CHALMERS

KANDIDATARBETESRAPPORT

Kollektiv styrning av autonoma robotar

ADO AGANOVIC
ANDREAS ANDERSSON
KARIN AHLBERG
MARCUS LILIEGÅRD
SIMON LARSON

Forskargrupp
Institutionen för signaler och system
Chalmers tekniska högskola
Göteborg 2012 SSYX02-12-25

Sammanfattning

I utvecklingen av ny teknik riktas fokus ofta mot effektiviseringen av system och hur de kan göras mer miljömässigt hållbara. Användandet av fler autonoma system är en stor del av att öka säkerheten och effektivisera framtidens industri och logistik. Systemet analyseras och optimeras för den process som skall utföras. Detta görs dels för att minimera kostnaden och energiåtgången men även för att förhindra olyckor och låsningar i systemet.

Denna typ av system kräver lösningar till flera tekniska problem. Styrningen av varje individuell enhet måste fungera för att fel inte skall fortplanta sig i systemet. Låsningar i systemet förhindras genom att implementera en schemaläggare som kontrollerar när olika händelser i processen skall utföras. För att möjliggöra kommunikation och samarbete mellan de olika enheterna måste ett protokoll för dataöverföring konstrueras.

I detta projekt analyseras och jämförs två algoritmer för schemaläggningen av diskreta händelser, i detta fall förflyttningar av autonoma fordon. Den första metoden består av ett bokningssystem med A*-inspirerad ruttberäkning (BAA). Den andra är en händelsestyrd schemaläggare baserad på automater (HSBA). Ett kommunikationsprotokoll med inspiration från polling implementeras för att kontrollera dataöverföringen mellan systemets enheter.

Algoritmerna för schemaläggning och kommunikationsprotokollet implementeras i en fallstudie där autonoma robotar rör sig i ett vägnät. I fallstudien används 3 stycken Pololu m3pi robotar som har försetts med en radio som sköter den trådlösa dataöverföringen.

Jämförelse mellan BAA och HSBA visade att HSBA är mindre känsligt för störningar. Inräkningen av väntetider gör att BAA resulterar i kortare systemtider. BAA har även kortare beräkningstider jämfört med HSBA.

Abstract

In the development of new technology focus is aimed towards efficiency and sustainability. To increase the efficiency and safety of future transportation and industry the possibility of using an automated system is often considered. The system is analyzed and optimized for the procedure at hand. This is done to minimize both the financial cost and the energy cost but also to prevent accidents and deadlocks in the system.

This kind of system poses several problems. The individual control of each autonomous unit has to work to allow the system to function. To eliminate the risk of deadlock a schedueler is implemented that controls when different events in the procedure should be performed. To enable the different parts of the system to communicate and work together, a protocol for data transfer needs to be constructed.

In this study two algorithms for the scheduling of discrete events are analyzed, in this case the movement of robots. The first algorithm consists of a booking system with route planning based on the A*-algorithm (BAA). The second is an event controlled scheduler implemented with the aid of automata (HSBA). A communication protocol is constructed with inspiration from systems using different kinds of polling between system units.

The communication protocol and the algorithms for scheduling are implemented in a case study where autonomous robots move within a system of roads. In the case study the robots used are m3pi from Pololu that have been equipped with a radio to handle the wireless data transfer.

A comparison of the two algorithms showed that HSBA is less sensitive to disturbances. The BAA system considers the waiting times which results in shorter system times. The computation complexity of BAA is lower than that of HSBA.

Förord

Detta kandidatarbete utfördes våren 2012 på institutionen för signaler och system på Chalmers tekniska högskola. Vi som författat detta arbete vill rikta ett varmt tack till vår handledare Oskar Wigström för sitt stöd, stora engagemang och för att han alltid ställt upp för oss.

Göteborg den 16 maj 2012

Innehåll

1 Inledning

I dagsläget läggs allt större vikt på att effektivisera system och göra dem mer miljömässigt hållbara. En viktig del i utvecklingen gällande utformningen av framtidens trafiknät är att få dessa att bli så tids- och energieffektiva som möjligt. Ett av de många alternativen för att uppnå detta är autonomt styrda fordon.

Autonomt styrda fordon i framtidens vägnät skulle kunna leda till ökad säkerhet i trafiken, tidsoptimerade transporter och med detta minskade utsläpp av avgaser. För att effektivisera autonoma transporter i större skala behövs någon form av kollektiv organisering och planering av de autonomt styrda fordonens färd. Genom att optimera färdvägen är det möjligt att förhindra trafikproblem både i form av kollisioner och köbildning. Autonom styrning av fordon har flera stora användningsområden, dels inom industriell verksamhet som t.ex. automatiska truckar men även inom transportindustrin.

Kommunikation mellan olika delar i ett autonomt system är oerhört viktig. Om överföringen av information angående ett autonomt styrt fordons omgivning inte når fram till kontrollenheten för fordonet kan icke-önskvärda händelser ske. Olyckor i form av kollisioner eller avkörningar är bara några av de följder som kan uppstå. Kommunikationen kan liknas vid den mellan olika datorer i moderna bilar där fel i kommunikationen kan leda till stora missöden. Det finns flertal typer av protokoll för hur kommunikation av denna sort byggs upp, inspiration till det protokoll som implementeras i projektet kommer till stor del från polling [8].

I detta projekt implementeras ett system för kollektiv styrning av AGVer (Automated Guided Vehicle). I systemet skall AGVer orientera sig självständigt i ett vägnät. Det kollektiva styrsystemet ska planera och optimera AGVernas rutter samt förhindra att kollisioner och låsningar uppstår. Hela systemet kan sedan ytterligare optimeras genom schemaläggning av rutterna. Att implementera detta kräver någon form av positioneringssystem, rutt- och schemaläggning samt kommunikation.

1.1 Syfte och mål

Projekts syfte är att undersöka två olika algoritmer för schemaläggning samt implementera ett protokoll för kommunikation och utvärdera deras prestanda i ett system som styr AGVer kollektivt.

Genom att utvärdera algoritmerna är det möjligt att fastställa vilken av dem som är mest önskvärd utifrån olika förutsättningar och kriterier. Förutsättningar som kommer studeras är olika antal AGVer i systemet samt storlek på vägsystemet. Kriterier som sedan studeras är vänte- och beräkningstider i

systemet, robusthet och sluttiden för det totala system samt för varje enskild AGV.

Ett kommunikationsprotokoll skall utvecklas inom projektet. Inspiration tas från existerande protokoll för liknande situationer, t.ex. polling [8]. De kriterier som utvärderas för kommunikationen är överföringshastigheten, antalet datapaket per sekund samt stabiliteten, d.v.s. om all data når fram till mottagaren.

I projektet kommer en fallstudie utföras där ett system av AGVer autonomt skall kunna orientera sig och anpassa sig efter varandra i ett fördefinierat trafiknät. I fallstudien skall AGVer kunna förflytta sig i vägnätet utan att kollidera eller skapa låsningar. För mer detaljerade mål vad gäller prestanda för fallstudien se Tabell 2 i Bilaga A. Projektet ämnar även att ur ett hållbarhetsperspektiv utföra en analys av systemet. Möjliga implementeringar för att minska energiåtgången i systemet kommer diskuteras.

1.2 Uppgift

För att uppfylla projektets mål ska flera AGVer parallellt kunna utföra individuella uppgifter. Detta kräver styrning på både individuell och kollektiv nivå. AGVerna skall inte bara kunna följa ett vägnät, de skall också anpassa sig till varandra för att undvika låsningar och kollisioner. För att uppfylla detta behöver AGVerna genom schemaläggning kunna planera rutter och undvika låsningar.

Att styra AGVer kollektivt är ett tekniskt problem som omfattar flera olika ingenjörsdiscipliner. Dessa ingenjörsområden består bland annat av reglerteknik, optimeringsteori, programmeringsteknik, signalbehandling samt digitalteknik. Utmaningen ligger i att integrera dessa områden för att skapa ett robust system. För att lyckas med detta måste ett gränssnitt utvecklas där systemarkitekturen möjliggör ett samspel mellan AGVns olika delar. Ett exempel är synkroniseringen av schemaläggningen samt kommunikationen mellan AGVerna. Dessa två delar har parallella trådar som måste fungera samtidigt för att systemet skall ha önskad funktion.

Projektet är upplagt så att fungerande kommunikation och det kollektiva styrsystemet i första hand skall tas fram. När detta är uppfyllt finns det möjlighet att implementera ytterligare funktioner i mån om tid. I projektet kommer störst vikt läggas vid programmeringen och styrningen av AGVerna.

För att lösa de ovannämnda problemen specificeras nu fyra deluppgifter.

