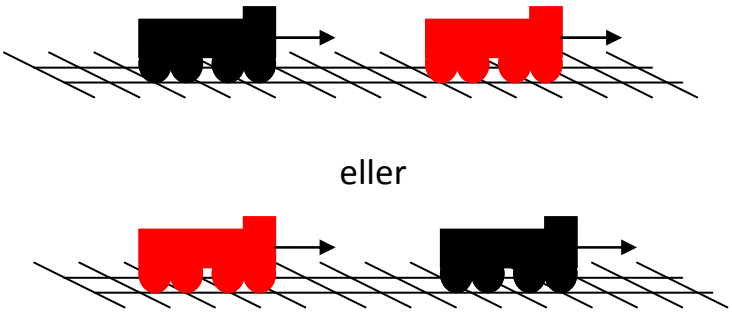
SLUTRAPPORT:

För projektet *Effektiv operativ omplanering av tåglägen vid driftstörningar* (EOT) med Dnr TRV 2010/29603



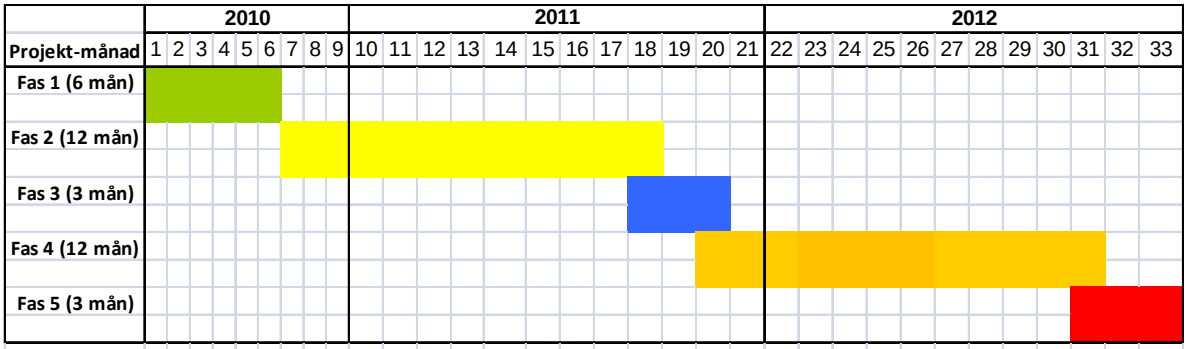
Johanna Törnquist Krasemann – johanna.tornquist@bth.se
Blekinge Tekniska Högskola
Karlskrona, 2012-11-29
www.bth.se/com/eot

Sammanfattning

Projektet "Effektiv operativ omplanering av tåglägen vid driftstörningar" (förkortas EOT) bygger vidare på arbetet som utfördes i de tidigare forskningsprojekten OAT och OAT+ (huvudsakligen finansierade av dåvarande Banverket) . Syftet med projekten är att utveckla beräkningsstöd som underlättar trafikledarnas arbetsuppgifter med avseende på framförallt omplanering av trafiken vid störningar. Projekten kan ses som komplement till projekten FTTS/STEG/STRATEG. Forskningsarbetet i EOT består av två huvudsakliga delar som syftar till att:

- Vidareutveckla och förbättra de metoder som togs fram i projektet OAT+.
- Utvärdera och studera beräkningsmetodernas prestanda från ett praktiskt perspektiv under driftslika förhållande och då i samverkan med framför allt Trafikverket.

Projektet har varit uppdelat i olika faser enligt tabellen nedan.



Fas 1	Mål: Etablera grundförutsättningar för att genomföra projektet,
Fas 2	Mål: Initial parallellisering av optimeringsalgoritm för omplanering av tåglägen
Fas 3	Mål: Förankring hos och återkoppling från relevant personal vid Trafikverket
Fas 4	Mål: Förbättring och vidareutveckling av parallella optimeringsalgoritmer samt utvärdering i samverkan med TrV
Fas 5	Mål: Kunskapsöverföring och diskussion om fortsatt utveckling och slutrapportering.

Arbetet i Fas 2 avsåg att vidareutveckla och förbättra den algoritm som BTH tidigare utvecklat inom OAT+ samt att implementera en parallelliserad version av denna (se kapitel 6). Algoritmen syftar till att omplanera tågtrafiken vid behov, vilket i vårt fall uppstår när tågtrafiken i drift avviker (eller man misstänker att den kommer att avvika) från tidtabellen i sådan omfattning att den måste omplaneras. Den är en s.k. girighetsalgoritm och eftersom den måste hitta en tillräckligt god första omplaneringslösning på ett fåtal sekunder är den därför av typen *djupet-först* (Depth-First Search, DFS). Detta innebär att först ska girighetsalgoritmen snabbt hitta en första bra tillåten omplaneringslösning och om det sedan finns tid kvar så försöker den hitta förbättrade alternativ till denna första lösning.

Att parallellisera en algoritm innebär att man skapar flera, snarlika kopior av algoritmen, vilka parallellt och beroende/oberoende av varandra löser problemet/uppgiften. Man kan se den principiella skillnaden mellan den sekventiella girighetsalgoritmen och den parallella versionen som att i den sekventiella sitter det *en trafikledare* som försöker lösa situationen ensam medan i den parallella versionen sitter det *många fler trafikledare* som arbetar med

samma problem parallellt och (delvis) oberoende av varandra. Trafikledarna försöker skapa ett antal *olika* omplaneringslösningar och utbyter information med varandra om vilka omplaneringsåtgärder som är bra och kan leda till en bra lösning samt vilka som är mindre bra. Därmed blir själva jakten på lösningar mer effektiv och det skapas också fler alternativa och kompletterande lösningar som de riktiga trafikledarna hos Trafikverket kan ta ställning till och modifiera vid behov. En central utmaning är då att undvika att dessa parallella beräkningar leder till likadana lösningar. Istället vill man effektivt kunna identifierar de bästa alternativa lösningarna och beräkna effekterna av dessa. En annan utmaning är att skapa ett effektivt informationsbyte mellan de parallella processerna som inte hämmar själva beräkningarna.

Den första fasen av utvecklingsarbetet och utvärderingen av den parallella algoritmen slutfördes under oktober 2011 och beskrivs i kapitel 6.3. Vi testade den parallella algoritmen på tre olika typer av störningssituationer (20 scenarier totalt):

(1) är en enskild initial störning hos ett visst tåg (som sedan ev. sprider sig till andra tåg),
(2) är en signifikant hastighetsnedsättning hos ett enskilt tåg medan
(3) är ett infrastrukturfel på linjen som medför att alla passerande tåg får en signifikant hastighetsnedsättning.

Den tredje typen – *infrastrukturfel* – är svårast att lösa med såväl den sekventiella som den parallella algoritmen. Med *infrastrukturfel* avser vi situationer som kan ge upphov till exempelvis en hastighetsnedsättning på en viss linjesträcka och som resulterar till att samliga tåg som passerar får ökade körtider. Infrastrukturfel kan också vara att ett av spåren på en dubbelspårsträcka är avstängt pga. t ex rälsbrott/oplanerat banarbete. Anledningen till att störningar som är av typen infrastrukturfel är svåra att lösa matematiskt är att det uppstår kraftig köbildning om trafiken är tät och som då påverkar många tåg. Därmed finns det många olika relevanta lösningsalternativ att utvärdera. Den teoretiska analysen av algoritmens prestanda visade därmed att den behövde förbättras ytterligare i särskilt svåra scenarier. Vi valde därför initialt att fokusera på dessa störningsscenarier vid vidareutvecklingen av den parallelliserade algoritmen. I Fas 4 genomfördes ytterligare förbättringar samtidigt som vi utförde experiment på en större mängd scenarier (100 stycken) för att synliggöra algoritmens svagheter (se kapitel 6.3-6.6). Experimenten har alla baserats på trafik och data för 2009 även om också trafiken under 2010-2012 har studerats i stor utsträckning. I Fas 4 var det initialt planerat att även utföra experiment baserat på utvalda trafikscenarier från 2011 men eftersom arbetet i denna fas till viss del fördröjdes (se nedan) så genomfördes dessa experiment inte fullt ut.

I Fas 4 planerade vi även att analysera och utvärdera hur algoritmen skulle fungera i ett mer praktiskt sammanhang. Syftet med Fas 3 var därför att etablera kontakt och samverka med relaterade projekt internt på Trafikverket. Genom våra kontaktpersoner på Trafikverket fick vi kontakt med medverkade från projektet NTL (Nationell TågLedning) och hade ett inspirerande och mycket givande möte med flera från projektgruppen i Göteborg under hösten 2011. Eftersom de båda projekten har flera beröringspunkter enades vi om att NTL-projektet vid behov skulle ge synpunkter på EOTs arbete med den praktiska utvärderingen och underlätta insamlandet av trafikdata. Under mötet diskuterades möjliga alternativ för att få utvärderingen av algoritmens prestanda under så driftslika förhållande som möjligt. Efter mötet tog BTH fram förslag på ett antal relevanta utvärderingskonfigurationer (se kapitel 7)

med tillhörande specifikation av informationsbehov och dessa diskuterades med medarbetare i NTL-projektet. Dessvärre fick EOT beskedet i februari (2012) att det är svårt att ge välstrukturerad realtidsdata till EOT-projektet utan att Trafikverket behöver göra större insatser själva i närtid. Det skulle kräva speciallösningar internt, mycket pga. t ex avsaknad av exportmöjlighet av relevant data från olika applikationer. Vi avsatte därför en summa av projektbudgeten i Fas 4 och 5 som skulle kunna användas för att finansiera den insats vi skulle kunna komma att behöva från Trafikverket. Under våren och sommaren 2012 diskuterade vi problemen med att realisera den praktiska utvärderingen under 2012 utan att hitta lämpliga lösningar. Vi insåg att vi hade underskattat svårigheten och tidsåtgången att skapa förutsättningar för och ”rigga” en dylik praktisk utvärdering. Projektets tidsplan och leveransplan för Fas 4 fördröjdes därmed, vilket ledde till att Trafikverket fick en förfrågan från BTH i oktober 2012 om förlängning av projektet t.o.m. september 2013 och som beviljades. Arbete under 2013 syftar därmed till att skapa goda förutsättningar för att i ett senare skede utföra en praktisk utvärdering av projektets beräkningsmetoder. Således har projektet vid denna tidpunkt (november 2012) inte kunnat leverera resultat från en praktisk utvärdering och därmed inte heller fullt ut kunnat anpassa beräkningsmetoderna i linje med de rekommendationer som utvärderingen skulle visat. Vissa analyser med avsikt att undersöka potentiella problem och frågeställningar som rimligtvis kommer att aktualiseras har dock genomförts. Dessa har huvudsakligen baserats på observationer via Trafikbilder och Opera samt material från STEG-projektets arbete. En uppenbar frågeställning som är viktig att beakta men som inte går att svara på i nuläget är t ex på vilken detaljnivå och med vilken noggrannhet (i tid) som tågen ska omplaneras och hur ska tågrörelser inne på mer komplexa stationer bör hanteras. I tidtabellskonstruktionen planerar man ju inte i detalj för hur trafiken ska interagera på stationerna, vilket då leder till att det saknas tillförlitlig data om hur alla tåg enligt plan ska trafikera stationerna. Därmed är tidtabellen inte garanterat konfliktfri i det avseendet (se kapitel 7 för vidare diskussion).

För att summera: Syftet med forskningsarbetet i EOT-projektet är att utveckla beräkningsmetoder som snabbt och effektiv kan skapa förslag på hur tågtrafiken bör omplaneras vid behov och som stöd för trafikledarna. Förslagen bör tas fram snabbt (på 10-30 s) och de bör vara så optimala som möjligt med avseende på de mål och kriterier som trafikledningen ska arbeta efter. Förslagen ska också vara trafiktekniskt lämpliga och genomförbara. Fokus på forskningsarbetet har varit att få beräkningsmetoderna att snabbt leverera förslag som minimerar trafikens totala försening men effekterna i form av antalet försenade tåg, fördelning av försening mm har också beaktats. Vi har ännu inte analyserat i detalj vilka prioriteringar mellan olika tåg som görs och om ”rätt” tåg hela tiden prioriteras. Detta blir en del av fortsatt arbete i samverkan med trafikledare och relaterade projekt såsom ”Operativa prioriteringskriterier”, STEG och NTL.

Det är de underliggande modellerna av trafikflödet som ska säkerställa att förslagen är trafiktekniskt genomförbara. Det är dock alltid en avvägning på vilken detaljnivå olika komponenter ska beskrivas och projektet har studerat olika nivåer av detaljrikedom. Eftersom gångtiderna i trafiken är relativt approximativa finns det ingen poäng i att planera trafiken på sekundnivå. Däremot finns det ett behov av större noggrannhet ju närmare tiden för exekvering planen kommer samt för mer komplexa stationer och knytpunkter. Man behöver också skilja på 1) den modell och beskrivning av trafiken man använder när man söker efter omplaneringsförslag och 2) den beskrivning man använder för att eventuellt detaljplanera, lägga tågvägar och säkerställer att den grövre planen från 1) håller. Hur dessa

båda nivåer bör samverka och effekten av detta är svår att utreda och besluta om på en generell nivå utan beror på det specifika fallet och de krav användaren ställer. I projektet har vi försökt lyfta och studera dessa aspekter men utan en mer omfattande praktisk utvärdering och diskussioner med presumtiva användare är det svårt att dra några generella slutsatser. Det blir istället del av fortsatt arbete som till viss del diskuteras och beskrivs i kapitel 8.

1 Inledning

Projektet *EOT* (Effektiv Operativ omplanering Av Tåglägen vid driftstörningar) pågår under tiden 2010-04-01—2013-09-30. Eftersom projektet formellt skulle avslutas enligt plan 2012-12-31 men blivit beviljat en förlängning till 2013-09-30, så har denna slutrapport sammanställts under november 2012 i linje med den ursprungliga planen. En kompletterande slutrapporteringen görs dessutom september 2013.

Projektet finansieras av Trafikverket där projektsponsor har varit Hans Dahlberg och senare Robin Edlund. Kontaktpersoner är Magdalena Grimm och Åke Lundberg. Vi vill rikta ett varmt tack till alla personer som direkt och indirekt deltagit och deltar i projektet och bidragit med tid, information och data och då framför allt medarbetare på Trafikverket.

Projektarbetet som beskrivs har utförts av projektledare Dr. Johanna Törnquist Krasemann, Prof. Håkan Grahn, doktorand Syed Muhammad Zeeshan Iqbal samt projektassistent Sara Soltani (oktober 2011-juni 2012) vid Blekinge Tekniska Högskola.

Denna slutrapport avser summera väsentliga delar av genomfört arbete och kompletteras av de publicerade artiklar och rapporter som återfinns som bilagor. Projektet har en hemsida www.bth.se/com/eot där relevanta publika dokument även finns tillgängliga.

