $\mathbf{s}$ är

$$\begin{aligned} C(\mathbf{s}) &= n_1(n_1+k) + n_2(n_2+k) \\ &= n_1^2 + n_2^2 + k^2 \\ &= (n_1+n_2)^2 - 2n_1n_2 + k^2 \\ &= 2k^2 - 2n_1n_2. \end{aligned} \tag{25}$$

För inte fastna i tekniska detaljer antar vi att antalet spelare k är jämnt, dvs. $k = 2n$ för något heltal $n \geq 2$. Vi undersöker den totala restiden för olika värden på n_1, n_2 genom att sätta $n_1 = n + m$, där m är ett heltal sådant att $-n \leq m \leq n$. Den totala restiden kan därmed skrivas som

$$2k^2 - 2n_1n_2 = 2k^2 - 2(n+m)(n-m) = 2k^2 - 2(n^2-m^2) = \frac{3}{2}k^2 + 2m^2. \tag{26}$$

Uttrycket i ekvation 26 minimeras när $m = 0 \Rightarrow n_1 = n_2 = n$, dvs. då hälften av spelarna väljer s_1 och övriga spelare väljer s_2. Den totala restiden blir $2k^2 - 2n^2 = \frac{3}{2}k^2$. Vi kallar denna strategiprofilen för $\mathbf{s}^*$ och visar nu att $\mathbf{s}^*$ också är den unika rena Nashjämvikten i Γ_1.

Proposition 2. *Låt Γ_1 och $\mathbf{s}^*$ vara som ovan. En ren strategiprofil $\mathbf{s}$ i Γ_1 är en ren Nashjämvikt om och endast om $\mathbf{s} = \mathbf{s}^*$.*

Bevis. Låt $\mathbf{s}$ vara en strategiprofil i Γ_1. Antalet spelare som väljer s_1, s_2 sammanfaller med n_1 resp. n_2, och kostnaderna för strategierna är $n_1 + k$ resp. $n_2 + k$. Antag att en spelare, säg spelare i, byter strategi från s_1 till s_2. Eftersom bytet ökar n_2 med 1 lönar sig bytet för spelare i om och endast om

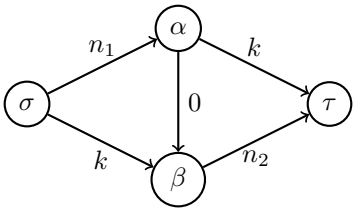
$$n_1 + k > (n_2 + 1) + k \Rightarrow n_1 > n_2 + 1. \tag{27}$$

Eftersom k är jämnt, dvs. $n_1 - n_2 = \pm 2m, m \in \mathbb{Z}$ innebär detta att spelare i byter till s_2 om och endast om $n_1 > n_2$. På grund av symmetri gäller även det omvända, dvs. ett byte från s_2 till s_1 lönar sig om och endast om $n_2 > n_1$. Alltså är $\mathbf{s}$ en Nashjämvikt om och endast om $n_1 = n_2$. □

Anmärkning 4. Eftersom $\mathbf{s}^*$ både är den enda rena Nashjämvikten och den strategiprofil som minimerar den totala restiden, har Γ_1 anarkipriset $\rho(\Gamma_1) = 1$.

3.1.2 Det utvidgade vägnätverket

Vi undersöker vad som händer då vi lägger till ett riktat vägstycke mellan α och β med restid 0. Vägstycket ger upphov till en ytterligare strategi $s_3 = (\sigma, \alpha, \beta, \tau)$ med restiden $n_1 + n_2$. Vi betecknar det nya spelet med Γ_2. Se Figur 6 för en illustration av det utvidgade spelet.



Figur 6: Utvidgning av vägnätverksspelet Γ_1 till Γ_2 med en kostnadsfri kant från α till β.

Vi antar återigen att $k = 2n$ för något heltal $n \geq 2$. De två huvudsakliga konsekvenserna av den här utvidgningen till Γ_2 som vi nu vill visa är

(i) Strategiprofilen $\mathbf{s}^*$ är inte längre en ren Nashjämvikt,

(ii) Strategiprofilen $\mathbf{s}^*$ minimerar fortfarande den totala restiden.

Bevis av (i). Det räcker att undersöka fallet där en spelare byter till s_3 (byte mellan s_1 och s_2 behandlades i 3.1.1). Antag att $\mathbf{s}^*$ är den nuvarande strategiprofilen, och att spelare i byter strategi från s_1 till s_3 (byte från s_2 till s_3 följer av symmetriskäl). Notera att bytet ökar n_2 till $n_2 + 1$ men lämnar n_1 oförändrad. Restiden för spelare i ändras enligt

$$c_i(s_1, \mathbf{s}^*_{-i}) = n_1 + k = \frac{3}{2}k > k + 1 = n_1 + (n_2 + 1) = c_i(s_3, \mathbf{s}^*_{-i}). \tag{28}$$

Eftersom spelare i minskar sin restid genom att byta till s_3 är $\mathbf{s}^*$ inte en ren Nashjämvikt. □

Bevis av (ii). Låt m_i beteckna antalet spelare som väljer strategi $i = 1, 2, 3$. Strategin s_3 utnyttjar både (σ, α) och (β, τ). Antalet spelare som använder vägstycket (σ, α) blir då summan av antalet spelare som väljer s_1 och s_3, dvs. $m_1 + m_3$. För en godtycklig strategiprofil $\mathbf{s}$ gäller alltså

$$\begin{cases} k = m_1 + m_2 + m_3, \\ n_1 = m_1 + m_3, \\ n_2 = m_2 + m_3. \end{cases} \tag{29}$$

Med dessa samband skriver formulerar vi om uttrycket för den totala restiden,

$$\begin{aligned} C(\mathbf{s}) &= m_1(n_1 + k) + m_2(n_2 + k) + m_3(n_1 + n_2) \\ &= m_1(m_1 + m_3 + k) + m_2(m_2 + m_3 + k) + m_3(m_1 + m_2 + 2m_3) \\ &= m_1(m_1 + m_3 + k) + m_2(m_2 + m_3 + k) + m_3(k + m_3). \end{aligned} \tag{30}$$

Vi tar ett nytt andetag och fortsätter med

$$\begin{aligned} &= k(m_1 + m_2 + m_3) + [m_1^2 + m_2^2 + m_3^2] + m_3(m_1 + m_2) \\ &= k^2 + [(m_1 + m_2 + m_3)^2 - 2(m_1m_2 + m_1m_3 + m_2m_3)] + m_3(m_1 + m_2) \\ &= k^2 + k^2 - 2m_1m_2 - m_1m_3 - m_2m_3 \\ &= 2k^2 - (2m_1m_2 + m_1m_3 + m_2m_3). \end{aligned} \tag{31}$$

Uttrycket (31) står nu på samma form som uttrycket i ekvation 25. Det räcker därmed att visa att den högra termen är mindre än $\frac{1}{2}k^2$, eftersom vi då har att $C(\mathbf{s}) \geq \frac{3}{2}k^2 = C(\mathbf{s}^*)$ för alla $\mathbf{s} \in \mathbf{S}$. Vi fortsätter omskrivningen av den andra termen med

$$\begin{aligned} 2m_1m_2 + m_1m_3 + m_2m_3 &= m_1m_2 + [m_1m_2 + m_1m_3 + m_2m_3] \\ &= m_1m_2 + \tfrac{1}{2}\left[(m_1 + m_2 + m_3)^2 - (m_1^2 + m_2^2 + m_3^2)\right] \\ &= \tfrac{1}{2}\left[k^2 - (m_1^2 - 2m_1m_2 + m_2^2 + m_3^2)\right] \\ &= \tfrac{1}{2}k^2 - \tfrac{1}{2}\left[(m_1 - m_2)^2 + m_3^2\right] \\ &\leq \tfrac{1}{2}k^2. \end{aligned} \tag{32}$$

Detta visar att $C(\mathbf{s}) \geq C(\mathbf{s}^*)$ för alla strategiprofiler $\mathbf{s} \in \mathbf{S}$. Alltså är $\mathbf{s}^*$ den strategiprofil som minimerar den totala restiden. □

Anmärkning 5. Den totala restiden för en godtycklig strategiprofil $\mathbf{s}$ i Γ_2 ges av

$$C(\mathbf{s}) = \tfrac{3}{2}k^2 + \tfrac{1}{2}\left[(m_1 - m_2)^2 + m_3^2\right].$$

Låt nu $\tilde{\mathbf{s}}$ beteckna strategin "alla spelare väljer s_3". Vi noterar att $\tilde{\mathbf{s}}$ är en ren Nashjämvikt, ty om någon spelare i byter strategi från s_3 till s_1 (byte till s_2 behandlas analogt) används kant (σ, α) fortfarande av $n_1 = k$ spelare, dvs. $c_i(s_1, \tilde{\mathbf{s}}) = 2k = c_i(s_3, \tilde{\mathbf{s}})$.

Den totala restiden i Γ_2 minimieras av $\mathbf{s}^*$, alltså när hälften av spelarna väljer s_1 och resten väljer s_2. Det följer av ekvation 31 att den totala restiden $C(\mathbf{s})$ aldrig kan överstiga $2k^2 = k(k+k) = C(\tilde{\mathbf{s}})$. Alltså är $\tilde{\mathbf{s}}$ den sämsta Nashjämvikten med avseende på den totala restiden. Därmed är anarkipriset för det utvidgade spelet

$$\rho(\Gamma_2) = \frac{C(\tilde{\mathbf{s}})}{C^*} = \frac{2k^2}{2k^2 - 2n^2} = \frac{k^2}{k^2 - \frac{k^2}{4}} = \frac{4}{3}. \tag{33}$$

Den totala restiden i en ren Nashjämvikt kan alltså öka med en faktor upp till och med $\frac{4}{3}$, trots att det enda vi har gjort är att erbjuda spelarna en ytterligare strategi. Hur är det möjligt? När en enskild spelare byter strategi tar hen ej hänsyn till hur det påverkar de andra spelarnas restider. I Γ_2 verkar den individuella förbättringen överskuggas av den ökade totala restiden.

