Kungliga Tekniska Högskolan
KTH CSC NADA

DD143X Examensarbete inom datalogi, grundnivå

Handelsresandeproblemet

Metoder för asymmetriska grafer

Författare:

Per Hagsten
hagsten@kth.se

Marcus Öberg
marcusob@kth.se

Handledare:

Vahid Mosavat

12 april 2013

Sammanfattning

I denna uppsats beskriver vi tre metoder för att lösa eller approximera en lösning till det asymmetriska handelsresandeproblemet. Vi försöker ta reda på huruvida det är en god ide att transformera en asymmetrisk instans till en symmetrisk instans av handelsresande problemet i relation till att lösa problemet med andra metoder. Innehållet uppsatsen är i huvudsak baserat på tidigare forskning. Som slutsats konstaterar vi att det är svårt att verifiera huruvida det finns några fördelar i medelfallet med att transformera en asymmetrisk instans till en symmetrisk instans. Dock uppmanar vi forskningsvärlden att fortsätta forska i denna riktning inom datalogin.

Abstract

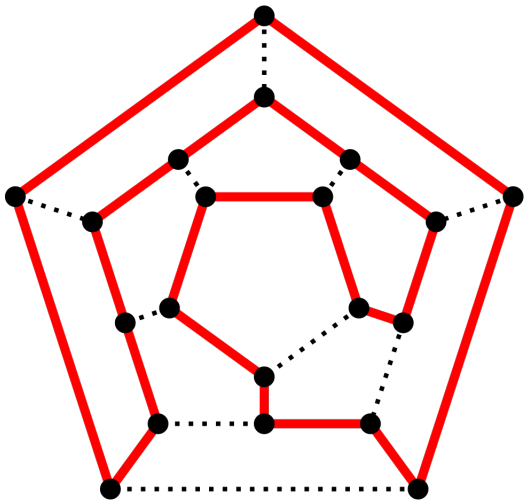
In this paper we describe three different methods for solving or approximating a solution to the asymmetric travelling salesman problem. We try to give an answer to whether it is a good idea to transform an asymmetric instance of the travelling salesman problem into a symmetric instance, in the relation of solving the problem with other methods. The content of our paper is mostly based on previous research. Our conclusion is that it is difficult at this time to give a straight answer to if there exists any benefits to transform from asymmetric to symmetric instance of the traveling salesman problem. Although we encourage the science community to continue research into this field of computer science.

Innehåll

1 Introduktion

Leonhard Euler publicerade 1736 sin bok Seven Bridges of Königsberg. Euler bevisade i sin bok att det inte gick att promenera över Königsbergs sju broar och återvända hem utan att gå över någon bro två gånger, en så kallad Eulercykel. Denna bok anses som det första vetenskapliga arbetet som behandlar det vi idag kallar för grafteori. Sedan dess har grafer blivit en stor del av matematiken och används inom många olika fält.

William Rowan Hamilton intresserade sig drygt hundra år senare för liknande problem som det Euler studerade, men istället för att begränsa resvägen med att varje bro (eller stig) bara får användas en gång, begränsade sig Hamilton till att varje ö (eller hörn) bara får besökas en gång. Andra vetenskapsmän hade studerat liknande problem tidigare men då Hamilton 1857 uppfann det Ikonsianska spelet [1], vilket blev producerat och sålt i stora delar av Europa, kom problemet att döpas efter honom (Hamilton cykel).I detta spel var målet att hitta en väg mellan alla hörn i en dodekaeder och återvända till det första hörnet utan att besöka något hörn mer än en gång, se figur 1.



Figur 1: Det Ikonsianska spelet av William R. Hamilton.

På 1930-talet formulerade och studerade Karl Menger det som idag är ett av de mest kända och mest välstuderade problemen inom datalogi: handelsresandeproblemet. På samma sätt som i Hamiltons spel söks det en väg som besöker alla hörn endast en gång i en godtycklig graf, men i detta fall vill man finna vilken väg som är kortast med avseende på sträckan mellan varje hörn. Handelsresandeproblemet definieras tydligare i kapitel 2.

[1] icosian game. Författarens översättning.

Vi ska i vår rapport studera specialfallet av handelsresandeproblemet där en väg mellan två städer är riktad samt inte nödvändigtvis är lika lång åt båda hållen . Detta kallas för det asymmetriska handelsresandeproblemet. Även om handelsresandeproblemet är mycket välstuderat är det asymmetriska fallet inte alls lika väl utrett. Vi kommer presentera tre olika sätt att lösa problemet och diskutera för- och nackdelar med de olika metoderna.

Det är intressant att studera det asymmetriska handelsresandeproblemet då det inte bara är ett teoretiskt problem utan då det kan även appliceras på många verkliga problem. Som omnämns i 3.

1.1 Problemformulering

Vi studerar klassiska lösningsmetoder för det asymmetriska handelsresandeproblemet för att sedan jämföra dem med tekniker för att översätta asymmetriska grafer till symmetriska grafer. För att sedan komma till en slutsats om vilken av dessa angreppssätt som kan anses effektivast med hänsyn till tidskomplexiteten för lösningsmetoderna.

1.2 Material

Vi har valt att endast studera forskningsrapporter, examensarbeten samt högkvalitativ facklitteratur inom ämnet, eftersom källor på lägre nivå var i det stora hela otillräckliga inom de resonemang, bevis och metoder vi intresserade oss för. Därför valdes källor som hade en hög kunskapströskel som material för våra slutsatser och resonemang.

1.3 Metod

Vi hade tänkt besvara frågeställningen genom att främst fokusera på att befintligt material och dra slutsatser utifrån dem. Vid mån av tid kommer vi även att implementera egna lösningar och tester på befintliga metoder och idéer, men detta här lägre prioritet eftersom problemet i sig är väl studerat sedan tidigare.

2 Handelsresandeproblemet - TSP

Handelsresandeproblemet eller Traveling salesman problem, som vidare kommer omnämnas som TSP, är ett välstuderat problem inom matematiken och datalogin som väckt många frågor och tankar bland forskare under årens lopp. Det är ett simpelt problem att förklara och förstå, men det har visat sig vara svårt att lösa på ett effektivt sätt. Det är sannolikt därför det idag är ett av de mest studerade och omdebatterade problemen inom datalogin.