Rapporten inleder med en presentation av bakgrunden till projektet och tidigare forskning. I Kapitel 4 beskrivs projektets syfte och omfattning. I Kapitel 5 följer en kortfattad analys och utvärdering av relaterad forskning medan Kapitel 6 och 7 ger en övergripande beskrivning av projektarbetet och dessa resultat. Kapitel 8 innehåller en diskussion av projektets resultat och förslag på fortsatt arbete. Kapitel 9 beskriver hur projektets frågeställningar och resultat har presenterats och kommunicerats i såväl forskningssammanhang som för samhället och branschen.

2 Bakgrund

Ett attraktivt och hållbart järnvägssystem ska hålla hög säkerhet, vara miljövänligt, ha god tillgänglighet och erbjuda pålitliga transporttjänster som håller tillräckligt hög tidsprecision. Samtidigt ska bankapaciteten utnyttjas i möjligaste mån på ett kostnadseffektivt sätt. Detta kräver en kontinuerlig balans mellan att skapa ett högt banutnyttjande och att bibehålla en tillräckligt hög robusthet, dvs. liten känslighet för störningar. Analyser av kapacitetsutnyttjandet på det svenska järnvägsnätet utförs kontinuerligt av Trafikverket. I den senaste analysen, se (Grimm och Wahlborg, 2012), tydliggörs hur tågtrafiken har ökat markant de senaste åren. Bland annat har transportarbetet ökat till 144 703 miljoner tågkm (2011) från 133 526 miljoner tågkm (2009). Den största ökningen (mätt i antal tågkm) har skett för resandetåg som utgör 96 361 miljoner tågkm (2011). Den höga belastningen på infrastukturens centrala delar som uppkommer under högtrafik åskådliggörs även i de grafer där rödmarkerade sträckor representerar bandelar där kapacitetsutnyttjande har nått sin maximala gräns. Exempel på sådana sträckor är Norrköping-Linköping och Hässleholm-Arlöv, vilka båda är delsträckor av Södra Stambanan.

När kapacitetsutnyttjandet är mycket högt så är det allmänt känt att små avvikelser från den planerade tidtabellen lätt kan få omfattande konsekvenser och många tåg merförsenas. Många orsaker till störningar kan förebyggas, vilket är något man arbetar aktivt med men oförutsedda händelser som skapar störningar är ofrånkomliga och konsekvenserna ska då kunna approximeras, minimeras samt kommuniceras till användarna. Värdefull information som ligger till grund för en god konsekvensanalys av en driftstörning och revidering av tågplanen finns dock inte alltid trafikledaren tillhands. Med nuvarande förutsättningar finns heller inte alltid en möjlighet att använda tillgänglig information eftersom trafikledaren på mycket kort tid måste agera och utan beslutstöd.

3 Om tidigare forskning

I de tidigare forskningsprojekten Omplanering Av Tåglägen (OAT, www.bth.se/tek/oat) och fortsättningsprojektet OAT+ vilka båda finansierats av f.d. Banverket, har matematiska modeller som beskriver hur trafiken kan omplaneras utvecklats. Projekten utvecklade även optimeringsbaserade metoder för att snabbt ge trafikledaren förslag på hur tågplanen kan revideras vid behov, baserat på relevant information om förutsättningarna och utefter de trafikstyrningsmål Trafikverket (TrV) har satt upp.

I OAT studerade och utvecklade vi i huvudsak optimeringsbaserade modeller och metoder för den operativa omplaneringen. Vi utförde en experimentell studie av hur olika typer av störningar sprider sig samt hur de kan lösas med hjälp av de utvecklade metoderna. I merparten av scenarierna erhöll vi en första lösning (dvs. ett förslag på en reviderad tidtabell) inom ett fåtal sekunder medan i mer komplicerade fall tog det längre tid att hitta en optimal lösning om en sådan överhuvudtaget kunde hittas. Målet är dock att en metod för operativ omplanering snabbt kan tillhandahålla en giltig, lämplig lösning även i de mest tidskritiska och mycket komplicerade situationerna.

Fortsättningsprojektet, OAT+, fokuserade därför på utvecklingen av en optimeringsalgoritm - som komplement till att tillämpa optimeringsmjukvara såsom CPLEX - för att säkerställa att vi alltid hittar en tillräckligt god lösning inom relativt kort tid (inom 30 sekunder) och oberoende av vilken typ av störningssituation som uppträder. Den algoritm som utvecklats är en s.k. girighetsalgoritm (Eng. *Greedy algorithm*) och eftersom den måste hitta en tillräckligt god första omplaneringslösning på ett fåtal sekunder är den av typen *djupet-först* (Depth-First Search, DFS). Detta innebär att först ska algoritmen snabbt hitta en bra tillåten omplaneringslösning och om det sedan finns tid kvar så försöker den hitta förbättrade alternativ till denna första lösning.

Inom OAT+ utförde vi en mängd datorbaserade experiment för att studera hur trafiken kan omplaneras under driftstörningar och då med hjälp av girighetsalgoritmen. Vi tillämpade algoritmen på tre olika typer av störningskategorier (20 scenarier totalt):

Kategori 1 innebär att ett tåg anländer till trafikdistriktet med en viss försening eller att det får en temporär försening på ett segment inom distriktet.

Kategori 2 innebär att ett tåg har en mer ”kronisk” försening såsom nedsatt accelerationsförmåga vilket resulterar i ökade gångtider.

Kategori 3 syftar på ett infrastrukturfel som ger upphov till exempelvis en hastighetsnedsättning på en viss linjesträcka och som resulterar till att samliga tåg som passerar får ökade gångtider.

För att utvärdera algoritmens prestanda har vi även löst scenarierna med motsvarande optimeringsmodell och tillämpat den kommersiella optimeringsmjukvaran IBM ILOG CPLEX Optimizer version 12.2 (med en maximal beräkningstid på 24 h). Vid en utvärdering av algoritmens prestanda och jämförelse med CPLEX kan vi först och främst se att CPLEX i merparten av fallen inte har kunnat generera en tillåten lösning alls trots tillåten söktid på 24 h. Vid en jämförelse med de lösningar som CPLEX genererat i de fall där ett optimum kunnat fastställas såg vi att CPLEX och algoritmen finner olika lösningar även om målfunktionsvärdena är snarlika. CPLEX tenderar att försena flertalet tåg innan de når sin slutdestination men ser till att de har en så liten slutlig försening som möjligt (dvs. i linje med målfunktionen). Algoritmens nuvarande sökstrategi, som har ett mer kortsiktigt perspektiv, undviker istället att försena ytterligare tåg såvida det inte verkligen medför en vinst. Vilken lösning som är den bästa från ett praktiskt perspektiv är svårt att säga generellt men det visar att det även finns fördelar med denna mer kortsiktiga sökstrategi. Det visar även hur komplicerat det kan vara att formulera en målfunktion som tar hänsyn till flera olika aspekter, vilket ger ytterligare stöd för att använda en skräddarsydd algoritm som komplement till, eller istället för, mjukvara såsom CPLEX eller Gurobi.

När det gäller algoritmens prestanda m a p de olika kategorierna visade resultaten att för kategori 1 och 2 så presterar algoritmen förhållandevis bra. För kategori 3, däremot, har det varit svårt att finna bra jämförelsevärden (eftersom problemen är för svåra att lösa för CPLEX) men det till trots ser vi att algoritmen har svårt att hitta tillräckligt bra lösningar. Det indikerar återigen att en lösningsmetod kan behöva anpassa sin sökstrategi och målfunktion beroende på vissa egenskaper hos störningssituationen.

4 Syfte, omfattning och genomförande

Arbetet i projektet "Effektiv operativ omplanering av tåglägen vid driftstörningar" består av två huvudsakliga delar som syftar till att:

- Vidareutveckla och förbättra beräkningsmetoden för omplanering och då delvis genom en parallellisering av girighetsalgoritmen och utvärdera effekten av denna.

- Utvärdera och studera beräkningsmetodens prestanda från ett praktiskt perspektiv under driftslika förhållande och då i samverkan med framför allt Trafikverket..

Som vi beskrivit ovan, påvisar våra tidigare resultat att det finns förbättringspotential gällande algoritmen generellt sett men framför allt gällande dess applicerbarhet vid hantering av störningar av kategori 3 - infrastrukturfel. En vidareutveckling syftar primärt till att få algoritmen mer effektiv så att den:

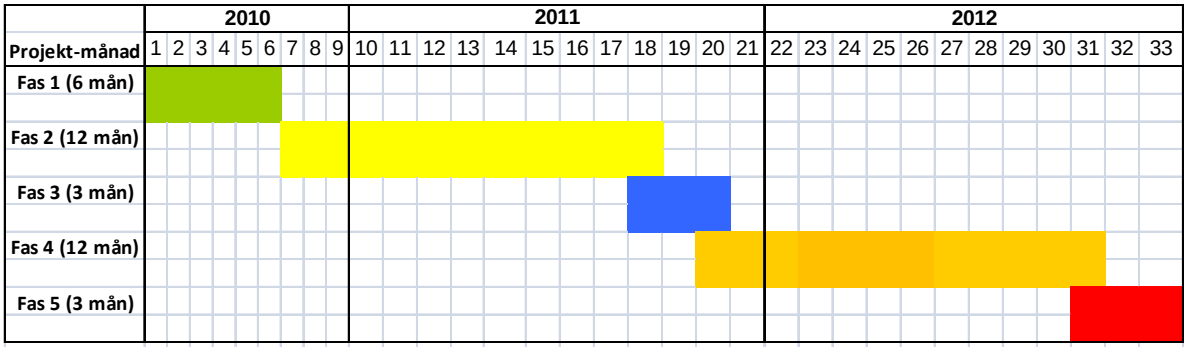
a) blir snabbare vid expansionen av sökträdet (dvs. att den kan undersöka fler noder per tidsenhet vilket kan uppnås genom förbättringar av koden samt genom att parallellisera sökprocessen med hjälp av multi-core processorer), samt
b) att dess sökstrategi förbättras och att den oftare väljer ”rätt” noder att expandera.

Att i större utsträckning undersöka skillnaderna mellan de lösningar CPLEX producerar med de från algoritmen skulle kunna ge ytterligare värdefull information om vilka avvägningar som bör göras i det kortsiktiga perspektivet för att uppnå samma lösningar som CPLEX genererar. Att använda multi-core processorer, se (Emer 2007), för att snabba upp sökningen innebär att flera grenar i trädet skulle kunna expanderas parallellt. I utvecklingen ska vi även anpassa den matematiska modellen till sträckan Katrineholm-Malmö på Södra Stambanan och trafiken enligt T11 och utvärdera omplaneringsmetoden på en större mängd scenarier än tidigare.

Den andra delen syftar till att belysa praktiska aspekter på beräkningsstödet för omplanering av tidtabellen. Centrala frågor är bl a :

- Vilka avvikelser från tidtabellen/störningsscenarier är relevanta att ett beslutstöd ska hantera generellt och i fallstudien?
- Hur påverkar kvaliteten på informationen beräkningarna (t ex tågens position och gångtider approximeras ofta)?
- Vilka justeringar av tågplanen i operativ drift vid oplanerade avvikelser är (inte) acceptabla och under vilka omständigheter gäller/rekommenderas detta?
- Hur kan vi bedöma föreslagna omplaneringslösningars relevans och hur kan/bör de värderas beroende på olika situationer?

Projektet följer nedanstående arbets- och tidsplan.



Fas 1	Mål: Etablera grundförutsättningar för att genomföra projektet,
Fas 2	Mål: Initial parallellisering av optimeringsalgoritm för omplanering av tåglägen
Fas 3	Mål: Förankring hos och återkoppling från relevant personal vid Trafikverket
Fas 4	Mål: Förbättring och vidareutveckling av parallella optimeringsalgoritmer samt utvärdering i samverkan med TrV
Fas 5	Mål: Kunskapsöverföring och diskussion om fortsatt utveckling och slutrapportering.

Tabell 1. Projektets arbets- och tidsplan.

5 Relaterad forskning

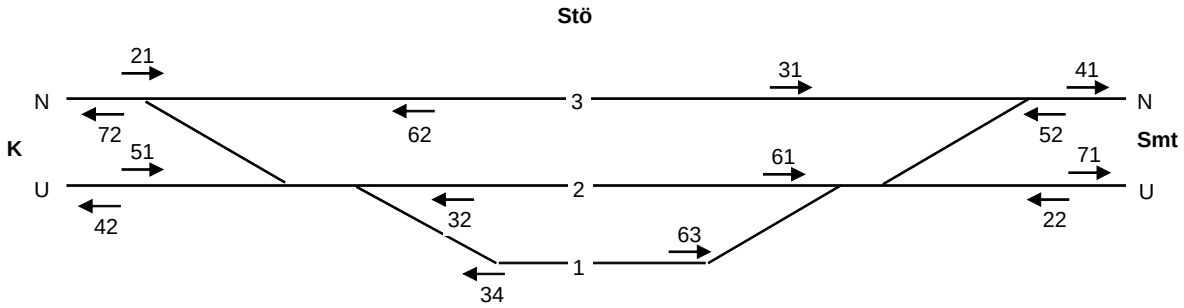
Forskningsintresset för metoder för planering av järnvägstrafik- och transporter har ökat markant de senaste åren. Även om problematiken på ett övergripande plan är densamma världen över, dvs. att resurser såsom spårkapacitet ska allokeras till en mängd operatörer och dess fordon, så skiljer sig förutsättningarna åt på flera sätt. Vad som dock skiljer svensk järnvägstrafik från den internationella är att avregleringen är längre framskriden i Sverige än inom övriga EU vilket gett effekter som ett mer splittrat transportsystem med fler operatörer som ska samsas om kapaciteten och vars trafik ska styras av en neutral part (Trafikverkets åtta driftledningscentraler). Vi har också en mer heterogen trafik där tåg med olika prestanda och behov (godståg, intercitytåg, snabbtåg och pendeltåg) använder samma spår samtidigt under allt längre perioder av dygnet. Många tågkoncept inom EU är periodiska (dvs. cykliska med t ex styva tidtabeller) och mer isolerade från annan tågtrafik, vilket gör att dessa system i nuläget inte behöver hantera denna ytterligare komplexitet i samma utsträckning som i Sverige. Vidare är det svenska järnvägsnätet anpassat för trafik i båda riktningar och man planerar även i tidtabellen ibland för parallelldrift (då är det oftast godstågen som går i motsatt körriktning och endast en kortare sträcka där det är trångt). Andra europeiska system verkar i högre utsträckning bedriva enkelriktad trafik på sina dubbelspår. Det kan man även se om man betraktar utformningen och konfigurationen av de simuleringsverktyg för järnvägstrafik som finns på marknaden.

