CHALMERS

GÖTEBORGS UNIVERSITET

Optimaltransport för styrning av en svärm av agenter

Optimal transport for controlling a swarm of agents

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet
Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Linnéa Holmberg
Emelie Lemann
Elias Sörstrand
Alfred Wärnsäter

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2022

Optimaltransport för styrning av en svärm av agenter

Examensarbete för kandidatexamen i matematik inom Matematikprogrammet, inriktning Tillämpad matematik, vid Göteborgs universitet

Linnéa Holmberg

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers

Emelie Lemann Elias Sörstrand Alfred Wärnsäter

Handledare: Axel Ringh

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2022

Förord

Denna kandidatuppsats är skriven vid Institutionen för Matematiska vetenskaper på Chalmers tekniska högskola och Göteborgs universitet. Vi vill tacka vår handledare Axel Ringh för hans stora engagemang och de intressanta diskussioner vi haft tillsammans.

Nedan listas den eller de huvudförfattare som skrivit de olika avsnitten och nödvändiga delar som tillhör rapporten. Utöver detta har individuella tidsrapporter skrivits samt en gruppdagbok som vi turats om att skriva och skicka in. Loggböckerna innehåller mer ingående detaljer kring varje persons bidrag till rapporten.

- **Linnéa Holmberg:** Jag har skrivit en stor del av den populärvetenskapliga presentationen samt varit medförfattare till både *Sammanfattning*, *Abstract* och *Förord*. Jag har skrivit delar av avsnitt *2.1 Grundläggande optimeringslära* och gjort figur 3 i avsnitt *3 Interpolation och styrning mellan fördelningar av agenter*. Jag har varit med och reviderat mycket i avsnitt *4.1 Härledning av Sinkhorniterationerna för bimarginella optimaltransportproblem med likhetsvillkor* och skrivit delar till avsnitt *4.2 Härledning av Sinkhorniterationerna för multimarginella optimaltransportproblem med likhets- och olikhetsvillkor*. Det var dock när vi började skriva på avsnittet i fråga och det var få delar som stannade kvar till slutversionen då det skrevs om. Jag har skrivit stora delar av avsnitt *6 Diskussion och slutsats*, men i ett utbrett samarbete med Emelie fått fram den slutgiltiga versionen. Jag har även varit medförfattare till *Appendix A* samt skrivit in bevis av lemmana i *Appendix B* med vår handledare Axel Ringhs hjälp.

- **Emelie Lemann:** Jag har reviderat en stor del av den populärvetenkskapliga texten, samt gjort några tillägg och omskrivningar av vissa delar. Jag har skrivit vissa mindre delar i *2.1 Grundläggande optimeringslära*. I *2.2 Matchningsproblemet* var jag med och påbörjade avsnittet, och skrev en ganska stor del som sedan skrivits om till slutversionen. Jag var medförfattare till avsnitt *2.3 Optimaltransport* och påbörjade hela det avsnittet. I det avsnittet skrev jag väldigt mycket, dock blev en del av min text omskriven då vi gjorde dubbelt arbete eftersom det även skrevs om optimaltransport i avsnitt 3 då. I avsnitt *3 Interpolation och styrning mellan fördelningar av agenter* har jag också varit med och skrivit en del i, men jag har främst gjort omformuleringar och en del omskrivningar av vissa av delarna. Jag var även medförfattare till *4.1 Härledning av Sinkhorniterationerna för bimarginella optimaltransportproblem med likhetsvillkor* och var med och reviderade en del i avsnitt 4.2. I avsnitt *6 Diskussion och slutsats* har jag även där reviderat en hel del och gjort några tillägg, samt gjort vissa omskrivningar.

- **Elias Sörstrand:** Jag var medförfattare till *Sammanfattning* och *Abstract*. Jag var också huvudförfattare för *1 Inledning* och skrivit vissa satser i *2.1 Grundläggande optimeringslära* som inte alla kom med i den slutliga versionen. *Avsnitt 2.2 Matchningsproblemet* var jag även medförfattare för, därtill skrev jag också en stor del i *2.3 Optimaltransport* som sedan blev struken då början av *3 Interpolation och styrning mellan fördelningar av agenter* var mycket lik. Jag har även reviderat en del i *4.1 Härledning av Sinkhorniterationerna för bimarginella optimaltransportproblem med likhetsvillkor*. Därtill har jag också vart medförfattare till *5.1 Tillämpningar med dynamik*. I *Appendix* har jag skrivit *C* samt lagt till korta förklaringar till koden i *D*. I början av projektet skrev jag också några sidor i rapporten om vad vi hade tagit upp på mötena med Axel. De sidorna handlade om det som skrivs om i *4.1 Härledning av Sinkhorniterationerna för bimarginella optimaltransportproblem med likhetsvillkor* samt någon sida om olikhetsvillkor. Utöver detta samarbetade jag med Alfred för att skriva lösaren till matchningsproblemet som visas i *Appendix D* i filen `matching-lp.jl`.

- **Alfred Wärnsäter:** Jag har skrivit delar av avsnitt *2.1 Grundläggande optimeringslära* samt reviderat och bearbetat mycket på avsnitt *2.2 Matchningsproblemet*. Är huvudförfattare till avsnitten *2.3 Optimaltransport*, *3 Interpolation och styrning mellan fördelningar av agenter*, *4.1 Härledning av Sinkhorniterationerna för bimarginella optimaltransportproblem med likhetsvillkor* och *4.2 Härledning av Sinkhorniterationerna för multimarginella optimaltransportproblem med likhets- och olikhetsvillkor*. Medförfattare till *5.1 Tillämpningar med dynamik*. Skrivit *5.2 Utrymning ur labyrint*. Medförfattare till *Appendix A*. Skrivit *Appendix*

B och ansvarat för *Appendix D*. Utöver detta har jag skrivit all kod i projektet (förutom delar av `matching-lp.jl` där jag samarbetade med Elias). Koden är skriven i programspråket Julia och visas i *Appendix D*. Koden har använts för att generera bilderna som visas i figur 1, 2, 4 och 5.

Utöver detta har vi tillsammans haft många möten med vår handledare och gått igenom viktiga koncept. Därtill har handledaren också kommit med feedback om rapporten som alla har varit med och reviderat. Dessutom har vi tillsammans skrivit planeringsrapporten samt gjort mycket litteraturstudier inom området.

Populärvetenskaplig presentation

Världen och dess invånare, djur som människor, är konstant i rörelse. Oavsett om det handlar om att djuren på savannen ska ta sig till vattenhålet eller att du på morgonen ska ta bilen till din arbetsplats. Rörelse, eller möjligtvis transport och det mest effektiva sättet att genomföra det är något vi kommer att prata mer om. Låt oss måla upp en bild där du antingen kan ta en kortare väg till arbetsplatsen men som oftare är mer trafikerad, eller en omväg med färre köer. Då kan vi ställa oss frågan: vilket alternativ går snabbast? Detta är något som skulle kunna beskrivas som ett optimaltransportproblem, vilket är vad vårt arbete handlar om.

När det kommer till liknande situationer kan man basera ett beslut på tidigare erfarenheter och ta flertalet för- och nackdelar i beräkning och ställa dessa mot varandra. Men när problemställningen skalas upp, och det inte längre handlar om sin egen bil och sin egen färd till arbetsplatsen, utan det i stället kan handla om alla bilar i din stad och vilka vägval de ska ta när de kör till arbetsplatsen på morgonen. En situation där vi har betydligt fler objekt, i vårt fall bilar, och där vi dessutom vet varje bils start- och slutpunkt, blir det mer komplicerat. Detta exempel kan i sin tur rikta sig till hur optimaltransport kan hjälpa till med att planera infrastrukturen i en stad på bästa sätt. Inte bara med att planera bilvägar utan även andra transportvägar till exempel spårvagnsräls, cykelvägar och gångbanor. För en enskild person blir det svårt att logiskt tänka ut ett förslag och i stället kan problemet ställas upp matematiskt. Det matematiska problemet kan sedan lösas av en dator vilket ger den snabbaste rutten för alla, totalt sätt.

Det finns flertalet problematiska aspekter med datorn och dess sätt att resonera, bland annat att den inte resonerar, utan gör vad den är tillsagd att göra. Ett exempel är det faktum att datorn inte automatiskt, som vi människor, tar i beaktande att bilarna måste hålla sig på vägen och att det inte är möjligt att köra genom ett hus. Datorn kan också ha svårt att veta om körvägen är den mest lämpliga för bilen och kan därför råka välja en väg som bilen rentav inte får köra på. Finns det dessutom tillräckligt med bilar och vägar växer problemet till ett extremt stort kombinatoriskt problem, speciellt om datorn rakt av ska testa alla körvägar för varje bil innan den kan bestämma sig för vilken väg som är den optimala. Men med omskrivningar, begränsningar av problemet och stegvisa algoritmer kan liknande problem som samhället stöter på lösas med hjälp av datorn på ett smidigt vis. Det kan vara omskrivningar i form av ett litet extra tillägg som ger en mer exakt lösning eller approximation. En ytterligare begränsning kan vara i form av ett extra krav på hur många punkter man får använda sig av i en algoritm. Dessa omskrivningar, nya uppställningar av det generaliserade problemet och metoden bakom att approximera det optimala värdet är något denna rapport kommer att fördjupa sig i.

Utöver nämnt exempel av optimaltransport finns det ett mycket djupare och bredare fält än vad det vid första anblick tycks vara och exemplen kan modelleras mer komplexa. Likt föregående problemställning kan datorn användas för att numeriskt hitta det optimala värde för det enklaste av exempel, men problemen kan även modellera hinder med en egenbestämd kostnad. Detta gör det intressant, är det kostnadsmässigt värt att riskera att få leriga skor för att ta en genväg genom skogen eller att komma några minuter försent för att i stället gå runt? Är det en lätt stig eller behövs det klättras ned för en bergsknall och skulle detta påverka beslutet? För att utveckla optimaltransportproblemet ytterligare kan faktorer som acceleration, tid, hinder med olika kostnader med mera göra det mer invecklat och komplicerat, vilket numeriskt kan lösas och kan därmed modellera verkligheten på ett mer riktigt vis. För detta krävs likt tidigare en matematisk modell. I denna rapport kommer vi härleda en liknande sådan modell för en klass av optimaltransportproblem som kan lösa flera olika typer av problem, och tillämpa modellen på snarlika exempel som vi beskrivit här. Men i stället för att undersöka bilar och deras färd till arbetsplatsen kommer vi undersöka hur så kallade agenter på smidigast sätt rymmer ut från en labyrint och hur en svärm agenter agerar när de behöver ta hänsyn till ett hinder i rörelse.

I en värld där effektivitet, produktivitet och snabbhet värdesätts, är det optimering vi söker. I en sådan värld kan optimering snart ses som människans nya bästa vän, om inte annat ett otroligt verktyg för att både individens, samhällets eller hela länders samverkan ska gå som en dans.

Sammanfattning

Syftet med denna rapport är att härleda och implementera en lösare för en typ av multimarginellt optimaltransportproblem. Denna typ av multimarginella optimaltransportproblem kan användas för att modellera och beräkna hur en svärm av agenter ska styras på ett optimalt sätt. För att göra detta används såväl interpolation, entropisk regularisering och Sinkhorniterationer. Lösaren testades på två olika fall. I det första fallet startade svärmen av agenter enligt en viss fördelning i ett 100×100 rutnät och deras mål var att fördela sig jämnt över hela rutnätet. Dessutom placerades ett hinder i modellen som förflyttade sig genom rutnätet över tid. I det andra fallet behövde lösaren hitta en optimal väg ut genom en labyrint.

Nyckelord: optimaltransport; matchningsproblem; tilldelningsproblem; agenter; interpolation; multimarginellt; entropisk regularisering; Sinkhorniterationer;

Abstract

The purpose of this report is to derive and implement a solver for a multimarginal optimal transport problem. This type of multimarginal optimal transport problem can be used to model and calculate how a swarm of agents should be controlled in an optimal way. Interpolation, entropic regularization and Sinkhorn iterations are used in order to do this. We applied the solver to two different cases. In the first case, the agents started according to a certain distribution inside a 100×100 grid and their goal was to evenly spread out. Furthermore, an obstacle was placed in the model that moved through the grid for each time step. In the last case, the algorithm was required to find an optimal way out through a maze.

Keywords: optimal transport; matching problem; assignment problem; agents; interpolation; multimarginal; entropy regularization; Sinkhorn iterations

Innehåll

1 Inledning

Optimaltransport är ett område inom matematik som handlar om att på ett så effektivt sätt som möjligt förflytta något från punkt A till B. En av de viktigaste händelserna i optimaltransport var 1781 [1] då den franske matematikern Gaspard Monge skrev artikeln *Mémoire sur la théorie des déblais et des remblais* vilket var den första rapporten skriven om optimaltransport [2, s. 29].

Artikeln besvarade frågan om hur man minimerar kostnaden för att flytta jord från x antal gruvor till y antal byggnadsplatser [2, s. 29]. Därefter var det inte många genombrott inom forskningen om optimaltransport fram till 1930 då A.N. Tolstoi publicerade en artikel som behandlade lösningar för optimaltransportproblemet [3] [4, s. 362]. Som en följd av Tolstois publikation ökade intresset för optimaltransport, speciellt under 1940-talet då många framsteg gjordes [5]. Stora framsteg inom fältet gjordes av Leonid Kantorovich, som ofta anses vara grundaren till linjärprogrammeringen [2, s. 31] [6, s. 220] [7, s. 591]. Andra framsteg gjordes bland annat av George Dantzig som under 1949 lyckades lösa optimaltransportproblemet numeriskt med hjälp av linjärprogrammering [5] [8].

Under det senaste årtiondet har optimaltransport utvecklats snabbt och används inte längre uteslutande till att flytta jord. I stället har optimaltransport hittat tillämpningar inom till exempel cancerscreening [9], färgöverföring mellan digitala bilder [10], maskininlärning [11], ekonomi [12] och utvecklingen av självkörande bilar [13].

I rapporten kommer vi betrakta det så kallade matchningsproblemet och se att detta kan ses som ett specialfall av det diskreta optimaltransportproblemet. I ett sådant problem är målet att två mängder av så kallade agenter bijektivt skall paras ihop med varandra på ett optimalt sätt. Agenterna kan representera nästan vad som helst beroende på användningsområdet. Konkreta exempel som skulle kunna dyka upp i praktiken är bilar, jordhögar, atomer eller liknande. Om agenterna är tillräckligt många kan de i stället representeras som en distribution eller som en massdensitet. Ett problem som uppstår är att den naiva lösningsmetoden till matchningsproblemet har en komplexitet som växer enligt $n!$, vilket är problematiskt redan vid relativt små beräkningar. För att underlätta beräkningarna går det att formulera om problemet med hjälp av linjärprogrammering, [14, s. 57-58] vilket är en viktig observation som mycket i rapporten bygger på.

Genom att länka samman fler optimaltransportproblem kan man formulera det så kallade interpolationsproblemet. Till skillnad från det enklare matchningsproblemet ska detta problem tolkas som att hitta den följd av fördelningar som agenterna ska gå igenom för att ta sig från någon given startfördelning till någon given slutfördelning. Problem av den här typen ger upphov till mycket stora linjärprogrammeringsproblem, och ett av projektets mål är att undersöka numeriska metoder för att lösa dessa approximativt. För detta kommer vi använda oss av begreppen entropisk regularisering och Sinkhorniterationer [15]. Teorin kommer demonstreras genom att vi implementerar en numerisk algoritm för att approximativt lösa interpolationsproblemet. Denna kommer sedan köras på storskaliga exempel för att belysa dess effektivitet och mångsidighet.