Schemaläggning

Varje enskild AGVs rutt till ett bestämt mål planeras så att låsningar med andra AGVer undviks. Detta kräver att schemaläggningen kan boka vissa vägsträckor. Rutterna kan även schemaläggas enligt olika algoritmer, d.v.s. att AGVernas prioritet ändras, så den totala systemtiden optimeras.

En annan utmaning är att AGVer vid ett eller annat tillfälle kommer åka samma sträcka samt att tidsfördröjningar kan uppkomma. Om detta inte åtgärdas kommer systemet att låsa sig. Denna typ av händelse måste kunna hanteras.

Kommunikation

För att brygga den kollektiva styrningen med AGVernas egna styrning behöver det finnas möjlighet för varje enhet att kunna skicka och ta emot data. För att lösa detta problem behövs hårdvara som kan skicka samt ta emot datan. Det behövs även en mjukvarulösning för att skicka data mellan de olika enheterna, med andra ord behövs ett kommunikationsprotokoll.

Ytterligare en viktig aspekt som utgör en utmaning är att två AGVer aldrig får skicka data samtidigt, eftersom detta kan göra att signalerna stör ut varandra eller att systemet i detta läge inte kan veta vilken AGV som skickat informationen. Ett kommunikationsprotokoll måste upprättas för att kunna hantera dessa svårigheter. Hur stora datapaket som får skickas mellan AGVerna måste också specificeras.

Miljömässig hållbarhet

Systemet skall analyseras ur ett hållbarhetsperspektiv där målet är att finna möjligheter för att minimera den totala energiåtgången i systemet. Det behöver därför utvärderas hur AGVernas körsätt påverkar energiåtgången. Ett tydligt exempel på när systemet kan effektiviseras är då AGVerna behöver stanna och vänta vid en korsning. Då acceleration är väldigt energikrävande kan t.ex. hastigheten regleras så att körningen blir effektiv i hänsyn till energiåtgång.

Fallstudie

För att utföra projektets praktiska del behöver vissa saker implementeras för de AGVer som används. Denna del av projektet kommer att avgränsas till att ingen egen hårdvara utvecklas, endast färdigkonstruerade hårdvarulösningar används. Färdiga komponenter kommer att köpas in och endast montering och programmering av AGVerna kommer ske.

AGVn ska kunna följa ett fördefinierat vägnät där den skall kunna göra förbestämda val i korsningar. Den skall kunna, ur indatan från sensorer, tolka vägnätet och när den kommer fram till en korsning utföra den manöver som programmerats, t.ex. svänga höger. I projektets implementation kommer AGVerna följa ett fast och tidigare känt vägnät.

2 Teori

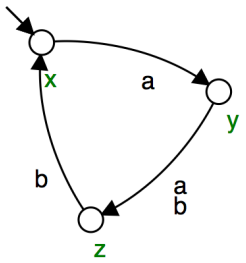
I detta avsnitt återfinns enklare teoretiska beskrivningar av de algoritmer och verktyg som används för att bygga upp studien.

Schemaläggning används som ett sätt att fördela resurser i ett system. I ett vägnät kan en väg ses som en resurs som delas mellan fordon. Schemaläggningens uppgift är att hitta en så bra lösning som möjligt, t.ex. kan färdtid minimeras och låsningar förhindras. I detta projekt modelleras systemet diskret. För planering av en AGVs rutt behövs algoritmer för att bestämma lämpligaste färdvägen enligt olika kriterier. Grunderna för den ruttberäkning som används beskrivs nedan tillsammans med grundläggande grafteori. I slutet av detta kapitel ges en beskrivning av den hårdvara som används i fallstudien.

2.1 Diskreta system

Ett diskret system kan modelleras med hjälp av automater. För en mer utförlig beskrivning av diskreta system och automater hänvisas till [2]. En automat kan beskrivas med en mängd diskreta tillstånd $Q = \{q_1,q_2,...,q_n\}$ och en mängd med händelser $E = \{e_1,e_2,...,e_n\}$. Övergång från ett tillstånd till ett annat kan ske då ett specifikt övergångsvillkor är uppfyllt. Övergångsvillkor består av en eller flera händelser som leder från ett tillstånd till ett annat, $f : Q \times E \rightarrow Q$. Ett av tillstånden i automaten måste markeras som initialtillstånd. Initialtillståndet är det tillstånd automaten utgår från. Ett exempel på en automat visas i Figur 1.

Händelser i ett system kan vara antingen kontrollerbara eller okontrollerbara. I projektet kommer enbart kontrollerbara händelser att behandlas.



Figur 1: Grafisk representation av en automat med diskreta tillstånd [x,y,z] som kan nås genom olika övergångar då händelserna [a,b] sker. Övergångarna illustreras med pilar mellan tillstånden. Initialtillståndet x markeras med en pil.

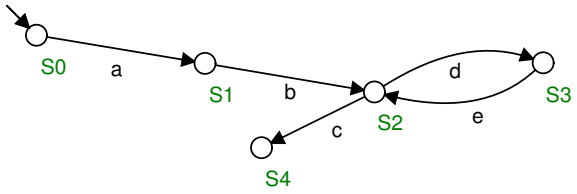
För ett givet system kan ett eller flera tillstånd anges vara önskvärda att ta sig till, dessa tillstånd benäms som systemets destination. I blockerande system finns det övergångar som gör det omöjligt att komma till en destination. Genom att förbjuda dessa övergångar skapas ett icke-blockerande system där det alltid går att nå en destination.

Flera automater kan synkroniseras och bilda en ny automat. Den synkroniserade automatens destination blir det tillstånd då alla individuella automater nått sin destination. En och samma händelse kan finnas i flera automater. För att en händelse skall kunna ske i den synkroniserade automaten måste denna händelse vara möjlig från de tillstånd automaterna befinner sig.

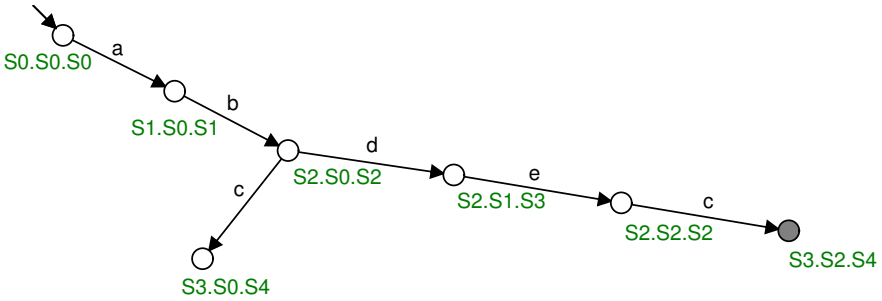
Synkronisering av automater visas med ett enkelt exempel. Två robotar, R_1 och R_2, kan plocka upp ett föremål. R_1 skall plocka upp föremålet för att hålla fast det medan R_2 borrar ett hål. Därefter skall R_1 lägga ner föremålet igen. Händelsemängden för robot R_1 blir { a = plockar upp föremål, b = håller fast föremålet, c = lägger ner föremålet } och för R_2 blir den { d = borrar ett hål, e = dra ur borr }. I initialtillståndet ligger föremålet på bordet. För varje Robot kan nu en automat sättas upp.

Figur 2: Robot R_1s (vänster) och R_2s (höger) händelsemängd visat i den ordning de kan utföras. Initialtillstånd markeras med pil, sluttillstånd markeras i grått.

I en tredje automat Z definieras det önskade händelseförloppet. Händelse d och e får endast ske då händelse b hänt. Detta visas i Figur 3. Dessa tre automater kan sedan synkroniseras för att bli en automat som visar hela systemet $R_1 \,||\, R_2 \,||\, Z$. I Figur 4 kan vi se att vissa tillstånd inte leder till vårt sluttillstånd, dessa är blockerade och kan plockas bort för att få ett icke-blockerande system.



Figur 3: I automaten Z har de möjliga förloppen specifierats. Borrning ska enbart kunna ske då föremålet hålls fast.

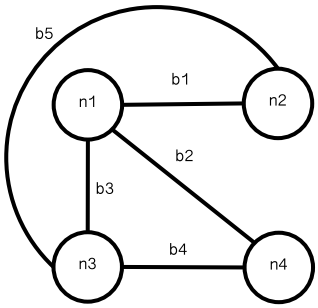


Figur 4: Synkronisering av de tre automaterna R_1, R_2 och Z. Tillstånd S3,S0,S4 är ett blockerat tillstånd varifrån sluttillståndet inte kan nås. Initialtillstånd markeras med pil, sluttillstånd markeras i grått.