Skillnader som dessa medför att det förekommer en mängd olika sätt att beskriva problemet på. I detta sammanhang är ofta en sådan problembeskrivning en *matematisk modell* som anger vilka restriktioner (dvs. *villkor*) som gäller för problemet och vilken typ av lösningen som eftersöks (t ex att maximera antalet rättidiga tåg) och som då uttrycks av en *målfunktion.* För att lösa problemet väljer man sedan en *lösningsmetod*. Det finns en mängd olika typer av metoder. I många fall är metoderna dock anpassade för att vara så effektiva som möjligt för den specifika variant av problemet som är i fokus. I Cordeau et. al. (1998), (Törnquist, 2005) och (Schachtebeck, 2009) finns en mer omfattande beskrivning av olika modeller och metoder för tilldelning av järnvägskapacitet.

Generellt sett så innehåller modeller för operativ omplanering av tågtrafik få detaljer och man särskiljer sällan på vilken tågväg ett tåg tar genom en station utan i bästa fall definierar man bara en mängd parallella spår varpå tåget väljer en. Dessa tågvägar är då inte i konflikt med varandra vare sig vid in- eller utfart. I praktiken är det dock annorlunda och i en störningssituation kan det vara viktigt att åskådliggöra vissa tågvägar i mer detalj och de marginaler som finns. Ett exempel är då ett X2000, som enligt plan ska köra på ner-spåret genom Strångsjö via spår 3 (se figuren nedan) utan uppehåll men tvingas istället gå in via spår 2 för en oplanerad förbigång av ett annat tåg som uppehåller sig på spår 3. X2000 måste då bromsa in för att köra genom en växel och förlorar märkbart med tid – något som då kan behöva beskrivas av en motsvarande matematisk modell som då kunde ha beräknat effekten av spårvalen för de båda tågen på stationen och istället gjort det omvända.

När vi vill skapa en modell över järnvägsnätet så måste vi alltså välja lämplig detaljnivå, dvs. vilka egenskaper hos tågen och trafiksystemet samt fysiska delar vi ska beskriva och inkludera i våra beräkningar. Radtke (2008) skiljer på olika detaljnivåer i de infrastrukturmodeller som används för järnvägstrafikplanering, där makromodeller har en

grövre representation med mindre detaljer medan mikromodeller innehåller mer detaljer om hur infrastrukturen är uppbyggd och dess restriktioner. Makromodeller använder ofta en aggregerad beskrivning av vissa resurser där exempelvis en station kan betraktas som en mängd parallella spår av begränsad eller oändlig kapacitet. I en mikromodell kan istället samma station ses som ett komplext nätverk av flera olika spårdelar som binds samman av signaler och växlar. Vilken nivå som är lämplig beror givetvis på syftet med modellen och i vilket avseende resultaten ska användas. Desto mer detaljerad en modell är, desto mer data och beräkningar krävs och för ett problem som ska lösas i real-tid kan det vara för tidskrävande att samla och bearbeta en stor mängd data. Dessutom ger en mer detaljerad modell potentiellt upphov till en större mängd alternativa lösningar att utvärdera. Desto mindre och enklare en modell är, desto lättare blir det att validera och verifiera den (dvs. att kontrollera om den är och beter sig korrekt och ger rimliga resultat) och egenskaper som inte påverkar resultatet (nämnvärt) bör därför inte inkluderas explicit i modellen. Likaså gäller då indata som krävs för att representera vissa egenskaper inte finns att tillgå eller är av för dålig kvalitet utan dessa måste då hanteras på annat sätt. Det kan därför vara effektivare att approximera vissa egenskaper så länge approximationerna ger tillräckligt tillförlitliga resultat.



Figur 1. Strångsjö station

En kartläggning som gjorts inom projektet visar att de flesta modeller för operativ omplanering av järnvägstrafik är makromodeller som inte inkluderar signaler eller växlar eller anger hur tåg färdas genom stationerna. I vissa fall, kan det dock vara nödvändigt att använda en mikromodell för t ex högtrafikerade områden kring storstäder och centrala nav i nätverket medan för andra mer glestrafikerade trafikområden kan det vara tillräckligt med en grov modell.

Projektet DisKon (*Disposition und Konfliktmanagement*) är ett samarbete mellan de tre tyska universiteten; RWTH Aachen, TU Dresden och Göttingens universitet, samt Deutsche Bahn AG. Projektet syftar primärt till att utveckla en metod för att hantera den tyska problematiken kring operativ trafikledning med fokus på att minimera effekten av brustna anslutningar. Metoden använder två olika detaljnivåer i sin modell av trafiken och infrastrukturen men inte beroende på att man anser det finnas behov av olika detaljnivåer i samma modell utan man delar upp problemet i två nivåer. Man löser alltså problemet först grovt med en optimeringsmodell på makronivå och när den lösningen är fastställd, dvs. vissa variabler i problemet är fixerade, så löser man problemet på en mer detaljerad nivå med en mikromodell. Det börjar bli ett alltmer förekommande sätt att dela upp problemet i

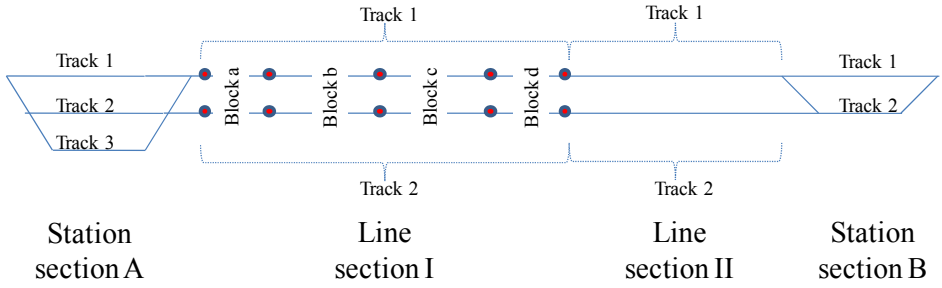
två nivåer som löses iterativt. Man kombinerar ofta en grov problembeskrivning (som t ex inte säkerställer att tågvägarna är konfliktfria inne på stationerna) men testar sedan lösningarna via en befintlig mjukvara som inkluderar en detaljerad beskrivning av infrastrukturen och som i vissa fall ger feedback till den grova modellen om vilka konflikter den identifierat.

Exempel på andra större internationella forskningsprojekt inom området är ARRIVAL (http://arrival.cti.gr/, avslutat) samt ON-TIME (pågående).

6 Projektets modeller och algoritmer

6.1 Modellering av järnvägstrafik

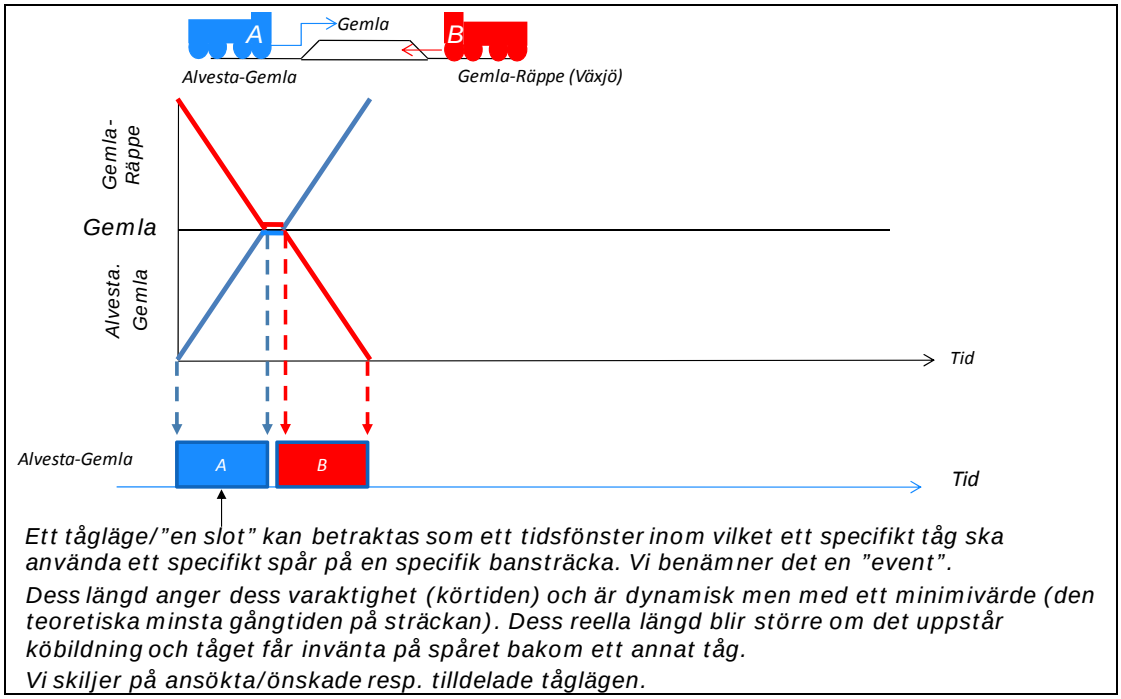
Precis som i det tidigare projektet OAT+ delar vi in infrastrukturen i olika *segment* där ett segment representerar en sträcka från en punkt till en annan och kan vara en station (anges som *station section* i Figur 2 nedan) liksom en sträcka mellan två stationer (anges som *line section)*. Vad som karakteriserar ett segment är att det består av ett eller flera parallella spår (anges som *track*). Varje spår kan vara uppdelat i en eller flera blocksträckor. Vi har använt data från Trainplan där segmenten då är av typen *Network Links* (NWK) eller *Locations* (LOC).



Figur 2. Modellering av infrastrukturen som resurser.

Vi betraktar problemet med att revidera (dvs. omplanera) tidtabellen i drift som ett s.k. *job shop scheduling problem (JSS)* där jobben som ska utföras på olika maskiner istället representeras av att tågen ska utföra sina resp. deltransporter (en deltransport betraktas som *ett* jobb) på ett visst segment (en maskin). Varje tåg ska därmed utföra en mängd jobb i en given sekvens på en mängd maskiner. Vi kallar varje sådan deltransport för en *event* (eller slot/tågläge). Varje tåg har då en sekvens (dvs. en lista) av *events* som ska utföras i en logisk ordning. Den första *eventen* motsvaras då av det uppehåll tåget har på sin avgångsstation eller bangård, medan eventen därefter då motsvaras av den första linjesträckan från avgångsstation till nästkommande station (eller linjesträcka, om det finns en växel emellan) på just det tågets rutt på järnvägslinjen. När ett tåg lämnar ett segment (dvs. avslutar en specifik event) så påbörjar tåget omedelbart ett annat segment (dvs. påbörjar nästa event). Enligt Trainplans datastruktur är då samtliga events av typen *Train Movement* (TMV, som sker på en NWK) eller *Train Stop* (TSP, som sker på en LOC). För det enkla exemplet i figuren nedan har Tåg A en lista med tre olika events som ska ske i

kronologisk ordning: {event 1 på Alvesta-Gemla; event 2 på Gemla; event 3 på Gemla Räppe}. Motsvarande eventlista för Tåg B blir då {event 1 på Gemla-Räppe; event 2 på Gemla; event 3 på Alvesta-Gemla}.



Figur 3. Illustration av hur vi modellerar ett tågläge/slot/event.

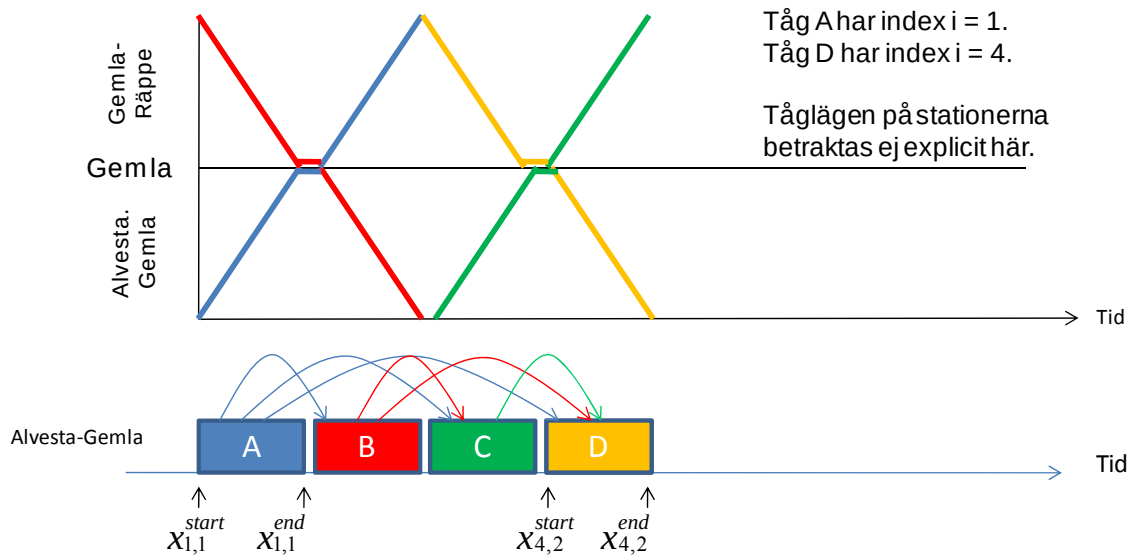
För att underlätta beskrivningen av modellen ges notationen nedan:

Låt T vara vår mängd tåg, B är mängden av segment, och E mängden av events där en event är tidsperioden då ett specifikt tåg planerar att köra på ett specifikt segment. Låt index i vara associerat med ett tåg, index j med ett segment och index k med en event. Varje event tillhör både ett tåg och ett segment, dvs. den representerar ett tåg på ett segment . Låt $K_i \subseteq E$ vara den sorterade mängden av events för tåg i ($i \in T$) och $L_j \subseteq E$ vara den sorterade mängden av events för segment j ($j \in B$). Vidare, vi använder $(k+1)$ för att representera den första event som följer efter event k medan $\hat{k}$ är *någon* av de event som följer event k i de sorterade mängderna, så att $k < \hat{k}$ gäller.

Varje segment j har ett visst antal spår $P_j = \{1,.., p_j\}$. Vi använder index t för att representera ett visst spår.