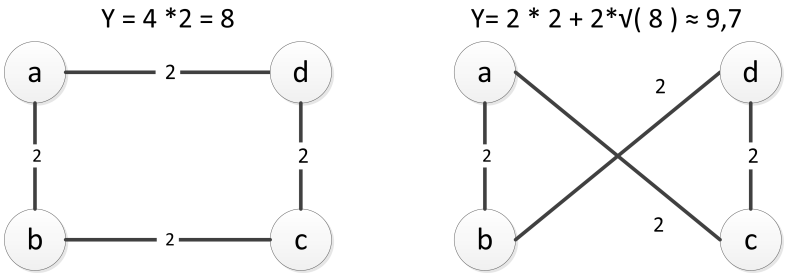
Problemet kan informellt definieras på följande sätt: Givet en positiv ändlig mängd städer och vägar, där en väg endast förbinder två städer och har en definierad restid, vill man finna en stig som besöker alla städer en, och endast en, gång för att sedan återvända till begynnelsestaden med kravet att den funna stigen ska vara så kort som möjligt. Man vill alltså besöka alla städer på minimal restid. Med en stig åsyftas den föreslagna permutationen för att besöka städerna.

Problemet fick sitt namn under 1930-talet på Princeton University. Tidigare har liknade varianter av problemet existerat där finns det föreslagna rutter genom fyrtiosju stycken städer i Tyskland och Schweiz i en handbok för köpmän från 1832 [1]. Idag existerar instanser på klart större turer, till exempel en som föreslår en optimal väg för att besöka alla städer i hela Sverige [1].

TSP har många olika användningsområden, exempelvis kan man planera en effektiv rutt för att leverera post inom ett utkörningsområde eller beräkna vilken rutt ett borrhuvud ska ta när man vill borra hål i ett kretskort. Andra mindre handfasta användningsområden är till exempel att man kan utnyttja TSP beräkningar för att effektivt observera flera galaxer och stjärnor i rymden med ett teleskop, då det är en tidskrävande process att justera om teleskopet mellan varje flytt om man vill observera flera himlakroppar.

2.1 Symmetrisk TSP

Det mest studerade fallet av TSP, är det symmetriska fallet då restiden mellan två städer är lika lång oavsett vilken utgångspunkt man väljer. Problemet går att representera som en oriktad graf där en stad motsvarar ett hörn i grafen och en väg motsvarar en kant i grafen. Restiden representeras som kanternas vikter i ett naturligt tal.



Figur 2: Beroende på vilken väg som väljs kan sträckan för att besöka alla hörn vara olika lång.

Givet en oriktad graf G där s_i tillhör $[s_1, s_2, \ldots, s_n]$, där n är antalet städer, låter vi alla permutationer π av $d(s_{\pi(i)}, s_{\pi(i+1)}) + d(s_{\pi(n)}, s_{\pi(1)})$ för varje hörn i vara en lösning som tillhör Ω. Vi söker den minsta totala summan som tillhör Ω.[2]. Med andra ord söker vi att minimera:

$$\sum_{i=0}^{n-2} d\left(s_{\pi(i)}, s_{\pi(i+1)}\right) + d\left(s_{\pi(n-1)}, s_{\pi(0)}\right). \tag{1}$$

för alla permutationer av gitiga turer π.

TSP kan lösas naivt med totalsökning genom att skapa alla möjliga permutationer av grafens n kanter för att sedan undersöka huruvida varje enskild permutation uppfyller kravet för en Hamiltoncykel och i så fall spara ner längden för den funna stigen. Till sist returneras den permutation med kortast stig längd. Denna metod medför dock att vi får $n!$ antal permutationer då vi kan besöka grafen i vilken ordning som helst och metoden fungerar endast på grafer av väldigt trivial storlek, eftersom då n går mot allt större naturliga tal så blir tidsåtgången för att lösa probleminstansen orimlig. Redan vid relativt små tal blir antalet permutationer väldigt stort. Till exempel i en graf med tio stycken städer kommer det finnas $10! = 3,\ 628{,}800$ olika permutationer av stigar.[3].

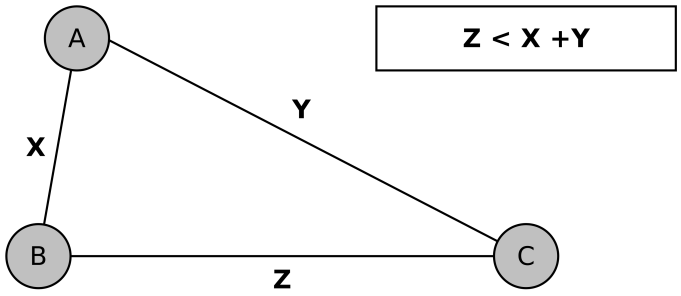
TSP kan formuleras som ett beslutsproblem som efterfrågar om det existerar en stig, kortare än någon längd l, i en oriktad graf. En föreslagen lösning på beslutsproblemet kan verifieras i polynomisk tidskomplexitet vilket innebär att TSP tillhör NP. Det har även visats att TSP är NP-svårt genom reduktion till andra NP-fullständiga problem [4].

Än så länge har det inte hittats ett effektivt sätt att lösa TSP på rimlig tid om ($P \neq NP$). Detta har lett till att man inom datalogin efterfrågar algoritmer för approximera en nära optimal lösning för problemet för att på så sätt uppnå en kortare körtid. En approximering ger en lösning som ligger mer eller mindre nära en optimal tur med avseende på tur-längd.

2.2 Triangelolikheten

Ett viktigt koncept inom TSP, som kommer vara antaget i alla fall av Asymmetrisk TSP i den här rapporten är det matematiska konceptet triangelolikheten. Triangelolikheten antas i många fall av TSP, då den förminskar lösningsrymden avsevärt och är en naturlig del i euklidiska grafer, se 2.5.

Triangelolikheten säger att om man har tre stycken städer, a,b och c, som är förbundna enligt grafen nedan så kan man tillämpa det matematiska konceptet triangelolikheten för att dra slutsatser och antaganden om de olika restiderna mellan städerna. I fallet när vi vill resa från a till b, så kan det inte vara billigare att först resa till c för att sedan resa vidare till b, sträckan direkt mellan två städer är alltid kortare eller lika med sträckan att resa via en annan stad till målet.



Figur 3: TriangelOlikheten. $BC < AC + AB$

Den formella definitionen av triangelolikheten följer nedan, där *d(a , b)* beteckar avståndet mellan a och b[4] :

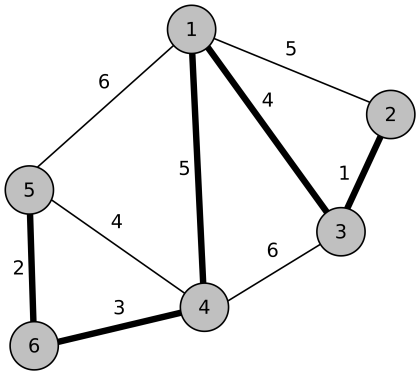
$$d(a\ ,\ b) \leq d(a\ ,\ c) + d(c\ ,\ b) \tag{2}$$

2.3 Linjärprogrammering

Linjärprogrammering innebär att man har någon form av målfunktion till exempel att man vill minimera en summa. Med tillhörande logiska villkor som begränsar målfunktionen, dessa kan antingen vara likheter och olikheter eller så existerar villkor av båda sorter. Sedan så försöker man optimera den givna målfunktionen med hjälp av de logiska villkoren.