2.2 Grafteori

Grafteori är den gren inom matematiken där grafers egenskaper studeras. För mer detaljerad beskrivning och fler exempel rekommenderas [10]. För vidare läsning om algoritmer inom grafteori, dels de som beskrivs nedan, hänvisas till kapitel 9 i [11]. En graf definieras av en mängd noder sammanbundna av en mängd bågar. Varje nod kan ansluta till flera bågar men varje båge ansluter endast till två noder. Figur 5 visar ett enkelt exempel på en graf med noderna $[n_1, n_2, n_3, n_4]$ som sammankopplas av bågarna $[b_1, b_2, b_3, b_4, b_5]$.

I en viktad graf tilldelas varje båge ett värde som representerar kostnaden att traversera denna båge. Dessa grafer används i detta projekt för att representera korsningar och vägarna mellan dem, kostnaden representerar längden



Figur 5: Ett enkelt exempel på en graf med noderna $[n_1, n_2, n_3, n_4]$ samt bågarna $[b_1, b_2, b_3, b_4, b_5]$.

på vägen. I följande avsnitt introduceras två stycken algoritmer för att finna den billigaste vägen mellan två noder i en viktad graf.

Dijkstras algoritm

Låt noderna $N = n_1, n_2 \ldots n_i$ tillsammans med bågarna $B = b_1, b_2 \ldots b_j$ med motsvarande vikt v_l, $l \in [1,..,j]$, definiera en viktad graf. Målet är att bestämma den billigaste rutten mellan de två noderna n_{start} och n_{stop}. Från början märks alla noder förutom n_{start} som okända, när vi sedan traverserar grafen kommer fler och fler noder bli kända. Att en nod är känd innebär att den billigaste rutt till denna nod från n_{start} är känd.

För att beräkna billigaste vägen från n_{start} till n_{stop} undersöks först alla grannoder till n_{start}, alltså alla noder som delar en båge med n_{start}. För varje nod noteras kostnaden för att ta sig dit, härefter benämnt g_n. Det vill säga att g_n är summan av alla v_l för de bågar som traverserats på vägen. Denna kostnad finns angiven för varje nod n_i som har undersökts och är den lägsta hittills kända kostnaden att förflytta sig mellan n_{start} till n_i. För varje nod anges även vilken nod som är dess föräldernod, d.v.s. den föregående noden. Då en nod som övervägs redan har g_n angivet jämförs denna med kostnaden för den nya rutten. Är det nya värdet lägre har en billigare väg till noden hittats. Det befintliga g_n och föräldern byts då ut mot det nya värdet och den nya föräldern.

För varje algoritmcykel väljs en ny nod att utgå ifrån, denna nod väljs som den nod med lägst g_n som ännu inte märkts känd. Från denna nod undersöks, och möjligtvis uppdateras, de olika värdena för dess grannar. Efter detta markeras noden som känd och en ny nod att besöka väljs på samma sätt i den uppdaterade listan.

När n_{stop} markeras som besökt har den billigaste rutten från n_{start} till n_{stop} hittats. Denna väg spåras genom att gå till föräldern till n_{stop}, och vidare bak till n_{start}.

A*-algoritmen

A*-algortimen används för att heuristiskt snabba på beräkningen av den optimala vägen från start nod till mål i en viktad graf. Metoden har sin grund i Dijkstras algoritm för att bedöma kostnaden för att ta sig från n_{start} till n_i, med tillägget av en heuristisk del som uppskattar den kvarvarande längden från n_i till n_{stop}. Den heuristiska delen gör att färre noder behöver utvärderas för att hitta kortaste vägen. I varje nod adderas en heuristisk uppskattningskostnad h_n med g_n. Detta ger en totalkostnad t_n, alltså $h_n + g_n = t_n$. h_n består av ett uppskattat värde på avståndet mellan n_i och n_{stop}. Om h_n ökar från föregående nod rör vi oss bort från n_{stop} och minskar den rör vi oss mot n_{stop}.

Till skillnad från Dijkstras är det inte den nod som har lägst g_n som besöks först utan den med lägst t_n. På detta vis utvärderar vi i första hand de noder som för oss mot slutnoden. Om g_{stop} har tilldelats ett värde klipps alla noder som har t_n större än detta värde av från lösningsrummet och på så sätt behöver inte lika många onödiga lösningar undersökas innan den optimala vägen hittats.

Det heuristiska värdet kan beräknas på olika vis beroende på tillämpning. I projektets implementation, där vikterna representerar längder på vägar, tas den euklidiska normen för de (x,y) koordinater som noderna representerar som den heuristiska kostnaden vilket garanterar att den funna lösningen faktiskt är den optimala då inga noder med en möjlig optimal lösning tas bort.

2.3 Komponentbeskrivning

I nedanstående stycken återfinns en enkel teknisk beskrivning av de hårdvarukomponenter som refereras till i senare delar av rapporten.

Pololu m3pi

M3pi [5] är en robotplattform från Pololu som är konstruerad för att följa linjer i marken. Roboten har en maxhastighet på ungefär 1 [m/s] och är utrustad med både LCD-skärm och en högtalare. Linjeföljningen utförs med

hjälp av fem stycken reflektionssensorer positionerade i framkant på plattformen. Roboten drivs av fyra stycken AAA-batterier och har en diameter på 95 [mm]. I Figur 6 visas roboten i sitt originalutförande.

Figur 6: Roboten Pololu m3pi i originalutförande. Bilden tagen från *Pololu m3pi User's Guide* [5].

Pololu wixel

Pololu wixel [6] är en microcontroller utrustad med 2,4 [GHz] radio och USB. Två sammankopplade wixelmoduler har mellan sig en trådlös överföringshastighet på 10 [kB/s]. I varje modul finns 32 [kB] flashminne samt 4 [kB] RAM. Vilken frekvens som radioenheterna sänder och tar emot på kan ställas om.

3 Metod

Utveckligen av de tre huvuddelarna: schemaläggning, kommunikation och implementation av fallstudien, har skett parallellt. I de kommande avsnitten förklaras och motiveras de metoder som använts inom de olika delarna av projektet. Vidare specificeras vad dessa metoder innebär rent konkret i projektets implementation.

3.1 Schemaläggning

För schemaläggning har två olika metoder använts. Den första metoden består av ett bokningssystem med A*-inspirerad algoritm (BAA). Den optimerar och schemalägger AGVernas rutter utifrån det som är känt om systemet.

I den andra metoden, händelsestyrd schemaläggare baserad på automater (HSBA), bestäms varje enskild rutt med A*-algortimen. Därefter skapas en synkroniserad automat som innehåller alla tillåtna tillstånd för AGVerna. Båda metoderna kommer simuleras i två olika vägnät, se Bilaga B.

3.1.1 Bokningssystem med A*-inspirerad algoritm

BAA beräknar kortaste vägen genom ett vägnät med A*-algortimen (se teoriavsnitt 2.2) som grund. Kostnaden av att ta sig mellan två noder valdes till ett relativt avstånd. Eftersom vägnätet konstruerats efter ett koordinatsystem är det avståndet mellan noderna som anger kostnaden. Avståndet i det verkliga vägnätet kommer således inte ha betydelse så länge proportionerna är de samma som i koordinatsystemet. Även den heuristiska delen grundar sig på att vägnätet konstruerats efter ett koordinatsystem. Den heuristiska delen kan beräknas på flera sätt. I projektet valdes att beräkna det kvarvarande avståndet som den euklidiska normen.

När rutterna bestämts förs dessa in i ett bokningssystem där varje vägsträcka bokas i de tidssteg då en AGV beräknas befinna sig där. I de fall flera AGVer samtidigt vill utnyttja en väg kommer AGVerna att prioriteras och de lägre prioriterade AGVerna kommer att få vänta tills vägsträckan är tillgänglig. För att minska systemtiden beräknar AGVerna sina rutter utifrån möjligheten att ta en annan väg eller att vänta på att en bokad vägsträcka inte längre används. Detta implementeras genom att en väntetidsvariabel, w_i, adderas till t_i, $t_i = g_i + h_i + w_i$. Den väg som tar kortast tid inklusive väntetider kommer då att väljas. För flödeschema över algoritmen se Bilaga C.