Vi har sett hur ett trängselspel med anarkipris $\rho = 1$ kan störas till den grad att den strategiprofil som minimerar den totala restiden inte längre är en ren Nashjämvikt, enbart genom att införa en ny strategi. Ett sätt att förhindra att sådana situationer uppstår är att tillåta samarbete. Spelarna kan uppnå den optimala restiden om de kommer överens om att fördela sig jämnt över s_1 och s_2. Situationen kommer dock att vara instabil eftersom varje enskild spelare skulle minska sin egen restid genom att bryta överenskommelsen.

I situationer med många spelare är det således kanske naivt att anta att spelarna faktiskt väljer att samarbeta. I 3.1.2 visade vi hur en ny resväg kan försämra restiden för alla. Det omvända gäller också, att vi kan manipulera spelets struktur så att spelarna istället dras mot en Nashjämvikt som är bättre för alla, genom att eliminera alternativ. Konsekvensen blir att vi istället tvingar fram en ren Nashjämvikt som är bättre för alla.

3.2 Övre begränsing av anarkipriset

För att beräkna anarkipriset för ett godtyckligt potentialspel behöver vi generellt sett information om spelets rena Nashjämvikter. Däremot går det i speciella fall att uppåt begränsa anarkipriset utan explicit information om alla rena Nashjämvikter. Följande sats är ett sådant resultat.

Sats 4 ([11]). *Låt $\Gamma = \langle N, F, \mathbf{S}, (w_f)_{f\in F}\rangle$ vara ett trängselspel med $w_f(n) := a_f \cdot n + b_f$ där $a_f, b_f \geq 0$ för alla $f \in F$. Då är*

$$\rho(\Gamma) \leq 5/2. \tag{34}$$

Vi använder följande lemma för att bevisa Sats 4.

Lemma 1. *Låt $n, m \in \mathbb{N}$. Då gäller att*

$$n(m+1) \leq \frac{5}{3}n^2 + \frac{1}{3}m^2. \tag{35}$$

Bevis av Lemma 1. Låt $g(n,m) := 5n^2 + m^2 - 3n(m+1)$ för alla heltal n och m. Ekvation 35 är ekvivalent med att $g(n,m) \geq 0$ för alla icke-negativa heltal n och m. Fallen $n = 0$ eller $m = 0$ och $m = n = 1$ är uppenbara. Om å andra sidan n eller m är större än 1 och $n, m \neq 0$ så är $m+n \geq 3$ och därav är

$$\begin{aligned} g(n,m) &= 5n^2 + m^2 - 3n(m+1) \\ &= 5n^2 + m^2 - 3nm - 3n \\ &= 4n^2 + m^2 - 4nm + n(n+m-3) \\ &= (2n-m)^2 + n(n+m-3) \geq 0. \end{aligned} \tag{36}$$

□

Bevis av Sats 4. Låt $\mathbf{s}^*$ vara en strategiprofil som minimerar C och $\mathbf{s}$ en godtycklig ren Nashjämvikt. Definiera för varje $f \in F$ $n_f(\mathbf{s}) := |\{i \in N \mid f \in s_i\}|$ och $n_f(\mathbf{s}^*) := |\{i \in N \mid f \in s_i^*\}|$, dvs. antalet spelare som använder resurs f givet de två strategiprofilerna $\mathbf{s}$ och $\mathbf{s}^*$. Notera att

$$C(\mathbf{s}) = \sum_{i\in N} c_i(\mathbf{s}) = \sum_{i\in N}\sum_{f\in s_i} a_f \cdot n_f(\mathbf{s}) + b_f = \sum_{f\in F} n_f(\mathbf{s}) \cdot [a_f \cdot n_f(\mathbf{s}) + b_f] \tag{37}$$

om vi summerar över F istället för N. Om spelare i byter från s_i till s_i^* kan antalet spelare som använder en godtycklig resurs f öka med som mest 1, dvs. $n_f(s_i^*, \mathbf{s}_{-i}) \leq n_f(\mathbf{s}) + 1$. Därav är

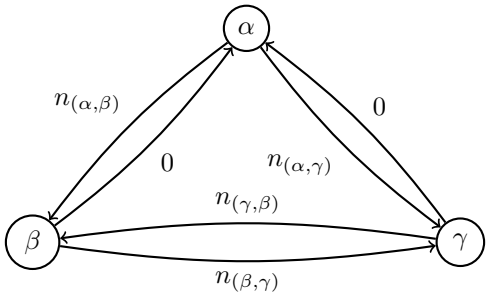
$$\sum_{i\in N} c_i(s_i^*, \mathbf{s}_{-i}) \leq \sum_{i\in N}\sum_{f\in s_i^*} a_f \cdot (n_f(\mathbf{s}) + 1) + b_f = \sum_{f\in F} n_f(\mathbf{s}^*) \cdot [a_f \cdot (n_f(\mathbf{s}) + 1) + b_f], \tag{38}$$

om vi summerar över F istället för N, likt ekvation 37. Eftersom $\mathbf{s}$ är en ren Nashjämvikt så är $c_i(\mathbf{s}) \leq c_i(s_i^*, \mathbf{s}_{-i})$ för alla $i \in N$. Lemma 1 ger oss att $n_f(\mathbf{s}^*)(n_f(\mathbf{s})+1) \leq \frac{5}{3}(n_f(\mathbf{s}^*))^2 + \frac{1}{3}(n_f(\mathbf{s}))^2$. Därav

$$\begin{aligned} C(\mathbf{s}) &\leq \sum_{i\in N} c_i(s_i^*, \mathbf{s}_{-i}) \\ &\leq \sum_{f\in F} n_f(\mathbf{s}^*) \cdot [a_f \cdot (n_f(\mathbf{s}) + 1) + b_f] \\ &= \sum_{f\in F} [a_f \cdot n_f(\mathbf{s}^*) \cdot (n_f(\mathbf{s}) + 1) + b_f \cdot n_f(\mathbf{s}^*)] \\ &\leq \sum_{f\in F} [a_f \left(\frac{5}{3}((n_f(\mathbf{s}^*))^2 + \frac{1}{3}(n_f(\mathbf{s}))^2\right) + b_f \cdot n_f(\mathbf{s}^*)] \\ &\leq \frac{5}{3}\sum_{f\in F} n_f(\mathbf{s}^*)(a_f \cdot n_f(\mathbf{s}^*) + b_f) + \frac{1}{3}\sum_{f\in F} a_f (n_f(\mathbf{s}))^2 \\ &\leq \frac{5}{3}C(\mathbf{s}^*) + \frac{1}{3}C(\mathbf{s}). \end{aligned} \tag{39}$$

Av ekvation 39 följer att $C(\mathbf{s}) \leq \frac{5}{2}C(\mathbf{s}^*)$ som avslutar beviset. □

Vi avslutar avsnittet med ett exempel där anarkipriset är maximalt samtidigt som det också finns en ren Nashjämvikt som minimerar den totala restiden.



Figur 7: Trängselspel med maximalt anarkipris

Exempel 10 ([3, s. 160])**.** Figur 7 visar ett vägnätverk i ett trängselspel. Spelet har 4 spelare som alla reser mellan olika destinationer. Vägnätverket har noderna α, β och γ som alla är sammankopplade i båda rikningarna. Alla vägstycken, förutom (γ, α) och (β, α) där restiden är 0, har restid lika med antalet spelare som färdas på vägen. Spelarnas strategirum är

$$\begin{aligned} &\mathbf{S}_0 = \{(\alpha,\beta),(\alpha,\gamma,\beta)\},\ \mathbf{S}_1 = \{(\alpha,\gamma),(\alpha,\beta,\gamma)\},\\ &\mathbf{S}_2 = \{(\beta,\gamma),(\beta,\alpha,\gamma))\},\ \mathbf{S}_3 = \{(\gamma,\beta),(\gamma,\alpha,\beta)\}. \end{aligned} \tag{40}$$

För enkelhetens skull ser vi det som att varje spelare kan välja mellan en "lång" (två vägstycken) och en "kort" (ett vägstycke) resväg. Låt ℓ_i och κ_i stå för "lång" respektive "kort" resväg för spelare i.

Sätt $\boldsymbol{\kappa} := (\kappa_0, \kappa_1, \kappa_2, \kappa_3)$. Vi ser att $c_i(\boldsymbol{\kappa}) = 1$ för alla spelare $i = 0, ..., 3$, och eftersom ingen spelare kan välja en väg med restid 0 måste $\boldsymbol{\kappa}$ både vara en ren Nashjämvikt, och en strategiprofil som minimerar den totala restiden. Även strategiprofilen $\boldsymbol{\ell} := (\ell_0, \ell_1, \ell_2, \ell_3)$ är en ren Nashjämvikt eftersom

$$c_i(\kappa_i, \boldsymbol{\ell}_{-i}) = c_i(\boldsymbol{\ell}), \text{ för alla } i = 0, 1, 2, 3. \tag{41}$$

Om vi jämför de totala restiderna för de två Nashjämvikterna ser vi att $C(\boldsymbol{\kappa}) = 4$ och $C(\boldsymbol{\ell}) = 10$ och därav är $\rho = \frac{C(\boldsymbol{\kappa})}{C(\boldsymbol{\ell})} = 5/2$.

Slutsatsen av det här exemplet är att vi har hittat ett trängselspel med linjära restidsfunktioner där anarkipriset, enligt Sats 4, är maximalt för den här typen av spel. Samtidigt existerar en ren Nashjämvikt då alla väljer sin "korta" resväg, där den totala restiden för alla spelare är minimal.