2.4 Minimalt spännande träd

Ett minimalt spännande träd är en delgraf som sammansluter alla hörnen i hela grafen med så få och så korta kanter som möjligt.



Figur 4: Minimalt spännande träd, alla hörn är förbundna med så få och korta kanter som möjligt

2.5 Euklidiska TSP

Givet n punkter i en dimension d, för ett specifikt d söks den kortaste turen som besöker alla punkter. Med andra ord saknar en euklidisk graf kantvikter utan kanternas längd är det faktiskt avståndet mellan två hörn.

3 Asymmetrisk TSP

Följande kapitel inleds med en formell definition av ATSP samt så föreslås metoder som vi valt att fokusera på senare i kapitlet.

Asymmetrisk traveling salesman problem (hädanefter förkortat till ATSP) är ett specialfall av det symmetriska TSP problemet. Till skillnad från den symmetriska varianten gäller dock inte att kantvikten från ett hörn A till ett annat hörn B är lika med kantlängden från B till A. Med andra ord kan man inte anta att $d(i, j) = d(j, i)$ för några hörn i och j.

Liksom det symmetriska fallet har ATSP bevisats vara ett NP-svårt problem samt tillhöra NP även då triangelolikheten gäller[8].

ATSP är långt ifrån lika välstuderat som den symmetriska versionen, dock finns det många verkliga fall av turer som är asymmetriska i sin natur. Paketbud använder sig till exempel av denna typ av beräkningar för att ta reda på vilket som är det mest energieffektivaste sättet att leverera paket.

Då TSP och ATSP är NP-svåra finns det inga effektiva metoder för att lösa dem exakt, därför försöker man ofta hitta heuristiker för att approximera en nära optimal lösning istället.

3.1 Definition

Givet en riktad graf G där s_i tillhör $[s_1, s_2, \ldots, s_n]$, där n är antalet städer, låter vi alla permutationer av $d(s_i, s_{i+1}) + d(s_n, s_1)$ för varje hörn och i vara en lösning som tillhör Ω. Vi söker den minsta totala summan som tillhör Ω.

3.2 Lösningsförslag

Det finns flera förslag på metoder för att lösa ATSP. I rapporten kommer vi endast att fördjupa oss i ett fåtal, dessa är:

- **Held-Karp**

  Held-Karp är en heuristik för att approximera en tur väldigt nära den optimala som bygger på minimalt spännande träd. Vi kommer förklara denna i mer detalj i kapitel 4.1.

- **ATSP till TSP transformering**

  Genom att transformera en ATSP instans till en vanlig TSP instans kan man till exempel använda Christofides även för att approximera ATSP. Vi kommer inte ge mer detaljer kring Christofides utan istället förklara hur transformationen från ATSP till TSP går till i kapitel 4.2

- **Branch and bound**

  Branch and bound algoritmen är en av de mest effektiva exakta metoderna för att finna den optimala lösningen för en ATSP instans. Metoden förklaras i mer detalj i kapitel 4.3.

4 Metod

I efterföljande underrubriker kommer vi beskriva tre stycken metoder för att lösa ATSP både med hjälp av approximationsheuristiker, samt exakta lösningsmetoder. Författare har föreslagit många metoder men vi har vi valt att ta upp följande tillvägagångssätt i rapporten, eftersom vi ansåg de välstuderade och hade en rad fördelaktiga egenskaper.

Först beskrivs Held-Karp, det är en klassisk approximationsheuristik från sent 60-tal. Följt av ATSP till TSP transformation som är ett sätt att översätta instanser av ATSP till det symmetriska TSP problemet som man sedan kan lösa med en approximativheuristik eller med en exakt lösningsmetod. Sedan kommer vi titta på lösningsmetoden Branch and Bound som är ett samlingssätt för att lösa olika problem exakt. Samtliga metoder används idag för att finna lösningar för ATSP och därför har vi ansett dem som aktuella för att lösa problemformuleringen.

4.1 Held-Karp

I detta kapitel beskrivs Held-Karp för ATSP, hur metoden är uppbyggd och hur algoritmen fungerar i enklare termer. Vi inleder med en kort historik följt av hur Held-Karp fungerar i det symmetriska fallet av TSP sedan i det asymmetriska fallet, då de är starkt relaterade till varandra. Sist i kapitlet rundar vi av med att reda ut hur pass bra skattningar av den undre gränsen Held-Karp genererar.

Held-Karp är en heuristik som kan approximera en lösning både för det symmetriska och det asymmetriska fallet som publicerades år 1969 i följande rapport[6], av de datavetenskapliga forskarna Richard M. Karp och Michael Held. Rapporten presenterar flertalet heuristiker som idag benämns som Held-Karp. De visade att man kunde framställa en approximering av TSP med hjälp av dynamisk programmering, samt att man även i det asymmetriska fallet kunde producera en gissning på ungefär samma sätt. De noterade även att heuristiken kunde skrivas om som ett program som nyttjar principerna för linjärprogrammering i det asymmetriska fallet [11]. Held och Karps forskning kan ses som en mycket viktig milstolpe inom TSP problemet, som lagt grunden för mera avancerade algoritmer.

Det viktigaste begreppet att komma underfund med när man vi använder Held-Karp är att lösningen vi får fram är en approximering av stig längden för den optimala lösningen av grafen. Held-Karp ger oss en undre gräns för den optimala lösningen men inte en vektor innehållande stigen i sig. I majoriteten av praktiska fall av problemet så är denna undre gräns för sträckan nära den optimala lösningen, vilket gör heuristiken fördelaktig än idag även om snabbare algoritmer har utvecklas sen 1969, algoritmen har en tidskomplexitet på $O(n^2 2^n)$[1].

Låt v vara en slumpvis valt men definierat hörn i grafen G. Ett "1-tree" (vidare kallat OT) definieras som följande: Låt $G*$ vara $G \setminus v$. Generera ett minimalt spännande träd över $G*$ och lägg till v med de två kortaste kanterna. Resultatet är ett "1-tree" vilket är en undre gräns till TSP.