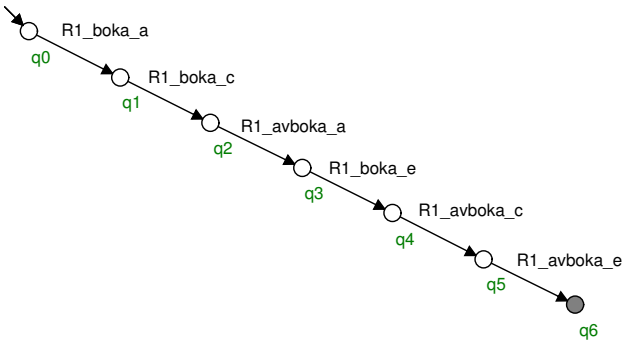
När två eller fler AGVer i systemet har bokningar som påverkar varandra kommer systemets totaltid påverkas av vilken AGV som har högst prioritet, d.v.s. hur schemaläggning av AGVernas rutter väljs. För att försöka minimera systemets totaltid kan olika schemaläggningsalgoritmer användas. I projektet utvärderas tre vanliga metoder. Den vanligaste och allra enklaste algoritmen First Come, First Served (FCFS) [7] innebär att AGVerna endast väntar om nästa sträcka för tillfället är otillåten. Shortest Job First (SJF) innebär att den kortaste rutten prioriteras först för att det ska uppkomma så få tillfällen som möjligt då rutterna korsas [7]. Den sista algoritmen som jämförs i projektet är Decreasing Time Algorithm (DTA) där de längsta rutterna prioriteras [1]. Med denna metod gör det inget om de kortare rutterna blir fördröjda så länge de tar kortare tid än den längsta rutten.

3.1.2 Händelsestyrd schemaläggare baserad på automater

För att skapa automater har Supremica API använts [9]. Supremica är ett javabaserat program som kan användas för att modellera diskreta system. I API:t finns det färdiga metoder för att konstruera automater och sedan skapa en synkroniserad och icke-blockerande automat. För en beskrivning av diskreta system och automater se avsnitt 2.1.

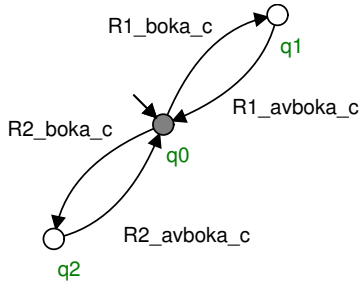
Målet med att använda automater är att få ut en synkroniserad automat där man från varje tillstånd kan se vilka händelser som är tillåtna. Händelserna kommer vara bokningar och avbokningar på de vägar som AGVerna ska ta. För att göra detta krävs två typer av automater.

Den första automaten beskriver en enskild AGVs rutt. AGVns rutt beräknas med hjälp av A*-algoritmen. När ordningen på vägarna är bestämd så kan boknings- och avbokningssekvensen skapas. Händelsesekvensen kommer vara sådan att varje AGV kommer alltid ha minst en väg bokad. Exempel på en automat för en AGVs rutt visas i Figur 7.



Figur 7: Automat för boknings- och avbokningssekvens för en AGV som ska köra vägarna a,c,e.

Den andra typen av automat ser till att varje väg enbart får bokas av en AGV åt gången. Endast AGVer som har vägen i sin rutt kommer läggas till i automaten. Exempel på denna automat visas i Figur 8.



Figur 8: Automat för hur bokning på vägen c får ske. Vid bokning av en väg hamnar automaten i ett nytt tillstånd varifrån ingen annan AGV kan boka. Automaten återgår till sitt initialtillstånd då vägen avbokas.

När alla individuella automater är skapade synkroniseras dessa och blockerande övergångar tas bort. Den synkroniserade automaten kan användas till att ta reda på vilken ordning AGVerna får lov att köra. När detta ska tillämpas praktiskt kan varje AGV skicka en förfrågan om att boka/avboka. Om denna händelse finns med bland de övergångar från det nuvarande tillståndet i automaten skickas en signal om godkännande till AGVn. I annat fall väntar AGVn till den fått tillåtelse att fortsätta.

3.1.3 Beräkningstider

Följande stycke beskriver tidskomplexiteten för de olika algoritmer som använts vid schemaläggningen av AGVer under projektet. För bakgrund till komplexitetsanalys och de datastrukturer som nämns rekommenderas [11]. Beräkningstiden för de två schemaläggningsmetoderna varierar beroende på antalet AGVer, storleken på vägnätet samt hur långa dess rutter är.

Tidskomplexiteten för BAA-algoritmen beräknas till $O(A \cdot B \cdot \log N)$, där A,B och N representerar antalet AGVer, antalet bågar respektive antalet noder. I denna beräkning antas att alla datastrukturer implementerats med $O(1)$ söktid samt att operationen som hittar minimumvärdet bland g-värdena har komplexitet $O(\log N)$. Hur dessa komplexitetskrav uppnås beskrivs grundligt i [11]. Denna tidskomplexitet är ett maximalt värde för beräkningstiden, tiden kommer givetvis att variera beroende på rutt.

3.1.4 Simulering av synkroniserad automat

För att det ska vara möjligt att jämföra resultatet från BAA med resultatet från HSBA behövdes en simulerad körning göras. Vid början och slut på varje korsning skickas en förfrågan från AGVn till basenheten om tillåtelse

att boka respektive avboka en sträcka. Om förfrågan beviljas skickas en accepterande signal till AGVn. Beviljas inte förfrågan väntar den och skickar nya förfrågningar med jämna mellanrum. Ur simuleringen fås alla tider som AGVerna befinner sig på vägarna. Även intressanta värden som väntetider för individuella AGVer och totaltid för hela systemet tas fram.

3.2 Kommunikation

AGVernas kommunikationsystem har implementerats med radiosändaren Pololu wixel. Wixeln har en helt separat processor från m3pi och de programmeras på två olika plattformar. Det finns däremot en koppling i hårdvaran samt i gränssnittet som möjliggör kommunikation mellan dem.

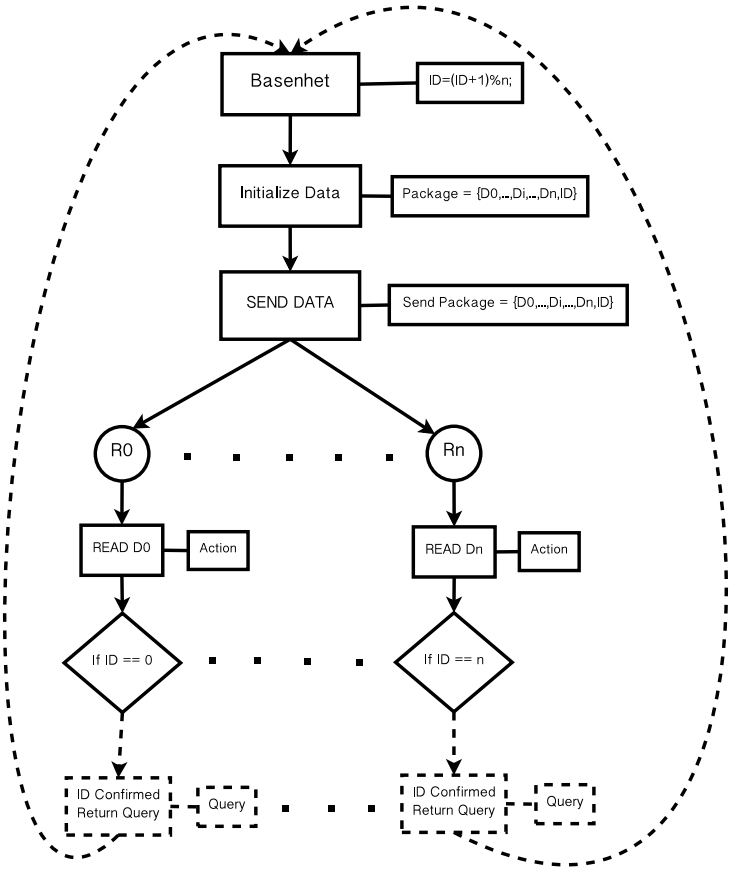
Protokollet som används i projektet är baserat på samma idé som polling. Kommunikation som använder polling innebär att en styrande enhet utför en kontinuerlig kontroll av andra program eller enheter för att se vilket tillstånd de är i, vanligen för att se ifall de fortfarande är anslutna eller vill kommunicera [8][3].

Vid multipoint- eller multidroppkommunikation [3] delar en kontrollerande enhet kommunikationskanal med flera underenheter. Den styrande enheten sänder ut ett meddelande till en enhet åt gången och frågar om enheten har något att kommunicera. På detta vis undviks att flera enheter försöker skicka samtidigt vilket kan leda till att data går förlorad.

Protokollet som använts grundar sig, precis som polling, i att det finns en basenhet som sänder ut data till övriga enheter, se Figur 9. All data som skickas sänds på en och samma frekvens. Detta innebär att bara en enhet åt gången får skicka data då de annars stör ut varandras försändelser. Varje gång som basenheten vill sända ut ett datapaket inkluderas även ett unikt ID. Detta ID representerar en specifik AGV. I datapaketet som sänds ut finns instruktioner till samtliga AGVer. Det innebär att en AGV kommer kunna få instruktioner från basenheten trots att dessa unika ID inte sänds. Fördelen med detta är att under en sändning skickas data ut till samtliga AGVer på en och samma gång. Detta innebär att överföringen av data mellan basenhet och AGVerna effektiviseras. Dessutom kan man på bara en datautsändning exempelvis få AGVerna att stanna. Efter varje dataförsändelse växlas det unika ID som skickas ut. Detta görs cykliskt bland de ID som existerar i systemet. Enbart AGVn som har det ID som skickats ut svarar basstationen, detta representeras av de streckade linjerna i flödesschemat (Figur 9).