3.3 Slutsats

Vi har redogjort för en del kända resultat inom studiet av trängselspel och anarkipriset. Dock kan antagandet om linjära kostnadsfunktioner ifrågasättas när det kommer till verkliga modeller av trafik. En möjlig fördjupning är att se hur väl dessa resultat generaliseras för andra typer av kostnadsfunktioner.

Det vore även upplysande att utforska förekomst av övre begränsningar av anarkpriset i spel med icke-linjära kostnadsfunktioner. Kan ρ anta större värden än $\frac{5}{2}$ i spel med exponentiella restidsfunktioner? Ger logaritmiska funktioner en lägre övre begränsning på ρ? Roughgardens [11] artikel om hur man kan bestämma anarkipriset i trängselspel skulle kunna vara en bra utgångspunkt för att besvara de här frågorna.

Innebörden av vad restid är i det här sammanhanget kan också diskuteras. Våra spel är avgjorda innan en förare lämnar sitt hem. En mer dynamisk syn på vad som händer då förare färdas i nätverket, och omvärderar sin strategi, vore intressant att studera. En möjlighet kan vara att istället definiera ett spel där spelare kan ändra sin strategi vid givna tidpunkter eller noder i nätverket.

4 Iterativt spelande i potentialspel

I det här avsnittet undersöker vi effektiviteten av en algoritm baserad på iterativt spelande. Vi beskriver först hur man kan konstruera potentialspel med godtyckliga värden på potentialfunktionen, och använder sedan detta för att generera *slumpade potentialspel*. Därefter definierar vi en algoritm och tillämpar den på ett stort antal slumpade potentialspel för att undersöka hur många iterationer som krävs för att nå en Nashjämvikt.

4.1 Konstruktion av potentialspel

Sats 2 i avsnitt 2.4 ger oss en metod att konstruera potentialspel med godtyckliga värden på potentialfunktionen. Vi konstruerar först ett koordinationsspel, där alla spelare har samma nyttofunktion, och lägger därtill dummyspelet, där varje enskilt byte av strategi lämnar utdelningen oförändrad. Vi börjar med att betrakta ett enkelt exempel.

Exempel 11 (Konstruktion av potentialspel)**.** Vi konstruerar ett potentialspel med 2 spelare dvs. $N = \{1, 2\}$, med 2 strategier vardera. Låt $C = (c_{ij})$ beteckna utdelningsmatrisen för ett koordinationsspel, och låt D_1, D_2 beteckna ett dummyspels utdelningsmatriser för spelare 1 och 2. I alla matriser representerar rad i och kolumn j valet av strategi för spelare 1 resp. 2. dvs. $\mathbf{S} = \{1, 2\} \times \{1, 2\}$.

Sätt $U_n := C + D_n$ för $n = 1, 2$. Enligt Sats 2 beskriver $\langle N, \mathbf{S}, U_n \rangle$ ett potentialspel med potentialfunktionen $\Psi(i, j) = c_{ij}$. Vi gör det konkret genom att sätta in värdena

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad D_1 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \quad \text{och} \quad D_2 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}. \tag{42}$$

I D_1 resp. D_2 är utdelningen oberoende av valet av rad resp. kolumn. I utdelningsmatrisrna

$$U_1 := C + D_1 = \begin{bmatrix} 1 & 3 \\ 3 & 5 \end{bmatrix}, \quad U_2 := C + D_2 = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} \tag{43}$$

väljer spelarna strategi genom att bestämma varsin koordinat i utdelningsmatrisen.

I spel med fler än två strategier utökar vi bara nyttomatrisen till motsvarande storlek. När vi konstruerar spel med fler än två spelare måste vi dock lämna matrisrepresentationen och återgå till den generella definitionen av nyttofunktionen.

4.1.1 Potentialspel med godtyckligt antal spelare

Låt Γ beteckna ett potentialspel med k spelare och n strategier vardera. Beteckna strategierna för varje spelare i enligt $S_i := (s_1^i, \ldots, s_n^i)$. Nyttofunktionen för spelare i får utseendet $u_i(\mathbf{s}) = u_i(s_j^i, \mathbf{s}_{-i})$ om spelare i väljer sin j:te strategi. Vi noterar att Γ har $|\mathbf{S}| = n^k$ möjliga strategiprofiler. Som vanligt låter vi Ψ beteckna potentialfunktionen.

Enligt Sats 2 kan nyttofunktionen för spelare i skrivas $u_i = \Psi + d_i$ där d_i är nyttunktionen i ett dummyspel. Det enklaste dummyspelet ges av $d_i := 0$ för alla i. Då vi under återstoden av avsnittet inte bryr oss om inbördes skillander mellan olika spelares nyttofunktioner sätter vi $u_i := \Psi$ för alla i.

Anmärkning 6. Man kan konstruera mer generella potentialspel, där dummyspelets nyttofunktioner inte är identiskt noll, genom att

(i) sätta godtyckliga värden på $d_i(s_1^i, \mathbf{s}_{-i}),\ i = 1, \ldots, k$,

(ii) sätta $d_i(s_j^i, \mathbf{s}_{-i}) := d_i(s_1^i, \mathbf{s}_{-i})$ för alla $j \neq 1$ och $i = 1, \ldots, k$.

I Appendix B.1 demonstrerar vi tillvägångssättet med en implementation i ett Pythonscript.

4.1.2 Slumpade potentialspel

För att kunna undersöka ett stort antal potentialspel genererar vi *slumpade potentialspel*, dvs. potentialspel med slumpmässiga värden på potentialfunktionen. Detta gör vi genom att för varje strategiprofil tilldela potentialfunktionen ett slumpmässigt värde ur en kontinuerlig likformig fördelning över intervallet $[0, 1)$.

Eftersom antalet strategiprofiler $|\mathbf{S}| = n^k$ ökar polynomiellt med antalet strategier n, och exponentiellt med antalet spelare k, når vi snabbt en övre begränsning med avseende på datorkapacitet. För t.ex. $n = k = 10$ behöver vi generera och lagra 10^{10} värden på potentialfunktionen.

För att hushålla med datorresurserna utnyttjar vi att värdena på potentialfunktionen kan representeras av en familj av *oberoende och identiskt fördelade* slumpvariabler. Detta gör att vi inte behöver generera alla värden på potentialfunktionen samtidigt, utan istället generera dem efterhand medan vi undersöker olika strategiprofiler $\mathbf{s} \in \mathbf{S}$.[2]

Vi knyter ihop säcken med en matematisk formulering av delavsnittet.

Definition 14. Låt $\mathbf{S}$ vara strategirummet i ett spel Γ, och låt $X_{\mathbf{s}} : \mathbf{s} \in \mathbf{S}$, vara en familj av kontinuerliga slumpvariabler likformigt fördelade på intervallet $[0, 1)$. Sätt

$$\Psi(\mathbf{s}) := X_{\mathbf{s}}. \tag{44}$$

Om $\Gamma = \langle N, \mathbf{S}, (u_i)_{i \in N} \rangle$ är ett spel sådant att $u_i = \Psi$ för alla $i \in N$ kallar vi Γ för ett **slumpat potentialspel** med en **slumpad potentialfunktion** Ψ.

Anmärkning 7. Vi väljer intervallet $[0, 1)$ som målmängd för att det är programmeringstekniskt praktiskt. Man kan givetvis definiera en slumpad potentialfuntktion med en annan målmängd.

4.2 En algoritm baserad på iterativt spelande

Vi påminner oss om att iterativt spelande inleds genom att välja en godtycklig strategiprofil. Sedan låter vi en spelare byta strategi under förutsättningen att bytet strikt förbättrar spelarens nyttofunktion. Vi upprepar förfarandet tills ingen enskild spelare längre kan förbättra sin utdelning genom att byta strategi.

En viktig komponent i proceduren, och något som vi fram tills nu har sopat under mattan, är en beslutsregel som avgör vilken av spelarna med förbättrande strategier som ska byta strategi. Vi väljer att låta den spelare som kan förbättra sin nyttofunktion mest byta strategi, eftersom vår intuition säger oss att detta borde göra algoritmen effektiv med avseende på antalet iterationer.

Vi utnyttjar definitionen av potentialfunktionen för att förenkla proceduren. Eftersom potentialfunktionen ökar precis lika mycket som nyttofunktionen då en spelare byter strategi, räcker det att undersöka potentialfunktionens värde i varje iteration. Algoritmen får följande utseende.

(1) Välj slumpmässigt en strategiprofil $\mathbf{s}^0$. Sätt $t := 0$.

(2) (a) För varje spelare i, sätt $s_i^* := \arg\max_{s_i \in S_i}(\Psi(s_i, \mathbf{s}_{-i}^t))$.

(b) Sätt $\mathbf{s}^{t+1} := \arg\max_{\mathbf{s} \in \mathbf{S}^*} \Psi(\mathbf{s})$, där $\mathbf{S}^* = \{(s_i^*, \mathbf{s}_{-i}^t) : i \in N\}$.

(3) Om $\Psi(\mathbf{s}^{t+1}) = \Psi(\mathbf{s}^t)$ är $\mathbf{s}^t$ en Nashjämvikt. Avbryt.

(4) Uppdatera $t := t + 1$ och upprepa från steg 2.

Anmärkning 8. Om flera strategiprofiler i steg 2 och 3 ger samma värden på potentialfunktionen kan man exempelvis välja slumpmässigt bland dessa. När vi implementerar algoritmen nedan väljer vi istället alltid den med lägst index.