Held Karp bygger på relationen mellan OT och MST för någon kostnadsfunktion c:

$$OT(c) \leq MST(c) \tag{3}$$

Att denna relation gäller kan man förstå genom att notera att en lösning till TSP består av två kanter kopplade till v samt ett minimalt spännande träd för $G \setminus v$. Ett OT är dock generellt en ganska svag undre gräns för TSP. För att få en bättre uppskattning för den undre gränsen låt π vara en funktion som ger hörn-vikter. Vi definierar en kostnadsfunktion c som:

$$c^{\pi}_{uv} = c_{uv} - \pi_u - \pi_v \tag{4}$$

Genom att uppdatera π kan vi beskriva relationen mellan OT och TSP som:

$$OT(c^{\pi}) \leq TSP(c^{\pi}) = TSP(c) - 2\pi(V) \tag{5}$$

Detta leder till att vi kan optimera med avseende på π för att få en så nära approximation som möjligt.

$$max_{\pi}(OT(c^{\pi}) + 2\pi(V)) \leq TSP(c) \tag{6}$$

För asymmetriska fallet så kan man använda Held-Karp som tidigare omnämnt men med en viss skillnad i metoden. Skillnaden är att man i grafen använder sig av "1-arborescences"istället, hädanefter omnämnt som en trädstruktur. En trädstruktur är en form av träd som man bildar i en riktad graf. Det är en riktad graf som utgår från rot-hörnet där alla kanter är riktade ut ifrån rot-hörnet, alltså att alla kanter pekar nedåt i trädet. Detta innebär att det inte kan existerar några cykler samt att det finns en och endast en riktad kant från roten till något annat hörn i grafen[4]. En 1-trädstruktur är ett specialfall av trädstruktur där man har lagt till en extra kant från rot-hörnet v som pekar tillbaka på rot-hörnet v i sig, det vill säga att man skapar en cykel i trädet. Med andra ord så innehåller en 1-trädstruktur alltid en och endast en cykel som innehåller hörnet v, sedan itererar vi över hela grafen på samma sätt som tidigare [11] för att förbättra stig längden. Det väsentliga med det här är att vi kan se att den minimala 1-trädstruktur egentligen är en tur i grafen, eftersom en minimum turen för trädstrukturen även är turen för den faktiska instansen. Därför vet vi att av den trädstrukturen även är den kortaste turen vi kan hitta i grafen. Med andra ord har vi tagit fram en undre gräns för approximationen av turen för den inskickade grafen.

I dagsläget så används inte Held-Karp lika flitigt för att beräkna symmetriska TSP:er utan andra algoritmer har tagit över fanan sedan dess. Dock så fungerar algoritmen relativt bra i det asymmetriska fallet. Idag så är andra algoritmer populärare då tar fram en lösning snabbare, dock så gäller det fortfarande att inte finns någon algoritm som är bättre att hitta en så nära på optimal lösning i det asymmetriska fallet [4].

Detta grundar sig i teorin om att man kan approximera lösningen där man endast approximerar en skattning av den optimala turens längd. Som tidigare omnämnt så konstruerar Held-Karp heuristiken en undre gräns för den optimala turen. Det har påvisats i tidigare forskning att denna undre gräns ligger väldigt nära den optimala lösningen i generella fall av ATSP [4]. Vi kan benämna den skattade lösningen som $\mathcal{OPT}_{\mathcal{A}}$, och den optimala lösningen som $\mathcal{OPT}_{\mathcal{O}}$. Korrektheten kan därför representeras som ett värde på följande vis.

$$\Delta = (\mathcal{OPT}_{\mathcal{O}}/\mathcal{OPT}_{\mathcal{A}}) \tag{7}$$

Där målet är att Δ i formel 7 ska konvergera mot talet ett, den tidigare forskningen visar att detta medelvärdet ligger på $\Delta = 1.008$ i generella fall. Det mest intressanta är dock att försöka få en lägre undre gräns som enligt de senaste rönen är: $O(logn/log(logn))$. Eftersom det är stor skillnad på övre och undre gränsen så kan vi anta att det finns mer att upptäcka om hur bra Held-Karp faktiskt presterar i det värsta fallet [4], så en tajtare övre och undre gräns kan fastställas.

4.2 ATSP till TSP

Ett mindre utforskat sätt att lösa ATSP grafer på är att använda sig av metoder för att översätta dem till symmetriska grafer som sedan beräknas med klassiska algoritmer, till exempel Christodifes algoritm som har en tidskomplexitet på $O(n^3)$ [7]. Vi kommer betrakta två metoder i följande del, en äldre och en nyare.

Kumar, Li samt Hahsler, Hornik

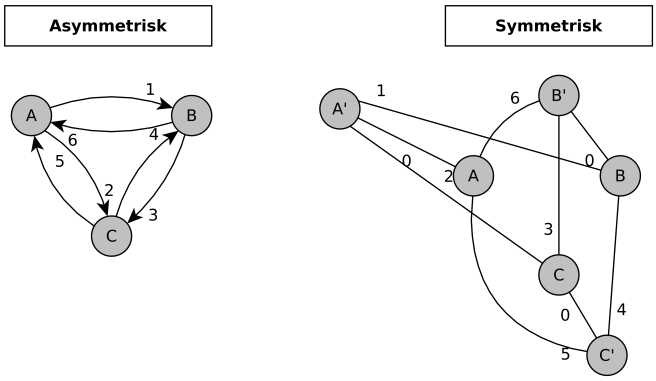
En ATSP graf där vi har antagit att triangelolikheten gäller kan representeras som en övergångsmatris där kanternas vikter är inlagda för varje matriselement, det vill säga sträckan mellan hörnen. Antar man att triangelolikheten gäller så kan man även härleda att probleminstansen består är en Euklidisk graf[9], matrisen går sedan att översätta till en symmetrisk matris. Gäller inte detta påstående så medför det att matrisen är asymmetrisk. Ett tomt eller positivt oändligt matriselement betecknar att det inte existerar en kant mellan de två hörnen. Vi kan transformera denna matris till en symmetrisk genom att dubbla matrisen, alltså går vi från en matris med N storlek till en $2N$ matris. De duplicerade hörnen har då ett fast avstånd mellan sig, till exempel noll för att explicit påvisa att det inte existerar ett avstånd mellan det faktiska hörnet och det duplicerade hörnet, vilket är relevant för att en annan algoritm ska kunna hitta en lösning för turen[5].

$$N = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} \infty & 1 & 2 \\ 6 & \infty & 3 \\ 5 & 4 & \infty \end{pmatrix} \end{array} \tag{8}$$

Figur 5: Den asymmetriska matrisen N

$$2N = \begin{array}{c} \\ A \\ B \\ C \\ A' \\ B' \\ C' \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & A' & B' & C' \end{array} \\ \begin{pmatrix} \infty & \infty & \infty & 0 & 6 & 5 \\ \infty & \infty & \infty & 1 & 0 & 4 \\ \infty & \infty & \infty & 2 & 3 & 0 \\ 0 & 1 & 2 & \infty & \infty & \infty \\ 6 & 0 & 3 & \infty & \infty & \infty \\ 5 & 4 & 0 & \infty & \infty & \infty \end{pmatrix} \end{array} \tag{9}$$