2 Bakgrund

I detta avsnitt presenteras bakgrundsmaterialet som resten av rapporten bygger på. Avsnittet är uppdelat i tre delar. Avsnitt 2.1 består av en introduktion till grundläggande optimeringslära där ett flertal välkända resultat presenteras. I avsnitt 2.2 presenteras matchningsproblemet där en viss relaxeringsprocess leder till formuleringen av ett optimaltransportproblem. Sist utvecklas teorin om optimaltransport i avsnitt 2.3.

2.1 Grundläggande optimeringslära

Matematisk optimering är det område inom matematiken som behandlar olika tekniker för hur man utifrån en given funktion kan hitta ett, i något avseende, optimalt värde på funktionen givet vissa begränsningar. I denna del av avsnittet ger vi en kortfattad sammanfattning av de delar av optimeringslära som är relevanta för projektet. Framställningen är delvis en adaption av [16, s. 1-2].

Ett generellt optimeringsproblem kan formuleras som

$$\begin{aligned} \text{minimera} \quad & f(x) \\ \text{då} \quad & f_i(x) \leq b_i, \quad \text{för } i = 1, \ldots, m, \end{aligned} \tag{2.1}$$

där variabeln $x \in \mathbb{R}^n$ i problemet kallas för optimeringsvariabeln. Funktionen $f : \mathbb{R}^n \to \mathbb{R}$ kallas för målfunktion och funktionerna $f_i(x) : \mathbb{R}^n \to \mathbb{R}$ för $i = 1, \ldots, m$ kallas för bivillkorsfunktioner. Bivillkorsfunktionerna ger tillsammans med konstanterna $b_i \in \mathbb{R}$ för $i = 1, \ldots, m$ problemets bivillkor. För att underlätta framställningen definierar vi mängden S av tillåtna lösningar som

$$S := \{x \in \mathbb{R}^n \, : \, f_i(x) \leq b_i, \text{ för } i = 1, \ldots, m\}.$$

Med denna notation kan vi skriva

$$\begin{aligned} \text{minimera} \quad & f(x) \\ \text{då} \quad & x \in S. \end{aligned} \tag{2.2}$$

Vi definierar nu vad som menas med begreppet global minimipunkt till (2.2).

Definition 2.1 (Global minimipunkt, [17, s. 82])**.** Vi säger att $x^* \in S$ är en global minimipunkt till (2.2) om

$$f(x^*) \leq f(x), \quad \forall\, x \in S.$$

För att kunna definiera begreppet lokal minimipunkt måste vi ha begreppet boll.

Definition 2.2 (Boll i $\mathbb{R}^n$)**.** En boll $B_r(x) \subseteq \mathbb{R}^n$ centrerad i punkten $x \in \mathbb{R}^n$ definieras som

$$B_r(x) := \{y \in \mathbb{R}^n \, : \, \|x - y\|_2 \leq r\},$$

där $\|\cdot\|_2$ betecknar den euklidiska normen.

Med hjälp av detta kan vi definiera begreppet lokal minimipunkt.

Definition 2.3 (Lokal minimipunkt, [17, s. 82])**.** Vi säger att $x^* \in S$ är en lokal minimipunkt till (2.2) om

$$\exists\, \varepsilon > 0 \text{ så att } f(x^*) \leq f(x), \quad \forall\, x \in S \cap B_\varepsilon(x^*).$$

Vi definierar även begreppen konvex mängd och konvex funktion.

Definition 2.4 (Konvex mängd, [17, s. 43])**.** Mängden S är konvex om

$$tx + (1 - t)y \in S,$$

för alla $x, y \in S$ och $t \in [0, 1]$.

Definition 2.5 (Konvex funktion, [17, s. 65])**.** Låt S vara en konvex mängd. En funktion $f : S \to \mathbb{R}$ är konvex om

$$f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y)$$

för alla $x, y \in S$ och $t \in [0, 1]$.

Vi säger att (2.2) är ett konvext optimeringsproblem om f är en konvex funktion och S är en konvex mängd. Om (2.2) i stället hade varit ett maximeringsproblem hade vi krävt att f skulle vara en konkav funktion, vilket innebär precis det som står i definition 2.5 fast med omvänd olikhet. Att begränsa sig till minimeringsproblem är inte en inskränkning eftersom vi alltid kan skriva

$$\underset{x \in S}{\text{maximera}} \; f(x) = -\left(\underset{x \in S}{\text{minimera}} \; -f(x)\right).$$

För konvexa optimeringsproblem har vi följande viktiga sats.

Sats 2.1 (Konvexoptimeringens fundamentalsats, [17, s. 82])**.** För ett konvext optimeringsproblem är varje lokal minimipunkt även en global minimipunkt.

Nedan ger vi några andra användbara resultat om konvexa funktioner.

Sats 2.2. [17, s. 94] Antag att $f : \mathbb{R}^n \to \mathbb{R}$ är konvex och differentierbar på $\mathbb{R}^n$. Då gäller det att

$$x^* \text{ är ett globalt minimum till } f \text{ på } \mathbb{R}^n \iff \nabla f(x^*) = \mathbf{0},$$

där $\mathbf{0} = (0, \ldots, 0)^T \in \mathbb{R}^n$.

Sats 2.3. Linjära funktioner är konvexa.

Bevis. Se Appendix A. □

Sats 2.4. Positiva linjärkombinationer av konvexa funktioner är konvexa.

Bevis. Se Appendix A. □

Om vi i (2.1) har att $f, f_1, \ldots, f_m$ är linjära funktioner kallar vi detta för ett linjärprogrammeringsproblem. Bivillkoren definierar i detta fall den så kallade polytopen

$$P := \{x \in \mathbb{R}^n \,:\, f_i(x) \leq b_i \text{ för } i = 1, \ldots, m\} \tag{2.3}$$

av tillåtna lösningar. Vi säger att $z \in P$ är en extrempunkt till P om $\nexists\, x, y \in P$ och $t \in (0, 1)$ så att $x \neq y$ och $z = tx + (1 - t)y$. Uttryckt i ord betyder detta att extrempunkterna till P är de punkter som inte går att skriva som strikta konvexkombinationer av andra punkter i P.

För linjärprogrammering har vi följande centrala resultat.

Sats 2.5. [18, sats 2.7] Anta att vi har ett linjärprogrammeringsproblem och att det finns åtminstone en extrempunkt i dess polytop P av tillåtna lösningar, där P definieras av (2.3). Anta vidare att det existerar en optimal lösning. Det existerar då en optimal lösning som är en extrempunkt till P.

Vi definierar nu begreppet dubbelstokastisk matris och permutationsmatris.

Definition 2.6 (Dubbelstokastisk matris, [19, s. 9-11])**.** En dubbelstokastisk matris $A \in \mathbb{R}_+^{N \times N}$ är en matris som uppfyller

$$A\mathbf{1} = \mathbf{1} \quad \text{och} \quad A^T\mathbf{1} = \mathbf{1},$$

det vill säga att varje rad och kolumn summerar till 1.

Definition 2.7 (Permutationsmatris)**.** En permutationsmatris är en dubbelstokastisk matris där varje element är antingen 0 eller 1. En permutationsmatris är med andra ord en kvadratisk binär matris där varje rad och varje kolumn har exakt en etta.

Polytopen R av alla dubbelstokastiska matriser definieras som

$$R := \left\{A \in \mathbb{R}_+^{N \times N} \,:\, A\mathbf{1} = \mathbf{1} \text{ och } A^T\mathbf{1} = \mathbf{1}\right\}.$$

Extrempunkterna till R definieras helt analogt med extrempunkterna till P, det är alltså de matriser som inte går att skriva som en strikt konvexkombination av andra matriser i R.

Slutligen återger vi en av Birkoffs satser om hur permutationsmatriser och polytopen av alla dubbelstokastiska matriser hänger ihop.

Sats 2.6. [20] Varje permutationsmatris är en extrempunkt till R och varje extrempunkt till R är en permutationsmatris.

2.2 Matchningsproblemet

Som en introduktion till optimaltransport betraktar vi först ett specialfall av detta som handlar om att beräkna optimala ihopparningar. Under denna del av avsnittet hänvisar vi till [21].

Låt

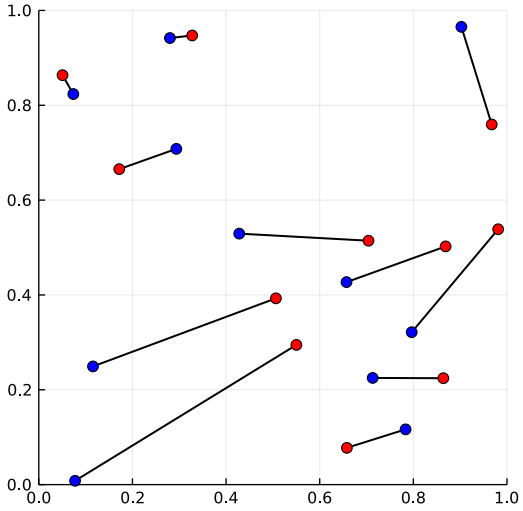
$$x^{(1)} := \left\{x_i^{(1)}\right\}_{i=1}^{N} \quad \text{och} \quad x^{(2)} := \left\{x_j^{(2)}\right\}_{j=1}^{N}$$

vara två mängder av agenter där $x_i^{(l)} \in \mathbb{R}^d$ för $i = 1, \ldots, N$; $l = 1, 2$ och $d \in \mathbb{N}$. Rummet som agenterna tillhör, vilket är $\mathbb{R}^d$ i detta fall, kallar vi för tillståndsrummet. Vi definierar matchningsproblemet mellan $x^{(1)}$ och $x^{(2)}$ som optimeringsproblemet

$$\underset{\phi \in \text{perm}(\{1,2,\ldots,N\})}{\text{minimera}} \sum_{i=1}^{N} \left\| x_i^{(1)} - x_{\phi(i)}^{(2)} \right\|_2^2, \tag{2.4}$$

där perm$(\{1, \ldots, N\})$ är mängden av alla permutationer av mängden $\{1, \ldots, N\}$. En permutation $\phi : \{1, ..., N\} \to \{1, ..., N\}$ är en bijektiv avbildning från mängden $\{1, ..., N\}$ till sig själv, och $\phi(i)$ är alltså bilden av i under ϕ.

Problemet som definierats i (2.4) ska tolkas som att hitta den bijektion mellan agenterna i $x^{(1)}$ och agenterna i $x^{(2)}$ som minimerar summan av det kvadrerade avståndet mellan paren med avseende på den euklidiska normen. I figur 1 visas ett exempel på hur en sådan optimal ihopparning ser ut.



Figur 1: En lösning på ett matchningsproblem, definierat i (2.4), där det finns 11 agenter, alltså $N = 11$, som ska paras ihop. Agenterna som paras ihop är de blåa och röda punkterna och linjen emellan illustrerar den optimala matchningen. Det optimala värdet fås då summa av alla längder i kvadrat är så liten som möjligt.

Det naiva sättet för att hitta lösningen till matchningsproblemet är att generera alla möjliga permutationer, beräkna deras kostnader och sedan jämföra för att se vilken permutation som är optimal. Detta leder till en kombinatorisk explosion och gör att även små exempel tar för lång tid att beräkna.

En annan metod för att lösa det kombinatoriska problemet börjar med att vi omformulerar matchningsproblemet som ett linjärprogrammeringsproblem med heltalsbivillkor. Låt därför $C = [C_{ij}]_{i,j=1}^{N} \in \mathbb{R}^{N \times N}$, där $C_{ij} = \left\| x_i^{(1)} - x_j^{(2)} \right\|_2^2$, vara en matris som representerar kostnaderna

för varje ihopparning. Matchningsproblemet kan då formuleras som

$$\begin{aligned} \underset{M\in\{0,1\}^{N\times N}}{\text{minimera}} \quad & \langle C, M\rangle \\ \text{då} \quad & \sum_{j=1}^{N} M_{ij} = 1, \quad \text{för } i = 1,\dots,N \\ & \sum_{i=1}^{N} M_{ij} = 1, \quad \text{för } j = 1,\dots,N, \end{aligned} \tag{2.5}$$

där vi använt den inre produkten

$$\langle C, M\rangle := \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}M_{ij}.$$

Bivillkoren i (2.5) kräver att M är en permutationsmatris. Det garanterar att varje agent i ena mängden paras ihop med exakt en agent i den andra mängden, vilket går att uttrycka som

$$M_{ij} = \begin{cases} 1, & \text{om agent } x_i^{(1)} \text{ paras ihop med } x_j^{(2)} \\ 0, & \text{annars.} \end{cases}$$

Det betyder i sin tur att $\langle C, M\rangle$ helt enkelt är kostnaden för ihopparningen som svarar mot matrisen M. Det är således enkelt att se att formuleringen (2.5) är ekvivalent med den i (2.4).

Betrakta nu det relaxerade problemet

$$\begin{aligned} \underset{M\in\mathbb{R}_+^{N\times N}}{\text{minimera}} \quad & \langle C, M\rangle \\ \text{då} \quad & \sum_{j=1}^{N} M_{ij} = 1, \quad \text{för } i = 1,\dots,N \\ & \sum_{i=1}^{N} M_{ij} = 1, \quad \text{för } j = 1,\dots,N, \end{aligned} \tag{2.6}$$

där vi tagit bort heltalsbivillkoren. Detta är ett linjärprogrammeringsproblem där polytopen av tillåtna lösningar är mängden av dubbelstokastiska matriser. Polytopen har åtminstone en extrempunkt, och eftersom den är kompakt och icke-tom vet vi även att det existerar en optimal lösning till (2.6). Således är förutsättningarna för att använda sats 2.5 uppfyllda, och vi kan dra slutsatsen att det existerar en optimal lösning till (2.6) som är en extrempunkt till polytopen av dubbelstokastiska matriser. Men vi vet enligt sats 2.6 att alla sådana punkter är permutationsmatriser. Med andra ord vet vi att det existerar en optimal lösning till (2.6) som är en permutationsmatris, vilket innebär att (2.4), (2.5) och (2.6) antar samma optimala värde, och om man har en optimal lösning till ett av problemen har man en optimal lösning till alla. Av dessa är (2.6) ett linjärprogrammeringsproblem som kan lösas effektivt.