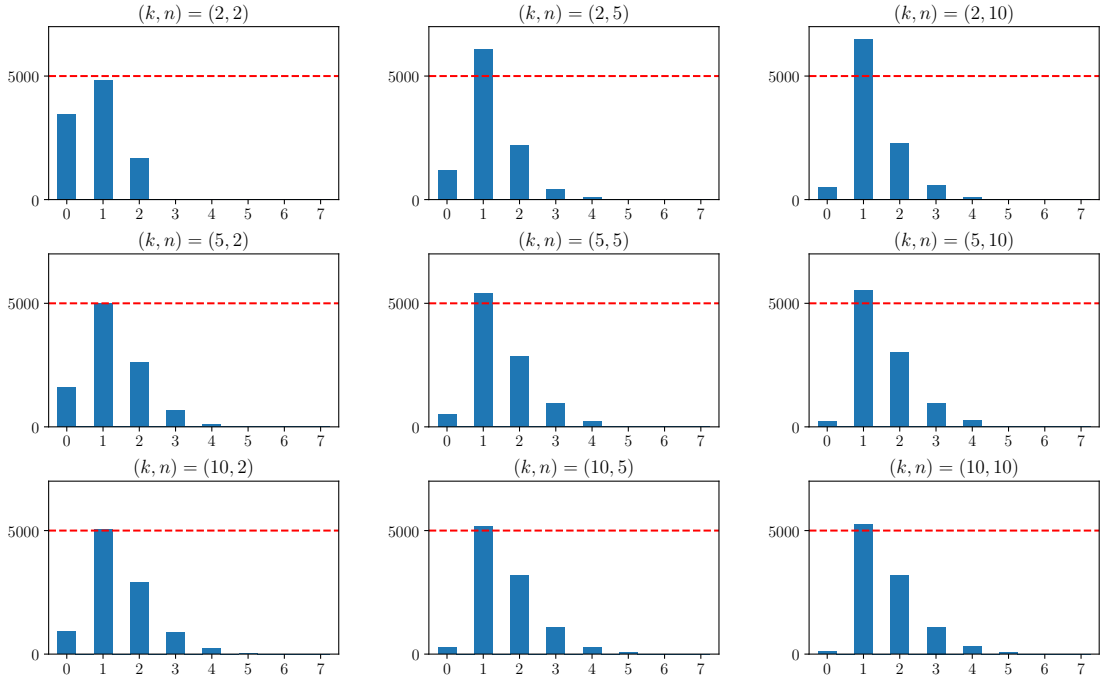
Under återstoden av avsnitt 4 låter vi $\mathcal{A}$ beteckna algoritmen vi just definierat.

[2] Vi lärde oss om den här metoden ur Durand & Gaujal [2] efter att de preliminära resultaten var producerade. Metoden hjälpte oss drastiskt förbättra prestandan på koden.

4.3 Simulering av potentialspel

Vi tillämpar nu algoritmen $\mathcal{A}$ från avsnitt 4.2 på slumpade potentialspel genererade enligt beskrivningen i avsnitt 4.1.2. Vi låter antalet spelare k och antalet strategier n variera och genererar 10,000 slumpade potentialspel för varje kombination av k och n.

För varje spel genererar vi en slumpmässig strategiprofil som startpunkt i $\mathcal{A}$, och för varje spel bokför vi antalet förbättrande steg som algoritmen tar innan den når en Nashjämvikt. Figur 8 visar histogram över antalet steg i 10,000 slumpade koordinationsspel för några olika värden på k och n. Se Appendix B.2.1 och B.2.2 för programkod.



Figur 8: Histogram över antalet steg i $\mathcal{A}$ för olika värden på k, n i 10,000 slumpade potentialspel.

En slående observation är att algoritmen når en Nashjämvikt efter endast ett steg i ungefär hälften av spelen, oavsett antal spelare och strategier. I allmänhet verkar algoritmen konvergera efter ett litet antal steg. En naturlig frågeställning som uppstår är huruvida det är algoritmen som är effektiv, eller om Nashjämvikter är vanligt förekommande.

En observation som talar mot det sistnämnda är att 0-stapeln i histogrammet verkar krympa då k och n blir större, eftersom detta betyder att den slumpade strategiprofilen $\mathbf{s}^0$ i de allra flesta fallen inte är en ren Nashjämvikt. Vi undersöker detta närmare.

4.3.1 Förekomst av Nashjämvikter

I varje iteration söker $\mathcal{A}$ bland alla k spelares tillgängliga strategier. Varje spelare har $n-1$ andra strategier att byta till, vilket gör att $1+k(n-1)$ strategiprofiler måste undersökas i varje iteration.

Sannolikheten att en given strategiprofil $\mathbf{s}^0$ är en Nashjämvikt sammanfaller med sannolikheten att den slumpade potentialfunktionen uppfyller $\Psi(\mathbf{s}^0) \geq \Psi(s^i_j, \mathbf{s}^0_{-i})$, för alla i och j. Notera att alla värden på potentialfunktionen Ψ är slumpade i iteration 0.

Låt $\mathbb{P}[E]$ beteckna sannolikheten att händelse E inträffar. Vi har att

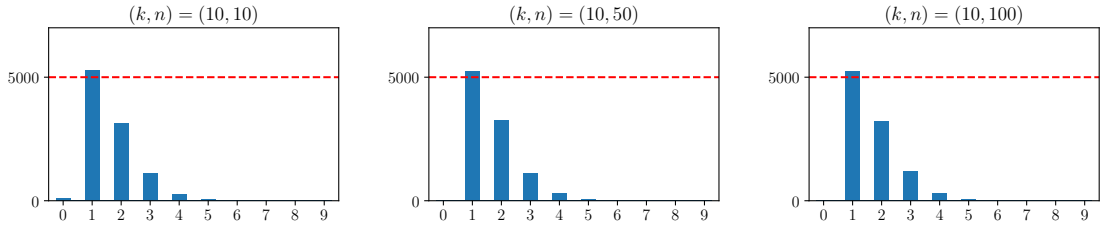
$$\mathbb{P}\left[\mathbf{s}^0 \text{ är en Nashjämvikt}\right] = \mathbb{P}\left[\Psi(\mathbf{s}^0) \geq \Psi(s^i_j, \mathbf{s}^0_{-i}),\ \forall i, j\right] = \frac{1}{k(n-1)+1} \to 0 \quad \text{då } k \text{ eller } n \to \infty, \tag{45}$$

vilket visar att Nashjämvikter är sällsynta i stora spel.

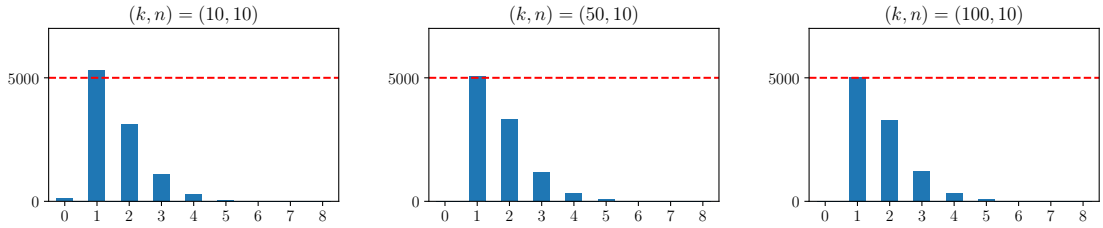
Anmärkning 9. Slumpade potentialspel har i genomsnitt $\dfrac{n^k}{k(n-1)+1}$ Nashjämvikter.

4.3.2 Större spel

Vi undersöker vad som händer för större värden på k och n. Figur 9 resp. 10 visar samma typ av histogram som ovan då vi varierar n resp. k.

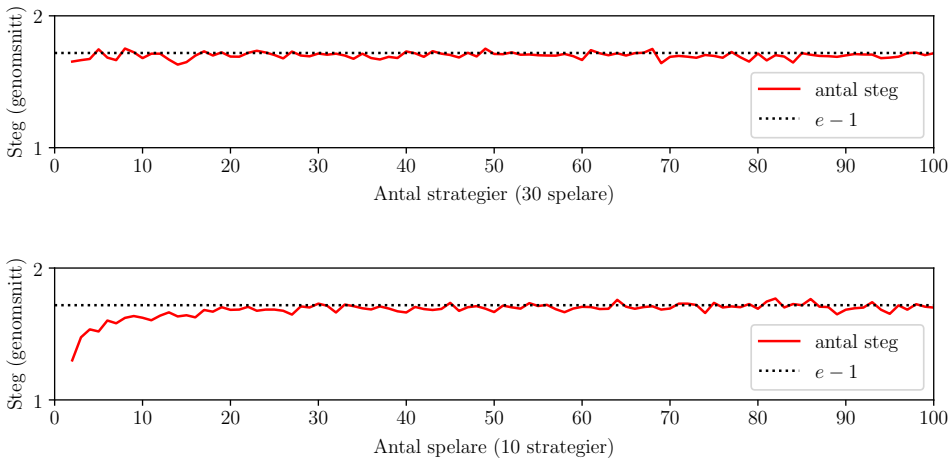


Figur 9: Histogram över antalet steg i $\mathcal{A}$ olika värden på n i 10,000 slumpade potentialspel.



Figur 10: Histogram över antalet steg i $\mathcal{A}$ olika värden på k i 10,000 slumpade potentialspel.

Histogrammen i Figur 9 och 10 är slående lika. Detta leder oss till att nämrare undersöka hur antalet steg beror på k och n genom att beräkna det genomsnittliga antalet steg i $\mathcal{A}$ då vi låter dem var för sig löpa igenom heltalen $2, 3, \ldots, 100$. Se Appendix B.2.1 och B.2.3 för programkod.



Figur 11: Genomsnittligt antal steg över 1000 slumpade spel då vi låter antalet strategier resp. spelare variera.