När AGVerna svarar basstationen skickas en bekräftelse på att datapaketet kommit fram. AGVn skickar även en extra byte ifall den vill meddela basstationen att den kommit fram till en korsning eller om den precis åkt ur en korsning. Denna försändelse görs så att basstationen skall veta var AGVn

befinner sig i vägnätet samt möjliggöra för basstationen att uppdatera instruktionerna till AGVerna.



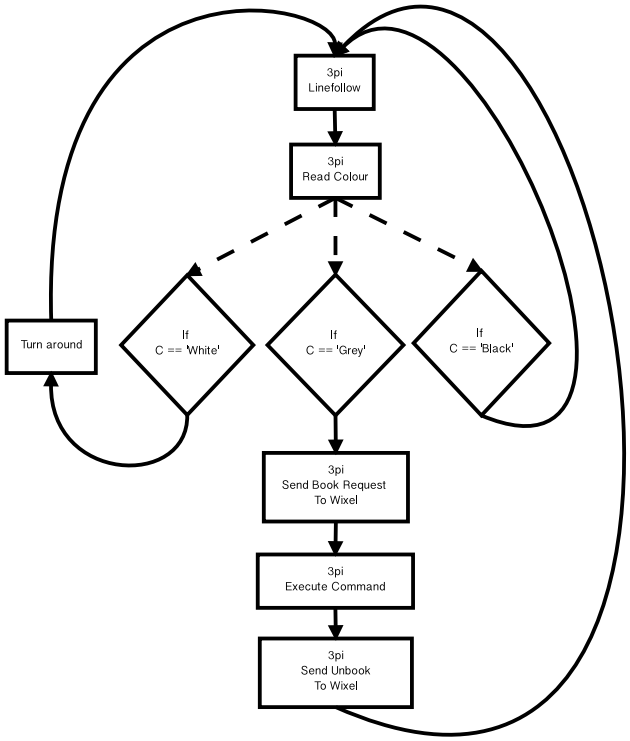
Figur 9: Flödesschema över kommunikationsprotokollet.

3.3 Fallstudie

AGV-systemet med tre stycken wixelbestyckade m3pi enheter implementerades i C kombinerat med Java. Figur 10 visar ett förenklat flödesschema som beskriver hur varje AGV går igenom sin programcykel. Denna kod beskrivs i mer detalj i det kommande avsnittet.

När rutterna beräknats och schemalagts måste de också översättas till direktiv för hur AGVn skall svänga i varje korsning. Detta görs genom att varje nod vet vilken riktning dennes grannar är åt. Varje nod i rutten kan på så sätt ta reda på vilken riktning den kommer från och vilken riktning den ska till. Från dessa två riktningar beräknas vilken sväng som ska göras. Svängarna sparas sedan i en lista som sedan ska skickas som instruktioner till AGVerna.

För att undvika låsningar i systemet vid störningar är svänglistan gemensam för AGVerna, och ett direktiv måste genomföras innan nästa instruktion i listan får köras. Uträkningen av rutt är alltså tidsberoende, men utförandet tar ingen hänsyn till tid.



Figur 10: En förenklad beskrivning av hur AGVernas program är uppbyggt och exekveras.

Individuell styrning

Den individuella styrningen av AGVerna har designats så att AGVn skall bete sig olika beroende på underlagets färg. Genom att bygga upp vägnätet med svarta vägar med grå korsningar möjliggörs kontrollen av vad som händer när en AGV når en korsning. De fem IR-sensorerna är placerade på rad i framkant av AGVn. De får in värden mellan 0 och 1000, där 1000 är lägsta möjliga reflektion och vise versa. De tre inre sensorerna är de som används för att reglera positionen på vägen, och strävar vid linjekörning mot att hålla mittsensorn centrerad, oavsett svart eller grått underlag. De två yttre sensorerna används endast för att upptäcka skarpa svängar. När en sväng upptäcks roterar AGVn i svängens riktning till dess vägen är centre-

rad under de tre mittsensorerna för att sedan fortsätta sin normala körning. Skillnaderna beroende på underlagets färg är:

- Svart underlag: Om de tre mittsensorerna får in ett totalvärde som är större än 1000 tolkas det som att AGVn är på svart underlag. Under detta förhållande följer AGVn endast linjerna.
- Grått underlag: Ifall AGVn får in ett totalvärde som är mellan 100 och 1000 tolkas det som att underlaget är grått. När detta inträffar skickar AGVn en signal till basstationen, stannar och inväntar kommando. Den har nu tre möjligheter: sväng höger, sväng vänster eller kör rakt fram. Då kommandot är utfört och underlaget skiftat till svart skickas en signal att AGVn lämnat korsningen.
- Vitt underlag: Om sensorerna får ett värde under 100 roterar AGVn till den hittar tillbaka till vägen.

Basstation

Basstationen fungerar som länken mellan schemaläggningsprogrammet och AGV-enheterna. Dess uppgift är att filtrera relevant data och skicka det till schemaläggaren samt kontinuerligt uppdatera mottagarenheterna med ny data. Delar av detta görs direkt i wixel-enheten men stora delar görs i separat Javakod som kommunicerar med wixeln via USB.

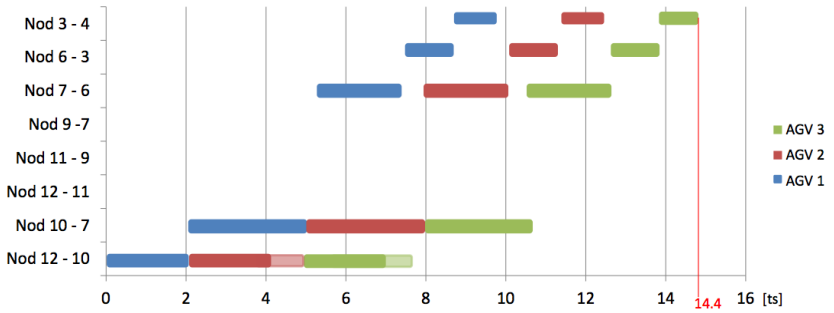
4 Resultat

Nedan redovisas de resultat som uppnåtts i projektet inom schemaläggning, kommunikation och fallstudien. Resultaten inom schemaläggning baserar sig på simuleringar med de två metoderna BAA och HSBA.

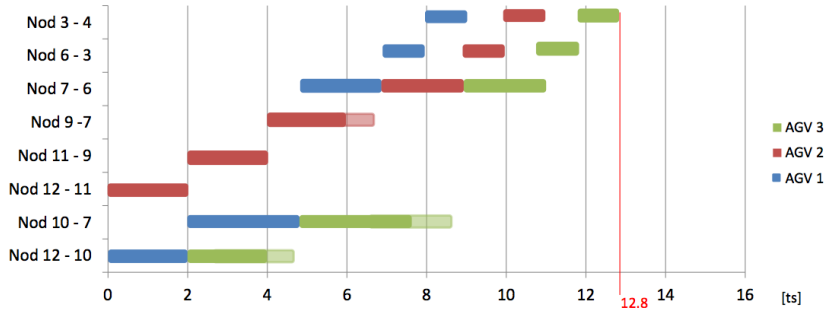
4.1 Optimering av systemtider

I följande exempel skall tre AGVer ta sig från nod 12 till nod 4 i vägnät 1, se Bilaga B. I detta exempel har samma rutt valts för samtliga AGVer för att garantera att väntetider kan uppstå. Denna sträcka tar för en enskild AGV 8.8 tidssteg [ts] att köra. Då systemet körs med HSBA kommer varje rutt beräknas enligt A*-algoritmen för att sedan schemaläggas allteftersom AGVerna kör. Alla AGVer kommer att välja samma väg vilket innebär att AGV_2 och AGV_3 kommer att få starta senare samt att väntetider uppkommer då vägsträckorna blockeras. Systemtiden blir 14.5 [ts] vilket är 5.6 [ts] längre än för en enskild AGV. Detta visas i Figur 11.

När systemet i stället körs med BAA kommer väntetider vägas in i kostnaderna för rutterna för att minska systemtiden. Detta leder till att AGV_3 kommer välja en annan rutt än den individuellt kortaste för att undvika väntetider. Det innebär också att AGV_2 kan starta direkt. Figur 12 visar att systemtiden nu har minskat till 12.8 [ts].