Problem (2.6) är ett specialfall av ett diskret bimarginellt optimaltransportproblem. Ett generellt sådant problem kan skrivas som

$$\begin{aligned} \underset{M\in\mathbb{R}_+^{N\times N}}{\text{minimera}} \quad & \langle C, M\rangle \\ \text{då} \quad & \sum_{j=1}^{N} M_{ij} = \mu_{1,i}, \quad \text{för } i = 1,\dots,N \\ & \sum_{i=1}^{N} M_{ij} = \mu_{2,j}, \quad \text{för } j = 1,\dots,N, \end{aligned} \tag{2.7}$$

där $\mu_1, \mu_2 \in \mathbb{R}_+^N$, och där vi kräver att

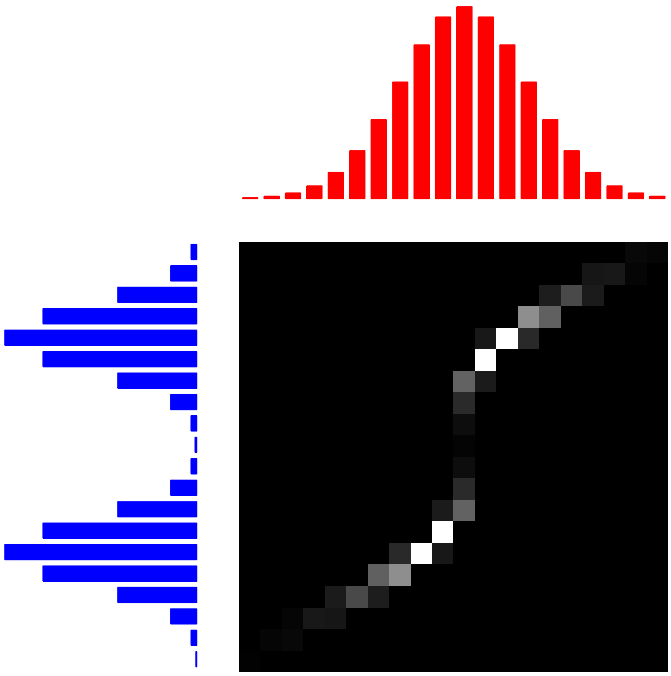
$$\sum_{j=1}^{N} \mu_{1,j} = \sum_{i=1}^{N} \mu_{2,i}.$$

De två vektorerna μ_1 och μ_2 kallas för problemets marginaler och M kallas för transportplanen. Specialfallet (2.6) fås genom att låta $\mu_1 = \mu_2 = \mathbf{1}$. Följaktligen kan vi ur detta perspektiv förstå optimaltransport som en form av mer generella matchningar mellan fördelningar.

2.3 Optimaltransport

Vi fortsätter även i denna del att hänvisa till [21]. I avsnitt 2.2 introducerades det diskreta bimarginella optimaltransportproblemet (2.7). Problemet ska tolkas som att hitta den transportplan M som minimerar kostnaden att transportera massan från den givna marginalen μ_1 till den andra givna marginalen μ_2. Notera att vi nu använder begreppet massa eftersom vi relaxerat heltalsbivillkoret.

En visualisering av ett exempel av diskret optimaltransport visas i figur 2, där kostnadsmatrisen $C_{ij} = |i-j|^2$ har använts. Transportplanen visas i figuren som ett rutnät där varje ruta motsvarar en massförflyttning från en punkt på μ_1 till en på μ_2, där ljusare färg motsvarar att mer massa förflyttats. Det syns tydligt att det är störst transportaktivitet vid de punkter på marginalerna där det finns som mest massa.



Figur 2: En visualisering av den optimala transportplanen för ett diskret optimaltransportproblem på formen (2.7). De röda staplarna är fördelningen av massa i μ_1 och den blåa fördelningen av massa i μ_2. Kvadraten i mitten är transportplanen, där ljusare färg representerar att mer massa flyttats.

Det är även möjligt att formulera en kontinuerlig variant av bimarginell optimaltransport. Denna tar formen

$$\begin{aligned} \underset{m \in \mathcal{M}_+(X\times X)}{\text{minimera}} \quad & \int_{X\times X} c(x_1,x_2) m(x_1,x_2)\, dx_1 dx_2 \\ \text{då} \quad & \int_X m(x_1,x_2)\, dx_2 = \nu_1(x_1) \\ & \int_X m(x_1,x_2)\, dx_1 = \nu_2(x_2), \end{aligned}$$

där $\mathcal{M}_+(\cdot)$ används för att beteckna mängden av icke-negativa funktioner. Notera att kostnadsfunktionen $c \in \mathcal{M}_+(X \times X)$ och att marginalerna $\nu_1, \nu_2 \in \mathcal{M}_+(X)$.

Denna form är praktisk eftersom det ofta är enklare att modellera problem på kontinuerlig form. För att det ska vara möjligt att använda numeriska metoder måste denna dock diskretiseras till ett diskret optimaltransportproblem. Detta görs genom följande procedur: Anta att $X = [a, b]$ är ett interval på den reella tallinjen. Vi definierar sedan en partitionering av detta intervall med hjälp av de ekvidistanta punkterna $x_i := a + h(i-1)$ för $i = 1, \dots, N+1$ där $h := (b-a)/N$. Denna partitionering kan vi ta hjälp av för att definiera

$$\begin{aligned} C_{ij} &:= c(x_i + h/2, x_j + h/2), && \text{för } i = 1, \dots, N;\ j = 1, \dots, N, \\ \mu_{1,i} &:= h\nu_1(x_i + h/2), && \text{för } i = 1, \dots, N, \\ \text{och} \quad \mu_{2,j} &:= h\nu_2(x_j + h/2), && \text{för } j = 1, \dots, N. \end{aligned}$$

Observera att $\nu_1, \nu_2 \in \mathcal{M}_+(X)$, medan $\mu_1, \mu_2 \in \mathbb{R}_+^N$. Vi approximerar sedan integraler med Riemannsummor, det vill säga vi gör approximationerna

$$\begin{aligned} \int_{X\times X} c(x_1, x_2) m(x_1, x_2)\, dx_1 dx_2 &\approx \sum_{i=1}^N \sum_{j=1}^N C_{ij} M'_{ij} h^2 = \sum_{i=1}^N \sum_{j=1}^N C_{ij} M_{ij}, \\ h \int_X m(x_i, x_2)\, dx_2 &\approx \sum_{j=1}^N M'_{ij} h^2 = \sum_{j=1}^N M_{ij}, \quad \text{för } i = 1, \dots, N, \\ \text{och} \quad h \int_X m(x_1, x_j)\, dx_1 &\approx \sum_{i=1}^N M'_{ij} h^2 = \sum_{i=1}^N M_{ij}, \quad \text{för } j = 1, \dots, N, \end{aligned}$$

där vi även infört de diskreta variablerna M'_{ij} och M_{ij} som relateras genom $M_{ij} := M'_{ij} h^2$ för $i = 1, \dots, N$; $j = 1, \dots, N$. Med de uppskattningar vi gjort får vi ett diskret optimaltransportproblem på formen (2.7). Analys av eventuella fel som kommer från denna diskretisering ligger utanför ramen för denna framställning.

Det diskreta bimarginella optimaltransportproblemet går att generalisera genom att tillåta godtyckligt antal marginaler. Ett sådant problem med L marginaler har formen

$$\begin{aligned} \underset{M \in \mathbb{R}_+^{N^L}}{\text{minimera}} \quad & \langle C, M \rangle \\ \text{då} \quad & P_l(M) = \mu_l, \quad \text{för } l \in \Gamma_=, \end{aligned} \tag{2.8}$$

där $\Gamma_= \subseteq \{1, \dots, L\}$ och där transportplanen M och kostnaden C nu är multidimensionella matriser, även kallade tensorer. Den inre produkten betyder alltså här

$$\langle C, M \rangle := \sum_{i_1=1}^N \cdots \sum_{i_L=1}^N C_{i_1,\dots,i_L} M_{i_1,\dots,i_L} \tag{2.9}$$

och $P_l(M) \in \mathbb{R}_+^N$ betyder projektionen av transporttensorn ner på marginal l, det vill säga

$$[P_l(M)]_i =: \sum_{i_1=1}^N \cdots \sum_{i_{l-1}=1}^N \sum_{i_{l+1}=1}^N \cdots \sum_{i_L=1}^N M_{i_1,\dots,i_{l-1},i,i_{l+1},\dots,i_N}, \quad \text{för } i = 1, \dots, N. \tag{2.10}$$

I denna formulering ska $M_{i_1,\dots,i_L}$ tolkas som mängden massa som förflyttas längs vägen som består av punkterna $i_1, \dots, i_L$ på marginalerna, där $i_l \in \{1, \dots, N\}$ för $l = 1, \dots, L$. Därmed låter den multimarginella formuleringen oss att hantera mer komplexa transportproblem med fler steg och marginaler.

3 Interpolation och styrning mellan fördelningar av agenter

Vi kommer nu att introducera det så kallade interpolationsproblemet för en svärm agenter. Problemet är ett optimeringsproblem som är uppbyggt av flera optimaltransportproblem och går att koppla till en viss grafstruktur. Problemet kan enkelt tillämpas praktiskt, vilket vi kommer att se

i avsnitt 5. Vi kommer att visa att interpolationsproblemet kan ses som en sammansättning av bimarginella optimaltransportproblem, och att det är möjligt att formulera detta som ett enda multimarginellt optimaltransportproblem. Avsnittet är en adaption av [21, s. 54-55, 58-62].

Vi börjar med att betrakta det kontinuerliga bimarginella optimaltransportproblemet

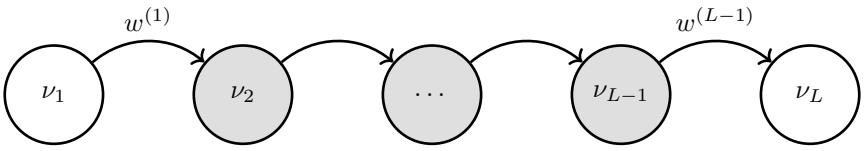
$$\begin{aligned} T(\nu_1,\nu_2) := \underset{w\in\mathcal{M}_+(X\times X)}{\text{minimera}} \quad & \int_{X\times X} q(x_1,x_2)w(x_1,x_2)\,dx_1dx_2 \\ \text{då} \quad & \int_X w(x_1,x_2)\,dx_2 = \nu_1(x_1) \\ & \int_X w(x_1,x_2)\,dx_1 = \nu_2(x_2). \end{aligned} \tag{3.1}$$

$T(\nu_1,\nu_2)$ ska i denna formulering tolkas som den minsta kostnaden för att transportera svärmen av agenter från en given startmarginal ν_1 till en given slutmarginal ν_2, med avseende på någon kostnadsfunktion q.

Betrakta nu interpolationsproblemet

$$\underset{\nu_l\in\mathcal{M}_+(X),\ l=2,\dots,L-1}{\text{minimera}} \quad \sum_{l=1}^{L-1} T(\nu_l,\nu_{l+1}), \tag{3.2}$$

där ν_1 och ν_L är givna och fixa. Se figur 3 för en schematisk bild över grafen som problemet motsvarar. Problemet kan tolkas som att hitta de mellanliggande marginalerna ν_l för $l=2,\dots,L-1$ som minimerar den totala kostnaden för att styra agenterna från startfördelningen ν_1 till målfördelningen ν_L. Om vi hittar dessa vet vi vilka steg som agenterna tar och kan därför styra dem från startfördelningen till målfördelningen på det mest effektiva sättet med avseende på kostnadsfunktionen q.



Figur 3: Grafstrukturen som problem (3.2) representerar. De vita noderna motsvarar kända fördelningar och de grå motsvarar okända fördelningar. Varje kant motsvarar ett transportsteg mellan två fördelningar. Målet i interpolationsproblemet är att hitta transportplanerna $w^{(l)}$ för $l=1,\dots,L-1$ (och således också de okända fördelningarna $\nu_2,\dots,\nu_{L-1}$) som minimerar den totala kostnaden för transporten med avseende på kostnadsfunktionen q.

Varje kontinuerligt problem $T(\nu_l,\nu_{l+1})$ diskretiseras nu i enlighet med beskrivningen i avsnitt 2.3 till ett diskret problem $\Theta(\mu_l,\mu_{l+1})$ på formen (2.7). Vi låter här $W^{(l)}\in\mathbb{R}_+^{N\times N}$ beteckna den diskreta transportplanen från μ_l till μ_{l+1} och $Q\in\mathbb{R}_+^{N\times N}$ den diskreta kostnaden. Det diskreta interpolationsproblemet kan alltså skrivas som

$$\underset{\mu_l,\ l=2,\dots,L-1}{\text{minimera}} \quad \sum_{l=1}^{L-1}\Theta(\mu_l,\mu_{l+1}). \tag{3.3}$$

Härnäst inser vi att problem (3.3) går att skriva som

$$\underset{\substack{W^{(l)}\in\mathbb{R}_+^{N\times N},\ l=1,\dots,L-1\\ \mu_l,\ l=2,\dots,L-1}}{\text{minimera}} \quad \sum_{l=1}^{L-1}\langle Q,W^{(l)}\rangle \tag{3.4a}$$

$$\text{då} \quad \sum_{j=1}^{N} W_{ij}^{(l)} = \mu_{l,i}, \qquad \text{för } i=1,\dots,N;\ l=1,\dots,L-1, \tag{3.4b}$$

$$\sum_{i=1}^{N} W_{ij}^{(l)} = \mu_{l+1,j}, \quad \text{för } j=1,\dots,N;\ l=1,\dots,L-1, \tag{3.4c}$$

där vi har skrivit ut definitionen av $\Theta(\mu_l, \mu_{l+1})$ och använt resultaten från appendix B, som kortfattat säger att minimering av en summa av minimeringsproblem kan skrivas om som ett enda stort minimeringsproblem över alla variabler.

Vi noterar nu att bivillkoren i (3.4b) för $i = 1, \ldots, N$; $l = 2, \ldots, L-1$ och i (3.4c) för $j = 1, \ldots, N$; $l = 1, \ldots, L-2$ inte påverkar mängden av tillåtna lösningar. Ett sätt att förvissa sig om detta är att tänka att vi löser optimeringsproblemet i (3.4) och ser vilka de optimala värdena är för $W_{ij}^{(l)}$ för $l = 1, \ldots, L-1$ och $i = 1, \ldots, N$; $j = 1, \ldots, N$. Dessa bestämmer helt värdet på målfunktionen. Givet dessa värden kan vi sedan alltid välja värdet på variablerna i högerleden i (3.4b) och (3.4c) så att bivillkoren blir uppfyllda, följaktligen kan man se det som att variablerna inte längre påverkar problemet.

Problem (3.4) kan då skrivas som

$$\begin{aligned} \underset{W^{(l)} \in \mathbb{R}_+^{N\times N},\, l=1,\ldots,L-1}{\text{minimera}} \quad & \sum_{l=1}^{L-1} \langle Q, W^{(l)} \rangle \\ \text{då} \quad & \sum_{j=1}^{N} W_{ij}^{(1)} = \mu_{1,i}, \qquad \text{för } i = 1, \ldots, N, \\ & \sum_{i=1}^{N} W_{ij}^{(L-1)} = \mu_{L,j}, \quad \text{för } j = 1, \ldots, N. \end{aligned} \tag{3.5}$$

För att få problemet på formen (2.8) introducerar vi tensorn $M \in \mathbb{R}_+^{N^L}$. Eftersom vi tolkar $M_{i_1,\ldots,i_L}$ som mängden massa som förflyttas längs vägen $i_1, \ldots, i_L$, där $i_l \in \{1, \ldots, N\}$ för $l = 1, \ldots, L$ definierar vi denna implicit genom att kräva att den uppfyller relationen

$$W_{i,j}^{(l)} = \sum_{i_1=1}^{N} \cdots \sum_{i_{l-1}=1}^{N} \sum_{i_{l+2}=1}^{N} \cdots \sum_{i_L=1}^{N} M_{i_1,\ldots,i_{l-1},i,j,i_{l+2}\ldots,i_L}. \tag{3.6}$$

Vi behöver även kostnadstensorn $C \in \mathbb{R}^{N^L}$. Eftersom $C_{i_1,\ldots,i_L}$ tolkas som kostnaden att förflytta en enhet massa längs vägen $i_1, \ldots, i_L$ definierar vi denna att helt enkelt vara summan av kostnaden för varje steg, det vill säga

$$C_{i_1,\ldots,i_L} := \sum_{l=1}^{L-1} Q_{i_l,i_{l+1}}, \tag{3.7}$$

där $i_l \in \{1, \ldots, N\}$ för $l = 1, \ldots, L$.