Problemet handlar om att fastställa ordningen på tågen på resp. segment och givet denna även fastställa start- och sluttider (x_{ik}^{begin} resp x_{ik}^{end}) för varje event k. Tilldelningen av dessa tider ska givetvis beakta de infrastrukturella begränsningarna som finns i form av att varje event ska tilldelas *ett* spår. Om event k använder spår t på det segment där den är planerad så antar den binära variabeln q_{ikt} värdet 1, annars 0. Om två tåg planerar använda samma spår måste de dessutom separeras i tid, vilket hanteras av de binära variablerna $\gamma_{k\hat{k}}$

(som antar värdet ’1’ om event k ska ske före event $\hat{k}$, annars 0) och $\lambda_{k\hat{k}}$ (som antar värdet ’1’ om event $\hat{k}$ ska ske före event *k*, annars 1). Vid enkelspårig sträcka behövs inte q_{ikt} eller $\lambda_{k\hat{k}}$ utan endast $\gamma_{k\hat{k}}$ eftersom det är uppenbart att tågen använder det enda spåret som finns. Se Figur 4 nedan för en illustration där samtliga γ-värden får värdet ’1’.



$$\gamma_{j,k,\hat{k}} = \begin{cases} 1, \text{ om event k är planerad att ske före event } \hat{k} \\ 0, \text{ annars (om event } \hat{k} \text{ är planerad att ske före event k)} \end{cases}$$

$$\gamma_{j,A,B} = \gamma_{1,1,1} = 1 \quad \gamma_{j,A,C} = \gamma_{1,1,2} = 1 \quad \gamma_{j,B,C} = \gamma_{1,2,1} = 1 \quad , osv...$$

Figur 4. Illustration av hur vi modellerar att resp. event ska skiljas i tid från övriga om de sker på samma spår, vilket är nödvändigt i detta exempel med en enkelspårsträcka mellan Växjö och Kalmar.

Vi skiljer också på två olika fall där tågen (dess events) behöver separeras om de använder samma spår. Det ena fallet i) är när tågen kör i samma riktning på ett spår som har flera blocksträckor och det andra ii) är när två tåg kör i olika riktning eller kör på ett segment med endast en blocksträcka.

I det första fallet ska tågens starttid på sträckan separeras av ett minsta tidsavstånd (headway) och likaså dess sluttider. I ii) måste först det ena tåget lämna segmentet, en viss ”clear time” för att släppa tågvägen ska passera, och först därefter får nästa tåg starta på det segmentet. Använder tågen inte samma spår (och tågvägarna inte på annat sätt är i konflikt) så spelar tågens inbördes ordning ingen roll.

I figuren tar vi inte upp events som sker på stationerna men i våra modeller och algoritmen behandlar vi events på stationer på i princip samma sätt som de på linjen mellan stationerna.

De variabler som nämns ovan ingår sedan i en mängd villkor som begränsar vilka lösningar som blir tillåtna och av de lösningar som är tillåtna söker lösningsmetoden efter den som är bäst utifrån en målfunktion. En målfunktion kan exempelvis vara att minimera avvikelsen mellan de tilldelade (av lösningsmetoden) avgångs- och ankomsttider och de önskade. Vi har använt den kommersiella mjukvaran IBM ILOG CPLEX Optimizer version 12.2 för att lösa optimeringsproblemet som vi formulerat. Modellen och notationen för problemet som helhet är baserad på den som återfinns i (Törnquist Krasemann, 2012).

6.2 Girighetsalgoritmen

Ett vanligt sätt att beskriva beroenden och hierarkin mellan olika element eller händelser är i egenskap av träd som består av ett antal noder som är länkade likt grenar till varandra så att det inte bildas någon loop. Varje nod, utom rotnoden, har föräldrar och varje nod kan ha ett eller flera s.k. barn. En nod som saknar barn kallas för löv och slutat på varje gren är alltså ett löv. Ett träd kan sägas bestå av en samling delträd. Det är utifrån denna struktur som algoritmen skapar och lagrar lösningar, dvs. planen för hur tågen i ordning ska exekveras på resp. sträcka de planerar trafikera.

Algoritmen skapar och utvärderar alltså omplaneringslösningar genom att successivt exekvera de events som står på tur enligt beskrivningen ovan. Det vill säga, finns det *n* tåg som har events som ej ännu är exekverade är det *n* events som undersöks i varje steg för att välja den event som är bäst lämpad att exekveras i nästa steg. De events som exekveras läggs till och bygger upp en gren där varje exekverad event är en nod och där den första noden, *roten*, är den första eventen som exekverats. Den sista eventen i varje gren, *ett löv*, är den som exekverats senast. Iterativt skapas nya lösningar, grenar, som då bildar *sökträdet*. Hela trädet sparas inte utan endast den bästa lösningen man har funnit hittills samt den aktiva gren i trädet som sökningen utgår ifrån. Man behåller dock i minnet vilka delgrenar man redan har undersökt för att undvika i möjligaste mån att man upprepar att utvärdera en lösning flera gånger. Varje nod i en gren har ett värde som är en optimistisk kostnadsestimering (en nedre gräns) av den lösning som den aktiva grenen ovan noden kan komma resultera i. Värdet på noder efter nod *i* är därmed alltså lika eller större vid ett minimeringsproblem.

Algoritmen har tre olika faser och har implementerats i Java. Fas 1 är en startfas (pre-processing phase), medan i Fas 2 utförs en djupgående sökning (Depth-First Search, DFS) för att snabbt hitta en första giltig och tillräckligt bra lösning, dvs. skapa den första kompletta grenen i trädet. I Fas 3 använder algoritmen den tid som återstår (av totalt 30 sekunder i detta fall) för att gå tillbaka i trädet och försöka hitta andra lösningar som är bättre än den första lösningen.

I Fas 1 exekveras först de events som motsvaras av de tåg som är aktiva när störningen inträffar så att starttiden för dem samt allokerat spår blir som planerat. Den första ej exekverade eventen i varje tågs kronologiskt sorterade eventlista läggs i en kandidatlista som är sorterad efter maxvärdet av 1) den tidigaste möjliga starttiden resp. 2) dess planerade starttid.

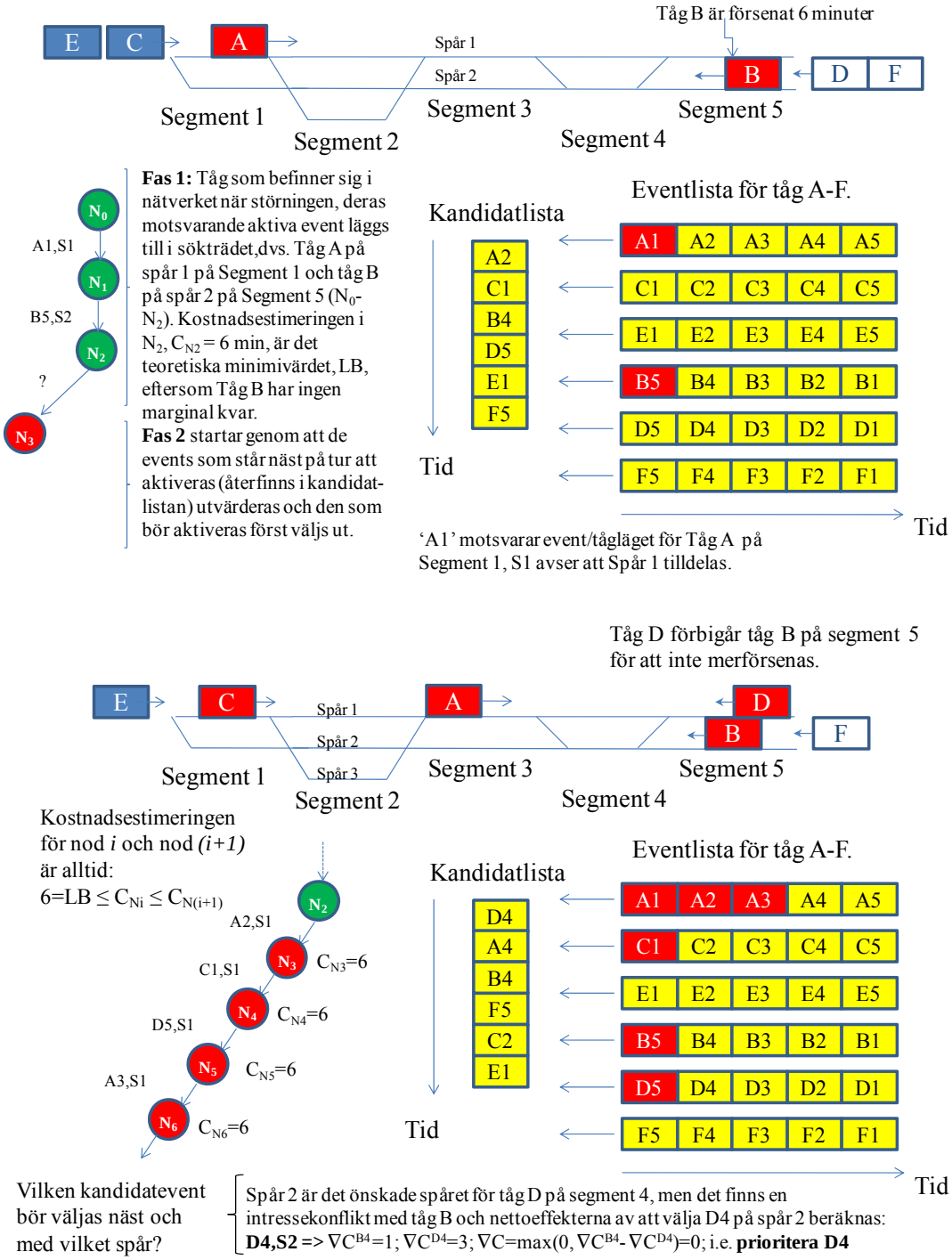
Fas 2 börjar sedan med att events i kandidatlistan utvärderas, där kandidateventen med den tidigaste starttiden utvärderas först. Först kontrollerar algoritmen om det finns något ledig

spår för eventen (med preferensen för det spår som tilldelats i tidtabellen). Om det inte finns något ledigt spår, så fortsätter den istället att utvärdera nästa event i kandidatlistan. Om tillgängligt spår finns utförs en *deadlock-analys* samt en analys av om det föreligger *intressekonflikt* gällande tillgång till spåret på just det segmentet. Om intressekonflikt finns mellan två eller fler events (dvs. tåg) som är planerade, så beräknas effekterna av de olika konfliktalternativen. De events som man identifierat att det finns en konflikt med för kandidateventen måste dock inte finnas i kandidatlistan. Så om tåg A är under utvärdering för att få ett spår allokerat före tåg B och om detta potentiellt skulle försena tåg B, så beräknas en nettominimikostnadsökning (*net minimum expected cost increase)* som tar hänsyn till aktuell försening och relevanta bufferttider. Om värdet är positivt, indikerar det att på kort sikt skulle tåg B drabbas mer än vad tåg A skulle vinna på att få tillgång till spåret först och följaktligen att det skulle vara mest kostnadseffektivt i det kortare perspektivet att prioritera tåg B.

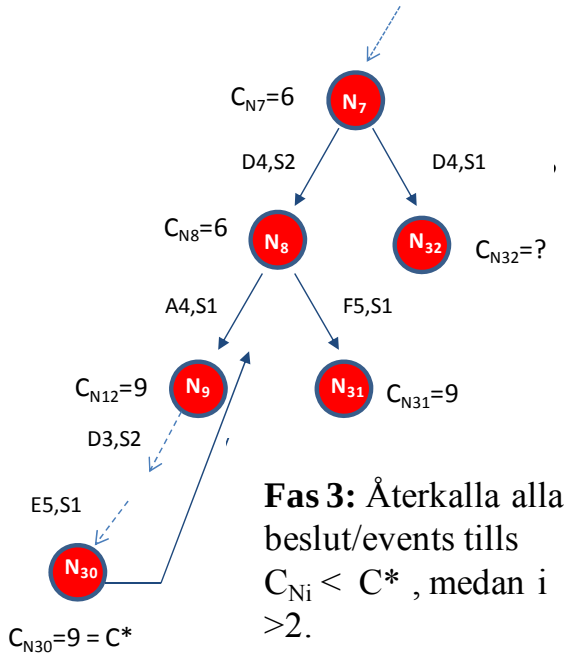
När den bästa eventen i kandidatlistan har identifierats så exekveras den (dvs. den blir aktiverad) och allokeras en starttid och ett spår i linje med gällande restriktioner. Eventuell föregående event för det berörda tåget avslutas följaktligen när den nya exekveras och den resurs som var allokerad frigörs. En nod som representerar att den nyaktiverade eventen allokerats ett visst spår bygger på trädet/grenen och nodens ”förälder” (dvs. dess *parent node*) lagrar i sitt minne vilka direkt underliggande noder (dvs. *child nodes*) den har haft. Denna information används sedan under *back-tracking* i Fas 3. En kostnadsestimering för den nya ej ännu kompletta lösningen beräknas genom att samtliga tågs minsta möjliga försening till slutstation summeras. Se Figur 5 nedan för en illustration av algoritmens arbetsprocess.

När samtliga events har exekverats (och avslutats) och lagts till som noder i trädet har en första komplett och giltig lösning hittats och den sista nodens värde är den verkliga kostnaden för lösningen, C*. Kostnaden kan vara angiven på valfritt sätt men vi har valt att i detta exempel summera samtliga tågs försening till dess slutdestination inom angivet trafikområde och tidshorisont (dvs. den sista eventen i resp. tågs i eventlista).

I Fas 3 använder sedan algoritmen resterande beräkningstid för att expandera trädet och söka vidare efter en bättre lösning. Undantaget är (dvs. då sökningen avbryts) om värdet på rotnoden är detsamma som den funna kompletta lösningens värde. Att värdena är lika innebär att det inte finns någon bättre lösning eftersom den klassiska principen om att alla nodvärden på noder under en nod *i* är lika bra eller sämre gäller. När algoritmen går upp i trädet (s k *back-tracking*) för att för första gången välja en lämplig nod att expandera vidare, så väljs en nod vars värde är lägre än gällande övre gräns (dvs. *upper bound* = *C**). Den övre gränsen är då kostnadsvärdet för den hittills bästa funna kompletta lösningen. Algoritmen fortsätter att expandera den aktiva grenen tills den hittar en ny komplett bättre lösning (som då sparas) eller en intermediär nod som har ett värde som är större eller lika med den övre gränsen. När back-tracking sker på nytt så väljs återigen en nod med ett värde som är lägre än den övre gränsen. För att undvika att samma nod väljs åter och åter igen så används den lagrade informationen om vilka direkt underliggande noder som tidigare besökts.



Figur 5. Beskrivning av girighetsalgoritmens Fas 1 och 2.