Graferna i Figur 11 är vid första anblicken överraskande eftersom de antyder att det genomsnittliga antalet steg i $\mathcal{A}$ närmar sig en konstant då k och n blir stora. Ju fler strategiprofiler vi undersöker utifrån $\mathbf{s}^0$, desto större värde på $\Psi(\mathbf{s}^1)$ kan vi förvänta oss att observera. Men om $\Psi(\mathbf{s}^1)$ är stort är sannolikheten att vi hittar en strategiprofil $\mathbf{s}^2$ sådan att $\Psi(\mathbf{s}^2) > \Psi(\mathbf{s}^1)$ förhållandevis liten, eftersom vi hämtar slumptalen ur en uppåt begränsad mängd.

Durand & Gaujal [2] gick djupare i studiet av slumpade potentialspel. I sin forskningsartikel ger de stöd för att det genomsnittliga antalet iterationer närmar sig talet e. Detta översätts till $e - 1$ förbättrande steg i $\mathcal{A}$, vilket vi också använder som referenslinje i Figur 11.

4.4 Två konkreta exempel

Våra empiriska undersökningar antyder att det genomsnittliga antalet steg i $\mathcal{A}$ närmar sig ett konstant värde i slumpade potentialspel med många spelare och strategier. Några sådana slutsatser kan förstås inte dras för potentialspel i allmänhet. Vi tillämpar här $\mathcal{A}$ på två exempel där antalet steg växer med antalet strategier resp. antalet spelare.

4.4.1 Ett "värsta scenario"

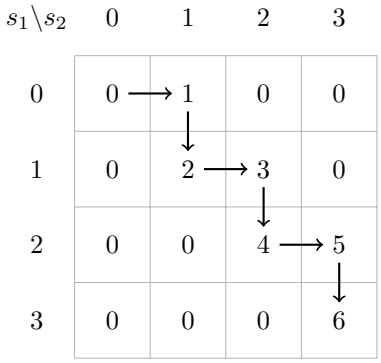
Betrakta ett potentialspel med 2 spelare och n strategier. Sätt $S_1 = S_2 = \{0, 1, ..., m\},\ m = n - 1$ och definiera potentialfunktionen enligt

$$\Psi(i,j) := \begin{cases} i+j, & \text{om } i \in \{j, j-1\}, \\ 0, & \text{annars.} \end{cases}, \ i,j = 0,\ldots,m \tag{46}$$

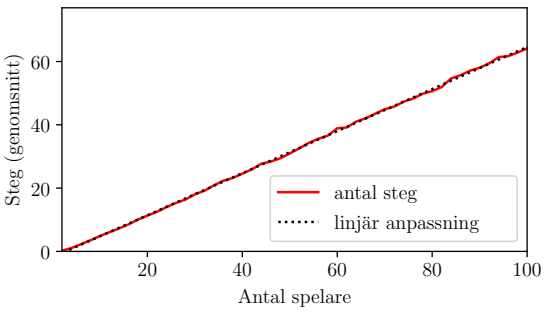
Låt $\mathcal{A}$ starta i $\mathbf{s}^0 = (0,0)$. Första iterationen leder till $\mathbf{s}^1 = (0,1)$, andra till $\mathbf{s}^2 = (1,1)$ osv. tills vi når Nashjämvikten $\mathbf{s}^{\tau-1} = (m,m)$, där $\tau = 2(n-1)$ är antalet steg som krävs för att hitta den rena Nashjämvikten. Figur 12a visar hur $\mathcal{A}$ konvergerar i ett sådant spel med 2 spelare och 4 strategier om vi startar i $\boldsymbol{s}^0 = (0,0)$.

4.4.2 Ett trängselspel

Vi avslutar avsnittet med ett att tillämpa $\mathcal{A}$ på vägnätverkspelet Γ_2 från avsnitt 3.1.2. Figur 12b visar samma typ graf som Figur 11. Här ser vi det genomsnittliga antalet steg växa linjärt med antalet spelare k. Minstakvadratmetoden ger $0.667k - 1.991$ som linjär approximation av det genomsnittliga antalet steg över 100 slumpade försök. Se Appendix B.3 för programkod.



(a) Ett "värsta scenario" med 2 spelare och 4 strategier. Ren Nashjämvikt nås efter 6 steg.



(b) Jämförelse mellan det genomsnittliga antal steg för $\mathcal{A}$ i Γ_2 över 100 slumpade strategiprofiler.

Figur 12: Algoritmen $\mathcal{A}$ på spelen i avsnitt 4.4.

4.5 Slutsats

Att algoritmen $\mathcal{A}$ vi definierade i 4 verkar konvergera efter ett konstant antal iterationer i slumpade potentialspel var för oss ett överraskande resultat. Med djupare kunskaper om stokastiska variabler hade vi kunnat ge ett starkare argument och eventuellt visa om det genomsnittliga antalet iterationer faktiskt konvergerar, och mot vad det konvergerar.

Man kan dock argumentera mot tillämpbarheten av resultatet genom att ifrågasätta hur troligt det är att en modell av en verklig situation kan anta formen av ett slumpat potentialspel. Vi har sett exempel på ett trängselspel där det genomsnittliga antalet iterationer växer linjärt med antalet spelare.

En naturlig fortsättning vore därmed att tillämpa $\mathcal{A}$ och andra algoritmer baserat på iterativt spelande på fler olika typer av potentialspel.

Referenser

[1] Braess, D. (1968). Uber ein paradoxon aus der verkehrsplanung. *Unternehmensforschung, 12*(1), 258 – 268. Engelsk översättning:
`http://www.uvm.edu/pdodds/files/papers/others/2005/braess2005a.pdf` [hämtad 2021-04-19]

[2] Durand, S. & Gaujal, B. (2016). *Complexity and Optimality of the Best Response Algorithm in Random Potential Games.* (Inria, RR-8925). Hämtad från Inria: `https://hal.inria.fr/hal-01330805`

[3] Karlin, A. & Peres, Y. (2016). *Game theory, alive*. Providence, Rhode Island: American Mathematical Society.

[4] Kolata, G. (1990, 25 december). What If They Closed 42d Street and Nobody Noticed? *The New York Times*.
Hämtad från
`https://www.nytimes.com/1990/12/25/health/what-if-they-closed-42d-street-and-nobody-noticed.html`

[5] Mendelson, E. (2004) *Introducing Game Theory and Its Applications.* New York: Chapman & Hall/CRC.

[6] Monderer, D. & Shapley, L. S. (1996). Potential games. *Games and economic behaviour. 14*(1), 124 – 143. doi: 10.1006/game.1996.0044

[7] Nash, J. F. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America, 36*(1), 48 – 49. Hämtad från `https://www.jstor.org/stable/88031`

[8] Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (2007). *Algorithmic Game Theory*. New York: Cambridge University Press.

[9] Papadimitriou, C. (2001). Algorithms, games, and the internet. *In Proceedings of the thirty-third annual ACM symposium on Theory of computing* (s. 749–753). New York: Association for Computing Machinery.

[10] Rosenthal, R. W. (1973). A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory, 2*(1), 65 – 67.

[11] Roughgarden, T. (2015). Intrinsic robustness of the price of anarchy. *Journal of the ACM. 32*, 513 – 522. doi: 10.1145/2806883

[12] Voorneveld, M., Borm, P., Van Megen, F., Tijs, S., Facchini, G. (2011). Congestion Games and Potentials Reconsidered. *International Game Theory Review. 1*(3), 283 – 299. doi: 10.1142/S0219198999000219

A Bevis från avsnitt 2

Vi presenterar här några resultat från spelteorin som ligger utanför ramarna för uppsatsen.

A.1 Nashs sats

Följande hjälpsats bevisas i avsnitt A.1.2.

Lemma 2. *Mängden av alla mixade strategiprofiler är begränsad, sluten och konvex.*

Nyckelargumentet i Nashs sats är beroende av *Brouwers fixpunktssats*. Dess bevis ligger dock utanför ramarna för vad som avhandlas i denna texten, och vi nöjer oss med att formulera resultatet.

Brouwers fixpunktsats. *Låt Ω vara en begränsad, sluten och konvex mängd. Varje kontinuerlig funktion $\varphi : \Omega \to \Omega$ har en fixpunkt, dvs. $\exists x \in \Omega : x = \varphi(x)$.*

A.1.1 Bevis

Nashs sats. *Varje ändligt spel har en mixad Nashjämvikt.*

Bevis. Vi börjar med att etablera ett villkor för Nashjämvikt med avseende på strategiprofiler. Sedan använder vi villkoret för att visa existens av mixad Nashjämvikt med hjälp av Brouwers fixpunktsats.

Låt $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ med $\mathbf{x}_i = (x_1^i, \ldots, x_{n_i}^i)^T$ vara en mixad strategiprofil i ett spel med k spelare. Antag att $n_i = |S_i| < \infty$ för alla $i = 1, \ldots, k$. För varje spelare i, sätt

$$\begin{cases} \delta_j^i = \max\left(u_i(s_j, \mathbf{X}_{-i}) - u_i(\mathbf{x}_i, \mathbf{X}_{-i}), 0\right), \\ x_j^{i*} = \dfrac{x_j^i + \delta_j^i}{1 + \sum_{j=1}^{n_i} \delta_j^i}. \end{cases} \qquad \forall j = 1, \ldots, n_i \tag{47}$$

Låt $\mathbf{X}^* := (\mathbf{x}_1^*, \ldots, \mathbf{x}_k^*)$ där $\mathbf{x}_i^* = (x_1^{i*}, \ldots, x_{n_i}^{i*})^T$ enligt ovan. Då är $\mathbf{X}^*$ en mixad strategiprofil, vilket läsaren lätt kan verifiera. Vi ska nu visa att $\mathbf{X}$ är en Nashjämvikt *om och endast om* $\mathbf{X} = \mathbf{X}^*$.