Vi visar nu att målfunktionen med avseende på de nya variablerna blir den inre produkten definierad i (2.9) mellan kostnadstensorn C och transporttensorn M. Vi har

$$\begin{aligned} \sum_{i_1=1}^{N} \cdots \sum_{i_L=1}^{N} C_{i_1,\ldots,i_L} M_{i_1,\ldots,i_L} &= \sum_{i_1=1}^{N} \cdots \sum_{i_L=1}^{N} (Q_{i_1,i_2} + \cdots + Q_{i_{L-1},i_L}) M_{i_1,\ldots,i_L} \\ &= \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} Q_{i_1,i_2} \sum_{i_3=1}^{N} \cdots \sum_{i_L=1}^{N} M_{i_1,\ldots,i_L} \\ &+ \sum_{i_2=1}^{N} \sum_{i_3=1}^{N} Q_{i_2,i_3} \sum_{i_1=1}^{N} \sum_{i_4=1}^{N} \cdots \sum_{i_L=1}^{N} M_{i_1,\ldots,i_L} \\ &+ \cdots \\ &+ \sum_{i_{L-1}=1}^{N} \sum_{i_L=1}^{N} Q_{i_{L-1},i_L} \sum_{i_1=1}^{N} \cdots \sum_{i_{L-2}=1}^{N} M_{i_1,\ldots,i_L} \\ &= \sum_{l=1}^{L-1} \sum_{i=1}^{N} \sum_{j=1}^{N} Q_{ij} W_{ij}^{(l)}, \end{aligned} \tag{3.8}$$

där vi i första likheten använt (3.7) och i den andra delat upp summan samt utnyttjat att varje kostnadsterm endast beror på två av indexen i summeringen. I sista likheten har vi bytt indexeringsvariabler. Vi har alltså

$$\sum_{l=1}^{L-1} \langle Q, W^{(l)} \rangle = \langle C, M \rangle. \tag{3.9}$$

Genom att substituera (3.6) och (3.9) i (3.5) får vi

$$\begin{aligned} \underset{M \in \mathbb{R}_+^{N^L}}{\text{minimera}} \quad & \langle C, M \rangle \\ \text{då} \quad & P_l(M) = \mu_l, \quad \text{för } l = 1, L, \end{aligned} \tag{3.10}$$

där vi har förenklat genom att använda notationen från (2.10) för projektionen av en tensor ner på en dimension. Vi noterar att (3.10) är på formen (2.8), det vill säga ett multimarginellt optimaltransportproblem, vilket var vad vi ville åstadkomma. Observera att vi här tekniskt sett borde säga att vi fortfarande optimerar över variablerna $W^{(l)}$ och har kvar villkoren (3.6) som bivillkor. Dessa kan dock tas bort med samma motivering som tidigare när (3.4) skrevs om som (3.5).

Notera att vi här endast har villkor på marginalerna då $l = 1, L$. Genom en liknande härledning kan vi formulera interpolationsproblemet med ytterligare bivillkor på agenterna för de resterande marginalerna som ett multimarginellt optimaltransportproblem. Det senare problemet har dessutom fortfarande samma grafstruktur i kostnadstensorn C. Villkoren behöver inte heller endast vara likhetsbivillkor, utan kan innehålla olikheter. Ett generellt sådant problem är

$$\underset{M \in \mathbb{R}_+^{N^L}}{\text{minimera}} \quad \langle C, M \rangle \tag{3.11a}$$

$$\begin{aligned} \text{då} \quad & P_l(M) \leq \mu_l, \quad \text{för } l \in \Gamma_{\leq}, && \text{(3.11b)} \\ & P_l(M) = \mu_l, \quad \text{för } l \in \Gamma_{=}, && \text{(3.11c)} \\ & P_l(M) \geq \mu_l, \quad \text{för } l \in \Gamma_{\geq}, && \text{(3.11d)} \end{aligned}$$

där mängderna $\Gamma_{\leq}, \Gamma_{=}, \Gamma_{\geq} \subseteq \{1, \ldots, L\}$ är disjunkta och där C fortfarande har formen (3.7). Här ska bivillkoren i (3.11b) tolkas som att det i varje interval i vår diskretisering endast får finnas en viss mängd agenter. Bivillkoren i (3.11c) ska tolkas som innan, vilket innebär att agenterna måste exakt uppfylla en viss fördelning. De sista bivillkoren i (3.11d) ska tolkas som att det i varje interval måste finnas minst en viss mängd agenter. Denna påbyggnad är ett användbart modelleringsverktyg som vi kommer att använda i avsnitt 5.

Observera att (3.11) är ett linjärprogrammeringsproblem med N^L variabler. Problem uppstår då antalet marginaler ökas eftersom antalet variabler växer exponentiellt i L. Detta gör att problemet fort blir ohanterbart beräkningsmässigt för vanliga lösningsmetoder för linjärprogrammeringsproblem. Om vi exempelvis vill interpolera i 50 marginaler i ett system där agenterna kan inta något av tillstånden i ett 100×100 rutnät skulle det innebära ett linjärprogrammeringsproblem med 10^{200} variabler. Detta kan jämföras med uppskattningar av antalet atomer i den observerbara delen av universum som ligger mellan 10^{78} och 10^{82}. Vi söker alltså nya metoder för att numeriskt kunna lösa (3.11), vilket är temat för det nästkommande avsnittet.

4 Entropisk regularisering och Sinkhorniterationer

Detta avsnitt är uppdelat i två delar. I första delen kommer vi att introducera begreppet entropisk regularisering och därefter kommer vi att härleda de så kallade Sinkhorniterationerna för ett bimarginellt optimaltransportproblem. Sinkhorniterationerna ger en effektiv numerisk metod för att beräkna approximativa lösningar till optimaltransportproblem. I andra delen generaliseras resultatet från första delen och här kommer Sinkhorniterationerna härledas för multimarginella optimaltransportproblem. Eftersom många problem, som exempelvis interpolationsproblemet från avsnitt 3, går att formulera som multimarginella optimaltransportproblem är detta ett viktigt steg mot att kunna tillämpa teorin praktiskt. Genom det här avsnittet hänvisar vi till [21].

4.1 Härledning av Sinkhorniterationerna för bimarginella optimaltransportproblem med likhetsvillkor

I denna del av avsnittet kommer vi att härleda Sinkhorniterationerna. Här kommer vi för enkelhetens skull göra detta för det bimarginella fallet där vi endast har likhetsbivillkor på marginalerna. Härledningen för det multimarginella fallet med blandade likhets- och olikhetsbivillkor görs i avsnitt 4.2.

Betrakta det diskreta bimarginella optimaltransportproblemet

$$\begin{aligned} \underset{M\in\mathbb{R}_+^{N\times N}}{\text{minimera}} \quad & \langle C, M\rangle \\ \text{då} \quad & \sum_{j=1}^{N} M_{ij} = \mu_{1,i}, \quad \text{för } i = 1,\dots,N, \\ & \sum_{i=1}^{N} M_{ij} = \mu_{2,j}, \quad \text{för } j = 1,\dots,N. \end{aligned} \tag{4.1}$$

Vi har redan konstaterat att det krävs mer effektiva numeriska metoder för att hitta lösningar till problem av formen (4.1). En relativt ny metod för detta beskrivs i [5, avsnitt 4] och går ut på att den så kallade entropiska regulariseringstermen $\varepsilon D(M)$ adderas till målfunktionen där $\varepsilon > 0$ och

$$D(M) := \begin{cases} \sum_{i=1}^{N}\sum_{j=1}^{N}\Big[M_{ij}\ln(M_{ij}) - M_{ij} + 1\Big], & \text{då } M_{ij} \geq 0 \;\forall\, i,j \\ \infty, & \text{annars,} \end{cases} \tag{4.2}$$

där vi definierar $0\ln(0) = 0$, eftersom $x\ln x \to 0$ då $x \to 0^+$. Termen $D(M)$ kallas för en entropiterm och ε kallas för en regulariseringsparameter. Vi får då problemet

$$\begin{aligned} \underset{M\in\mathbb{R}_+^{N\times N}}{\text{minimera}} \quad & \langle C, M\rangle + \varepsilon D(M) \\ \text{då} \quad & \sum_{j=1}^{N} M_{ij} = \mu_{1,i}, \quad \text{för } i = 1,\dots,N, \\ & \sum_{i=1}^{N} M_{ij} = \mu_{2,j}, \quad \text{för } j = 1,\dots,N. \end{aligned} \tag{4.3}$$

När $\varepsilon \to 0$ går lösningen M till (4.3) mot den optimala lösningen till (4.1) [5, proposition 4.1]. Valet att addera just $\varepsilon D(M)$ för att få (4.3) kan tyckas vara godtyckligt, men det kommer att visa sig att detta leder till en effektiv numerisk metod för att lösa problem av formen (4.1).

Härledningen börjar med att vi Lagrangerelaxerar alla bivillkor i (4.3), som vi vet är ett konvext problem till följd av lemma 4.2 samt sats 2.4. Vi får då Lagrangefunktionen

$$\mathcal{L}(M, \lambda_1, \lambda_2) = \langle C, M\rangle + \varepsilon D(M) - \lambda_1^T(\mu_1 - M\mathbf{1}) - \lambda_2^T(\mu_2 - M^T\mathbf{1})$$

där $\lambda_1, \lambda_2 \in \mathbb{R}^N$ är Lagrangemultiplikatorer. Den duala Lagrangefunktionen är, per definition

$$\varphi(\lambda_1, \lambda_2) := \min_{M\in\mathbb{R}_+^{N\times N}} \mathcal{L}(M, \lambda_1, \lambda_2) \tag{4.4}$$

och det duala problemet kan nu formuleras som

$$\max_{\lambda_1,\lambda_2\in\mathbb{R}^N} \varphi(\lambda_1, \lambda_2). \tag{4.5}$$

Vi visar nu att (4.4) är ett konvext optimeringsproblem genom att använda följande olikhet.

Lemma 4.1 (Logaritmsummaolikheten, [22, s. 31])**.** Låt $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ vara icke-negativa tal och låt $a = \sum_{i=1}^{n} a_i$ samt $b = \sum_{i=1}^{n} b_i$. Då gäller det att

$$\sum_{i=1}^{n} a_i \ln \frac{a_i}{b_i} \geq a \ln \frac{a}{b},$$

med likhet om och endast om det existerar c så att $a_i = cb_i$ för $i = 1, 2, \ldots, n$.

Vi kommer även att ha användning av följande hjälpsats.

Lemma 4.2. Låt $D : \mathbb{R}^{N\times N} \to \mathbb{R}$ och definierad enligt (4.2). Då gäller det att D är konvex, det vill säga

$$D(tA + (1-t)B) \leq tD(A) + (1-t)D(B)$$

för alla $A, B \in \mathbb{R}_+^{N\times N}$ och $t \in [0, 1]$.

Bevis. Vi har

$$\begin{aligned}
D(tA + (1-t)B) &= \sum_{i=1}^{N}\sum_{j=1}^{N}\Big[(tA_{ij} + (1-t)B_{ij}))\ln(tA_{ij} + (1-t)B_{ij})) - (tA_{ij} + (1-t)B_{ij})) + 1\Big] \\
&\leq \sum_{i=1}^{N}\sum_{j=1}^{N}\Big[(tA_{ij}\ln(A_{ij}) + (1-t)B_{ij}\ln(B_{ij}) - (tA_{ij} + (1-t)B_{ij})) + t + (1-t)\Big] \\
&= t\sum_{i=1}^{N}\sum_{j=1}^{N}\Big[A_{ij}\ln(A_{ij}) - A_{ij} + 1\Big] + (1-t)\sum_{i=1}^{N}\sum_{j=1}^{N}\Big[B_{ij}\ln(B_{ij}) - B_{ij} + 1\Big] \\
&= tD(A) + (1-t)D(B),
\end{aligned}$$

där vi använt olikheten från lemma 4.1. □

Med dessa förberedelser visar vi nu att (4.4) är ett konvext optimeringsproblem.

Sats 4.3. Optimeringsproblemet

$$\min_{M\in\mathbb{R}_+^{N\times N}} \mathcal{L}(M, \lambda_1, \lambda_2)$$

är konvext.

Bevis. Vi måste visa dels att $\mathcal{L}(M, \lambda_1, \lambda_2)$ är en konvex funktion i M och dels att $\mathbb{R}_+^{N\times N}$ är en konvex mängd. Vi noterar att det senare följer direkt från att summor och produkter av icke-negativa tal är icke-negativa och fortsätter med att bevisa det tidigare.

Eftersom $\mathcal{L}(M, \lambda_1, \lambda_2)$ är en linjärkombination med positiva koefficienter av linjära och konvexa funktioner i M vet vi enligt sats 2.3 och 2.4 att $\mathcal{L}(M, \lambda_1, \lambda_2)$ är en konvex funktion i M. Alltså är påståendet visat. □

Enligt sats 2.1 vet vi nu att lokala minimipunkter till optimeringsproblemet (4.4) även är globala minimipunkter. Eftersom $\mathcal{L}(M, \lambda_1, \lambda_2)$ är konvex och differentierbar i M vet vi också enligt sats 2.2 att varje minimipunkt är antingen en stationär punkt eller en punkt på randen. Med detta som motivering deriverar vi nu Lagrangefunktionen med avseende på varje matriselement. Vi får

$$\frac{\partial}{\partial M_{ij}}\mathcal{L}(M, \lambda_1, \lambda_2) = C_{ij} + \lambda_{1,i} + \lambda_{2,j} + \varepsilon \ln M_{ij}.$$

För att undersöka om det finns en stationär punkt $M^* \in \mathbb{R}_+^{N\times N}$, som därmed skulle vara optimal, sätter vi alla partiella derivator till noll. Det ger ekvationen

$$0 = C_{ij} + \lambda_{1,i} + \lambda_{2,j} + \varepsilon \ln M^*_{ij}, \quad \text{för } i = 1, \ldots, N;\ j = 1, \ldots, N.$$

Vi löser för M^*_{ij} och får

$$M^*_{ij} = \exp\left(-\frac{C_{ij} + \lambda_{1,i} + \lambda_{2,j}}{\varepsilon}\right) \geq 0$$

för alla värden på C_{ij}, $\lambda_{1,i}$ och $\lambda_{2,j}$, alltså är detta en global minimipunkt till (4.4).

Målfunktionen φ i det duala problemet (4.5) kan nu skrivas som

$$\begin{aligned}
\varphi(\lambda_1,\lambda_2) &= \langle C, M^*\rangle + \varepsilon D(M^*) - \lambda_1^T(\mu_1 - M^*\mathbf{1}) - \lambda_2^T(\mu_2 - (M^*)^T\mathbf{1}) \\
&= \sum_{i=1}^N\sum_{j=1}^N C_{ij}\exp\left(-\frac{C_{ij}+\lambda_{1,i}+\lambda_{2,j}}{\varepsilon}\right) \\
&+ \varepsilon\sum_{i=1}^N\sum_{j=1}^N\left[\exp\left(-\frac{C_{ij}+\lambda_{1,i}+\lambda_{2,j}}{\varepsilon}\right)\left(-\frac{C_{ij}+\lambda_{1,i}+\lambda_{2,j}}{\varepsilon}\right)-\exp\left(-\frac{C_{ij}+\lambda_{1,i}+\lambda_{2,j}}{\varepsilon}\right)+1\right] \\
&- \sum_{i=1}^N\lambda_{1,i}\left(\mu_{1,i} - \sum_{j=1}^N\exp\left(-\frac{C_{ij}+\lambda_{1,i}+\lambda_{2,j}}{\varepsilon}\right)\right) \\
&- \sum_{j=1}^N\lambda_{2,j}\left(\mu_{2,j} - \sum_{i=1}^N\exp\left(-\frac{C_{ij}+\lambda_{1,i}+\lambda_{2,j}}{\varepsilon}\right)\right) \\
&= -\varepsilon\sum_{i=1}^N\sum_{j=1}^N\left[\exp\left(-\frac{C_{ij}+\lambda_{1,i}+\lambda_{2,j}}{\varepsilon}\right)+1\right] - \sum_{i=1}^N\lambda_{1,i}\mu_{1,i} - \sum_{j=1}^N\lambda_{2,j}\mu_{2,j}.
\end{aligned} \tag{4.6}$$

Problemet lämpar sig nu för numerisk lösning via koordinatvis optimering. För att se detta börjar vi med att visa följande resultat.