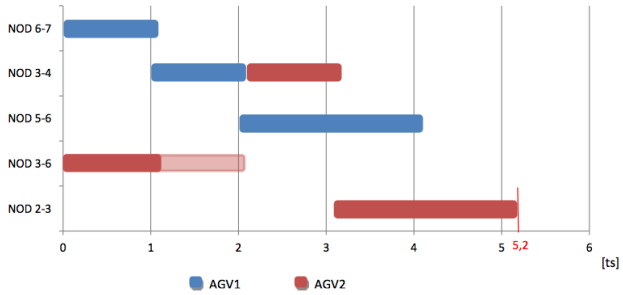


Figur 11: Bokning för AGVer med HSBA. Bokningar som är transparenta är väntetid.

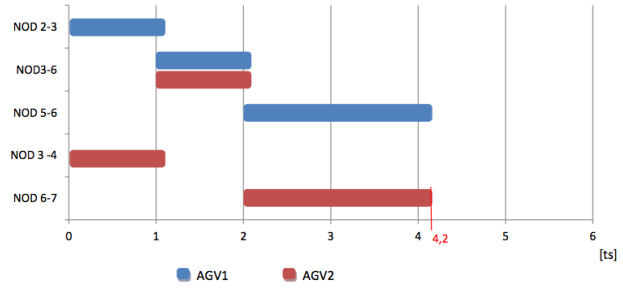


Figur 12: Bokning för AGVer med BAA. Bokningar som är transparenta är väntetid.

För att ytterligare minska systemets totaltid kan riktningsbokningar implementeras i BAA. Detta innebär att flera AGVer tillåts färdas på samma sträcka samtidigt under förutsättning att de färdas åt samma håll. I detta exempel färdas två AGVer varsin rutt i vägnät 1, se Bilaga B, med en gemensam väg, sträckan mellan nod 3 och 6. I Figur 13 ser vi att systemtiden blir 5.2 [ts] då bokningar sker utan riktningsbokningar. AGV_2 får en väntetid på sträckan mellan nod 3 och 4 då den måste vänta till AGV_1 avbokat sträckan mellan nod 3 och 6. I Figur 14 har riktningsbokningar används och systemtiden har minskats till 4.2 [ts].



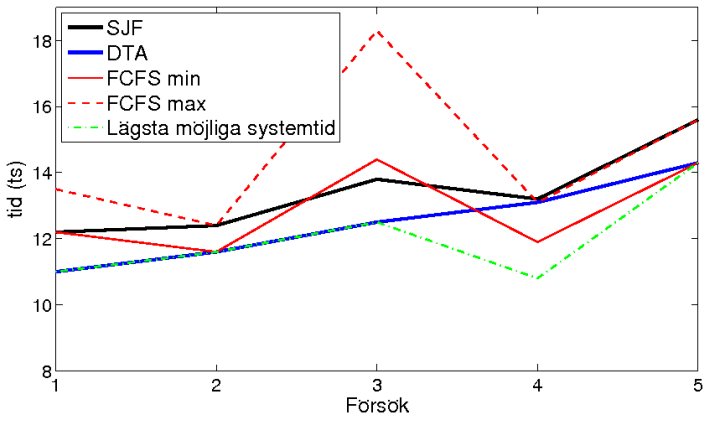
Figur 13: BAA utan riktningsbokning. Bokningar som är transparenta är väntetid. AGV_2 väntar mellan nod 3 och 4 till AGV_1 har avbokat sträckan mellan nod 3 och 6.



Figur 14: BAA med riktningsbokning. Sträckan mellan nod 3 och 6 är bokad för både AGV_1 och AGV_2 i samma tidssteg.

4.2 Prioriteringspolicyer

I följande exempel jämförs de tre prioriteringspolicyerna FCFS, SJF och DTA (se avsnitt 3.1.1). Samtliga AGVer färdas olika rutter. Varje rutt är lagd så att den korsar minst en annan rutt. I BAA är val av policy möjligt medan HSBA alltid körs med FCFS. Vid FCFS kan alla typer av policyer inträffa, även SJF och DTA. För att kunna jämföra de olika policyerna kommer dock de varianter då AGVerna noteras i en ordning som resulterar i SJF- eller DTA-prioritering att räknas bort från FCFS.

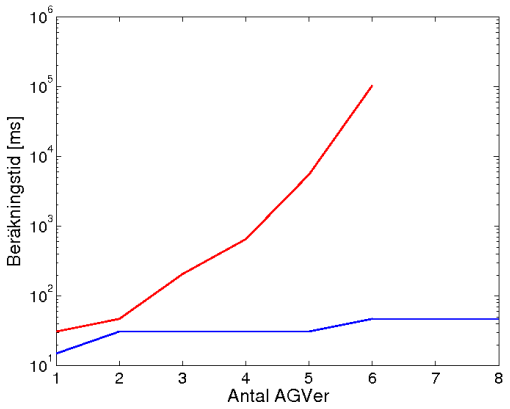


Figur 15: Jämförelse mellan prioritering enligt SJF, DTA och FCFS (möjliga kombinationer av prioritering borträknat SJF och DTA). Minsta möjliga systemtid är den individuella körtiden för den längsta rutten.

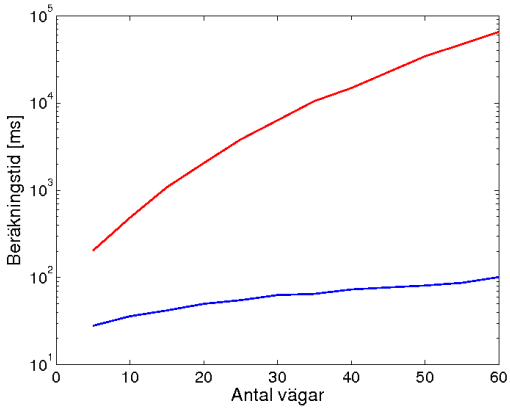
I Figur 15 noteras systemtiderna för fem olika försök med tre stycken AGVer. I försök 1, 2, 3 och 5, som körs i vägnät 1 (se Bilaga B), blir systemtiden med prioritering enligt DTA den samma som längsta individuella tid. Försök 4 körs i vägnät 2 (se Bilaga B) som på grund av sina få vägar kan skapa många väntetider eller tvinga lågt prioriterade AGVer att vänta med att starta sina rutter.

4.3 Beräkningstider

Vid användning av HSBA är tiden som det tar att räkna ut rutter och skapa individuella automater försumbart i jämförelse med tiden det tar att skapa den synkroniserade automaten. Tiden det tar att skapa den synkroniserade automaten ökar exponentiellt med antalet AGVer, vilket syns tydligt i Figur 16. Vid ökning av antalet vägar som ingår i varje AGVs rutt ökar tidskomplexiteten för HSBA polynomiellt, detta illustreras i Figur 17.



Figur 16: Logaritmen av beräkningstiden [ms] plottad för de två algoritmerna BAA (blå), HSBA (röd) mot antalet AGVer i systemet. Här syns tydligt att tiden för HSBA ökar exponentiellt.



Figur 17: Logaritmen av beräkningstiden [ms] plottad för de två algoritmerna BAA (blå), HSBA (röd) mot antalet vägar per schemalagd rutt. Observera att ingen av algoritmerna ökar exponentiellt när denna parameter ändras.

4.4 Kommunikation

Det kommunikationsprotokoll som beskrivs i avsnitt 3.2 har implementerats med hjälp av wixel-enheter, där en har fungerat som styrande enhet och resterande som mottagande enheter. Flera försök gjordes för att utvärdera kommunikationens hastighet samt stabilitet.

För att undersöka wixelns egenskaper gjordes ett enkelt försök med endast en mottagarenhet. Överföringshastighet beräknades till ca 7 [kB/s] vilket är väldigt nära den, enligt manualen, maximala överföringshastigheten på 10 [kB/s]. Försök gjordes för att undersöka stabiliteten hos dataöverföringen där en enhet skickade och en andra enhet lyssnade och räknade hur många datapaket den tog emot. Utav hundra skickade paket togs det i genomsnitt emot 95 stycken vid enkelriktad dataöverföring.

Det tidigare beskrivna protokollet implementerades och utvärderades. Detta gjordes genom att antalet mottagarenheter varierades och mätningar gjordes för att undersöka ifall detta skulle leda till sämre stabilitet eller lägre hastigheter. Resultaten från de mätningar som gjordes återfinns i Tabell 1.

Antal mottagarenheter	missade paket [%]	hastighet [kB/s]
1	3	7
2	4	6.5
3	3	6.5

Tabell 1: Resultat av försök med kommunikationsprotokoll implementerat med wixel-enheter.