Antag först att den mixade strategiprofilen $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ är en Nashjämvikt. Då har vi att $u_i(\mathbf{x}_i, \mathbf{X}_{-i}) \geq u_i(\mathbf{x}_i^*, \mathbf{X}_{-i})$ för alla $\mathbf{x}_i^*$ och alla i. Men då är $\delta_j^i = 0$ i ekvation 47 för alla i och j. Alltså har vi att $\mathbf{X} = \mathbf{X}^*$. För att visa implikation åt andra hållet tar vi hjälp av följande lemma, som vi bevisar i avsnitt A.1.2.

Lemma 3. *Antag att strategiprofilen* $\mathbf{X}$ inte *är en Nashjämvikt. Då gäller följande.*

(i) Det finns en ren strategi $s_j \in S_i$ sådan att $u_i(s_j, \mathbf{X}_{-i}) > u_i(\mathbf{x}_i, \mathbf{X}_{-i})$.

(ii) Det finns en ren strategi $s_{j^*} \in S_i$ med $x_{j^*}^i > 0$ sådan att $u_i(s_{j^*}, \mathbf{X}_{-i}) \leq u_i(\mathbf{X})$.

Vi ser att (i) medför att $\delta_j^i = \max\left(u_i(s_j, \mathbf{X}_{-i}) - u_i(\mathbf{x}_i, \mathbf{X}_{-i}), 0\right) > 0$ i ekvation 47. Från (ii) följer å andra sidan att $\delta_{j^*}^i = 0$. Dessa två samband ger tillsammans

$$0 < x_{j^*}^{i*} = x_{j^*}^i / \left(1 + \textstyle\sum_{j=1}^{n_i} \delta_j^i\right) < x_{j^*}^i. \tag{48}$$

Speciellt får vi $x_{j^*}^{i*} \neq x_{j^*}^i \Rightarrow \mathbf{X}^* \neq \mathbf{X}$, och därmed är vi klara med första delen av beviset.

Vi ska nu visa att en sådan strategiprofil alltid existerar. Definiera $\varphi : \mathbf{X} \mapsto \mathbf{X}^*$, med $\mathbf{X}^*$ som i ekvation 47, och notera att φ är kontinuerlig ($x_j^i \mapsto \delta_j^i$ är kontinuerlig eftersom u_i är kontinuerlig). Vidare är mängden $\boldsymbol{\Delta}$ av alla mixade strategiprofiler begränsad, sluten och konvex enligt lemma 2. Eftersom φ är en kontinerlig funktion från $\boldsymbol{\Delta}$ till sig själv, uppfyller φ villkoren i Brouwers fixpunktssats. Alltså existerar en mixad strategiprofil $\mathbf{X}$ sådan att $\varphi(\mathbf{X}) = \mathbf{X}$. Detta visar att varje ändligt spel har en mixad Nashjämvikt.

□

A.1.2 Bevis av hjälpsatser

Bevis av lemma 2. Låt Δ^n beteckna mängden av alla n-dimensionella sannolikhetsvektorer. Vi observerar att Δ^n är begränsad av enhetshyperkuben $[0,1]^n$. För att visa slutenhet observerar vi att

$$\Delta^n = [0,1]^n \cap \{\mathbf{x} \in \mathbb{R}^n;\ \sum_{i=1}^{n} x_i = 1\},$$

dvs. snittet av två slutna mängder, en hyperkub och ett hyperplan. Alltså är Δ^n sluten.

Låt $\mathbf{x}, \mathbf{y} \in \Delta^n$ och sätt $\mathbf{z} = \mathbf{x}(1-t) + \mathbf{y}t,\ \in [0,1]$. För att visa konvexitet räcker det att visa att $\mathbf{z} \in \Delta^n$. Vi har

$$\sum_{i=0}^{n} z_i = \sum_{i=0}^{n}(x_i(1-t) + y_i t) = \sum_{i=0}^{n} x_i + t\sum_{i=0}^{n}(y_i - x_i) = 1 + t\left(\sum_{i=0}^{n} x_i - \sum_{i=0}^{n} y_i\right) = 1, \tag{49}$$

och från definitionerna av x_i, y_i, t;

$$z_i = x_i(1-t) + y_i t \geq 0,\ i = 1, \ldots, n. \tag{50}$$

Med detta ser vi att $\mathbf{z} \in \Delta^n$. Alltså är Δ^n konvex. Låt nu $\mathbf{\Delta}$ beteckna mängden av alla strategiprofiler och sätt $n_i := |S_i|,\ i = i, \ldots, k$. Vi har att

$$\mathbf{\Delta} = \prod_{j=1}^{k} \Delta^{n_i}. \tag{51}$$

Produkten av ändligt många begränsade, slutna och konvexa mängder är begränsad, sluten och konvex. Därav är $\mathbf{\Delta}$ begränsad, sluten och konvex. □

Bevis av Lemma 3. Antag att $\mathbf{X}$ *inte* är en Nashjämvikt, och antag att $\mathbf{y}_i = (y_i, \ldots, y_{n_i})$ är en mixad strategi för en spelare i sådan att $u_i(\mathbf{y}_i, \mathbf{X}_{-i}) > u_i(\mathbf{x}_i, \mathbf{X}_{-i})$. Ett sådant i och $\mathbf{y}_i$ existerar enligt definition 6. Vi bevisar båda påståendena med motsägelsebevis och med hjälp av upprepade tillämpningar av Proposition 1.

(i) Antag att $u_i(s_j, \mathbf{X}_{-i}) \leq u_i(\mathbf{x}_i, \mathbf{X}_{-i})$ för alla $s_j \in S_i$. Det följer att

$$u_i(\mathbf{y}_i, \mathbf{X}_{-i}) = \sum_{j=1}^{n_i} y_j u_i(s_j, \mathbf{X}_{-i}) \leq \sum_{j=1}^{n_i} y_j u_i(\mathbf{x}_i, \mathbf{X}_{-i}) = u_i(\mathbf{x}_i, \mathbf{X}_{-i}) = u_i(\mathbf{X}) \tag{52}$$

vilket motsäger antagandet om $\mathbf{y}_i$. Alltså måste $u_i(s_j, \mathbf{X}_{-i}) > u_i(\mathbf{x}_i, \mathbf{X}_{-i})$ för någon ren strategi $s_j \in S_i$.

(ii) Antag nu att $u_i(s_j, \mathbf{X}_{-i}) > u_i(\mathbf{X})$ för alla j med $x_j^i > 0$. Då får vi

$$u_i(\mathbf{X}) = \sum x_j u_i(s_j, \mathbf{X}_{-i}) > u_i(\mathbf{X}), \tag{53}$$

dvs. $u_i(\mathbf{X}) \neq u_i(\mathbf{X})$, motsägelse. Alltså måste det finnas en strategi s_{j^*} med $x_{j^*}^i > 0$ sådant att $u_i(s_{j^*}, \mathbf{X}_{-i}) \leq u_i(\mathbf{X})$.

□

A.2 Varje ändligt potentialspel definierar ett trängselspel

Definition 15 (Isomorfi mellan spel)**.** Låt $\Gamma = \langle N, \mathbf{S}, (u_i)_{i\in N}\rangle$ och $\Delta = \langle N, \mathbf{T}, (v_i)_{i\in N}\rangle$ vara två spel med k spelare. Vi säger att att Γ är **isomorf** med Δ om det för varje spelare $i \in N$ och $\mathbf{s} = (s_1, ..., s_k) \in \mathbf{S}$ existerar en bijektiv funktion $g_i : S_i \to T_i$ sådan att

$$u_i(s_1, ..., s_k) = v_i(g_1(s_1), ..., g_n(s_k)). \tag{54}$$

Lemma 4 ([12])**.** *Varje koordinationsspel är isomorft med ett trängselspel.*

Bevis. Låt $G = \langle N, \mathbf{S}, (u)_{i\in N}\rangle$ vara ett godtyckligt koordinationsspel med k spelare. Definiera en resurs $f(\mathbf{s})$ för varje $\mathbf{s} \in \mathbf{S}$ så att $\mathbf{s} \neq \mathbf{s}' \implies f(\mathbf{s}) \neq f(\mathbf{s}')$. Vi konstruerar nu funktionen $g_i(\mathbf{s}_i) = \bigcup_{\mathbf{s}_{-i}\in S_{-i}}\{f(s_i, \mathbf{s}_{-i})\}$. Definiera trängselspelet $\langle N, F, \prod_{i\in N} X_i, (w_i)_{f\in F}\rangle$ där $F = \bigcup_{\mathbf{s}\in\mathbf{S}} f(\mathbf{s})$, $X_i = \{g_i(s_i)|s_i \in S_i\}$ och

$$w_f(r) = \begin{cases} u(s) & , \text{ om } r = x \\ 0 & , \text{ om } r \neq 0. \end{cases} \tag{55}$$

Vi visar nu att g_i är en bijektion för varje $i \in N$. Att g_i är surjektiv följer per definition av X. För att visa att g_i är injektiv; antag att $s_i \neq s_i'$. Då gäller för varje $\mathbf{s}_{-i} \in \mathbf{S}_{-i}$ att $f(s_i, \mathbf{s}_{-i}) \neq f(s_i', \mathbf{s}_{-i})$ och därav är $g_i(s_i', \mathbf{s}_{-i}) \neq g_i(s_i, \mathbf{s}_{-i})$. Kontraposition ger att g_i är injektiv för alla $i \in N$.