Figur 6: Den översatta symmetriska matrisen 2N

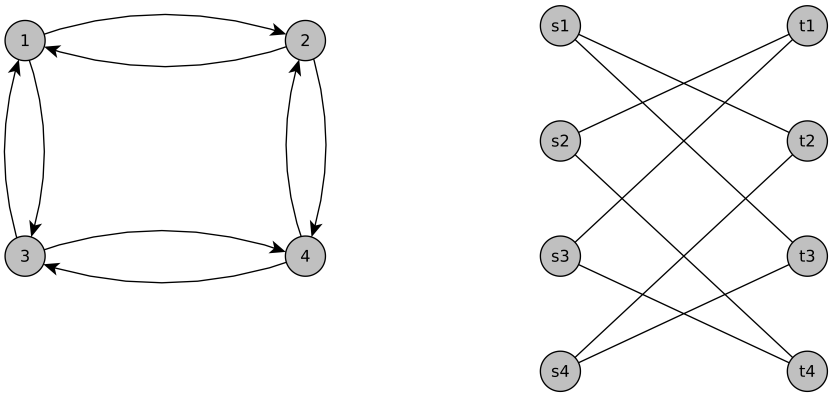


Figur 7: Grafen transformeras från en Asymmetrisk till en symmetrisk graf

Palbom, Engebretsen

Som tidigare omnämnt så kan en ATSP graf representeras som en avståndsmatris för att sedan transformeras om till en symmetrisk matris. Ett annat sätt att transformera ATSP grafer har tagits fram av Anna Palbom och Lars Engebretsen via NADA här på KTH år 2006[8]. Metoden är intressant eftersom den öppnar upp ett helt nytt angreppssätt att lösa TSP-grafer, eftersom den översätter grafen till en bipartit graf.

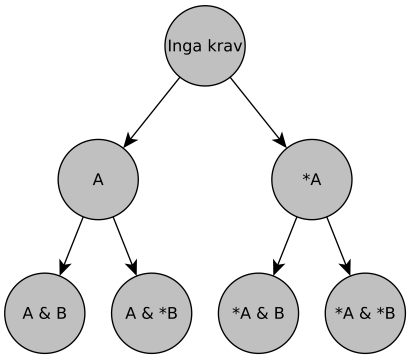
En ATSP graf som följer antagandet om triangelolikheten går att transformera till en bipartit graf, där kanterna i den bipartita grafen däremot inte följer triangelolikheten. Detta spelar dock ingen roll då det även går att se att turernas längder som man finner i båda instanserna skiljer sig med en konstant faktor. Transformationen görs genom att varje hörn i den riktade asymmetriska grafen representeras av två stycken hörn i den bipartita grafen, ett originalhörn *1* som är källan *s1* och ett duplicerat hörn som är sänka *t1*. Hörnen förbinds med den tillhörande riktade kanten i originalgrafen där den riktade kanten representeras som en oriktad kant i den bipartita grafen, eftersom vi representerar det med hjälp av en källa och en sänka så har vi implicit riktat kanten från källan till sänkan. Detta medför att kanternas vikter i avståndsmatrisen kommer kunna föras över till den bipartita grafens kanter, där kanterna får samma vikt som sin motsvarande kant i originalgrafen. En matchning för den bipartita grafen kommer då representera till en hörntäckning i ATSP-grafen. Detta innebär implicit att det nu räcker med att ta fram en algoritm som kan lösa ut en tur ur den bipartita grafen, eftersom den turen kommer att vara en mer eller mindre bra approximation av lösningen för ATSP-grafen som vi utgick ifrån.[8]. I figuren nedan visas en beskriven transformation av en ATSP graf till en bipartit graf.



Figur 8: En riktad graf med in-valens två och ut-valens två transformeras till en oriktad bipartit graf.

4.3 Branch and Bound

Branch and bound algoritmen (vidare kallad BnB) är en generell metod för att hitta lösningen till flera typer av optimeringsproblem. Generellt kan man beskriva algoritmen som följer: Låt $f(x)$ definiera en funktion som beräknar en undre gräns för probleminstansen. Bygg sedan upp ett träd av möjliga lösningsinstanser där varje hörn har två löv som består av komplementära krav på lösningen, nivå för nivå. Varje löv i trädet beräknar en undre gräns för den specifika lösningsinstansen och skapar nya löv som består av den nuvarande hörnets krav samt ett ytterligare komplementärt krav. Då en lösning hittas blir den lösningen en övre gräns och alla löv som har en undre gräns större eller lika med den övre gränsen kan ignoreras och behöver således inte evalueras.



Figur 9: Träd av möjliga lösningsinstanser där varje nivå har ett mer krav än den tidigare nivån. A och B är kanter i en graf.

För TSP och ATSP kan man använda den ungerska algoritmen[2] för att beräkna den under gränsen. Den ungerska algoritmen beskrivs närmare i 4.3. I grunden används den ungerska algoritmen för att lösa tilldelningsproblemet[3] vilket är en förenkling av TSP. TSP kan reduceras till tilldelningsproblemet genom att ta bort kravet att lösningen ska vara en komplett rutt. Eftersom lösningen på tilldelningsproblemet kan bestå av antingen en komplett rutt (en lösning till TSP problemet) eller flera sub-turer så ställer vi krav på vilka kanter som ska finnas med i lösningen och bygger upp det komplementära trädet, se figur 9. För varje nivå i trädet beräknar man den undre gränsen och genererar nya löv om ingen giltig lösning hittades. Det som gör att detta fungerar är det faktum att för varje hörn i träden kan den undre gränsen aldrig vara mindre än förälderns undre gräns. Detta kan bevisas genom ett enkelt tanke experiment:

Låt probleminstansen P vara roten på trädet med en optimal lösning δ. Låt nu P_1 vara en delmängd av P där kant i måste vara en del av lösningen och P_2 vara en delmängd av P där kant i inte finns med. Låt δ_1 vara den undre gränsen för P_1 och δ_2 vara undre gränsen för P_2. Då P_x inte innehåller några kanter som inte finns i P kan den omöjligt innehålla en tur som är kortare än den kortaste turen i P. Detta är sant för varje nivå i trädet.

Komplexitet

Den ungerska algoritmens tidkomplexitet O(n^3)[10]. I värsta fall kommer Branch and Bound algoritmen att tvingas söka igenom hela lösningsrymden innan en undre gräns hittas och således är det värsta fallet för Branch and Bound lika med totalsökning, det vill säga exponentiellt tidskomplexitet. I verkliga fall har man dock sett att BnB presterar väl i jämförelse med andra lösningar [12].