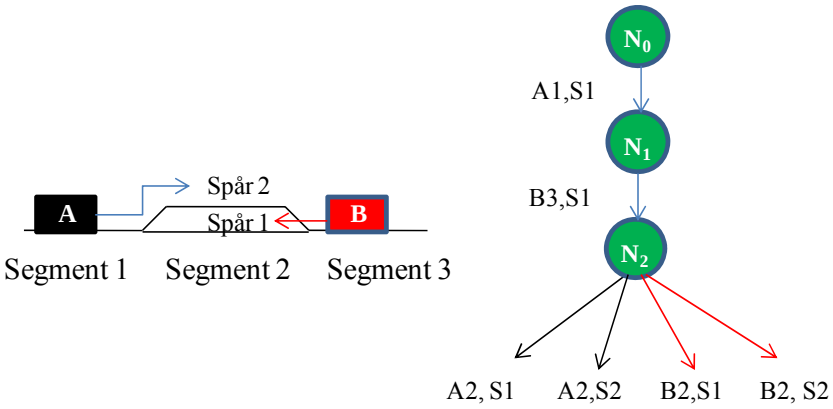


Fas 3: Återkalla alla beslut/events tills $C_{Ni} < C^*$, medan i >2.

Expandera endast nod N_{32} om dess värde är bättre än C*, eller gå tillbaka till nod N_7 och välj en annan nod att exekvera. Om alla noder nedanför N_7 inte medför en förbättring m a p C* återgå till nod N_6.

Figur 6. Beskrivning av girighetsalgoritmens hantering av trädsökning i Fas 3.

Vad som bland annat förbättrats i vidareutvecklingen av algoritmen är dess strategier för att undvika att utvärdera lösningar som är mycket lika tidigare lösningar. Som exempel kan vi ta den enkla situation som uppstår om vi har två tåg i motsatt riktning på en enkelspårsträcka och där båda ska stanna på en station för ett uppehåll. Stationen har två spår enligt figuren nedan och tillåter samtidig infart. Denna typ av station kallar vi symmetrisk eftersom val av spår har mindre betydelse. Om vi har situationen där tågen är på väg in till stationen och algoritmen ska avgöra, enligt sökträdsförfarandet, i vilken ordning de ska exekveras och spårval, så finns fyra olika alternativ. Datastrukturmässigt är det fyra olika lösningar men från ett praktiskt perspektiv utgör de mycket snarlika lösningar. Alla behöver därmed inte undersökas på just den nivån i trädet i Fas 3 (eftersom algoritmen i första hand tilldelar den önskade stationsspåret). En begränsning sätts därför på antalet utvärderingsalternativ på symmetriska stationer. Algoritmen beskrivs i mer detalj i (Törnquist Krasemann, 2012) samt (Grahn och Törnquist Krasemann, 2011).



Figur 7. Exempel på symmetrisk station och effekten av redundans.

6.3 Parallellisering av girighetsalgoritmen

Att parallellisera en algoritm innebär i princip att man skapar flera, snarlika kopior av algoritmen, vilka parallellt och beroende/oberoende av varandra försöker hitta bra lösningar till problemet. Man kan se skillnaden mellan den sekventiella girighetsalgoritmen och den parallella versionen som att i den sekventiella sitter det *en trafikledare* som försöker lösa situationen ensam medan i den parallella versionen sitter det *många fler trafikledare* som arbetar med samma problem parallellt och (delvis) oberoende av varandra. Trafikledarna försöker skapa ett antal *olika* omplaneringslösningar och utbyter information med varandra om vilka omplaneringsåtgärder som är bra och kan leda till en bra lösning samt vilka som är mindre bra. Därmed blir själva jakten på lösningar mer effektiv och det skapas också fler alternativa och kompletterande lösningar som de riktiga trafikledarna hos Trafikverket kan ta ställning till och modifiera vid behov. En central utmaning är då att undvika att dessa parallella beräkningar leder till likadana lösningar. Istället vill man effektivt kunna identifiera de bästa alternativa lösningarna och beräkna effekterna av dessa. En annan utmaning är att skapa ett effektivt informationsbyte mellan de parallella processerna som inte hämmar själva beräkningarna.

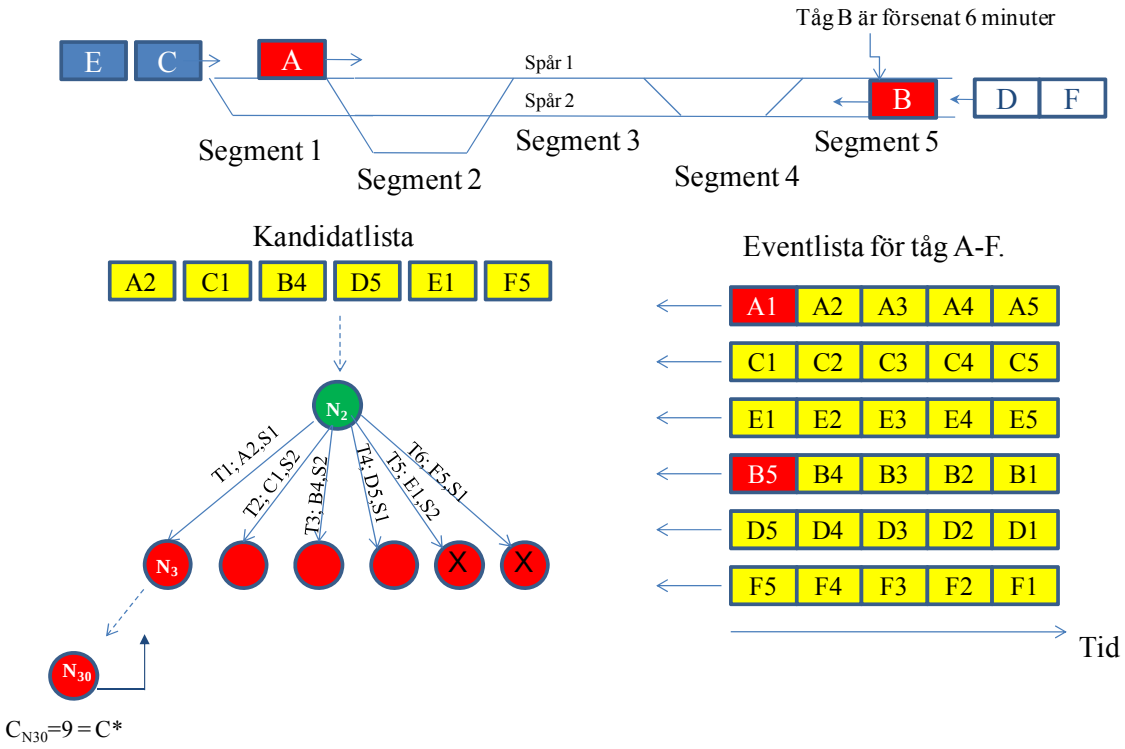
Vi använder en så kallad *master-slave* strategi där det finns en övergripande koordinering som sköts av en s k *master.* Denna *master* fördelar arbetet mellan de parallella processerna som benämns *trådar/slaves/workers.*

I initieringsfasen utför först vår *master* Fas 1 precis som i den sekventiella girighetsalgoritmen. Sedan skapas den första kandidatlistan samt ett valfritt antal parallella processer (dvs. trådar, som är sex stycken i exemplet nedan och Figur 8). Varje tråd, T, tilldelas en tillåten event från kandidatlistan att utgå ifrån så att Tråd 1 (T1) får den första tillåtna kandidaten i listan (A2). Tråd 2 (T2) tar sedan den kandidatevent som står på tur näst (C1) och så vidare. T1 startar först Fas 2 med att exekvera A2 och hitta en tillåten lösning medan Tråd 2-6 väntar. Först när T1 hittat en tillåten lösning så startar den parallella bearbetningen. Det vill säga, när T1 påbörjar Fas 3 på samma sätt som i den sekventiella girighetsalgoritmen, så startar T2-T8 Fas 2 med utgångspunkt från sin resp. tilldelade start event C1-F5 och söker på samma sätt som girighetsalgoritmen. I en senare version av parallelliseringen (se kapitel 6.6) får dock övriga trådar starta efter en viss tid, även om T1 inte än, hittat en tillåten lösning. Detta för att minska risken för att systemet kollapsar om T1 av någon anledning inte når en tillåten lösning tillräckligt fort.

I figuren nedan markeras två noder med ”X”. Dessa noder motsvarar tillståndet efter att tråd 5 och 6 skulle starta event E1 resp. F1 och ”X” illustrerar att dessa två events inte kan exekveras direkt efter nod N_2. Detta beror på att motsvarande tåg – tåg E och tåg F - ligger i kolonn efter tåg C och tåg D resp. med ett minsta tidsavstånd enligt headway-principen. Det innebär att event C1 och D5 måste exekveras först.

När parallelliseringen startat och trådarna fortsätter att - oberoende av varandra men med olika utgångslägen - leta efter förbättrade lösningar så informerar de varandra om förbättringar och uppdaterar den globala bästa lösningen via en s k *white board*. I grundimplementationen av den parallella girighetsalgoritmen har de olika trådarna samma

sökbeteende, men i senare version (se kapitel 6.6) skapar vi olika strategier för dem så att de på ett bättre sätt tar höjd för den variation av störningsscenarier som uppträder och bättre kompletterar varandra. Mekanismen och funktionen för att utbyta information mellan trådarna är också under utveckling.



Figur 8. Beskrivning av parallelliseringen av girighetsalgoritmen.

Tre centrala frågeställningar i vidareutvecklingen av den parallella versionen av girighetsalgoritmen har varit:

a) Hur många trådar vi ska använda?
b) När ska vi initiera parallelliseringen?
c) Hur ska vi sortera kandidatlistan så att tilldelningen av ”startevents” till trådarna blir mest optimal?

Eftersom antalet trådar motsvarar antalet parallella processer så beror deras effektivitet på hur många processorer (kärnor) den datorn som man använder har. Har man fler trådar än kärnor så behöver trådarna turas om att få tillgång till kärnorna. Denna resursdelning kan i värsta fall leda till att en tråd, A, som är nära att hitta en förbättrad lösning pausas temporärt till förmån för en annan tråd, B. Den gren som tråd B utgår ifrån i sökträdet kanske hade betraktas som icke-intressant om tråd A hade fått hitta den förbättrade lösningen först. Å andra sidan kanske detta hade tagit mycket lång tid. Ett ökat antal trådar medför dock en

möjlighet att sprida riskerna och undersöka många olika strategier och delar av det stora sökträdet.

Val av parallelliseringspunkt är viktigt av den anledningen att om man börjar högt upp i trädet (som i grundimplementationen) är risken att trådarna utför mycket snarlikt arbete och undersöker snarlika lösningar (jmf. effekten av symmetriska stationer). Initierar man å andra sidan parallelliseringen för sent kan man redan ha fattat mindre optimala beslut som då påverkar möjligheterna att expandera trädet och hitta bättre lösningar.

Det är alltså en viktig balansgång och förhållandet mellan dessa tre aspekter när man designar parallelliseringen analyseras bäst via upprepade experiment med olika konfigurationer. I våra initiala experiment kom vi fram till att om vi inte skapar en tråd för varje kandidatevent i den första kandidatlistan så kan vi missa att expandera viktiga grenar i sökträdet. Vi använde åtta kärnor och som mest 52 trådar och detta förhållande - 8/52 - är generellt sett inte så effektivt från ett resurstilldelningsperspektiv. Om man utgår från att *varje* tråds resultat är synnerligen viktigt för att hitta bättre lösningar - snarare än att eventuellt bara identifiera en eventuellt ointressant gren - så borde konfigurationen av b) och c) säkerställa att det räcker med ett lika stort antal trådar som kärnor. En utmaning ligger därmed i att hitta den optimala sorteringen/tilldelningen av start events och startpunkt för parallellisering (som ju också kan ske gradvis). Beskrivningen av algoritmutvecklingen och experimenten beskrivs i sin helhet i (Iqbal et. al., 2012a) i Bilaga 1.

I nästa steg skapade vi därför ett antal olika strategier för att sortera kandidatlistan så att de absolut lämpligaste kandidaterna ligger först i listan oberoende av vilket störningsscenario som avses. Vi valde att kalla den initiala sorteringsstrategin för ”strategy 0”, förkortat s0. Nedan listas de olika strategier vi valde att använda och utvärdera effekten av:

- s0 = Sortering baserad på tidigaste möjliga starttid för resp. kandidatevent
- s1α = Sortering baserad på tidigaste möjliga sluttid (”releasetid”) för resp. kandidatevent beräknad på den planerade kör-/uppehållstiden.
- s1β = Sortering baserad på tidigaste möjliga sluttid (”releasetid”) för resp. kandidatevent beräknad på den minsta möjliga kör-/uppehållstiden.
- s2 = Sortering baserad på tidigaste möjliga starttid för resp. kandidatevent men tar också hänsyn till om kandidateventen har någon buffertid.
- s3 = Sortering baserad på 1) tidigaste möjliga starttid för resp. kandidatevent samt 2) minsta möjliga kör-/uppehållstiden.

s0 har visat sig effektiv att hitta den första tillåtna lösningen snabbt men är sedan inte i alla scenarier lika effektiv att hitta förbättringar. Detta beror delvis på att sorteringsstrategin inte tar hänsyn till när en kandidatevent förväntas avslutas utan endast hur tidigt den kan starta. Strategi s1 – i de två olika men mycket snarlika utförandena - sorterar istället baserat på tidigaste ”releasetid” och prioriterar de events i ”kön” som kan avslutas fortast. s2 är mycket snarlik s0 men tar också hänsyn till om en event har någon bufferttid. s3 är också mycket lik s0 men har som andra kriteriet att jämföra minimikörtiden och prioriterar den event som har kortast minimikörtid om två events har samma tidigaste möjliga starttid.

För att jämföra effekten av de olika sorteringsstrategierna utförde vi en mängd experiment baserat på de 20 scenarier som redovisas i nästa kapitel samt 80 ytterligare baserat på samma tidsperiod och nätverk som de andra 20. Utförande och resultat redovisas i sin helhet i (Iqbal et. al., 2012b) som återfinns i Bilaga 1.

6.4 Simuleringsexperiment och utvärdering av algoritmen

Experimenten vi har genomfört för att utvärdera algoritmernas prestanda har utgått från de scenarier som användes i OAT+ för att kunna göra en jämförelse med tidigare experiment med den sekventiella versionen av algoritmen.