Sats 4.4. Optimeringsproblemet 4.5 är konvext i varje koordinat, det vill säga optimeringsproblemen

$$\max_{\lambda_1\in\mathbb{R}^N}\quad \varphi(\lambda_1,\lambda_2) \quad\text{och}\quad \max_{\lambda_2\in\mathbb{R}^N}\quad \varphi(\lambda_1,\lambda_2) \tag{4.7}$$

är konvexa.

Bevis. Mängden av tillåtna lösningar är $\mathbb{R}^N$ för båda problemen. Denna mängd är konvex, vilket innebär att det endast återstår att visa att målfunktionerna är konkava.

För detta åberopar vi sats A.1 som säger att den duala Lagrangefunktionen alltid är konkav. Alltså

$$\varphi(t\alpha+(1-t)\gamma, t\beta+(1-t)\delta) \geq t\varphi(\alpha,\beta)+(1-t)\varphi(\gamma,\delta)$$

för alla $\alpha,\beta,\gamma,\delta\in\mathbb{R}^N$ och $t\in[0,1]$. Speciellt har vi då

$$\varphi(tx+(1-t)y,\lambda_2) \geq t\varphi(x,\lambda_2)+(1-t)\varphi(y,\lambda_2)$$

för alla $x,y\in\mathbb{R}^N$ och $t\in[0,1]$, med andra ord är målfunktionen i det första problemet i (4.7) konkav. Konkavitet för den andra målfunktionen följer analogt. □

För undersöka om det finns en stationär punkt deriverar vi uttrycket för φ i (4.6) med avseende på $\lambda_{1,i}$ och sätter resultatet till noll. Vi får ekvationerna

$$0 = \frac{\partial\varphi}{\partial\lambda_{1,i}} = \sum_{j=1}^N\exp\left(-\frac{C_{ij}+\lambda_{1,i}+\lambda_{2,j}}{\varepsilon}\right) - \mu_{1,i}, \quad \text{för } i=1,\ldots,N. \tag{4.8}$$

Vi gör samma sak för $\lambda_{2,j}$, vilket ger ekvationerna

$$0 = \frac{\partial\varphi}{\partial\lambda_{2,j}} = \sum_{i=1}^N\exp\left(-\frac{C_{ij}+\lambda_{1,i}+\lambda_{2,j}}{\varepsilon}\right) - \mu_{2,j}, \quad \text{för } j=1,\ldots,N. \tag{4.9}$$

För att förenkla dessa ekvationer definierar vi

$$K_{ij} := \exp\left(-\frac{C_{jk}}{\varepsilon}\right), \quad u_i := \exp\left(-\frac{\lambda_{1,i}}{\varepsilon}\right) \quad\text{och}\quad v_j := \exp\left(-\frac{\lambda_{2,j}}{\varepsilon}\right).$$

Med denna notation blir (4.8)

$$\sum_{j=1}^{N} u_i K_{ij} v_j = \mu_{1,i}, \quad \text{för } i = 1, \dots, N.$$

Löser vi för u_i, får vi

$$u_i = \frac{\mu_{1,i}}{\sum_{j=1}^{N} K_{ij} v_j}, \quad \text{för } i = 1, \dots, N. \tag{4.10}$$

Låter vi ./ beteckna elementvis division av vektorer kan vi slå samman ekvationerna till

$$u = \mu_1 ./ (Kv). \tag{4.11}$$

Om vi i stället börjar med (4.9) får vi analogt

$$v = \mu_2 ./ (Ku). \tag{4.12}$$

Iterationsstegen som ekvation (4.11) och (4.12) beskriver kallas för Sinkhorniterationer [21, s. 63-64]. Genom att alternera de två stegen får vi en algoritm som maximerar den duala Lagrangefunktionen i en koordinat åt gången. Algoritmen konvergerar linjärt (se [5, sats 4.2]), vilket innebär att vi nu har en numerisk lösningsmetod till dualproblemet (4.5), vilket är precis det vi sökte.

För att försäkra oss om att en optimal lösning till (4.3) direkt implicerar en optimal lösning till (4.5) undersöker vi slutligen så kallad stark dualitet genom att ta hänsyn till Slaters villkor [16, avsnitt 5.2.3]. Stark dualitet innebär att de optimala värdena till de två problemen är lika med varandra, med andra ord att dualitetsgapet är noll [17, s. 165]. Därtill vet vi att om vi har stark dualitet och hittar $(\lambda_1^*, \lambda_2^*)$ optimal till det duala problemet (4.5) ger detta M^* optimal till (4.3).

I vårt fall är $S = \mathbb{R}_+^{N \times N}$ mängden där $D(M) \neq \infty$ för det primala problemet. Detta ger $\text{relint}(S) = \{A \in \mathbb{R}^{N \times N} : A > 0\}$, det vill säga mängden av alla strikt positiva matriser. Slaters villkor säger att stark dualitet råder om det existerar $M^* \in \text{relint}(S)$ så att bivillkoren i (4.3) är uppfyllda. Så länge vi har $\mu_{l,k} > 0$ för $l = 1, 2$; $k = 1, \dots, N$ är det enkelt att se att detta är uppfyllt.

Skulle vi ha en probleminstans där $\mu_{l,k} = 0$ för några $l \in \{1, 2\}$ och $k \in \{1, \dots, N\}$, säg exempelvis för $l = 1$ och $k = 1$, kan vi inte använda Slaters villkor för att visa stark dualitet eftersom vi då hade haft $M_{1,j} = 0$ för $j = 1, \dots, N$. Det hade i det fallet inte existerat något tillåtet M i $\text{relint}(S)$, vilket gör att Slaters villkor blir obrukbart.

Men eftersom vi vet att $M_{1,j} = 0$ för $j = 1, \dots, N$ kan vi konstruera ett nytt problem med dessa variabler fixerade och som ger ett motsvarande dualproblem. För detta problempar har vi då stark dualitet enligt Slaters villkor. Vi kan sedan för dualproblemet följa härledningen ovan för att ta fram en ny variant av Sinkhorniterationerna. Detta kommer i slutändan leda till att vi i (4.10) endast får uppdateringsregler för $i = 2, \dots, N$. Men fixerar vi $u_1 = 0$ (eftersom det är detta val som ger att första raden i M blir noll) och inför den artificiella regeln

$$u_1 = \frac{\mu_{1,1}}{\sum_{j=1}^{N} K_{1,j} v_j},$$

som enligt definition bara säger att $0 = 0$, kan Sinkhorniterationerna skrivas på exakt samma form som innan, det vill säga (4.11) och (4.12). Vi har ovan antagit $\mu_{l,k} = 0$ för just $l = k = 1$, men hela proceduren går att utföra helt analogt för andra och fler val av l och k.

Vi noterar även att nämnarvektorerna i de elementvisa divisionerna alltid är positiva om vi antar att $u, v \neq 0$ eftersom $K > 0$. Fallet $u = 0$ inträffar då $\mu_1 = 0$, vilket resulterar i att även $\mu_2 = 0$ och således att $v = 0$. Det innebär att $u = 0$ om och endast om $v = 0$, och $u = v = 0$ om och endast om $\mu_1 = \mu_2 = 0$. Detta leder till division med noll, men eftersom ett optimaltransportproblem med nollmarginaler inte är intressant kommer vi alltid att anta att minst ett element i varje marginal är strikt positivt.

4.2 Härledning av Sinkhorniterationerna för multimarginella optimaltransportproblem med likhets- och olikhetsvillkor

I den här delen härleder vi Sinkhorniterationerna för det multimarginella fallet där vi även har olikhetsvillkor på marginalerna. Eftersom härledningen är mycket lik den som gjordes i föregående avsnitt fokuserar vi på de huvudsakliga skillnaderna mellan fallen i stället för detaljerna.

Betrakta det generella diskreta multimarginella optimaltransportproblemet givet i (3.11). Som innan börjar vi med att addera den entropiska regulariseringstermen $\varepsilon D(M)$ till målfunktionen, vilket ger

$$\begin{aligned} \underset{M\in\mathbb{R}_+^{N^L}}{\text{minimera}} \quad & \langle C, M\rangle + \varepsilon D(M) && \text{(4.13a)}\\ \text{då} \quad & P_l(M) \le \mu_l, \quad \text{för } l \in \Gamma_{\le}, && \text{(4.13b)}\\ & P_l(M) = \mu_l, \quad \text{för } l \in \Gamma_{=}, && \text{(4.13c)}\\ & P_l(M) \ge \mu_l, \quad \text{för } l \in \Gamma_{\ge}, && \text{(4.13d)} \end{aligned}$$

där

$$D(M) := \begin{cases} \displaystyle\sum_{i_1=1}^{N} \cdots \sum_{i_L=1}^{N} \Big[M_{i_1,\ldots,i_L} \ln(M_{i_1,\ldots,i_L}) - M_{i_1,\ldots,i_L} + 1 \Big], & M_{i_1,\ldots i_L} \ge 0 \;\forall\, i_1,\ldots i_L \\ \infty, & \text{annars.} \end{cases}$$

Låt $\Gamma = \Gamma_{\le} \cup \Gamma_{=} \cup \Gamma_{\ge}$. Som innan relaxerar vi alla bivillkor och tecknar Lagrangefunktionen

$$\mathcal{L}(M, \Lambda) = \langle C, M\rangle + \varepsilon D(M) - \sum_{l\in\Gamma} \Lambda^{(l)}(\mu_l - P_l(M))$$

där $\Lambda^{(l)} \in \mathbb{R}^N$ för $l \in \Gamma$ är Lagrangemultiplikatorer. Den duala Lagrangefunktionen blir

$$\varphi(\Lambda) := \min_{M\in\mathbb{R}_+^{N^L}} \mathcal{L}(M, \Lambda) \tag{4.14}$$

och det duala problemet kan formuleras som

$$\begin{aligned} \max_{\Lambda^{(l)},\, l=1,\ldots,L} \quad & \varphi(\Lambda) \\ \text{då} \quad & \Lambda^{(l)} \in \mathbb{R}_+^N, \quad \text{för } l \in \Gamma_{\le} \\ & \Lambda^{(l)} \in \mathbb{R}^N, \quad \text{för } l \in \Gamma_{=} \\ & \Lambda^{(l)} \in \mathbb{R}_-^N, \quad \text{för } l \in \Gamma_{\ge}. \end{aligned}$$

Notera att vi nu på grund av olikhetsvillkoren måste tänka på de tillåtna mängderna för Lagrangemultiplikatorerna.

Nu deriverar vi Lagrangefunktionen med avseende på varje tensorelement. Detta ger

$$\frac{\partial}{\partial M_{i_1,\ldots,i_L}} \mathcal{L}(M, \Lambda) = C_{i_1,\ldots,i_L} + \sum_{l\in\Gamma} \Lambda_{i_l}^{(l)} + \varepsilon \ln M_{i_1,\ldots,i_L}.$$

Om vi på samma sätt som innan sätter derivatorna till noll och löser de ekvationer som uppkommer får vi den stationära punkten

$$M^*_{i_1,\ldots,i_L} = \exp\left(-\frac{C_{i_1,\ldots,i_L} + \sum_{l\in\Gamma} \Lambda_{i_l}^{(l)}}{\varepsilon} \right) \ge 0.$$

Med samma resonemang som i beviset till sats 4.3 kan vi visa att (4.14) är ett konvext optimeringsproblem. Eftersom $\mathcal{L}(M, \Lambda)$ är konvex och differentierbar i M vet vi också enligt sats 2.2

att varje stationär punkt är en minimipunkt. Det följer då att M^* är en minimipunkt till (4.14). Målfunktionen i det duala problemet kan då, på samma sätt som i (4.6), skrivas som

$$\varphi(\Lambda) = \mathcal{L}(M^*, \Lambda) = -\varepsilon \sum_{i_1=1}^{N} \cdots \sum_{i_L=1}^{N} \left[\exp\left(-\frac{C_{i_1,\ldots,i_L} + \sum_{l\in\Gamma} \Lambda_{i_l}^{(l)}}{\varepsilon} \right) + 1 \right] - \sum_{l=1}^{L} \sum_{i=1}^{N} \Lambda_i^{(l)} \mu_{l,i}. \quad (4.15)$$

Analogt med beviset i sats 4.4 kan vi visa att de koordinatvisa optimeringsproblemen

$$\max_{\Lambda^{(l)} \in \mathbb{R}_+^N} \varphi(\Lambda), \quad \text{för } l \in \Gamma_{\leq}, \qquad \max_{\Lambda^{(l)} \in \mathbb{R}^N} \varphi(\Lambda), \quad \text{för } l \in \Gamma_{=}, \quad \text{och} \quad \max_{\Lambda^{(l)} \in \mathbb{R}_-^N} \varphi(\Lambda), \quad \text{för } l \in \Gamma_{\geq}$$

är konvexa. Vi deriverar därför (4.15) med avseende på varje element i Λ. Vi får ekvationerna

$$0 = \frac{\partial \varphi}{\partial \Lambda_{i_l}^{(l)}} = \sum_{i_1=1}^{N} \cdots \sum_{i_{l-1}=1}^{N} \sum_{i_{l+1}=1}^{N} \cdots \sum_{i_L=1}^{N} \exp\left(-\frac{C_{i_1,\ldots,i_L} + \sum_{k\in\Gamma} \Lambda_{i_k}^{(k)}}{\varepsilon} \right) - \mu_{l,i_l}, \quad \text{för } i_l = 1, \ldots, N. \quad (4.16)$$

Inför vi koordinatbytena

$$K_{i_1,\ldots,i_L} := \exp\left(-\frac{C_{i_1,\ldots,i_L}}{\varepsilon} \right) \quad \text{och} \quad u_i^{(l)} := \exp\left(-\frac{\Lambda_i^{(l)}}{\varepsilon} \right) \quad (4.17)$$

och löser för $u_{i_l}^{(l)}$ får vi

$$u_{i_l}^{(l)} = \mu_{l,i_l} \Bigg/ \sum_{i_1=1}^{N} \cdots \sum_{i_{l-1}=1}^{N} \sum_{i_{l+1}=1}^{N} \cdots \sum_{i_L=1}^{N} \left[K_{i_1,\ldots,i_L} \prod_{k\in\Gamma\setminus l} u_{i_k}^{(k)} \right], \quad \text{för } i_l = 1, \ldots, N$$

som våra Sinkhorniterationer (jämför med (4.11) och (4.12) i avsnitt 4.1). Genom att förlänga bråket med $u_{i_l}^{(l)}$ och skriva om som en elementvis ekvation med vektorer får vi

$$u^{(l)} = u^{(l)} \odot \mu_l ./ P_l(K \odot U), \quad (4.18)$$

där $\odot$ betecknar elementvis multiplikation och där vi definierat tensorn

$$U_{i_1,\ldots,i_L} := \prod_{l\in\Gamma} u_{i_l}^{(l)},$$

som ger sambandet $M = K \odot U$. Algoritmen för att koordinatvis maximera dualen är alltså att i tur och ordning för $l \in \Gamma$ utföra iterationsstegen som beskrivs i (4.18). Eftersom vi nu har olikhetsbivillkor måste vi dock se vad som händer om den stationära punkten från (4.16) hamnar utanför den tillåtna mängden.