Den ungerska algoritmen

Låt matrisen M representera hörnen $v_1, v_2 \dots v_n$ varje M_{ij} representerar avståndet mellan hörnen v_i och v_j.

$$M = \begin{pmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,n} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n,1} & v_{n,2} & \cdots & v_{n,n} \end{pmatrix} \tag{10}$$

[2]Hungarian algorithm
[3]Assignment problem, problembeskrivning utelämnas

Diagonalen i matrisen är inte definierade då de representerar loopar vilket inte är tillåtet i problemformuleringen för TSP. Algoritmen kan delas upp i fyra steg på följande sätt:

1. För varje rad i matrisen M subtrahera varje element i raden med radens minsta värde. Försök sedan göra en tilldelning genom att först titta på varje rad huruvida det finns en och endast en nolla, markera i sådana fall den nollan. Gör sedan samma sak fast kolumn alternerande antingen till alla rader och alla kolumner har en markerad nolla eller att det inte går att markera några fler nollor då det finns fler än en nolla i någon rad och kolumn. Om tilldelningen lyckades så ger de markerade nollorna svaret på tilldelningsproblemet, annars fortsätt med steg 2.

$$\phi 1 = \begin{pmatrix} Min(M_{11}, M_{12} \dots M_{1n}) \\ Min(M_{21}, M_{22} \dots M_{2n}) \\ \vdots \\ Min(M_{n1}, M_{n2} \dots M_{nn}) \end{pmatrix} \tag{11}$$

$$M1 = \begin{pmatrix} M_{1,1} - \phi 1_1 & M_{1,2} - \phi 1_1 & \cdots & M_{1,n} - \phi 1_1 \\ M_{2,1} - \phi 1_2 & M_{2,2} - \phi 1_2 & \cdots & M_{2,n} - \phi 1_2 \\ \vdots & \vdots & \ddots & \vdots \\ M_{n,1} - \phi 1_n & M_{n,2} - \phi 1_n & \cdots & M_{n,n} - \phi 1_n \end{pmatrix} \tag{12}$$

2. För varje kolumn subtrahera varje element i kolumnen med kolumnens minsta värde. Försök göra en tilldelning på samma sätt som i steg 1. Om det misslyckas gå vidare till steg 3.

   Låt M1' = $M1^T$

$$\phi 2 = \begin{pmatrix} Min(M1'_{11}, M1'_{12} \dots M1'_{1n}) \\ Min(M1'_{21}, M1'_{22} \dots M1'_{2n}) \\ \vdots \\ Min(M1'_{n1}, M1'_{n2} \dots M1'_{nn}) \end{pmatrix} \tag{13}$$

$$M2' = \begin{pmatrix} M1'_{1,1} - \phi 2_1 & M1'_{1,2} - \phi 2_1 & \cdots & M1'_{1,n} - \phi 2_1 \\ M1'_{2,1} - \phi 2_2 & M1'_{2,2} - \phi 2_2 & \cdots & M1'_{2,n} - \phi 2_2 \\ \vdots & \vdots & \ddots & \vdots \\ M1'_{n,1} - \phi 2_n & M1'_{n,2} - \phi 2_n & \cdots & M1'_{n,n} - \phi 2_n \end{pmatrix} \tag{14}$$

   M2 = $M2'^T$

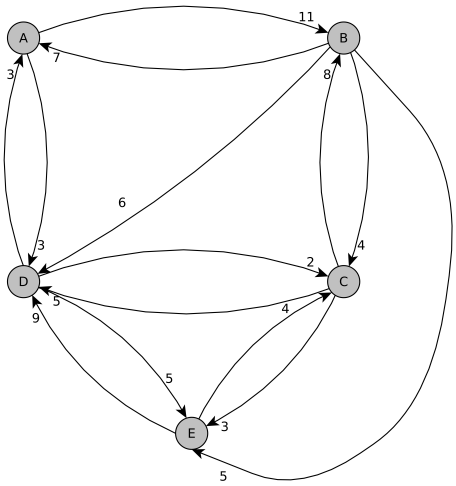
3. Börja med att tilldela så många nollor som möjligt i M2. Markera sedan alla rader som inte har någon tilldelad nolla i sig. Efter det markeras alla kolumner som har en nolla i den markerade raden. Repetera med rader och kolumner tills inga fler rader eller kolumner kan markeras. Invertera markeringarna på raderna så att alla rader som saknar markering blir markerade och ta bor markeringen från de markerade raderna.

4. Nu tittar man på alla element som inte ligger på en markerad rad eller kolumn och bestämmer det minsta värdet θ av dessa element. För varje element i matrisen gör följande:

   - Om elementet tillhör en markerad rad och en markrad kolumn, θ adderas till elementet.
   - Om elementet tillhör en markerad rad eller en markerad kolumn, låt värdet vara oförändrat.
   - Om elementet inte tillhör en markerad rad eller kolumn, subtrahera θ från elementet.

   Försök sedan att göra en tilldelning på den resulterande matrisen så som i steg 1, om detta misslyckas upprepa från steg 3.

Exempel beräkning

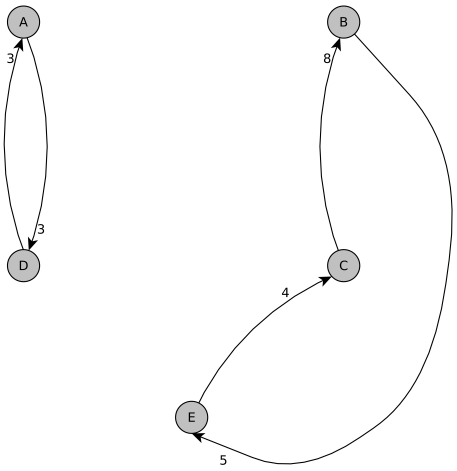


Figur 10: Slumpmässigt genererad graf.

Matrisen G beskriver grafen i figur 10.

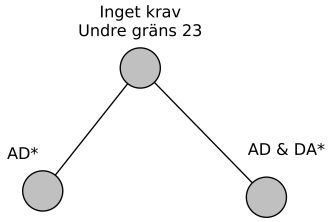
$$G = \begin{pmatrix} \infty & 11 & \infty & 3 & \infty \\ 7 & \infty & 4 & 6 & 5 \\ \infty & 8 & \infty & 5 & 3 \\ 3 & \infty & 2 & \infty & 5 \\ \infty & \infty & 4 & 9 & \infty \end{pmatrix} \tag{15}$$

Genom att beräkna den undre gränsen på G med hjälp av den ungerska algoritmen så får vi en undre gräns på 23, dock får vi inte någon lösning till TSP utan lösningen består av två stycken turer, se figur 11.



Figur 11: Tilldelningsproblemet löst på grafen i figur 10

Då lösningen inte bestod av en komplett rutt lägger vi till krav på lösningen. Eftersom det finns en minimal sub-tur mellan AD och DA och en lösning kräver att turen på något sätt kopplas samman med resten av grafen så är det lämpligt att låta ett krav vara att kanten AD inte får finnas i lösningen eller att kanten AD måste finnas i lösning (Om kanten måste finnas i lösningen så kräver det att en annan kant i sub-turen inte finns i lösningen annars genereras samma lösning om igen).