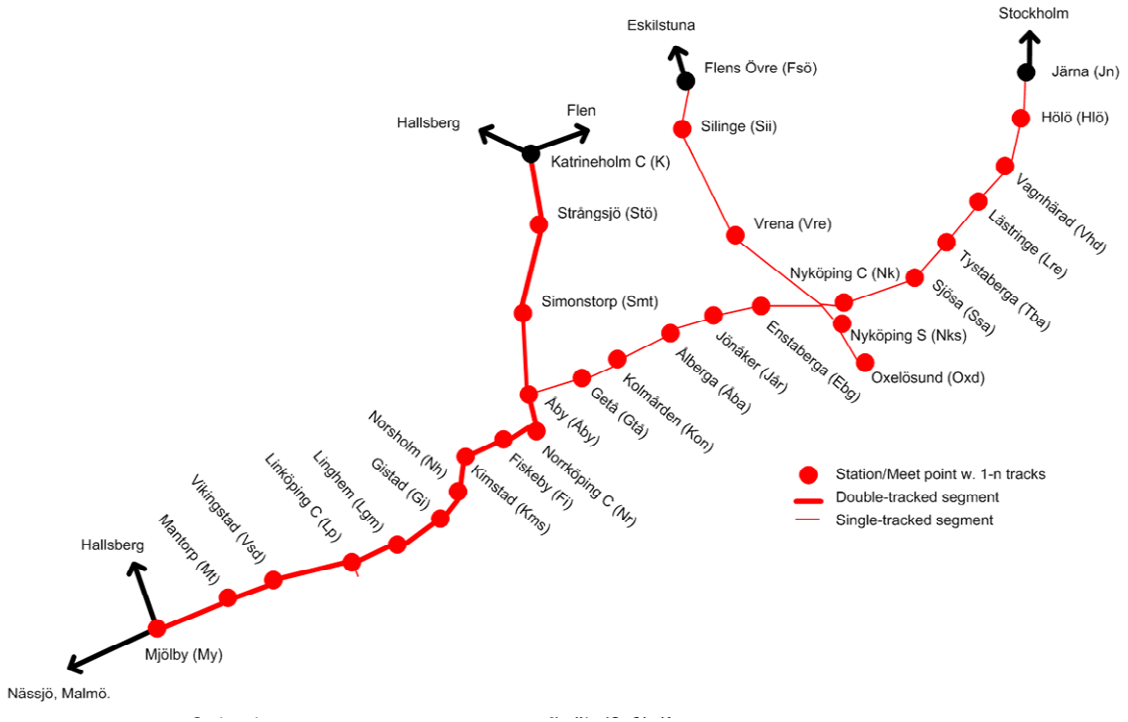
Vi har fokuserat på trafiken i området för Norrköpings distrikt (se figuren nedan) med en mer detaljerad modell av sträckan Norrköping-Katrineholm. Alla spår tillåter trafik i båda riktningar och sträckan Katrineholm-Åby-Mjölby är dubbelspårig medan övriga linjer är enkelspåriga. Stationerna har mellan två och 14 spår med undantag från Norsholm (Nh) som bara har ett. Linjesträckorna mellan Katrineholm och Åby innehåller sju konsekutiva blocksträckor vardera medan sträckan mellan Åby och Norrköping innehåller två konsekutiva linjesträckor precis som sträckorna Ålberga-Kolmården samt Linköping-Linghem. För stationerna Åby, Simonstorp samt Strångsjö är endast tillåtna tågvägarna inkluderade medan för övriga stationer där modellen inte innehåller denna information så antar vi att alla tågvägar in till och ut från stationerna är tillåtna. Vi har tillämpat algoritmen på tre olika typer av störningskategorier:

Kategori 1 innebär att ett tåg anländer trafikdistriktet med en viss försening eller att det får en temporär försening på ett segment inom distriktet.

Kategori 2 innebär att ett tåg har en mer ”kronisk” försening såsom nedsatt accelerationsförmåga vilket resulterar i ökade gångtider.

Kategori 3 syftar på ett infrastrukturfel som ger upphov till exempelvis en hastighetsnedsättning på en viss linjesträcka och som resulterar till att samliga tåg som passerar får ökade gångtider.

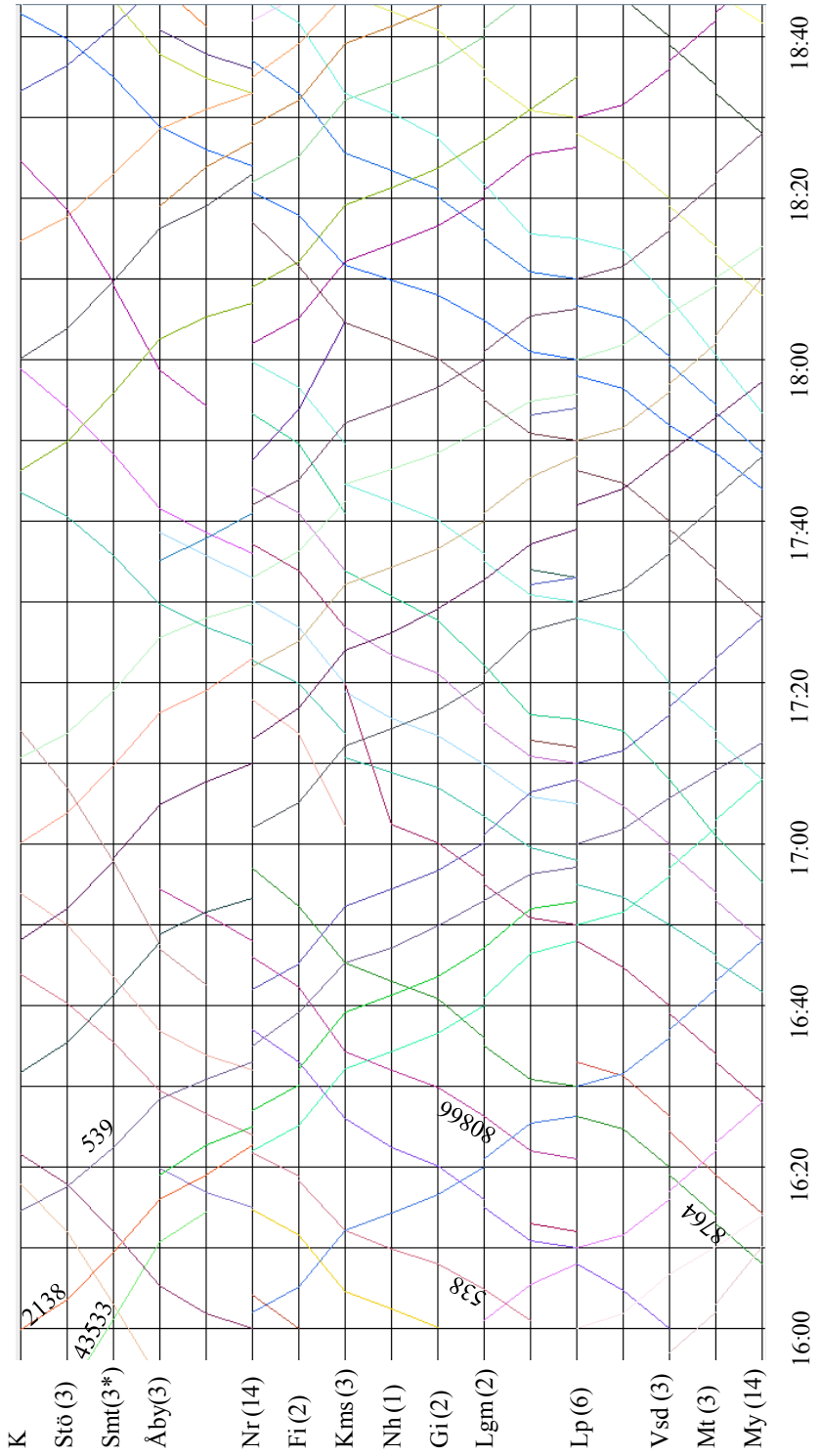
Vi har använt tidtabellsdata från TrainPlan och simulerat att störningarna inträffar i rusningstrafik på eftermiddagen den 23 april 2009. Vi har omplanerat trafiken inom en tidshorisont på 90 minuter med utgångspunkt från då resp. störning inträffat samt vi har tillåtit en maximal beräkningstid på 30 s. En beskrivning av resp. scenario finns i Tabell 2 och Figur 10. Scenario 1-10 tillhör kategori 1 medan 11-15 tillhör kategori 2 och 16-20 tillhör kategori 3.



Figur 9. Nätverket som använts i den första experimentella utvärderingen av den parallella algoritmen.

Scenario	
1	Tåg 538 (norrgående X2000), försenat 12 min Linköping-Linghem.
2	Tåg 538 (norrgående X2000), försenat 6 min Linköping-Linghem.
3	Tåg 2138 (södergående intercity), försenat 12 min Katrineholm-Strångsjö
4	Tåg 2138 (södergående intercity), försenat 6 min Katrineholm-Strångsjö
5	Tåg 80866 (norrgående intercitytåg), försenat 12 min Linköping-Linghem.
6	Tåg 80866 (norrgående intercitytåg), försenat 6 min Linköping-Linghem.
7	Tåg 8764 (norrgående intercitytåg), försenat 12 min Mjölby-Mantorp.
8	Tåg 8764 (norrgående intercitytåg), försenat 6 min Mjölby-Mantorp.
9	Tåg 539 (södergående X2000), försenat 12 min Katrineholm-Strångsjö
10	Tåg 539 (södergående X2000), försenat 6 min Katrineholm-Strångsjö
11	Tåg 538 (norrgående X2000) med kronisk hastighetsnedsättning som medför 50% ökade körtider på alla linjesektioner med start Linköping-Linghem
12	Tåg 2138 (södergående intercity) med kronisk hastighetsnedsättning som medför 50% ökade körtider på alla linjesektioner med start Katrineholm-Strångsjö
13	Tåg 80866 (norrgående intercity) med kronisk hastighetsnedsättning som medför 50% ökade körtider på alla linjesektioner med start Linköping-Linghem
14	Tåg 8764 (norrgående intercity) med kronisk hastighetsnedsättning som medför 50% ökade körtider på alla linjesektioner med start Mjölby-Mantorp
15	Tåg 539 (södergående X2000) med kronisk hastighetsnedsättning som medför 50% ökade körtider på alla linjesektioner med start Katrineholm-Strångsjö
16	Hastighetsnedsättning för alla tåg mellan Strångsjö och Simonstorp (alla tåg får en körtid på 27 min, jmf. 5-10 min planerad körtid) med start för det södergående godståget 43533.
17	Hastighetsnedsättning för alla tåg mellan Åby och Simonstorp (alla tåg får en körtid på 20 min) med start för tåg 2138.
18	Hastighetsnedsättning för alla tåg mellan Åby och Norrköping (alla tåg får en körtid på 8 min) med start för tåg 2138.
19	Hastighetsnedsättning för alla tåg mellan Mjölby och Mantorp (alla tåg får en körtid på 20 min) med start för tåg 8764.
20	Hastighetsnedsättning för alla tåg mellan Linköping och Linghem (alla tåg får en körtid på 15 min) med start för tåg 538.

Tabell 2. Beskrivning av scenario 1-20.



Figur 10. Beskrivning av trafikbilden på delsträckan Katrineholm(K) – Mjölby (My) där störningsscenarierna inträffar.

6.5 Utvärdering av alternativa sorteringsstrategier

För att jämföra effekten av de olika sorteringsstrategierna som beskrivits i kapitel 6.3 utförde vi en mängd experiment baserat på de 20 scenarier som redovisas i kapitel 6.4 samt 80 ytterligare baserat på samma tidsperiod och nätverk som de andra 20. Utförande och resultat redovisas i sin helhet i (Iqbal et. al., 2012b) som återfinns i Bilaga 1.

I Tabell 3 nedan visas en summering av resultatet från de experiment när den sekventiella algoritmen har omplanerat trafiken under 30 s. De olika kolumnerna anger resultaten då sorteringsstrategin har varit olika enligt beskrivningen i kapitel 6.3. Det vill säga:

- s0 = Sortering baserad på tidigaste möjliga starttid för resp. kandidatevent
- s1α = Sortering baserad på tidigaste möjliga sluttid (”releasetid”) för resp. kandidatevent beräknad på den planerade kör-/uppehållstiden.
- s1β = Sortering baserad på tidigaste möjliga sluttid (”releasetid”) för resp. kandidatevent beräknad på den minsta möjliga kör-/uppehållstiden.
- s2 = Sortering baserad på tidigaste möjliga starttid för resp. kandidatevent men tar också hänsyn till om kandidateventen har någon buffertid.
- s3 = Sortering baserad på 1) tidigaste möjliga starttid för resp. kandidatevent samt 2) minsta möjliga kör-/uppehållstiden.

Som framgår av Tabell 3 nedan och de resultat som återfinns i artiklarna i Bilaga 1, så är det inte en enda sorteringsstrategi som är dominerande i alla scenarier. Strategierna som baseras på tidigaste starttid (s0) samt tidigaste ”releasetid” fungerar bäst (s1α och s1β) och kompletterar varandra väl. Strategin som tar hänsyn till bufferttiden (s2) fungerade i sin implementation sämst. I likhet med tidigare resultat är fortfarande flera av de störningar som är av typen infrastrukturfel svårast att lösa.

Eftersom det heller inte är möjligt att identifiera någon tydlig korrelation mellan typen av störningsscenario och val av sorteringsstrategi är det lämpligt att använda flera parallella algoritmer med olika sorteringsstrategier och därmed öka möjligheterna att hitta flera bra lösningar.

Sc#	Bästa funna lösning (i sekunder) av sekventiell alg.					Minsta värde	CPLEX 12.2 24h
	S0	S1α	S1β	S2	S3		
1	1175	995	1103	1489	1103	995	855
2	437	288	396	751	396	288	226
3	781	686	740	1150	740	686	570
4	421	326	380	790	380	326	210
5	930	1111	1300	1604	1300	930	686
6	53	53	68	592	68	53	30
7	499	499	568	1109	568	499	486
8	207	332	401	1003	401	207	176
9	800	768	837	1405	837	768	731
10	269	269	338	874	338	269	256
11	1233	1084	1192	1233	1192	1084	1022
12	680	585	639	1049	639	585	469
13	2245	2231	2488	2900	2504	2231	2230,5
14	1519	1677	1856	1999	1958	1519	1112,5
15	1659	1850	1913	2264	1912	1659	1598,5
16	13850	13850	13850	13850	13850	13850	13850
17	7069	7105	9128	7069	9128	7069	7038
18	4295	-	4739	4130	4739	4130	4130
19	28883	28883	28883	28883	28883	28883	28740 (41.77%)
20	23587	21898	-	22954	-	21898	18971 (36.8%)

Tabell 3. Simuleringsresultat från studien av betydelsen av val av sorteringsstrategi för kandidatlistan. Tidshorisonten är 90 minuter för samtliga scenarier och algoritmen har haft en max tillåten beräkningstid på 30 s. Den sekventiella algoritmen utgår ifrån den version som beskrivs i (Törnquist Krasemann, 2012).

6.6 En multistrategibaserad parallell girighetsalgoritm

Som resultaten i Tabell 3 ovan visar så är en kombination av de mest framgångsrika sorteringsstrategierna ett lämpligt sätt för att göra beräkningsmetoden mer effektiv och robust. I det senaste forskningsarbetet har vi kombinerat de olika strategierna på tre olika sätt; ”Approach 1-3” där varje ”worker” (dvs. parallell tråd) tilldelas en av de strategierna som beskrivits ovan och sorterar sin kandidatlista baserat på denna.