En spelares nyttofunktion i trängselspelet är

$$v_i(x) = \sum_{f\in x_i} w_f(n_f(x)). \tag{56}$$

Låt $s \in S$ vara en godtycklig strategiprofil och i en godtycklig spelare.

$$v_i(g_i(s_i)) = \sum_{f\in x_i} w_f\left(n_f[(g_i(s_i))_{i\in N}]\right) = \sum_{f\in\bigcap_{i\in N} g(s_i)} w_f\left(n_f[(g_i(s_i))_{i\in N}]\right) = u(s), \tag{57}$$

då $\bigcap_{i\in N} g(s_i) = \{f(s)\}$. □

Lemma 5 ([12])**.** *Varje dummyspel är isomorft med ett trängselspel.*

Bevis. Låt $G = \langle N, \mathbf{S}, (u_i)_{i\in N}\rangle$ vara ett dummyspel. Resurserna är $f(\mathbf{s}_{-i})$ för varje $i \in N$ och $\mathbf{s}_{-i} \in \mathbf{S}_{-i}$. Låt

$$F = \cup_{i\in N} \cup_{\mathbf{x}_{-i}\in X_{-i}} \{f(\mathbf{x}_{-i})\}. \tag{58}$$

Definiera funktionerna h_i där

$$h_i(s_i) = \{f(\mathbf{s}_{-i})|\mathbf{s}_{-i} \in S_{-i}\} \cup \{f(y_{-j})|j \in N \setminus \{i\} \wedge (y_{-j} \in S_{-j} \wedge y_i \neq s_i)\} \tag{59}$$

och $w_f : \mathbb{N} \to \mathbb{R}$ för varje $f(\mathbf{s}_{-i}) \in F$ där

$$w_{f(\mathbf{s}_{-i})}(r) = \begin{cases} u_i(s_i, \mathbf{s}_{-i}) & , \text{ om } r = 1 \text{ och } s_i \in S_i \text{ är godtycklig} \\ 0 & , \text{ annars.} \end{cases} \tag{60}$$

Trängselspelet är följande: $\langle N, F, X, (w_f)_{f\in F}\rangle$ där $X_i = \{h_i(s_i)|s_i \in S_i\}$ och $X = \prod_{i\in N} X_i$.

Vi visar att h_i är injektiv, surjektivitet följer per definition av X_i. Antag att $h_i(s_i) = h(\bar{s}_i)$. Om $s_i \neq \bar{s}_i$ så finns det $y_i = s_i$ och $y_i \neq s_i$. Därav existerar $\mathbf{y}_{-j}$ så $f(\mathbf{y}_{-j}) \in h_i(s_i)$ men $f(\mathbf{y}_{-j}) \notin h_i(\bar{s}_i)$ vilket motsäger att $h_i(s_i) = h(\bar{s}_i)$. Därav är h_i injektiv för alla $i \in N$.

Sist visar vi att för nyttofunktionen, v_i, för varje spelare i i trängselspelet och för varje strategiprofil $\mathbf{s} \in \mathbf{S}$ så gäller

$$v_i(h_1(s_1), ..., h_n(s_n)) = u_i(s_1, ..., s_n). \tag{61}$$

Tag godtyckligt $i \in N, \mathbf{s}_{-i} \in \mathbf{S}_{-i}$ och $s_i \in S_i$. Låt $\mathbf{x} = (h_1(s_1), ..., h_n(s_n))$ Då är $f(\mathbf{s}_{-i}) \in h_i(s_i)$ och för varje $j \in N \setminus \{i\} : f(\mathbf{s}_{-i}) \notin h_j(s_j)$ så i är den enda spelaren som använder $f(\mathbf{s}_{-i})$ givet strategiprofilen $\mathbf{x}$. Det gäller också att alla andra resurser i $h_i(s_i)$ används av fler än 1 spelare. Detta påståendet visas genom två utömmande fall.

1. Låt $f = f(\mathbf{y}_{-i})$ för godtycklig $\mathbf{y}_{-i} \in X$. Om $\mathbf{y}_{-i} \neq \mathbf{s}_{-i}$ så måste $y_j \neq s_j$ för minst ett $j \in N \setminus \{i\}$. Därav $f \in h_j(s_j)$, alltså används f av minst 2 spelare.

2. Låt $f = f(\mathbf{y}_{-j})$ för något $j \in N \setminus \{i\}$ och strategin $\mathbf{y}$ är sådan att $\mathbf{y}_{-j} \in X_{-j}$ och $y_i \neq s_i$. Då är $f \in h_j(s_j)$ och, som i fall 1, f används av minst 2 spelare.

Därav är det enbart en term i summan som bildar $v_i(h_1(s_1), ..., h_n(s_n))$ som är nollskild, $u_i(s_1, ..., s_n)$. □

Sats 5 ([12])**.** *Varje potentialspel är isomorft med ett trängselspel.*

Bevis. Antag att $\Gamma = \langle N, \mathbf{S}, (u_i)_{i \in N} \rangle$ är ett potentialspel. Låt Γ_k och Γ_d vara det koordinationsrespektive dummyspel som korresponderar mot Γ enligt Sats 2. Använd Lemma 4 och 5 för att avbilda Γ_k och Γ_d till trängselspelen T_k och T_d där g_i och h_i är deras respektive isomorfier för alla $i \in N$. Vi kan, utan förlust av generalitet, anta att båda trängselspelens resursmängder är disjunkta. Vi konstruerar nu trängselspelet T där mängden av resurser är unionen av resurserna från T_k och T_d, unionen av nyttofunktionerna definierade av Lemma 4 och 5 samt strategirummet $Y_i = \{g_i(s_i) \cup h_i(s_i) | s_i \in S_i\}$. □

B Programkod

Detta avsnitt innehåller den programkod som använts för att simulera spel och generera de plottar som presenteras i avsnitt 4. All kod är körbar i Python 3.8.5.

B.1 Konstruktion av potentialspel

Koden nedan demonstrerar förfarandet i 4.1.1 genom att konstruera ett potentialspel med 3 spelare och 3 strategier vardera genom att tilldela potentialfunktionen och dummyspelets nyttofunktioner slumpmässiga heltalsvärden mellan 0 och 4.

```
from numpy.random import randint
from numpy import array,zeros

'''    This script is a simple demonstration of how to construct a potential game
    with k players and n strategies with random integer payoffs. '''

k,n = 3,3
bound = 5

CoordinationGame = randint(0,bound,k*(n,))
DummyGame = k*[0,]

for i in range(k):
    # Randomize along "all but player i's" dimension and cast to correct size
    DummyGame[i] \
        = zeros(k*(n,),dtype=int) \
        + randint(0,bound,i*(n,) + (1,) + (k-i-1)*(n,))

# functions u and Psi accept k-tuple of integer values in range(0,n) as index
def u(i):
    # Player i:s utility function. Accepts k-tuple of integer values in
    # rane(n) as indices.
    return CoordinationGame + DummyGame[i]

def Psi(s):
    # Potential function.
    return CoordinationGame[s]
```

B.2 Iterativt spelande i slumpade potentialspel

Här implementerar vi algoritmen $\mathcal{A}$ på slumpade potentialspel. Modulen importeras i programkoden i B.2.2 och B.2.3 som `potentialgame_sparse.py`.

B.2.1 Implementation av algoritmen i avsnitt 4

```
''' This module implements the best response algorithm in random potential
    games with k players and n strategies, starting in strategy profile s.
    The value of the potential function is generated each time a new strategy
    profile is visited.
'''

from random import random

def BRD(s,k,n):
    # s is a k-tuple of integers in range(n).
    Psi = dict()
    Psi[s] = random()
    iterations = 0
    while True:
        maximum, best = Psi[s],s
        for i in range(k):
            # List adjacent strategy profiles.
            S_i = [ s[:i] + (j,) + s[i+1:] for j in range(n) ]
            for s_i in S_i:
                if s_i not in Psi.keys():
                    Psi[s_i] = random()
                # Find most improving strategy
                if Psi[s_i] > maximum:
                    maximum, best = Psi[s_i], s_i
        # Find most improving player
        if Psi[best] > Psi[s]:
            s = best
            iterations += 1
        else:
            break
    return iterations
```

B.2.2 Olika antal spelare och strategier

Följande kod genererar histogrammen i Figur 8, 9 och 10, och importerar modulen i B.2.1 som `potentialgame_sparse.py`.

```
''' This script produces histograms displaying the number of steps
    in the best response algorithm in random potential games.
'''

from random import seed as random_seed

from numpy.random import randint,seed as numpy_seed
import matplotlib.pyplot as plt

from potentialgame_sparse import BRD

K = (2,5,10)    # players
N = (2,5,10)    # strategies
sample = 10000

random_seed(314159)
numpy_seed(314159)

iterations = [ [ tuple() for n in N ] for k in K ]
max_iter = 0
for i,k in enumerate(K):
    for j,n in enumerate(N):
        max_iter = 0
        for foo in range(sample):
            s = tuple(randint(0,n,k))
            iter_eq = BRD(s,k,n)
            iterations[i][j] += (iter_eq,)

max_iter = max( max([ [ max(j) for j in i ] for i in iterations ] ) )

plt.rc('text', usetex=True)
plt.rc('font', family='serif',size=12)

box = 100*len(K)+10*len(N) if len(N) > 1 else 100*len(N)+10*len(K)
for i in range(len(K)):
    for j in range(len(N)):
        plt.subplot(box+len(N)*i+j+1)
        plt.hist(
            iterations[i][j],
            bins=[ k/2 for k in range(2*max_iter+2)],
            align='left')
        plt.plot([-0.5,max_iter+0.5],2*[0.5*sample],'r--')
        plt.axis([-0.5,max_iter+0.5, 0,0.7*sample])
        plt.title(f'$(k,n) = ({K[i]},{N[j]})$')
        plt.xticks(ticks=range(max_iter+1))
        plt.yticks(ticks=[0,0.5*sample])
        plt.subplots_adjust(wspace=0.1, hspace=0.2)

plt.show()
```