Figur 12: Beslutsträdet som begränsar lösningsrymden för grafen. AD får inte finnas eller AD måste finnas.

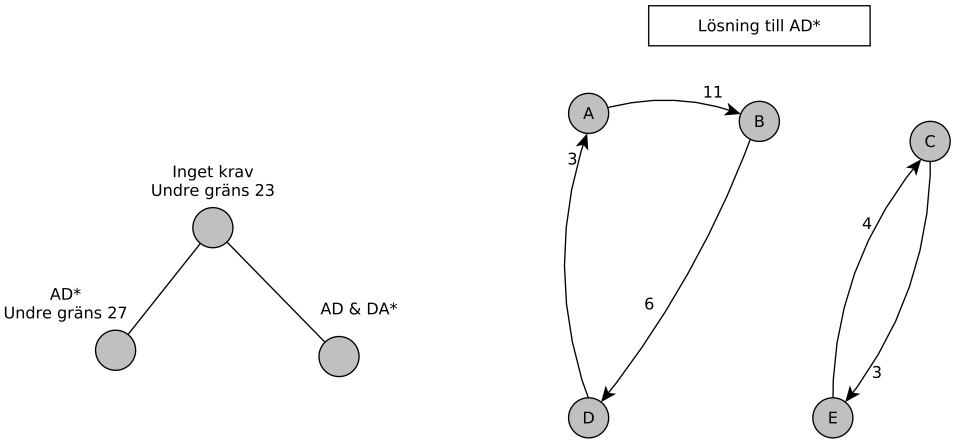
I fallet då AD inte finns i lösningen låter vi matrisen G förändras så att kanten mellan A och D är ∞.

$$G_{AD*} = \begin{pmatrix} \infty & 11 & \infty & \infty & \infty \\ 7 & \infty & 4 & 6 & 5 \\ \infty & 8 & \infty & 5 & 3 \\ 3 & \infty & 2 & \infty & 5 \\ \infty & \infty & 4 & 9 & \infty \end{pmatrix} \tag{16}$$

I fallet att AD måste finnas i lösningen så låter vi alla andra värden på första raden och fjärde kolumnen vara ∞ och låter även elementet som representerar kanten DA vara ∞.

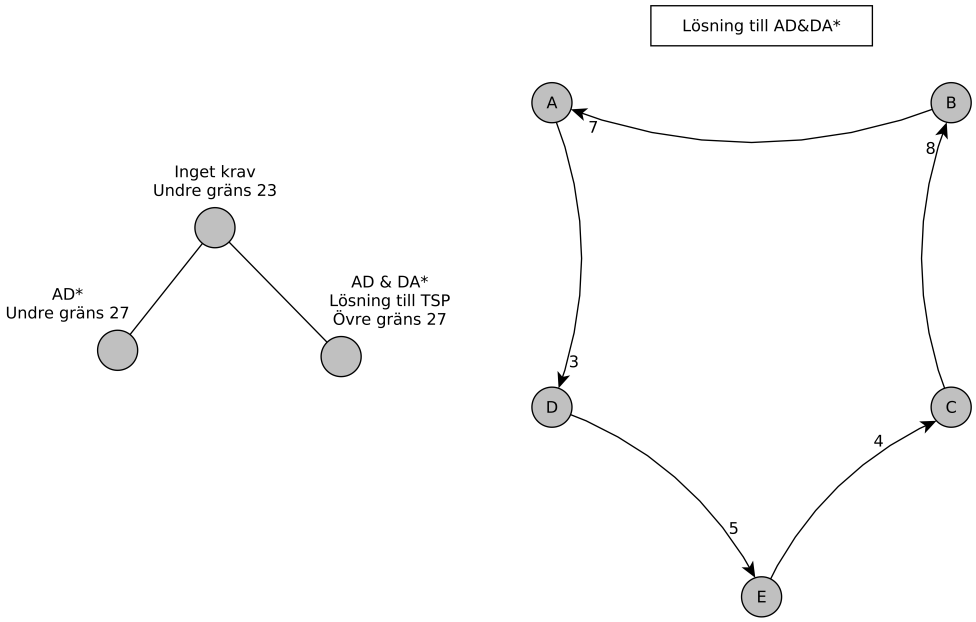
$$G_{AD\&DA*} = \begin{pmatrix} \infty & \infty & \infty & 3 & \infty \\ 7 & \infty & 4 & \infty & 5 \\ \infty & 8 & \infty & \infty & 3 \\ 3 & \infty & 2 & \infty & 5 \\ \infty & \infty & 4 & \infty & \infty \end{pmatrix} \tag{17}$$

Vi löser återigen tilldelningsproblemet denna gång för G_{AD*} och ser i figur 13 att detta genererar en lösning som även den innehåller sub-turer men har en undre gräns på 27.



Figur 13: Lösningen som bildas om AD inte finns med består återigen av sub-turer men har en undre gräns på 27.

I nästa steg löser vi tilldelningsproblemet för $G_{AD\&DA*}$ och ser i figur 14 att detta genererar en lösning till TSP problemet med en tur-längd av 27 vilket blir en övre gräns. Eftersom grenen AD* i trädet har en undre gräns som är lika med den övre gränsen så behöver den grenen inte utvärderas vidare. Detta innebär att hela trädet är utvärderat och vi har hittat en optimal lösning till ATSP problemet.



Figur 14: Lösningen som bildas om AD måste finnas med i lösningen är en komplett rutt och således en lösning till ATSP.

5 Diskussion

Vi har berört tre metoder för att fastställa en lösning eller approximation till ATSP, i det här kapitlet diskuterar vi och knyter samman teorierna för att besvara frågeställningen.

Vi har i denna studie haft som syfte att undersöka om det finns effektivare sätt att lösa ATSP på genom att utnyttja faktumet att man kan översätta graferna till symmetriska instanser. Tidigare forskning inom ämnet har påvisat vikten av att hitta effektivare sätt att lösa ATSP på grund av den höga tidkomplexiteten. Problemet är väl utforskat men det material vi har gått igenom pekar nästan alltid på samma slutsats. Eftersom ATSP är så pass svårt att lösa så har inte forskningen gått nämnvärt framåt det senaste åren, istället har man gjort små förbättringar och effektiviseringar av befintlig forskning som kan ses som gammal jämfört med resten av datalogin. Resultatet i denna studie styrker därför tidigare forskning i och med att vi inte hittat ett klart sammanband som leder en inpå nya rön som kan lösa problemet bättre.