- *Approach 1: 1 strategi och 1 worker/kandidatevent*

  Här är antalet ”workers” samma som antalet events i kandidatlistan vid tidpunkten T_0 (när störningen inträffar). Alla workers använder samma strategi så denna approach har 5 olika varianter (en per strategi).

- *Approach 2: 5 olika strategier och 1 worker/strategi:*

  Här skapar vi 5 workers som får varsin av de fem olika strategierna.

- *Approach 3: 5 workers/kandidatevent:*

  Här skapar vi fem workers per event i kandidatlistan vid T_0 (i.e. minimum 5x50 = 250 workers i våra probleminstanser) där varje worker får varsin av de fem olika strategierna.

Det är i Approach 1 och Approach 3 vi kan få problem med resurstillgången (som diskuterats i kapitel 6.3) för de olika trådarna. Samtidigt är Approach 3 den som skapar flest olika möjliga lösningar. I Tabell 4 nedan återfinns en summering av resultaten som också håller på att sammanställas i en artikel till RailCopenhagen 2013. Vi kan se här att för Approach 3, som rimligtvis borde hitta samma eller bättre lösningar än i de andra två utföranden, så verkar resurstilldelningen bli ett problem. Detta visar sig också stämma när vi utökar beräkningstiden från 30 sekunder till över 15 minuter och då även Approach 3 hittar de bästa lösningar (dvs. i scenario 1,2 och 13).

Sc#	Parallel approach 1					Parallel approach 2	Parallel approach 3
	S0	S1α	S1β	S2	S3		
1	1172	995	1103	1486	1103	995	1103
2	437	288	396	751	396	288	366
3	781	686	740	1150	740	686	686
4	421	326	380	790	380	326	326
5	701	956	1338	1190	878	930	701
6	53	53	68	592	68	53	53
7	499	499	568	1109	568	499	499
8	207	332	401	1003	401	207	207
9	744	768	837	1349	837	768	744
10	269	269	338	874	338	269	269
11	1233	1084	1192	1233	1192	1084	1084
12	680	585	639	1049	639	585	585
13	2245	2231	2488	2900	2504	2231	2245
14	1519	1677	1888	1999	1888	1519	1519
15	1659	1844	1913	2264	1912	1659	1659
16	13850	13850	13850	13850	13850	13850	13850
17	7069	7105	9128	7069	9128	7069	7069
18	4295	4242	4739	4130	4739	4130	4130
19	28883	28883	28883	28883	28883	28883	28883
20	23144	21898	-	22954	-	21898	21898

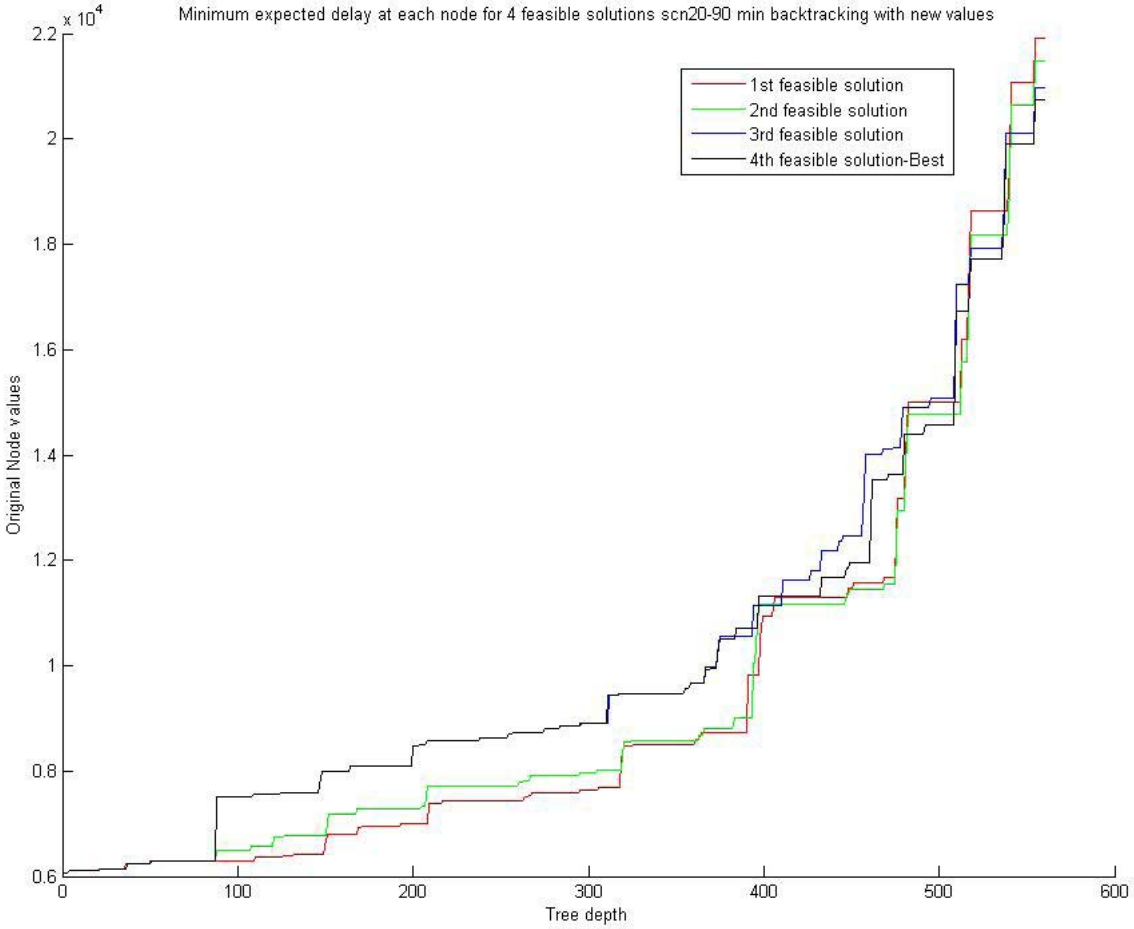
Tabell 4. Simuleringsresultat från analysen av olika parallelliseringsstrategier. Tidshorisonten är 90 minuter för samtliga scenarier och algoritmen har haft en max tillåten beräkningstid på 30 s. Den sekventiella algoritmen utgår ifrån den version som beskrivs i (Törnquist Krasemann, 2012).

Resultaten i Tabell 4 indikerar att ett lämpligt nästa steg är att anpassa Approach 2 så att åtta trådar används (dvs. lika många som antalet kärnor i vår nuvarande laborationsmiljö) och där flera trådar tillämpar samma strategi av de bättre strategierna. Det kan dock finnas

behov av ytterligare sorteringsstrategier och rutiner för algoritmen att inkludera beroende på vilka operativa prioriteringskriterier och andra riktlinjer som kommer att gälla framöver. Detta är dock en aspekt för fortsatt arbete och diskuteras inte vidare i denna rapport.

6.7 Förbättringar av den sekventiella girighetsalgoritmen

Basen för den parallella algoritmen är som bekant den sekventiella algoritmen. I dess initiala utformning (som beskrivs i Törnquist Krasemann, 2012) så har vi visat i föregående kapitel att den behöver justeras med avseende på sorteringsstrategin för att fungera bättre för vissa typer av störningar. Vi har också genomfört vissa andra förbättringar som genererar bättre lösningar i vissa scenarier men framför allt som gör den mer stabil. En viktig komponent som vi har studerat är hur algoritmen kan skapa bättre prognoser och approximationer av de följdeffekter ett visst beslut får längre ner i beslutsträdet. Figur 11 nedan visar hur approximation av lösningens slutvärde (dvs. den totala förseningen för samtliga tåg vid slutstation) beräknas i varje steg, så att ju längre ner i trädet (dvs. ju färre events som återstår att planera) desto mer exakt blir approximationen. Som vi kan se så är det svårt att få en tydlig och konsekvent bild av hur det approximativa värdet förhåller sig till slutvärdet.



Figur 11. Beskrivning av hur approximationen av målfunktionsvärdet för olika omplaneringsförslag utvecklar sig ju närmare fixering av alla tågrörelser algoritmen kommer i resp. förslag. I det ideala fallet vill man att de olika kurvorna inte skär varandra.

Ett sätt är att mäta andra egenskaper i dellösningen som kan ge ytterligare information om huruvida algoritmen är på väg ner i rätt nod och gren av trädet eller ej. I en mindre studie har projektet testat ett antal sätt. Studien och resultaten återfinns i Bilaga 2.

7 Utvärdering av algoritmen från ett driftperspektiv

I den teoretiska analysen, i Fas 2 resp. Fas 4, har vi utvärderat hur stabil och snabb algoritmen är och hur bra lösningar den genererar med avseende på teoretiska optimalvärden. Men minst lika viktigt är att analysera i vilken utsträckning algoritmens modell avspeglar trafiken och infrastrukturen i ett omplaneringsskede korrekt och ger förslag på rationella omplaneringsåtgärder. Denna typ av utvärdering planerade vi att göra framför allt i Fas 4.

7. 1 Utvärderingsaspekter

Primärt avser vi undersöka hur algoritmen beter sig med avseende på bland annat:

- Prioriteringen mellan olika tåg vid konflikter och hanteringar av sena resp. tidiga tåg.
- Val av tågvägar, spår och plattformar och trafikteknisk lämplighet.
- Representation av gångtider och dynamiken i trafiken.
- Beräkning och bedömning av omplaneringsförslagens effekter.

7.2 Planerat genomförande

I utvärderingen, med avseende på de aspekter som nämnts, planerade vi att tillämpa algoritmen i dess olika utföranden på en mängd simulerade störningsscenarier och fallstudier samt analysera dess beteende baserat på;

(i) De rekommendationer och riktlinjer som ges av relevant dokumentation från Trafikverket.

(ii) Analyser av trafikutfallet via Opera-data samt systematiska observationer via Trafikbilder.

(iii) Diskussioner med Trafikverket och då i synnerhet med personer som arbetar med, eller tidigare har arbetat med, tågtrafikledning och då med fördel på Södra Stambanan.

En viktig del av utvecklingsarbetet är naturligtvis att utvärdera om algoritmerna resulterar i relevanta lösningar och hur gränssnittet i tid och mot andra bandelar än de modellerade påverkar. Därför vore det önskvärt att även utföra någon typ av driftnära off-line utvärdering**.** Ett första steg (1) i en sådan vore att återskapa en trafiksituation där avvikelser förekommer samt omplanera denna trafiksituation ”off-line” med hjälp av algoritmen och jämföra utfallet i praktiken. I ett andra steg (2) önskar vi att i realtid få tillgång till

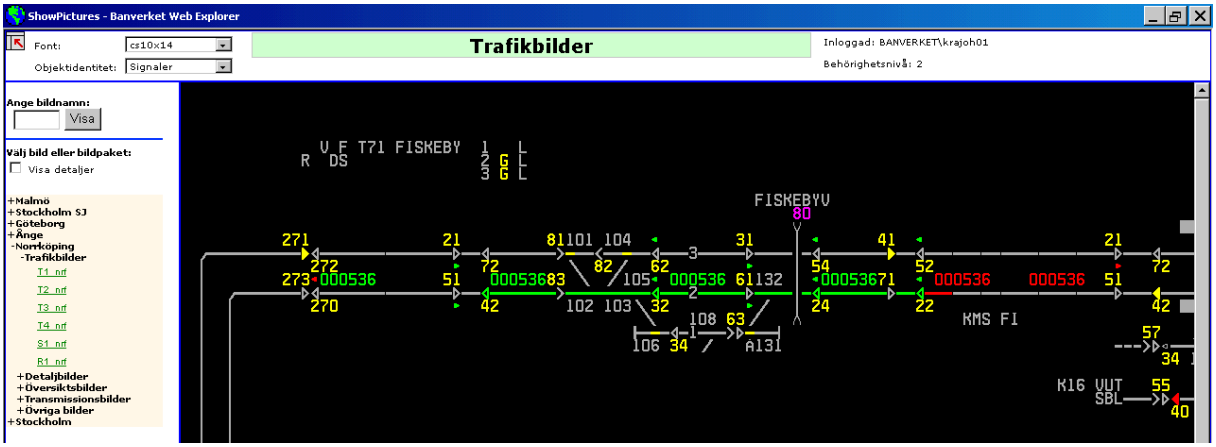
realtidsdata och ”vid sidan om” den ordinarie trafikledningen låta algoritmen iterativt revidera tidtabellen och ge omplaneringsförslag.

För att kunna återskapa hur tågtrafiken gick på en avgränsad del av järnvägsnätet under ett visst tidsintervall finns det i huvudsak två alternativa tillvägagångssätt där alternativ A- är det vi har möjlighet att genomföra med nuvarande tillgång till informationssystem. Alternativ A- är enormt arbetsintensivt och kräver mycket handpåläggning. Alternativ A+ är det vi ser som lämpligt för Steg 1 medan Alternativ B krävs för att genomföra Steg 2.

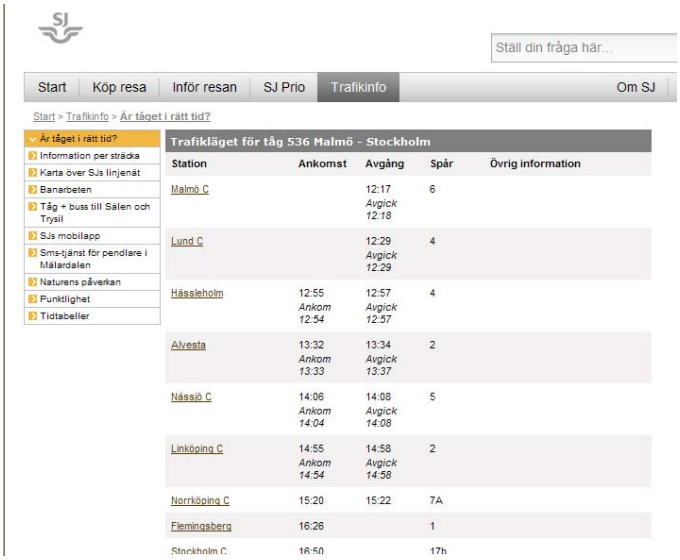
Alternativ A-
Den administrativa processen för att samla trafikdata följer:

a) Filtrera en textfil som innehåller hela tidtabellen (från TrainPlan) för det år som studeras (t ex T11) och sammanställ en lista som innehåller alla de tågnummer som har en planerad aktivitet under det tidsavsnitt och på den bandel som utgör fokus (t ex den 8 september 2011 kl 05:00-13:00 på sträckan Malmö-Norrköping-Katrineholm).
b) Sök manuellt ut i Opera för varje tågnummer enligt ovan och givet datum de passagetider som loggats. Passagetiderna anges i två olika format – ett där varje TPOS/ATL som loggats visas och ett där endast ankomst/avgång för stationer/trafikplatser visas. Ett tågs tidtabell beskrivs i TrainPlan i princip på motsvarande detaljnivå som den enklare loggen från Opera.