B.2.3 Stora spel

Följande kod genererar plottarna i Figur 11, och importerar modulen i B.2.1 som `potentialgame_sparse.py`.

```
''' This script plots the average number of steps in the best response
    algorithm in random potential games.
'''

from random import seed as random_seed
from math import e

from numpy.random import randint,seed as numpy_seed
import matplotlib.pyplot as plt

from potentialgame_sparse import BRD

sample = 1000
random_seed(314159); numpy_seed(314159)

K,n = range(2,101,1),10     # players, strategies
iterations = [ tuple() for n in K ]
for i,k in enumerate(K):
    for foo in range(sample):
        s = tuple(randint(0,n,k))
        iterations[i] += (BRD(s,k,n),)
average = [ sum(stuff)/sample for stuff in iterations ]
plt.figure(figsize=(10,2.1))
plt.subplot().set_aspect(15)
plt.plot(K,average,'red',label='antal steg')
plt.plot([0,max(K)],2*[e-1],'black',linestyle='dotted',label='$e-1$')
plt.axis([0,max(K),1,2])
plt.ylabel('Steg (genomsnitt)')
plt.xlabel('Antal spelare (10 strategier)')
plt.yticks(ticks=[1,2])
plt.xticks(ticks=range(0,101,10))
plt.legend(loc='lower right')

N,k = range(2,101,1),30     # strategies,players
iterations = [ tuple() for n in N ]
for i,n in enumerate(N):
    for foo in range(sample):
        s = tuple(randint(0,n,k))
        iterations[i] += (BRD(s,k,n),)
average = [ sum(stuff)/sample for stuff in iterations ]
plt.figure(figsize=(10,2.1))
plt.subplot().set_aspect(15)
plt.plot(N,average,'red',label='antal steg')
plt.plot([0,max(N)],2*[e-1],'black',linestyle='dotted',label='$e-1$')
plt.axis([0,max(N),1,2])
plt.ylabel('Steg (genomsnitt)')
plt.xlabel('Antal strategier (30 spelare)')
plt.yticks(ticks=[1,2])
plt.xticks(ticks=range(0,101,10))
plt.legend(loc='lower right')
plt.show()
```

B.3 Iterativt spelande i trängelspel

I detta avsnittet finns programkod som definierar klasser för att bygga upp vägnätverksspel, samt det script som genererar plotten i Figur 12b.

B.3.1 Grafstruktur

Modulen nedan bygger upp en generell grafstruktur och ett vägnätverksspel.

```
'''    This module contains classes to build a graph structure and define an
    atomic selfish routing game, along with functions to run a best response
    algortithm finding pure Nash equilibria. '''

from sys import float_info


class Edge:
    # Edge objects keep track of their number of drivers and latency.

    def __init__(self,origin,target,latency):
        self.origin = origin
        self.target = target
        self.set_latency(latency)
        self.drivers = 0

    def __str__(self):
        return f'{self.origin},{self.target}'

    def __repr__(self):
        return f'({self.origin},{self.target})'

    def __lt__(self,other):
        return self.origin < self.target

    def set_latency(self,latency):
        if type(latency) == type(lambda x:x):
            self.latency = latency
        else:
            self.latency = lambda x : latency


class Vertex:
    # Vertex objects provide structure for Graph objects.

    def __init__(self,ID):
        self.ID = ID
        self.neighbours = set()
        self.outgoing = {}
        self.nearest = None

    def __str__(self):
        return f'{self.ID}:{self.neighbours}'

    def __repr__(self):
        return f'{self.ID}'

    def add_neighbour(self,other):
```

```
        self.neighbours |= {other}

    def add_outgoing(self,other,edge):
        self.outgoing[other] = edge


class Graph:
    # Graph object is built from a set of vertices V and a list of edges E.
    # An element of E is a 3-tuple with (v_1,v_2,w), where v_1,v_2 are
    # elements of V and w is a lambda function or a real number.
    # Use 'digraph' as optional argument to construct a directed graph.

    def __init__(self,V,E=[],*optional):
        self.digraph = True if 'digraph' in optional else None
        V = range(V) if type(V) == int else V
        self.V = { v:Vertex(v) for v in set(V) }
        self.E = []

        for e in E:
            self.add_edge(e)

    def __iter__(self):
        return self.V.values()

    def add_edge(self,edge):
        origin,target,latency = edge

        e = Edge(origin,target,latency)
        self.E.append(e)

        # Store neighbours and reference to corresponding edge in each Vertex.
        V = self.V
        V[origin].add_neighbour(V[target])
        V[origin].add_outgoing(V[target],e)
        if not self.digraph:
            V[target].add_neighbour(V[origin])
            V[target].add_outgoing(V[origin],e)

    def Dijkstra(self,start,finish):
        # Dijkstra's algorithm finds the shortest path from start to finish.
        V = self.V
        s,t = V[start],V[finish]
        labelling = { v: (None,float_info.max) for v in V.values()}
        labelling[s] = None,0
        labelled = {s}
        v = s
        while v != t:
            for n in v.neighbours - labelled:
                edge = v.outgoing[n]
                new_label = labelling[v][1] + edge.latency(edge.drivers + 1)
                if new_label < labelling[n][1]:
                    labelling[n] = v,new_label
            minimum = float_info.max
            for u in set(V.values()) - labelled:
                if labelling[u][1] < minimum:
                    minimum = labelling[u][1]
```

```
                    w = u
                labelled |= {w}
                v = w
            v,path = t,(t,)
            while s.ID != v.ID:
                path += (labelling[v][0],)
                v = labelling[v][0]

            return Path(self,path[::-1])

        def Routes(self,v,t):
            # Returns all possible paths from v to t.
            V = self.V
            if v == t:
                return [(t,)]
            else:
                routes = []
                for n in V[v].neighbours:
                    for path in self.Routes(n.ID,t):
                        routes.append((v,) + path)
                return routes


    class Path:
        # Just another support class for Graph objects.
        def __init__(self,graph,path): #,cost):
            V = graph.V
            self.path = path
            self.edges = \
                set( path[i].outgoing[path[i+1]] for i in range(len(path)-1) )

        @property
        def cost(self):
            return sum( edge.latency(edge.drivers) for edge in self.edges )

        def __str__(self):
            return f'Cost {self.cost}, {self.path}'

        def __eq__(self,other):
            return self.path == other.path


    class Driver:
        # Class representing drivers in RoutingGame below.
        def __init__(self,graph,route,*path):
            self.G = graph
            self.start = route[0]
            self.finish = route[1]
            self.best = None
            if path:
                path = tuple( graph.V[p] for p in path[0] )
                self.path = Path(graph,path)
            else:
                self.path = graph.Dijkstra(self.start,self.finish)
            self.history = None
```

```
    @property
    def cost(self):
        return self.path.cost

    def clear_path(self):
        for edge in self.path.edges:
            edge.drivers -= 1

    def populate_path(self):
        for edge in self.path.edges:
            edge.drivers += 1

    def BestResponse(self):
        self.clear_path()
        self.best = self.G.Dijkstra(self.start,self.finish)
        cost_decrease = self.path.cost - self.best.cost
        self.populate_path()
        return cost_decrease

    def SwitchToBest(self):
        self.clear_path()
        self.path = self.best
        self.populate_path()


class RoutingGame:
    # This class defines an atomic routing game with len(routes) drivers.
    # Arguments routes and paths are k-tuples containing (start,finish)
    # and initial path respectively for player 1,...,k.

    def __init__(self,G,routes,paths):
        self.G = G
        self.drivers \
            = { k:Driver(G,routes[k],paths[k]) for k in range(len(routes)) }
        for driver in self.drivers.values():
            for edge in driver.path.edges:
                edge.drivers += 1

    def Cost(self):
        return sum( self.drivers[d].cost for d in self.drivers )

    def BRD(self):
        # Best response maximum gain algorithm.
        drivers = self.drivers.values()
        N = len(self.drivers.keys())
        steps = 0
        while True:
            improvements = [ driver.BestResponse() for driver in drivers ]
            max_gain = max(improvements)
            if max_gain > 0:
                max_driver = improvements.index(max_gain)
                self.drivers[max_driver].SwitchToBest()
                steps += 1
            else:
                break
        return steps
```

B.3.2 Trängselspelet

Följande kod genererar plotten i Figur 12b och importerar modulen i B.3.1 som `routing_game.py`

```
''' This script plots the average number of steps in the best response
    algorithm when applied on the road network in Braess' paradox.
'''

from random import choice,seed

from numpy import array,ones,vstack
from numpy.linalg import lstsq
from matplotlib import pyplot as plt

from routing_game import *

seed(314159)

V = a,b,c,d = 'a','b','c','d'        # vertices
start,finish = a,d                   # start/end points
S = [(a,b,d),(a,c,d),(a,b,c,d)]      # available paths
K = range(2,101,2)                   # players
sample = 100
iterations = [ tuple() for k in K ]

for i,k in enumerate(K):
    E = {   (a,b,lambda x: x),\
            (a,c,k),\
            (b,c,0),\
            (b,d,k),\
            (c,d,lambda x: x) }
    G = Graph(V,E,'digraph')
    max_iter = 0
    routes = k*((start,finish),)
    for foo in range(sample):
        s = [ choice(S) for i in range(k) ]
        Game = RoutingGame(G,routes,s)
        iterations[i] += (Game.BRD(),)
        for edge in G.E:
            edge.drivers = 0

average = [ sum(iterations[i])/sample for i in range(len(K)) ]

plt.figure(figsize=(5,3))
plt.subplot().set_aspect(2/3)
A = vstack([array(K),ones(len(K))]).T   # see next line
a,b = lstsq(A,average,rcond=None)[0]    # least squares solution
plt.plot(K,a*array(K)+b,'black',linestyle='dotted',label='linjär anpassning')
plt.plot(K,average,'red',label='antal steg')

plt.axis([min(K),max(K),0,1.2*max(average)])
plt.ylabel('Steg (genomsnitt)')
plt.xlabel('Antal spelare (10 strategier)')
plt.legend(loc='lower right')
plt.show()
```