Vi konstaterade tidigt att det finns många olika sätt att angripa ATSP problemet, både i det symmetriska och asymmetriska fallet. Dock påvisade resultatet i denna studie inga större samband eller bevis för att det skulle vara effektivare att översätta asymmetriska grafer till symmetriska grafer då de algoritmer vi valt att titta på har liknande tidskomplexitet. Svårigheten att ge ett entydigt svar på frågeställningen ligger framförallt i att dessa algoritmer har en värsta falls komplexitet som är väldigt hög, dock har man sett i tester att medelfallet är långt mycket snabbare än värsta fallet. Det bör även noteras att den forskning som vi valt att fokusera på oftare lägger större vikt på att få en tajtare undre och övre gräns för den faktiska längden av turen som approximationsalgoritmerna genererar.

Resultaten från tidigare forskning ger dock flera grundstenar som kan vara tillhanda i framtiden. Både äldre och nyare studier är överens om att Held Karp i det generella fallet genererar lösningar som är väldigt nära den optimala lösningen. Det är därför vår uppfattning med stöd av tidigare forskning att Held Karp kan användas för att jämföra korrektheten av lösningar som andra lösningsmetoder producerar.

Ett annat sätt att att ta fram lösningar är att använda sig av Branch and Bound metodik. Fördelen med BnB är att den agerar som en form av begränsad totalsökning i grafen där man i det generella fallet hittar en lösning klart snabbare, än en totalsökning. Det vill säga att även om BnB i värsta fallet har en exponentiell tidskomplexitet så hittar man oftast en optimal lösning klart snabbare. En annan fördel med BnB är att den producerar turen för den optimala lösningen till skillnad från Held Karp som genererar en uppskattad stig längd.

Man kan även dra slutsatsen att man skulle kunna jämföra effektivitet mellan nya lösningsmetoder genom att jämföra de mot en BnB lösning. Den stora fördelen med BnB framför Held Karp är att den ger en exakt lösning istället för en approximativ.

Det mest intressanta rönet är dock att det går att omforma riktade grafer till bipartita. Detta medför som tidigare nämnt ett nytt angreppssätt att lösa ATSP, även om det bipartita graferna i sig inte följer triangelolikheten. Det finns emellertid en viss osäkerhet i detta påstående då vi inte hittade eller själva utförde någon vidare forskning inom detta. Författarna av teorin hade inte heller genomfört någon vidare forskning inom detta. Vi kan således föreslå att fokus läggs på att konstruera en approximations algoritm som löser symmetrisk TSP med en bipartit graf som utgångspunkt, då detta möjligtvis skulle leda till en effektivare approximering av ATSP. Men även denna slutsats måste tolkas med försiktighet då vi i dagsläget inte kan presentera några belägg för påståendet. Därför krävs det mera forskning för att fastställa hypotesen, någonting som vi hade utforska mera, ifall vi hade haft mer tid och resurser än de som tilläts under projektet. I framtiden rekommenderar vi att man bör lägga mera fokus på bipartita grafer, då det fortfarande är av stor vikt att hitta ett bättre sätt att lösa ATSP.

6 Slutsats

I detta projekt har vi undersökt tillvägagångssätt för att lösa problemet ATSP, med syftet att bestämma huruvida det är effektivt eller inte att översätta ATSP instanser till symmetriska grafer jämfört med andra angreppssätt. Det tydligaste resultatet som framkom var att det fortfarande finns mycket att lära om inom problemet. Vi har inte hittat någon metod som är klart effektivare än en annan. Detta tyder på att man behöver komma på nya sätt att angripa ATSP, denna åsikt stöds av flertalet av de källor som vi valt att använda som material. Vi har därför inte kunnat bekräfta eller förkasta hypotesen att en transformering från ATSP till TSP skulle vara effektivare än befintliga lösningsmetoder för ATSP.

En svaghet i studien är framförallt att vi ha haft begränsat om tid och resurser för att utföra egna beräkningar och mätningar på befintliga metoder. En annan begränsning är den höga teoretiska inlärningströskeln för materialet som har funnits tillhanda, därför har mycket av tid lagts på inläsning av fakta och befintliga studier.

I mån av tid hade vi velat undersöka frågan om det skulle kunna finnas en effektiv strategi för att lösa bipartita grafer, då detta var det nyaste och minst välstuderade ämnet vi hittat. I hopp om att hitta ett effektivare sätt att lösa problemet. En framtida studie som lägger sitt huvudfokus på detta område är därför av stort värde. Därför hoppas vi att detta material kan guida andra till att ta sig an problemet på ett nytt sätt. Samt att ha väckt intresset för problemet i studenter som är nya inom området TSP.

Den kanske viktigaste slutsatsen vi kan dra för dagens forskning är dock att ATSP lösningar generellt kan brytas ner i två stycken faktorer. Hur exakta approximeringarna är med hänsyn till dess tidsåtgång att beräkna, eftersom man i dagsläget måste offra snabbhet för exakthet och vice versa. Därför lämnar vi frågan öppen om man kan hitta ett klart effektivare sätt att lösa ATSP på i framtiden.

Referenser

[1] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, 1 2007.

[2] Marcus Öberg David Falk, Klaus Nicosia. Advanced algorithms project 2. Hand in, KTH, 1 2013.

[3] Jeffrey H Kingston. *Algorithms and Data Structures Design, Correctness, Analysis*. Pearson Education, second edition, 1997.

[4] Per Mattsson. The asymmetric traveling salesman problem. Master thesis, Uppsala Universitet, 10 2010.

[5] Kurt Hornik Michael Hahsler. Tsp infrastructure for the traveling salesperson problem. Report, Computer Science and Engineering & Department of Finance, Accounting and Statistics, Southern Methodist University & Wirtschaftsuniversität Wien, 2007.

[6] Richard M. Karp Michael Held. The traveling-salesman problem and minimum spanning trees. Report, IBM Systems Research Institute, New York & University of California, Berkeley, California, 1969.

[7] Christian Nilsson. Heuristics for the traveling salesman problem. Report, Linköping University, 2003.

[8] Anna Palbom. *On Approximating Asymmetric TSP and Related Problems*. PhD thesis, KTH, 2006.

[9] Haomin Li Ratnesh Kumar. On asymmetric tsp: Transformation to symmetric tsp and performance bound. Report, Department of Electrical Engineering, University of Kentucky, 1994.

[10] Subhash Suri. Bipartite matching & the hungarian method. Notes, Department of Computer Science, University of California, Santa Barbara, 8 2006.

[11] David Paul Williamson. Analysis of the held-karp heuristic for the traveling salesman problem. master thesis, Massachusetts Institute if Technology, 06 1990.

[12] Weixiong Zhang. Branch-and-bound seach algorithms and their computational complexity. Report, Information Science Institute, Department of Computer Science, University of Southern California, 5 1996.