På grund av den koordinatvisa differentierbarheten hos φ vet vi att lokala maxpunkter endast kan inträffa i stationära punkter eller på randen. Den tillåtna mängden för $\Lambda_i^{(l)}$ för $l \in \Gamma_{\leq}$ är $[0, \infty)$, vilket innebär att om den stationära punkten hamnar utanför denna måste maximipunkten vara $\Lambda_i^{(l)} = 0$, vilket motsvarar $u_i^{(l)} = 1$. Eftersom bilden av $[0, \infty)$ under koordinatbytet i (4.17) är $[0, 1]$ justerar vi Sinkhorniterationerna till att vara

$$u^{(l)} = \min(u^{(l)} \odot \mu_l ./ P_l(K \odot U), \mathbf{1}), \quad \text{för } l \in \Gamma_{\leq},$$

och eftersom bilden av $(-\infty, 0]$ är $[1, \infty)$, får vi analogt

$$u^{(l)} = \max(u^{(l)} \odot \mu_l ./ P_l(K \odot U), \mathbf{1}), \quad \text{för } l \in \Gamma_{\geq},$$

där $\min(a, b)$ och $\max(a, b)$ i uttrycket ovan är elementvisa minimum och maximum mellan vektorerna a och b.

Observera att projektionerna av $P_l(K \odot U)$ kräver N^{L-1} additioner om de beräknas naivt enligt (2.10). Detta är inte hanterbart. För strukturen hos interpolationsproblemet har vi dock följande resultat som underlättar beräkningarna.

Sats 4.5. [21, sats S1, s. 65] Med beteckningarna ovan gäller det att

$$P_l(K \odot U) = u_l \odot \hat{\varphi}_l \odot \varphi_l,$$

där

$$\hat{\varphi}_l = Q^T \operatorname{diag}(u_{l-1}) Q^T \dots Q^T \operatorname{diag}(u_2) Q^T u_1$$

och

$$\varphi_l = Q \operatorname{diag}(u_{l+1}) Q \dots Q \operatorname{diag}(u_{L-1}) Q u_L,$$

där $\operatorname{diag}(\cdot)$ betecknar diagonalmatrisen av en vektor.

5 Tillämpningar

Den teori som presenterats i de föregående avsnitten har givit oss en effektiv metod för att hitta approximativa lösningar till multimarginella optimaltransportproblem. I detta avsnitt tillämpar vi teorin på konkreta exempel av interpolationsproblemet för att visa metodens effektivitet och mångsidighet. Eftersom dynamiska system är centrala i modelleringssammanhang visar vi i avsnitt 5.1 hur dynamik kan introduceras genom att välja en viss kostnadsfunktion. Därefter körs algoritmen på ett storskaligt exempel. I avsnitt 5.2 visar vi sedan hur det är möjligt att modellera agenter som tar sig ut ur en labyrint.

5.1 Tillämpningar med dynamik

Hittills har vi låtit kostnadsfunktionen, som bestämmer hur dyrt det är för agenterna att röra sig mellan olika tillstånd, vara abstrakt. I detta avsnitt skall vi visa hur man genom ett specifikt val av kostnadsfunktion kan introducera dynamik för agenterna samt hur detta kan användas för att med hjälp av interpolationsproblemet formulera vad som menas med optimal styrning av en svärm agenter. Under detta avsnitt hänvisar vi till [21, s. 53-54].

I avsnitt 2.2 använder vi den euklidiska normen för att formulera ett matchningsproblem mellan två mängder av agenter. Vi observerar att

$$\begin{aligned}
\|x_i^{(1)} - x_j^{(2)}\|_2^2 = \underset{\gamma \in C([0,1])}{\text{minimera}} \quad & \int_0^1 \|\dot{\gamma}(t)\|_2^2 \, dt = \underset{u \in L_2([0.1])}{\text{minimera}} \quad \int_0^1 \|u(t)\|_2^2 dt \\
\text{då} \quad & \gamma(0) = x_i^{(1)} \qquad \text{då} \quad \dot{x}(t) = u(t) \\
& \gamma(1) = x_j^{(2)}, \qquad\qquad x(0) = x_i^{(1)}, \quad x(1) = x_j^{(2)}
\end{aligned}$$

där problemet i högerledet är ett så kallat optimalstyrningsproblem med simpel dynamik. Problemet i mellanledet kan ses som att vi vill hitta den kurva från $x_i^{(1)}$ till $x_j^{(2)}$ som minimerar kurvlängden. Betrakta nu det dynamiska systemet med linjär tidsinvariant dynamik

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{5.1}$$

där $A \in \mathbb{R}^{d_x \times d_x}$, $B \in \mathbb{R}^{d_x \times d_u}$ och där paret (A, B) är styrbart. Genom att helt enkelt definiera kostnadsfunktionen

$$\begin{aligned}
q^{A,B}(x_1, x_2) := \underset{u \in L_2([0.1])}{\text{minimera}} \quad & \int_0^1 \|u(t)\|_2^2 dt \\
\text{då} \quad & \dot{x}(t) = Ax(t) + Bu(t) \\
& x(0) = x_1 \\
& x(1) = x_2,
\end{aligned} \tag{5.2}$$

och använda denna i ett optimaltransportproblem på formen (3.1) har vi introducerat dynamik i modellen. Istället för att vi nu minimerar summan av de kvadrerade euklidiska avstånden som agenterna rör sig minimerar vi nu integralen av den kvadrerade normen av styrsignalen. Amplituden hos styrsignalen är i många fall förknippad med nog form av energiförbrukning. Ta exempelvis en bil: Om bilen gasar kraftigt (hög styrsignal) förbrukar den mer bränsle (energi). Alltså kan man

nu se det som att vi minimerar energin som krävs för att styra agenterna från $x_i^{(1)}$ till $x_j^{(2)}$ när de lyder under dynamiken (5.1).

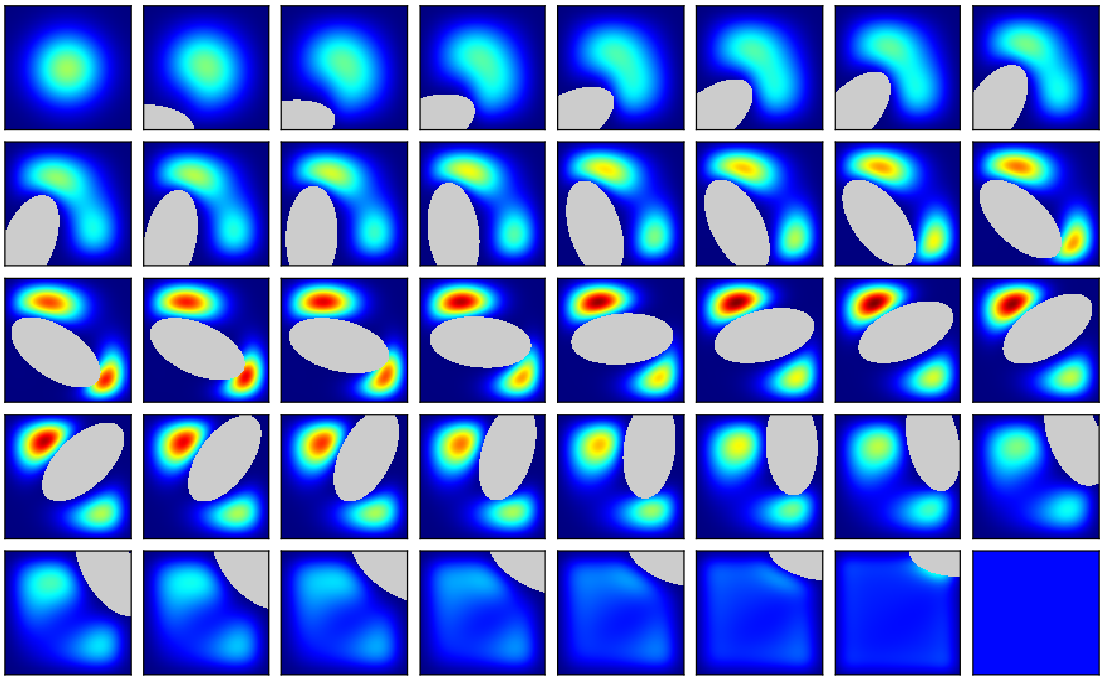
Det är även möjligt att skriva (5.2) på den slutna formen

$$q^{A,B}(x_1, x_2) = \big(x_2 - \operatorname{expm}(A)x_1\big)^T \Sigma_{A,B}^{-1} \big(x_2 - \operatorname{expm}(A)x_1\big), \tag{5.3}$$

där expm$(\cdot)$ är matrisexponentialfunktionen. Därtill är $\Sigma_{A,B} \in \mathbb{R}^{d_x \times d_x}$ den så kallade styrbarhetsgramianen som definieras enligt

$$\Sigma_{A,B} := \int_0^1 \operatorname{expm}(A(1-\tau))BB^T \operatorname{expm}(A^T(1-\tau))d\tau$$

för ett styrbart par (A, B) [23, s. 145]. Härledningen från (5.2) till (5.3) kommer inte demonstreras i denna rapport.



Figur 4: Lösning för ett interpolationsproblem för agenter med dynamik. Varmare färger representerar tätare fördelning av agenter. Agenterna börjar i en normalfördelning kring medelvärdet $(1/2, 1/2)$, längst upp till vänster, och skall i slutet vara jämnt utspridda över hela enhetskvadraten, längst ner till höger. Det grå området representerar det område där agenterna inte får befinna sig.

I figur 4 ser vi den approximativa lösningen till ett interpolationsproblemet över enhetskvadraten $[0, 1] \times [0, 1]$. Här är startmarginalen μ_1 och slutmarginalen μ_L givna och visas högst upp till vänster respektive längst ner till höger. Startmarginalen är modellerad efter en normalfördelning kring medelvärdet $(1/2, 1/2)$ och slutfördelningen är en likformig fördelning över alla tillstånd i rutnätet. I detta exempel lyder agenterna under dynamiken

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{och} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

det vill säga

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}.$$

Vi kan alltså här direkt kontrollera agenternas hastigheter. Denna enkla dynamik gör att kostnadsfunktionen förenklas till $q^{A,B}(x_1, x_2) = \|x_1 - x_2\|_2^2$ (se appendix C för uträkningar). Tillståndsrummet har diskretiserats med hjälp av ett rutnät bestående av 100×100 punkter och

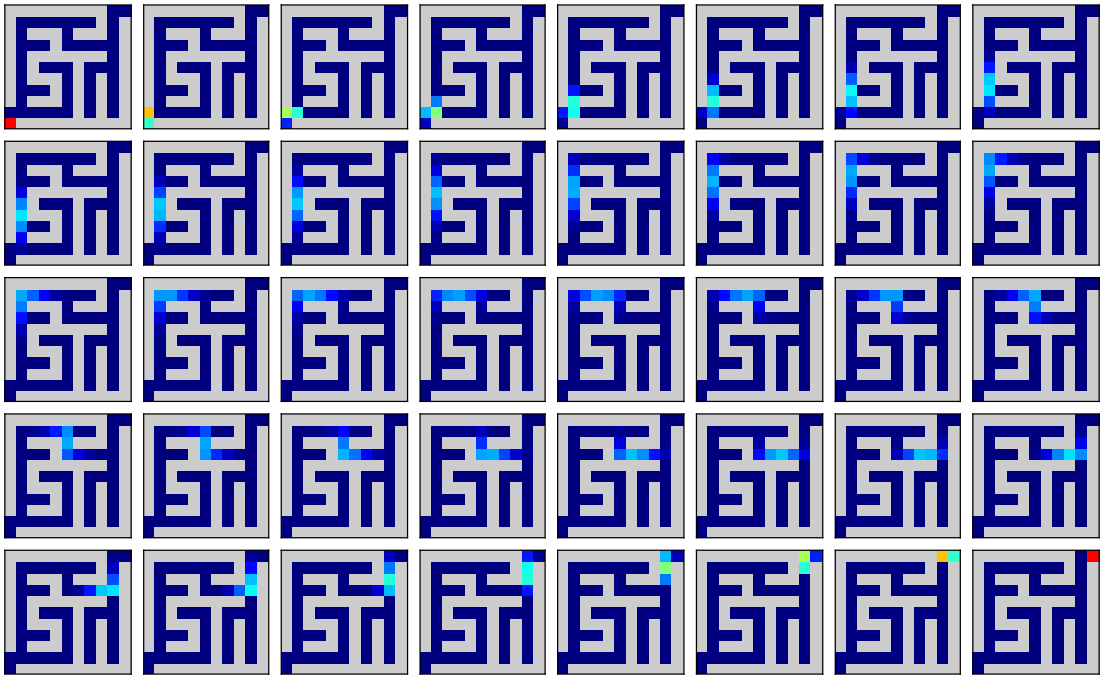
antalet marginaler är $L = 40$. Den använda regulariseringsparametern är $\varepsilon = 10^{-2}$. Hindret (det grå området) som dyker upp vid $l = 2$ och som försvinner vid $l = 39$ är modellerat med hjälp av olikhetsbivillkor på formen (4.13b) där $\mu_{l,k} = 0$ för de tillstånd där agenterna inte får vara och $\mu_{l,k} = a$ för de tillstånd där de får vara. Här är a ett tal större än den totala massan.

5.2 Utrymning ur labyrint

Som ett andra exempel visar vi hur ett problem där agenterna skall utrymma en labyrint kan formuleras. Vi låter här kostnadsmatrisen $Q = [Q_{ij}]_{i,j=1}^{N}$ i interpolationsproblemet vara definierad som

$$Q_{ij} = \begin{cases} |i-j|, & \text{då} \quad |i-j| \leq 1 \\ \infty, & \text{annars,} \end{cases} \tag{5.4}$$

där $|i-j|$ betyder avståndet mellan tillstånd i och j. Denna kostnadsfunktion gör att vi bara tillåter agenterna att röra sig till de tillstånd som ligger närmast det nuvarande tillståndet. Eftersom vi i slutet av avsnitt 4.1 antog att $K > 0$, kan vi egentligen inte använda ∞ som ett värde här. Detta kan lösas genom att istället använda ett tillräckligt stort tal b. Eftersom kostnaden då agenterna gör lagliga steg (första fallet i (5.4)) som störst är a, där a är den totala massan, och antalet steg maximalt är $L-1$ ser vi att $b > a(L-1)$ räcker (vilket även kan ses från (3.8)).



Figur 5: Lösning för ett interpolationsproblem för agenter vars rörelse styrs av kostnadsfunktionen (5.4). Varmare färger representerar tätare fördelning av agenter. Agenterna börjar i nedre vänstra hörnet av labryinten och skall i slutet befinna sig i det övre högra hörnet. De grå områdena (det vill säga labyrintens väggar) representerar tillstånd där agenterna inte får befinna sig.