4.5 Fallstudie

Den fallstudie som projektet utfört har mestadels givit önskvärda resultat. Körning med flera AGVer simultant fungerade. Viss dataförlust förekom, detta löstes med hjälp av införandet av att data skickades till dess att mottagandet bekräftats.

Vid körning av AGVerna har problem påträffats med IR-sensorerna. IR-sensorerna kalibrerar sig så att den identifierar svart, grått och vitt bra vid de ljusförhållanden som råder på den platsen. Detta gör att om AGVerna t.ex. åker från ett skuggat område till ett upplyst kommer sensorerna få andra utslag än tidigare. AGVn tolkar därför färgen på underlaget felaktigt vilket har lett till problem vid körning.

Implementationen av BAA i fallstudien fungerade. I vissa fall misslyckades schemaläggaren att förhindra kollisioner.

Körning med hjälp av HSBA som schemaläggare fungerade inte som väntat. AGVerna kunde beräkna rutt och boka vägar men avbokningar fungerade inte. Om flera robotar skall besöka samma väg kommer vägen inte vara tillgänglig efter första besöket. Problemet ligger inte i schemaläggningen utan i samspelet mellan kommunikationen och schemaläggaren.

5 Diskussion

I detta kapitel förs en diskussion kring de resultat som projektet genererat. Även de metoder som använts diskuteras samt för- och nackdelar belyses. Diskussionen utgår från simulering av de båda schemaläggnings metoderna.

5.1 Optimering av systemtider

I de fall systemet önskas optimeras med avseende på systemtiden har BAA en klar fördel då rutterna bestäms efter kortast systemtid istället för, som med HSBA, kortast individuella tid. Möjligheten att bestämma bokningsprioritet är också en stor del för att minska systemtiderna. Detta kan enbart göras med BAA eftersom HSBA är händelsestyrt.

Uppkomsten av situationer där låsningar eller störningar i systemet uppkommer är viktigt i projektet för att kunna visa hur systemet löser dessa problem. Om riktningsbokningar används uppstår dessa situationer mer sällan. På grund av detta har riktningsbokning inte fått en central roll i projektet. Riktningsbokning ger även upphov till en rad nya problem. För att undvika att AGVer blir osynliga för systemet när de färdas på riktningsbokade sträckor måste möjlighet att bilda köer av bokningar som överlappar varandra i tiden finnas. Detta gör att vägen fortfarande är bokad åt rätt riktning även då den första AGVn avbokat sträckan. Ett annat möjligt problem är att fler AGVer samtidigt vill färdas på sträckan än vad som fysiskt får plats på vägen. Önskas en optimering av systemets totaltid är riktningsbokningar ett effektivt sätt att minska tiden, förutsatt att dessa problem löses.

5.2 Prioriteringspolicyer

Jämförelsen, i Figur 15, mellan prioritering enligt SJF, DTA och FCFS visade att DTA gav lägst systemtid i fyra av fem fall. Systemtiden blev i dessa fall den samma som den längsta individuella ruttiden, vilket är en undre gräns

för den optimala systemtiden. I det fall då DTA inte var den bästa prioriteringsmetoden kördes systemet i vägnät 2 som är ett mycket litet vägnät. Detta gjorde att några AGVer tvingades vänta till högre prioriterade AGVer hade kört färdigt vilket resulterade i den långa systemtiden. Utifrån detta anses DTA-policyn vara lämpligast då systemtiden skall minimeras.

5.3 Beräkningstider

Det största problemet som uppkom vid användandet av HSBA var skapandet av synkroniserade automater för system med många AGVer. Antalet tillstånd i den färdigsynkroniserade automaten ökar exponentiellt när antalet AGVer ökar. Antalet vägar i de rutter AGVerna tar spelar också roll men komplexiteten ökar inte lika kraftigt. Beräkningarnas komplexitet gör att tiden det tar att skapa automaten blir väldigt lång och även minnesproblem för datorn kan uppkomma. Att skapa en automat behövs dock bara göras en gång vid varje uppstart och det finns möjlighet att spara ner automater som går att använda senare. I vårt fall kan systemets storlek minskas genom att utesluta AGVer ur synkroniseringen om de inte delar vägar med de andra AGVerna. Denna AGV kommer alltid att ha tillåtelse att boka sina vägar. Även vägsträckor som endast bokas av en AGV kan uteslutas ur synkroniseringen.

5.4 Robusthet

I BAA bokas AGVernas rutter redan vid systemstart och den gemensamma händelsesekvensen för samtliga AGVer är därmed bestämd. HSBA utgår istället från hur situationen ser ut i varje tidssteg. Om en störning uppkommer då AGVerna körs efter BAA kan en låsning uppstå. Körs AGVerna med HSBA kan de AGVer som inte påverkas av störningen fortsätta sina rutter. Detta gör HSBA mer robust mot störningar.

I det diskreta systemet modelleras händelser som momentana, det vill säga att tiden de tar inte är definierad utan att händelsen kan ske mellan två diskreta tidssteg. En bokning är momentan och i och med den diskretiserade tidsaxeln kan flera bokningar ske i samma diskreta tidpunkt. Detta medför att två AGVer i BAA kan byta vägsträckor i ett enda tidssteg. AGVerna blir osynliga i noderna (korsningarna i det fysiska vägnätet) och det uppstår risk för kollision. För att förhindra detta infördes en kort tilläggstid i slutet av varje bokning. Den förlängda bokningen gör att AGVn hinner köra igenom efterkommande korsning och in på en ny väg innan en annan AGV kan boka sträckan. Dessutom kontrollerades hur länge AGVn kan vara kvar på den aktuella vägen innan den måste flyttas för en annan AGV. Dessa åtgärder löste delvis problematiken med momentana bokningar. Problemet kvarstår

dock när en AGV som har högst prioritet skall åka in på en väg som en annan AGV står och väntar på. Detta kan lösas med en kontroll huruvida bokningen av den väg AGVn med lägre prioritet önskar åka på går till den nod där den måste vänta. Om så är fallet måste AGVn beräkna en ny rutt för att inte blockera AGVn med högre prioritet. Detta innebär att lägre prioriterade AGVer väljer längre rutter för att förhindra blockering av högre prioriterade AGVer. En annan möjlighet för att undvika låsningar eller att AGVer blir osynliga i korningar skulle vara att utföra schemaläggningen med dynamisk programmering.

5.5 Kommunikation

Kommunikationsprotokollet som tagits fram i projektet är designat för att maximera mängden data som kan skickas ut till varje enhet. För att uppnå detta utnyttjas möjligheten att samtliga mottagarenheter kan ta emot ett och samma datapaket för att sedan på egen hand filtrera ut vilken data som är relevant för just den enheten. I och med detta växer datamängden i varje försändelse från basstationen, men med den överföringshastigheten som uppnåtts är detta inte en begränsande faktor. Ett tillfälle då detta skulle kunna skapa mer problem än det gör nytta är ifall systemet har en så stor mängd AGVer att tiden det tar att föra över datan till AGVerna gör att systemet inte uppdateras tillräckligt ofta. Detta kan dock lösas på flera olika vis, till exempel kan subsystem införas som opererar på en annan frekvens och enbart kommunicerar med resterande delar av systemet genom en basstation.

Protokollet som utvecklats skiljer sig på väldigt få punkter från traditionell polling [8]. Den absolut största skillnaden ligger i den ökade dataöverföringen till mottagarenheterna som nämns ovan. Vid polling frågas varje enhet kontinuerligt ifall de har något att skicka. Om enheten önskar skicka något tilldelas den tillåtelse att använda kommunikationskanalen. Detta skiljer sig från det utvecklade protokollet där enheten direkt skickar den data den har vid förfrågning. Det här tillvägagångssättet var av fördel eftersom systemets datapaket var oerhört små, som störst 5 byte, vilket gjorde det överflödigt och ineffektivt att först fråga om det fanns något i paketet.

Protokollet som tagits fram är designat för att hantera just den situationen projektet innebar, d.v.s. ett fåtal enheter samt minnesmässigt små datapaket. Under dessa omständigheter är fördelarna gentemot polling uppenbara, fler uppdateringar av mottagarenheterna samt färre icke-överförande kommunikationer. Protokollet har nackdelen att det är väldigt specialanpassat för en specifik situation till skillnad från polling som är oerhört mångsidigt och flexibelt. Kommunikationen skulle även kunna köras utan pollning, där basstationen ständigt väntar på invärden och övriga enheter kan skicka när

de vill. Detta leder visserligen till snabbare kommunikationstider, eftersom enheten inte behöver vänta på sin tur innan den kan skicka. Dock är ett sådant system mer känsligt för störningar då flera enheter kan skicka samtidigt och störa ut varandras signaler.