I figur 5 ser vi lösningen som algoritmen producerar. Längst upp till vänster syns startfördelningen μ_1 och längs ner till höger syns slutfördelningen μ_L. Agentsvärmen börjar alltså längst ner i vänstra hörnet av labyrinten och skall röra sig till det övre högra hörnet. Labyrintens väggar (de grå områdena) har modellerats med olikhetsbivillkor på formen (4.13b). På samma sätt som i föregående exempel väljs $\mu_{l,k} = 0$ för de tillstånd som ska vara otillgängliga och $\mu_{l,k} = a$ för de övriga tillstånden. Labyrinten är 11×11 rutor stor, och antalet interpolationssteg är $L = 40$. Den använda regulariseringsparametern är $\varepsilon = 1/4$. Vi ser att agenterna klarar av att lösa labyrinten.

6 Diskussion och slutsats

Vi har nu kommit fram till och implementerat en lösare för en klass av optimaltransportproblem, vilken kan användas för att lösa många olika typer av problem. Vårt främsta fokus har legat på det multimarginella fallet. Under rapportens gång har vi härlett en lösare till dessa med hjälp av interpolation, entropisk regularisering och Sinkhorniterationer, och vi har sett att dessa hjälpmedel kunnat bidra effektivt till att lösa optimaltransportproblem. Optimaltransport i sig är dock både bredare och djupare än det vi presenterat, där andra typer av problem såväl som andra metoder går att använda sig av, men där principen av att approximera ett optimalt värde med avseende på en viss målfunktion kvarstår.

Optimaltransport, som nämnt i både den populärvetenskapliga texten som inledningen, är ett aktuellt och populärt verktyg som går att tillämpa inom många ämnesområden. Ämnets användbarhet leder till att det forskas mycket inom detta ämne och fältet utvecklas ständigt. Mycket av utvecklingen handlar om att effektivisera numeriska metoder, så att man med samma beräkningskraft kan hitta bättre lösningar snabbare. Realistiska problem inkluderar ofta många aspekter, såsom tid, hastigheter, riktningar och hinder. Detta leder ofta till att problemet snabbt växer, som vi också sett i denna rapport. I detta projekt har lösarens förmåga att hantera flertalet dimensioner varit en avgränsning, men att arbeta fram en metod som kan utföra beräkningar av detta problem i flera dimensioner är något dagens matematiker gemensamt forskar inom och är ett pågående forskningsfält med stor potential.

Referenser

[1] Monge G. Mémoire sur la théorie des déblais et des remblais. Histoire de l'Académie Royale des Sciences de Paris, avec les Mémoires de Mathématique et de Physique pour la même année. 1781:666-704.

[2] Villani C. Optimal transport: Old and New. 1st ed. Berlin: Springer; 2009.

[3] Tolstoi AN. Methods of finding the minimal total kilometrage in cargo transportation planning in space. TransPress of the National Commissariat of Transportation. 1930;1:23-55.

[4] Schrijver A. Combinatorial optimization: Polyhedra and Efficiency. Berlin: Springer; 2003.

[5] Peyré G, Cuturi M. Computational Optimal Transport. arXiv e-prints. 2018 Mar:arXiv:1803.00567.

[6] Lindbeck A. Economic sciences, 1969-1980: the Sveriges Riksbank (Bank of Sweden) prize in economic sciences in memory of Alfred Nobel. Singapore: World scientific; 1992.

[7] Vershik AM, Kutateladze SS, Novikov SP. Leonid Vital'evich Kantorovich (on the 100th anniversary of his birth). Russian Mathematical Surveys. 2012;67(3):589-97.

[8] Dantzig GB. Programming of Interdependent Activities: II Mathematical Model. Econometrica, Journal of the Econometric Society. 1949:200-11.

[9] Wang W, Ozolek JA, Slepčev D, Lee AB, Chen C, Rohde GK. An optimal transportation approach for nuclear structure-based pathology. IEEE transactions on medical imaging. 2010;30(3):621-31.

[10] Alghamdi H, Grogan M, Dahyot R. Patch-based colour transfer with optimal transport. In: 2019 27th European Signal Processing Conference (EUSIPCO). IEEE; 2019. p. 1-5.

[11] Flamary R, Courty N, Gramfort A, Alaya MZ, Boisbunon A, Chambon S, et al. Pot: Python optimal transport. Journal of Machine Learning Research. 2021;22(78):1-8.

[12] Galichon A. Optimal Transport Methods in Economics. Princeton, New Jersey: Princeton University Press; 2016.

[13] Gordon T, Lidberg M. Automated driving and autonomous functions on road vehicles. Vehicle System Dynamics. 2015;53(7):958-94.

[14] Lundgren J, Rönnqvist M, Värbrand P. Optimization. Lund: Studentlitteratur; 2010.

[15] Cuturi M. Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances. arXiv e-prints. 2013 Jun:arXiv:1306.0895.

[16] Boyd S, Vandenberghe L. Convex optimization. Cambridge: Cambridge University Press; 2004.

[17] Andréasson N, Evgrafov A, Patriksson M. An Introduction to Continuous Optimization. 3rd ed. Lund: Studentlitteratur; 2005.

[18] Bertsimas D, Tsitsiklis JN. Introduction to linear optimization. Belmont, Massachusetts: Athena Scientific; 1997.

[19] Gagniuc PA. Markov Chains: From Theory to Implementation and Experimentation. Hoboken, New Jersey: Wiley; 2017.

[20] Birkhoff G. Tres observaciones sobre el algebra lineal. Univ Nac Tucuman, Ser A. 1946;5:147-54.

[21] Haasler I, Karlsson J, Ringh A. Control and Estimation of Ensembles via Structured Optimal Transport. IEEE Control Systems Magazine. 2021;41(4):50-69.

[22] Cover TM, Thomas JA. Elements of information theory. 2nd ed. Hoboken, New Jersey: Wiley; 2006.

[23] Chen CT. Linear System Theory and Design. 3rd ed. New York: Oxford University Press; 1999.

Appendix

A Satser och bevis angående konvexitet

Bevis av sats 2.3 . Låt $f : \mathbb{R}^n \to \mathbb{R}$ vara en linjär funktion. Då har vi

$$f(ax + by) = af(x) + bf(y),$$

för alla $x, y \in \mathbb{R}^n$ och $a, b \in \mathbb{R}$. Speciellt har vi då

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y),$$

för alla $x, y \in \mathbb{R}^n$ och $t \in [0, 1]$ vilket är definitionen av konvexitet. □

Bevis av sats 2.4. Låt $w_i, \ldots, w_m \in \mathbb{R}_+$ och låt $f_i : \mathbb{R}^n \to \mathbb{R}$ för $i = 1, \ldots, m$ vara konvexa funktioner. Definiera även $g : \mathbb{R}^n \to \mathbb{R}$ som

$$g(x) := \sum_{i=1}^{m} w_i f_i(x).$$

Vi har då

$$\begin{aligned} g(tx + (1-t)y) &= \sum_{i=1}^{m} w_i f_i(tx + (1-t)y) \leq \sum_{i=1}^{m} w_i (tf_i(x) + (1-t)f_i(y)) \\ &= t\sum_{i=1}^{m} w_i f_i(x) + (1-t)\sum_{i=1}^{m} w_i f_i(y) = tg(x) + (1-t)g(y), \end{aligned}$$

där vi i olikheten använt definitionen av konvexitet. □

I rapporten använder vi oss av Lagrangerelaxering samt dualitet. I den nedanstående texten fram till och med sats A.1 hänvisar vi till [17, s. 160]. Vi börjar med att låta

$$\varphi(\lambda) := \operatorname*{infimum}_{x \in X} \mathcal{L}(x, \lambda)$$

vara den duala Lagrangefunktionen, definierad som det största minstavärdet av Lagrangefunktionen över X. Vi kan nu skriva det duala problemet som

$$\begin{aligned} \operatorname*{maximera}_{\lambda} \quad & \varphi(\lambda) \\ \text{då} \quad & \lambda \geq \mathbf{0}. \end{aligned}$$

Med dessa förberedelser kan vi formulera sats A.1 som säger att det duala problemet alltid är konvext.

Sats A.1 (Dualproblemet är konvext, [17, s. 160])**.** Låt D_φ vara den faktiska domänen av φ, det vill säga $D_\varphi := \{\lambda \in \mathbb{R}^m : \varphi(\lambda) > -\infty\}$. Då är D_φ konvex och φ är konkav på D_φ.

B Kompletterande resultat för interpolationsproblemet

I detta avsnitt visar vi mer utförligt hur vi går från (3.3) till (3.4) i avsnitt 3. För att underlätta notationen definierar vi mängderna

$$U(\mu_l, \mu_{l+1}) := \left\{ X \in \mathbb{R}_+^{N\times N} \,:\, X\mathbf{1} = \mu_l \text{ och } X^T\mathbf{1} = \mu_{l+1} \right\}, \quad \text{för } l = 1, \ldots, L-1.$$

Med denna notation blir det vi vill visa

$$\underset{\mu_l,\, l=2,\ldots,L-1}{\text{minimera}} \left[\sum_{l=1}^{L-1} \underset{W^{(l)}\in U(\mu_l,\mu_{l+1})}{\text{minimera}} \langle Q, W^{(l)} \rangle \right] = \underset{\substack{W^{(l)}\in U(\mu_l,\mu_{l+1}),\ l=1,\ldots,L-1 \\ \mu_l,\, l=2,\ldots,L-1}}{\text{minimera}} \quad \sum_{l=1}^{L-1} \langle Q, W^{(l)} \rangle. \tag{B.1}$$

För att visa (B.1) kommer vi att använda följande två lemman och följdsats.

Lemma B.1.

$$\min_x \min_y f(x,y) = \min_{x,y} f(x,y)$$

Bevis. För att visa påståendet visar vi att vänsterledet är mindre än eller lika med högerledet och tvärtom. För detta ändamål, låt $(x^*, y^*) = \operatorname{argmin}_{x,y} f(x,y)$, vilket betyder att $f(x^*,y^*) \le f(x,y)$ $\forall\, x, y$. Vi börjar med att visa att vänsterledet är mindre än eller lika med högerledet. Notera att

$$\min_x \min_y f(x,y) \le \min_x f(x,y^*) \le f(x^*,y^*) = \min_{x,y} f(x,y),$$

där den första olikheten kommer från att minimivärdet över y per definition är mindre än eller lika med värdet av funktionen i punkten y^*. Den andra olikheten följer på motsvarande sätt. Härnäst visar vi att högerledet är mindre än eller lika med vänsterledet. Det faktum att $f(x^*,y^*) \le f(x,y)$ $\forall\, x, y$ implicerar att

$$f(x^*,y^*) \le \min_y f(x,y), \quad \forall\, x.$$

Men $\min_y f(x,y)$ är en funktion av x, som vi kallar för $g(x)$. Det följer därför att

$$f(x^*,y^*) \le g(x), \quad \forall\, x \quad \Longrightarrow \quad \min_{x,y} f(x^*,y^*) \le \min_x g(x) = \min_x \min_y f(x,y).$$

Detta visar att $\min_x \min_y f(x,y) = \min_{x,y} f(x,y)$. □

Lemma B.2.

$$\min_{x,y}\{g(x) + h(y)\} = \min_x\{g(x)\} + \min_y\{h(y)\}.$$

Bevis. För att visa detta kan vi, till exempel, använda lemma B.1, välja $f(x,y) = g(x) + h(y)$, och sedan notera att $g(x)$ är konstant med avseende på y, samt att $h(y)$ är konstant med avseende på x. □

Korollarium B.3. Det gäller att

$$\min_x \min_{y\in V(x)} f(y) = \min_{\substack{x \\ y\in V(x)}} f(y),$$

där $V(x)$ är en mängd som beror på variabeln x.

Bevis. Genom att införa indikatorfunktionen

$$I_{V(x)}(y) = \begin{cases} 0, & \text{då } y \in V(x), \\ \infty, & \text{annars} \end{cases}$$

kan vi skriva

$$\min_x \min_{y\in V(x)} f(y) = \min_x \min_y \Big[f(y) + I_{V(x)}(y) \Big] = \min_{x,y} \Big[f(y) + I_{V(x)}(y) \Big] = \min_{\substack{x \\ y\in V(x)}} f(y),$$

där vi i andra likheten använt lemma B.1. □

Genom upprepad användning av lemma B.2 kan vi skriva

$$\underset{\mu_l,\, l=2,\ldots,L-1}{\text{minimera}} \left[\sum_{l=1}^{L-1} \underset{W^{(l)} \in U(\mu_l, \mu_{l+1})}{\text{minimera}} \langle Q, W^{(l)} \rangle \right] = \underset{\mu_l,\, l=2,\ldots,L-1}{\text{minimera}} \left[\underset{W^{(l)} \in U(\mu_l, \mu_{l+1}),\, l=1,\ldots,L-1}{\text{minimera}} \sum_{l=1}^{L-1} \langle Q, W^{(l)} \rangle \right].$$

Identifierar vi nu x med $\mu_2, \ldots, \mu_{L-1}$, y med $W^{(1)}, \ldots, W^{(L-1)}$, $V(x)$ med $U(\mu_1, \mu_2) \times \cdots \times U(\mu_{L-1}, \mu_L)$ och $f(y)$ med

$$\sum_{l=1}^{L-1} \langle Q, W^{(l)} \rangle$$

kan vi använda korollarium B.3 för att skriva

$$\underset{\mu_l,\, l=2,\ldots,L-1}{\text{minimera}} \left[\underset{W^{(l)} \in U(\mu_l, \mu_{l+1}),\, l=1,\ldots,L-1}{\text{minimera}} \sum_{l=1}^{L-1} \langle Q, W^{(l)} \rangle \right] = \underset{\substack{W^{(l)} \in U(\mu_l, \mu_{l+1}),\, l=1,\ldots,L-1 \\ \mu_l,\, l=2,\ldots,L-1}}{\text{minimera}} \sum_{l=1}^{L-1} \langle Q, W^{(l)} \rangle,$$

vilket avslutar beviset för (B.1).

C Förenkling av kostnadsfunktionen från avsnitt 5.1

I detta avsnitt visar vi hur kostnadsfunktionen (5.3) kan förenkals till den euklidiska normen om vi väljer

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{och} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Om vi börjar med att beräkna $\Sigma_{A,B}$. Detta ger att

$$\begin{aligned}
\Sigma_{A,B} &= \int_0^1 \operatorname{expm}(A(1-\tau))BB^T \operatorname{expm}(A^T(1-\tau))d\tau \\
&= \int_0^1 \operatorname{expm}\left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}(1-\tau)\right)\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T \operatorname{expm}\left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}^T(1-\tau)\right)d\tau \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.
\end{aligned}$$

Vidare kan vi nu beräkna kostnadsfuntionen $q^{A,B}(x_1, x_2)$, vilket ger

$$\begin{aligned}
q^{A,B}(x_1,x_2) &= \big(x_2 - \operatorname{expm}(A)x_1\big)^T \Sigma_{A,B}^{-1}\big(x_2 - \operatorname{expm}(A)x_1\big) \\
&= \left(x_2 - \operatorname{expm}\left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right)x_1\right)^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\left(x_2 - \operatorname{expm}\left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right)x_1\right) \\
&= \left(x_2 - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}x_1\right)^T \left(x_2 - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}x_1\right) \\
&= (x_2 - x_1)^T(x_2 - x_1) \\
&= \|x_2 - x_1\|_2^2.
\end{aligned}$$

D Kod

Nedan återfinns koden som använts för att generera alla bilder som använts i rapporten. All kod är skriven i programspråket Julia. Koden för algoritmen som beskrivs i avsnitt 4 finns i `displacement-interpolation.jl`.

Programmet `matching-lp.jl` användes för att generera figur 1 med hjälp av linjärprogrammering.

matching-lp.jl

```julia
using JuMP
using GLPK
using Plots

# Denna funktion beräknar och plottar en lösning till matchningsproblemet på enhetskvadraten
# med linjärprogrammering.
function matching_lp(n, filename)
    x0 = rand(Float64, n)
    y0 = rand(Float64, n)
    x1 = rand(Float64, n)
    y1 = rand(Float64, n)
    c(i, j) = (x0[i]-x1[j])^2 + (y0[i]-y1[j])^2

    model = Model(GLPK.Optimizer)
    @variable(model, 0 <= M[1:n, 1:n])
    @objective(model, Min, sum(sum(c(i, j)*M[i, j] for j in 1:n) for i in 1:n))
    @constraint(model, c1, M*ones(n) .== ones(n))
    @constraint(model, c2, M'*ones(n) .== ones(n))
    optimize!(model)
    best_perm_mat = value.(M)
    Plots.plot()
    for i = 1:n
        for j = 1:n
            if best_perm_mat[i, j] > 0
                Plots.plot!([x0[i], x1[j]], [y0[i], y1[j]], color = "black", lw=1.5)
            end
        end
    end
    Plots.scatter!(x0, y0, color = "blue", markersize = 5)
    display(Plots.scatter!(x1, y1, color = "red", size = (400, 400), markersize = 5,
        xlims = (0, 1), ylim = (0, 1), legend = false))
    savefig(filename)
end

matching_lp(11, "plots/matching.pdf")
```

Programmet `bimarginal-transport.jl` användes för att generera figur 2 med hjälp av linjärprogrammering.

bimarginal-transport.jl

```julia
using JuMP
using GLPK
using Random, Distributions
using Plots
using Plots.PlotMeasures

# Denna funktion plottar resultatet till exemplet som genereras nedan.
function transport_map_marginal_plot(marginal_1, marginal_2, transport_map, filename)
    margin = -20;
    bw = 0.8
    m1 = bar(transpose(marginal_1), orientation = :vertical, bar_width=bw, yflip=false,
        bottom_margin=margin*Plots.px, showaxis=false, ticks=false, legend = false,
        color = "red", linecolor= "red")
    m2 = bar(transpose(marginal_2), orientation = :horizontal, bar_width=bw, xflip=true,
        right_margin=margin*Plots.px, showaxis=false, ticks=false, legend = false,
        color = "blue", linecolor= "blue")
    hm = Plots.heatmap(transport_map, color = :greys, showaxis=false, ticks=false,
        legend = false, framestyle = :box)

    l = @layout[_ a; b c{0.7w, 0.7h}]

```

```
    display(Plots.plot(m1, m2, hm, size = (600, 600), layout = l, link = :both))
    savefig(filename)
end

# Denna funktion genererar och löser ett exempel av bimarginell optimaltransport.
function bimarginal_transport(N)
    a = pdf.(Normal(N/2, 3/20*N), 0:1:N-1)
    b = pdf.(Normal(N/4, 1.5/20*N), 0:1:N-1)/3 + pdf.(Normal(N*3/4, N*1.5/20), 0:1:N-1)/3
    b = b .* sum(a) ./ sum(b)

    n = length(a)

    c(i, j) = abs(i - j)^2

    model = Model(GLPK.Optimizer)
    @variable(model, 0 <= M[1:n, 1:n])
    @objective(model, Min, sum(sum(c(i, j)*M[i, j] for j in 1:n) for i in 1:n))
    @constraint(model, c1, M*ones(n) .== b)
    @constraint(model, c2, M'*ones(n) .== a)
    optimize!(model)
    best_perm_mat = value.(M)

    transport_map_marginal_plot(transpose(a), transpose(b), best_perm_mat,
        "plots/bimarginal_transport.pdf")
end

bimarginal_transport(20)
```

I filen `displacement-interpolation.jl` finns själva algoritmen som behandlades i avsnitt 4. Denna fil kan inte köras för sig utan används i filerna `dynamics-example.jl` och `maze-example.jl`.

displacement-interpolation.jl

```
using LinearAlgebra

# Denna funktion löser interpolationsproblemet med hjälp av Sinkhorniterationer.
function compute_interpolation(C, mu, types, epsilon, tol)
    N = size(mu, 2)
    L = size(mu, 1)

    K = exp.(-C / epsilon)
    K_inv = inv(K)
    K_trans = transpose(K)
    u = ones(L, N)

    function gen_phi()
        phi = ones(L, N)
        phi[L-1, :] = K * u[L, :]
        for l in (L-2):(-1):1
            phi[l, :] =  K * Diagonal(u[l+1, :]) * phi[l+1, :]
        end
        return phi
    end

    count = 1
    max_diff = Inf
    # I denna loop används Sinkhorniterationerna för att beräkna lösningen.
    while max_diff > tol
        print("Iterationsrunda $(count): Maxdifferens: $(max_diff), " *
            "Vald tolerans: $(tol)\n")
        u_prev = copy(u)
        phi_hat = 1
        phi = gen_phi()
        for l in 1:L
            proj = u[l, :] .* phi_hat .* phi[l, :]
            tmp = mu[l, :] ./ proj[:]
            replace!(tmp, NaN => 1)
            u_tmp = u[l, :] .* tmp
            if types[l] == '<'  # Vi delar upp i fall beroende på typ av bivillkor.
                u[l, :] = min.(u_tmp, 1)
            elseif types[l] == '='
                u[l, :] = u_tmp
```

```
            elseif types[l] == '>'
                u[l, :] = max.(u_tmp, 1)
            end
            phi_hat = (l != 1 ? K_trans * Diagonal(u[l, :]) : K_trans * u[l, :]) * phi_hat
        end
        count = count + 1
        max_diff = maximum(abs.(u - u_prev))
    end

    phi_hat = 1
    phi = gen_phi()
    projs = zeros(L, N)
    for l in 1:L
        projs[l, :] = u[l, :] .* phi_hat .* phi[l, :]
        phi_hat = (l != 1 ? K_trans * Diagonal(u[l, :]) : K_trans * u[l, :]) * phi_hat
    end

    return projs
end
```

Funktionen i filen `plotting.jl` används för att rita figur 4 och 5, men gör inga beräkningar utan behöver indata för att rita ut något. Denna fil kan inte heller köras för sig utan används i filerna `dynamics-example.jl` och `maze-example.jl`.

plotting.jl

```
using CairoMakie

# Denna funktion plottar agenternas fördelning och eventuella hinder.
function plot_results(data, obstacle, filename, is_maze)
    L = size(data, 1)
    n_cols = 8
    n_rows = ceil(Int32, L / n_cols)
    replace!(x -> isapprox(x, 0) ? -1 : 0, obstacle)
    replace!(x -> (x > 0) ? 0 : x, obstacle)
    size_cm = 4 .* (n_cols, n_rows)
    size_pt = 28.3465 .* size_cm
    fig = Figure(resolution = size_pt, fontsize = 12)
    count = 1
    done = false
    for row in 1:n_rows
        for col in 1:n_cols
            if count > L
                done = true
                break
            end
            dist = reshape(data[count, :], isqrt(size(data[count, :], 1)), :)
            if !is_maze
                dist = dist .+ (count == 1 || count == L ? 0 : obstacle[count-1, :, :])
                ax, _ = CairoMakie.heatmap(fig[row, col][1, 1], dist, colormap=:jet,
                    colorrange=(0, maximum(data)), lowclip=:grey80, highclip=:red)
            else
                dist = dist .+ obstacle[count, :, :]
                ax, _ = CairoMakie.heatmap(fig[row, col][1, 1], dist, colormap=:jet,
                    colorrange=(0, 0.9), lowclip=:grey80, highclip=:red)
            end
            rowgap!(fig.layout, 10)
            colgap!(fig.layout, 10)
            hidedecorations!(ax)
            count = count + 1
        end
        if done
            break
        end
    end
    display(fig)
    save(filename, fig, pt_per_unit = 1)
end
```

Filen `dynamics-example.jl` bygger upp probleminstansen från avsnitt 5.1 och använder sedan `displacement-interpolation.jl` för att approximativt lösa problemet samt `plotting.jl` för att generera figur 4.

dynamics-example.jl

```julia
using QuadGK

include("displacement-interpolation.jl")
include("plotting.jl")

# Denna funktion beräknar kostnadsmatrisen som motsvarar dynamiken definierad av A och B.
function cost_matrix(grid_points, A, B)
    N = size(grid_points, 1)
    # Numerisk integrering.
    sigma, _ = quadgk(s -> exp(A*(1-s)) * B*transpose(B) * exp(transpose(A)*(1-s)), 0, 1)
    sigma_inv = inv(sigma)
    exp_A = exp(A)
    function cost(x_0, x_1)
        tmp = x_1-exp_A*x_0
        return transpose(tmp)*sigma_inv*tmp
    end
    # Här beräknas kostnaderna mellan alla par av möjliga tillstånd.
    C = [cost(grid_points[i, :], grid_points[j, :]) for i in 1:N, j in 1:N]
    return C
end

# Denna funktion beräknar rutnätet utifrån de givna punkterna.
function gen_grid_points(points)
    N = length(points)-1
    grid = points[1:N] .+ 0.5/N  # Vi samplar i mitten av varje ruta.
    grid_points = hcat(repeat(grid, inner = N), repeat(grid, outer = N))
    return grid_points
end

# Denna funktion genererar hindret för agenterna.
function gen_obstacle(N, n_steps)

    # Parametriserad ellips.
    r = 0.2
    x_curve(t) = 2*r*cos.(t)
    y_curve(t) = r*sin.(t)
    x_start, x_end = 0, 1
    y_start, y_end = 0, 1

    h = 1 / N
    n_a_pts = 10 * N
    n_r_pts = 10 * N

    inside(x) = all((1 .<= x .<= N))
    rotate(p, theta) = [p[1]*cos(theta)-p[2]*sin(theta), p[1]*sin(theta)+p[2]*cos(theta)]
    snap(x) = round.(Int32, x / h) .+ 1

    function transform(pts, t_vec, r_angle)
        t_pts = zeros(size(pts))
        for i in 1:size(pts, 1)
            t_pts[i, :] = rotate(pts[i, :], r_angle) + t_vec[:]
        end
        return t_pts
    end

    function gen_obs(pts)
        obs = zeros(N, N)
        pts = snap(pts)
        for i in 1:size(pts, 1)
            if inside(pts[i, :])
                obs[pts[i, :]...] = 1
            end
        end
        return obs
    end

```

```
    a_pts = range(0, 2*pi, n_a_pts)
    r_pts = range(0, 1, n_r_pts)
    obs_pts = zeros(n_r_pts*n_a_pts, 2)
    for i in 1:n_r_pts
        obs_pts[(i-1).*n_a_pts .+ (1:n_a_pts), :] = r_pts[i] .*[x_curve(a_pts) y_curve(a_pts)]
    end

    xs = range(x_start, x_end, n_steps)
    ys = range(y_start, y_end, n_steps)
    rs = range(0, 2*pi, n_steps)
    obstacle = zeros(n_steps, N, N)

    for l in range(1, n_steps)
        t_pts = transform(obs_pts, [xs[l], ys[l]], rs[l])
        obstacle[l, :, :] = gen_obs(t_pts)
    end

    return obstacle
end

# Denna funktion definierar exemplets parametrar, startar lösaren och plottar resultatet.
function dynamics_example(N, L, epsilon, tol)
    # Definition av matriserna som definierar det dynamiska systemet.
    A = [
        0 0
        0 0
    ]
    B = [
        1 0
        0 1
    ]
    # Parametrar för startfördelningen.
    m = [0.5, 0.5]
    s = [0.2, 0.2]
    # Exempelfördelning. Baserad på täthetsfunktionen för en tvådimensionell normalfördelning.
    f(x, m, s) = exp(-((x[1]-m[1])^2/(2*s[1]^2) + (x[2]-m[2])^2/(2*s[2]^2)))
    pts = range(0, 1, N+1)  # Ekvidistanta punkter som definierar diskretiseringen.
    grid_pts = gen_grid_points(pts)
    println("Genererar kostnadsmatris...")
    C = cost_matrix(grid_pts, A, B)
    mu = zeros(L, N*N)
    types = fill('-', L)
    mu[1, :] = reshape([f(row, m, s) for row in eachrow(grid_pts)], 1, :)  # Startfördelningen.
    types[1] = '='
    total_mass = sum(mu[1, :])
    obstacle = gen_obstacle(N, L-2)  # Vi lägger till ett hinder för agenterna.
    obstacle = (obstacle .- 1) * (-total_mass)
    for l in 2:(L-1)  # Hindret modelleras med olikhetsbivillkor.
        mu[l, :] = reshape(obstacle[l-1, :, :], 1, :)
        types[l] = '<'
    end
    mu[L, :] = reshape(total_mass * ones(N, N) / N^2, 1, :)  # Slutfördelningen.
    types[L] = '='
    println("Beräknar interpolation...")
    data = compute_interpolation(C, mu, types, epsilon, tol)
    println("Plottar...")
    plot_results(data, obstacle, "plots/dynamics-example.pdf", false)
    return
end

dynamics_example(100, 40, 0.01, 0.01)
```

Filen `maze-example.jl` bygger upp probleminstansen från avsnitt 5.2 och använder sedan `displacement-interpolation.jl` för att approximativt lösa problemet samt `plotting.jl` för att generera figur 5.

maze-example.jl

```
include("displacement-interpolation.jl")
include("plotting.jl")

function cost(x_1, y_1, x_2, y_2)
```

```
    if x_1 == x_2 && y_1 == y_2
        return 0
    elseif abs(x_1-x_2) + abs(y_1-y_2) == 1
        return 1
    else
        return Inf
    end
end

# Denna funktion definierar exemplets parametrar, startar lösaren och plottar resultatet.
function maze_example(L, epsilon, tol)
    maze = [  # Labyrinten som agenterna ska utrymma.
        1 1 0 0 0 0 0 0 0 0 0
        0 1 1 1 1 1 1 1 1 1 0
        0 1 0 1 0 0 0 1 0 1 0
        0 1 0 1 0 1 0 1 0 1 0
        0 1 0 0 0 1 0 0 0 1 0
        0 1 1 1 1 1 0 1 1 1 0
        0 0 0 0 0 0 0 1 0 1 0
        0 1 1 1 1 1 0 1 0 1 0
        0 0 0 0 0 1 0 1 0 0 0
        0 1 1 1 1 1 1 1 1 1 1
        0 0 0 0 0 0 0 0 0 0 1
    ]
    N = size(maze, 1)
    obstacle = zeros(L, N, N)
    for l in 1:L
        obstacle[l, :, :] = maze
    end
    pts = hcat(repeat(1:N, inner = N), repeat(1:N, outer = N))
    println("Genererar kostnadsmatris...")
    C = [cost(pts[i, 1], pts[i, 2], pts[j, 1], pts[j, 2]) for i in 1:(N*N), j in 1:(N*N)]
    mu = zeros(L, N*N)
    types = fill('<', L)
    mu[1, 1] = 1  # Agenterna börjar i ena hörnet.
    types[1] = '='
    for l = 2:(L-1)
        mu[l, :] = reshape(maze, 1, :)
    end
    mu[L, N*N] = 1  # Agenterna slutar i det motsatta hörnet.
    types[L] = '='
    println("Beräknar interpolation...")
    data = compute_interpolation(C, mu, types, epsilon, tol)
    println("Plottar...")
    plot_results(data, obstacle, "plots/maze-example.pdf", true)
    return
end

maze_example(40, 0.25, 0.01)
```