Implementationen och testningen av kommunikationsprotokollet visade potential för både höga hastigheter samt robust överföring. Om man jämför de resultat som erhölls när två wixlar kommunicerade syns inga större förändringar i varken hastighet eller stabilitet mot när fyra wixlar sätts i systemet. Dock uppkommer en liten nergång i överföringshastighet när antalet wixlar ökades vilket mycket troligt beror på att andra partier i processorernas exekvering tar mer tid då kommunikationen skall utföras mellan flera enheter. Säkerheten och stabiliteten av kommunikationen förändras inte märkbart då man ökar antalet enheter, åtminstone inte i den utsträckning som testats.

5.6 Fallstudie

Problemen som uppkommit vid implementeringen av systemet av AGVer har till stor del varit kopplat till hårdvara. Förlusterna i kommunikationen är inte större än de förluster som uppmättes då enbart ett par av wixel-enheterna kommunicerade, vilket indikerar att detta problem härstammar från hårdvaran. Svårigheterna med att AGVerna uppfattar färgen på underlaget fel beror på den känslighet till olika ljusnivåer som IR-sensorerna tydligt uppvisar.

5.7 Miljömässig hållbarhet

I projektet har mycket fokus lagts på att tidsoptimera systemen. En annan aspekt är att optimera systemet med hänsyn till energiåtgång. Projektets resultat är inte direkt applicerbara för energisnåla system men indikerar att det finns möjligheter att minska energiåtgången för systemet. Om AGVernas hastigheter sänks så att väntetiderna övergår i körtid skulle energiåtgången minska eftersom kraftiga inbromsningar och acceleration som är väldigt energikrävande undviks [12]. Att, som i BAA, välja längre vägar så länge de minskar systemtiden är inte alltid lämpligt för att minimera energiåtgången. Att ta bort väntetidstillägget och införa riktningsbokningar är, ur detta perspektiv, en bättre lösning.

5.8 Validitet

I simuleringsresultaten tenderar systemets sluttid att bli den samma som den längsta individuella ruttiden. Denna tid är även en undre gräns för den globala systemtiden. Att dessa två tider är samma tyder på att den globala

systemtiden är optimal. BAA garanterar emellertid inte denna typ av optimalitet, dock blir sluttiden för varje enskild AGV optimal. För de flesta fall som studerats verkar dock BAA vara optimal när det gäller sluttid för det globala systemet.

5.9 Generaliserbarhet

Vid en framtida vidareutveckling och implementation av de två schemaläggningsalgoritmerna gäller det att ha vissa saker i åtanke. För verkliga system med stor komplexitet och många fordon, såsom högtrafikerade storstadskorsningar, kommer HSBA inte fungera då beräkningskomplexiteten ökar exponentiellt. En lösning på detta problem är att partitionera systemet i mindre delsystem. Därmed planeras inte varje fordons rutt längs hela tidshorisonten och på så vis hålls beräkningskomplexiteten nere. När det gäller BAA är den känslig för störningar eftersom den schemalägger händelser i en diskret ordning. Små variationer i verkligheten kan göra BAA till en ineffektiv metod då ett visst fordon kan behöva vänta på ett annat i onödan och därmed inte få sin handling tillgänglig. Detta kan lösas genom att BAA kontinuerligt uppdateras och i realtid räknar om schemaläggningen.

6 Slutsats

I projektet har de två schemaläggningsmetoderna BAA och HSBA jämförts. BAA utmärks av korta beräknings- och systemtider men lider av stor känslighet för naturliga variationer. DTA var den prioriteringspolicy som resulterade i lägst systemtider för BAA. HSBA fungerar bra för implementering i system där störningar är sannolika, dock ökar algoritmens beräkningskomplexitet polynomiellt med antal vägar och exponentiellt med antalet AGVer vilket kan bli ett problem i större system. Önskas implementering av HSBA i större vägnät krävs att dessa kan partitioneras till mindre delsystem. Kommunikationsprotokollet som framarbetats visar på robust samt snabb och effektiv dataöverföring.

Genom att reglera AGVernas hastighet kan väntetid övergå i körtid och kraftiga accelerationer förhindras. På detta och liknande sätt kan energiåtgången i systemet minskas, därmed ökar systemets miljömässiga hållbarhet.

Det finns flera möjligheter att fortsätta studierna kring både den praktiska och teoretiska delen av arbetet. Att jämföra och utveckla fler algoritmer för schemaläggningen är ett i princip outömligt område för vidareutveckling. Ett sätt att minska systemtiden kan vara att rutterna beräknas i realtid med hänsyn till vilka vägar som är tillgängliga. Implementering och analys av mer

avancerade kollektiva uppdrag för AGVerna i systemet är en både teoretisk och praktisk utökning. Vid en fortsatt utveckling av den praktiska studien rekommenderas ej linjeföljning med hjälp av IR-sensorer som grundläggande orienteringssystem, då detta har visat sig instabilt i ljusskiftande miljöer.

Referenser

[1] Bowen, L. (2000) *Introduction to Contemporary Mathematics, Scheduling Algorithms*. http://www.ctl.ua.edu/math103/ (12-05-2012).

[2] Cassandras, C. G. och Lafortune, S. (2008). *Introduction to Discrete Event Systems*, New York: Springer.

[3] Kurose, J., Schwartz, M. och Yemini, Y. (1984). Multiple-access protocols and time-constrained communication, *ACM Computing Surveys (CSUR)*, nr 16, s.43-70.

[4] Pearl, J. (1984). *Heuristics intelligent search strategies for computer problem solving*. Michigan: Addisson-Wesley.

[5] Pololu Corporation. (2011). *Pololu m3pi User's Guide*. http://www.pololu.com/docs/pdf/0J48/m3pi.pdf (07-05-2012).

[6] Pololu Corporation. (2011). *Pololu Wixel User's Guide*. http://www.pololu.com/docs/pdf/0J46/wixel.pdf (12-05-2012).

[7] Qui, L., Hsu, W., Huang, S. och Wang, H. (2002). Scheduling and routing algorithms for AGVs: a survey. *International Journal of Production Research*, vol 40, nr 3, ss. 745-760.

[8] Rom, R. och Sidi, M. (1990). *Multiple Access Protocols: Performance and Analysis*, New York: Springer-Verlag.

[9] Supremica (2007) http://www.supremica.org (2012-05-07)

[10] Trudeau, R. J. (1994). *Introduction to Graph Theory*, New York: Dover Publications.

[11] Weiss, M. A. (2007). *Data structures and Algorithm Analysis in Java*. 2nd ed. Florida: Peason Addison-Wesley.

[12] Wigstrom, O., Lennartson, B., Vergnano, A. och Breitholtz, C. (2012). *High level scheduling of energy optimal trajectories*, Accepted for publication in IEEE Transactions on Automation Science and Engineering.

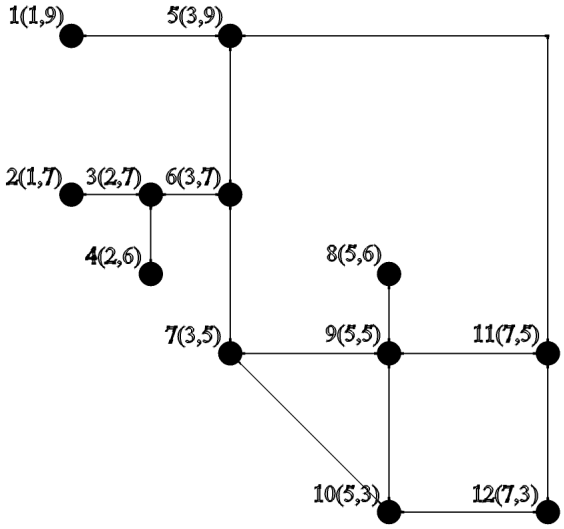
Bilagor

A Tabeller

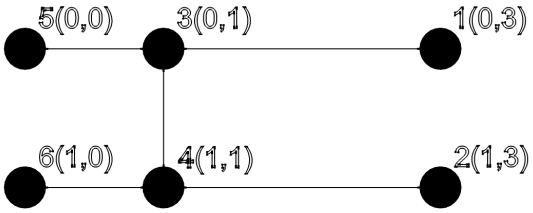
Beskrivning	**Önskad prestanda**
Hastighet AGV	
Medelhastighet	>0,2 [m/s]
Kommunikation	
Överföringshastighet	>0,5 [kB/s]
Antal kommunikationer / sekund	>50 [st]

Tabell 2: Specifikation för de prestandamål som satts upp för systemet.

B Vägnät



Figur 18: Vägnät 1. Noder betecknas med nummer samt koordinater inom parantes.



Figur 19: Vägnät 2. Noder betecknas med nummer samt koordinater inom parantes.

C Flödesschema för BAA