   Eftersom vi i nuläget inte har någon ”TPOS/ATL-karta”, dvs. en beskrivning över hur resp. loggad TPOS ska ”mappas” mot infrastrukturen samt den beskrivning vi har av resp. tågs tidtabell så kan vi inte använda den data (den utökade Opera-loggen) som då skulle kunna ge information om vilka spår tågen använt. Information om spåranvändning får vi istället bara ut delvis genom att i realtid studera hur tåg går via Trafikbilder (se Figur 10) samt Trafikinfo på webben (se Figur 11). Underlaget blir dock således inte komplett och godstågens tågvägar är mycket svåra att återskapa.



Figur 10. Screenshot av tågtrafiken kl 15.15 vid trafikplats Fiskeby söder om Norrköping och norr om Kimstad och Linköping.



Figur 11. Screenshot av ”Trafikinfo”-vyn där trafikerat spår för tåg 536 endast anges på de trafikplatser där tåget har haft ett planerat uppehåll.

Alternativ A+
Den administrativa processen för att samla trafikdata följer:

i) För angiven sträcka och tidsperiod sök automatiskt ut samtliga tågpassagetider (TPOS/ATL) kopplat till resp. tågnummer.
ii) Matcha resp. TPOS/ATL till en plats i järnvägsnätet baserat på en numerisk alt. grafisk ”TPOS/ATL-karta”.

Alternativ B
I alternativ B krävs någon form av stabil koppling mot TPOS/ATL-databank hos TrV som då ersätter i) ovan, så att omplaneringsalgoritmerna iterativt får nya passagetider och därmed kan uppdatera resp. tågs aktuella position.

7.3 Genomfört arbete och resultat

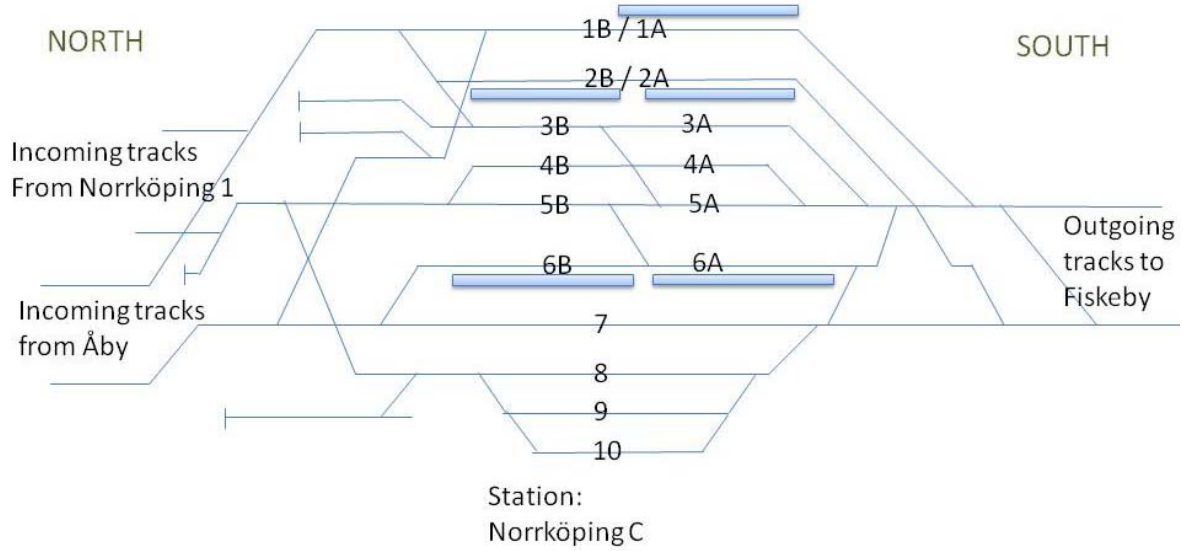
Under projektets Fas 3 och Fas 4 förde vi diskussioner med NTL-projektet för att få hjälp med att möjliggöra och genomföra den praktiska utvärderingen i olika steg, enligt beskrivningen i föregående kapitel. Det visade sig dock inte möjligt för Trafikverket att under den tidsperioden tillhandahålla den realtidsdata och koppling till informationssystemen som utvärderingen kräver eftersom det skulle kräva speciallösningar internt, mycket pga. t ex avsaknad av nuvarande exportmöjlighet från olika applikationer.

Vi hade alltså underskattat svårigheten och tidsåtgången att skapa förutsättningar för och ”rigga” en dylik praktisk utvärdering. Projektets tidsplan och leveransplan för Fas 4 fördröjdes därmed, vilket ledde till att Trafikverket fick en förfrågan från BTH i oktober 2012 om förlängning av projektet t.o.m. september 2013 och som beviljades. Arbetet under 2013 syftar därför till att skapa goda förutsättningar för att i ett senare skede utföra en praktisk utvärdering av projektets beräkningsmetoder. Således har projektet vid denna

tidpunkt (november 2012) inte kunnat leverera resultat från en praktisk utvärdering och därmed inte heller fullt ut kunnat anpassa beräkningsmetoderna i linje med de rekommendationer som utvärderingen skulle visat. Vissa analyser med syfte att undersöka potentiella problem och frågeställningar som rimligtvis kommer att aktualiseras har dock genomförts. Dessa analyser har baserats på observationer via Trafikbilder och Opera samt material från STEG-projektets arbete. En uppenbar frågeställning som är viktig att beakta men som inte går att svara på fullt ut i nuläget är t ex på vilken detaljnivå och med vilken noggrannhet (i tid) som tågen bör omplaneras och hur tågrörelser inne på mer komplexa stationer bör beskrivas och planeras. I tidtabellskonstruktionen planerar man ju inte i detalj hur trafiken ska interagera på stationerna, vilket då leder till att det saknas tillförlitlig data om hur alla tåg enligt plan ska trafikera stationerna.

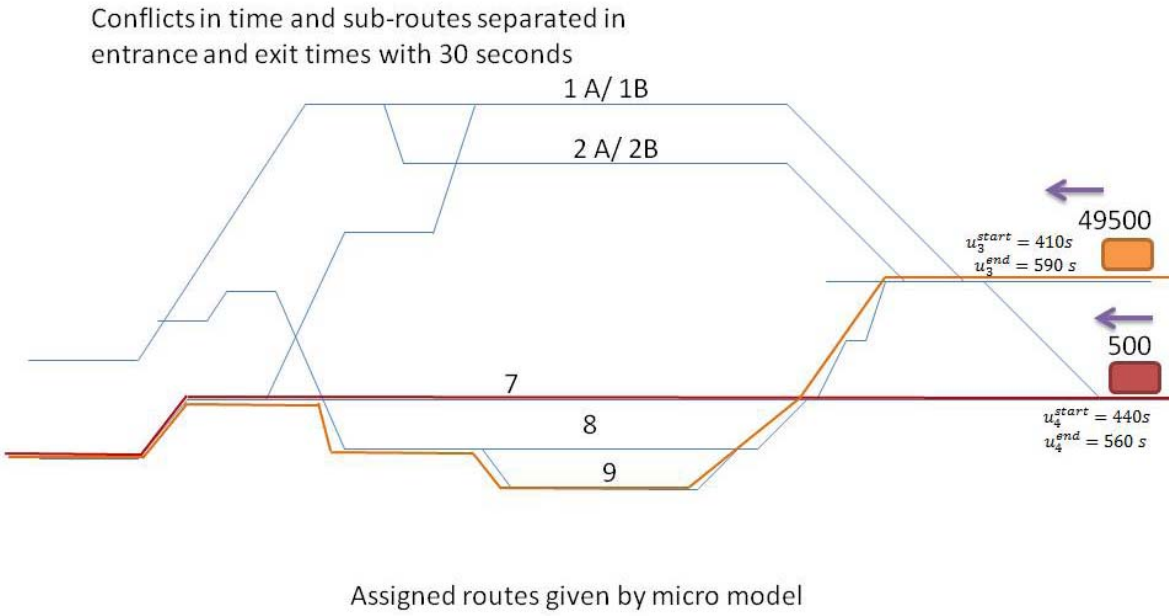
Det är ju de underliggande modellerna, vilka beskriver hur tågen får röra sig i nätverket och i förhållande till varandra, som ska säkerställa att omplaneringsförslagen är trafiktekniskt genomförbara. Det är dock alltid en avvägning på vilken detaljnivå olika komponenter ska beskrivas och projektet har studerat olika nivåer av detaljrikedom. Eftersom gångtiderna i trafiken är relativt approximativa finns det ingen poäng i att planera trafiken på sekundnivå. Däremot finns det ett behov av större noggrannhet ju närmare tiden för exekvering av planen man kommer samt för mer komplexa stationer och knytpunkter. Man behöver också skilja på 1) den beskrivning av trafiken man använder när man söker efter omplaneringsförslag och 2) den beskrivning man använder för att eventuellt detaljplanera, lägga tågvägar och säkerställer att den grövre planen från 1) håller. Hur dessa båda nivåer bör samverka och effekten av detta är svår att utreda och besluta om på en generell nivå utan beror på det specifika fallet och de krav användaren ställer. Eftersom projektet har fokuserat på sträckan Katrineholm-Norrköping-Mjölby samt sträckan Norrköping-Nyköping-Järna så har vi studerat Åby och Norrköping i mer detalj (se Bilaga 3 för en mer utförlig beskrivning av arbetet och resultat).

När det gäller Norrköpjng så är det en relativt komplex station, även om det finns flera mycket mer komplexa stationer i det svenska järnvägsnätet. Komplexiteten uppstår på grund av de många alternativa tågvägar som finns och som kan vara i konflikt med varandra genom att tågvägar korsas, vissa stationsspår inte tillåter samtidiga infarter och utfarter mm, se Figur 12.



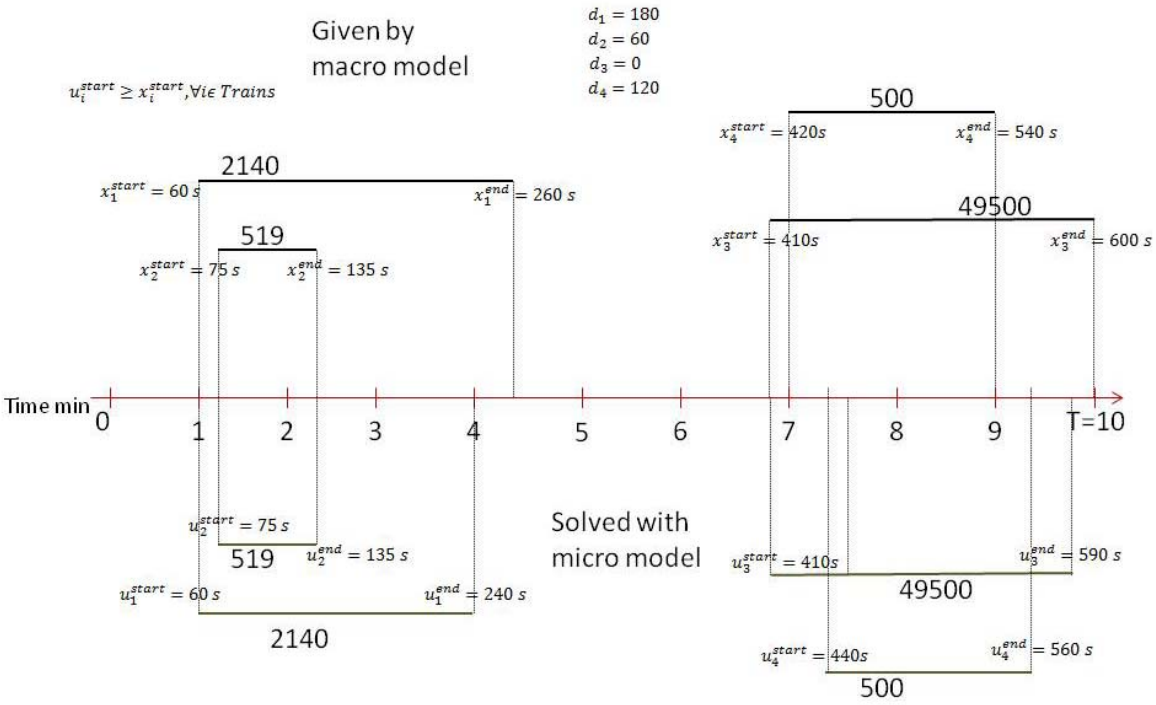
Figur 12. Schematisk bild över spårlayout för Norrköpings järnvägstation.

I våra experiment har vi primärt tillämpat en grov beskrivning av Norrköping, där vi anger att stationen har 14 individuella spår, 2 infarter/utfarter från båda håll och där samtliga spår når samtliga in- och utfarter. Så länge tågen inte använder samma stationsspår eller infart/utfart anses de inte vara i konflikt. Som det framgår av Figur 12 och Figur 13 finns det därmed en risk att icke-samtidiga tågrörelser inplaneras felaktigt av grova modellen pga. otillräckligt noggrannhet i problembeskrivningen. Problemet tydliggörs i Figur 13 som visar ett snitt av Norrköpings station där tågvägarna för tåg 500 och 49500 är i konflikt och behöver separeras i tid både vid infart och ufart.



Figur 13. Illustration över hur tågvägarna för de två fiktiva tågen 500 och 49500 är i konflikt och behöver separeras både vid infart och ufart.

Figur 14 illustrerar hur ankomst- och avgångstiderna för fyra olika tåg - däribland 500 och 49500 från exemplet ovan - är planerade på Norrköping station enligt den grövre modellen (dvs. ”macro model”) medan en mer detaljrik modell (”micro model”) särskiljer tågrörelserna utifrån makromodellens beslut om tågordning, spårval mm så att planen blir trafiktekniskt genomförbar. Tågvägarna åtskiljs inte tillräckligt av enbart makromodellen utan behöver justeras lite ytterligare av mikromodellen, vilket framgår av figuren.



Figur 14. Schemaläggning av tågrörelser på Norrköpings järnvägstation i två olika steg – på makronivå och mikronivå.

Sannolikt är detta sätt – att dela upp problemet i två olika nivåer – lämpligaste för större och mer komplexa stationer medan t ex Åby är tillräckligligt liten för att beskrivas fullt ut i makromodellen. Den typen av mikromodell som utvecklats för och tillämpats på Norrköping kräver dock att alla relevanta tågvägar och deras beroenden finns väl beskrivet. I projektet har vi inte haft tillgång till denna typ av data utan kartläggningen och beskrivningen av stationen har varit manuell och mycket arbetsintensiv.

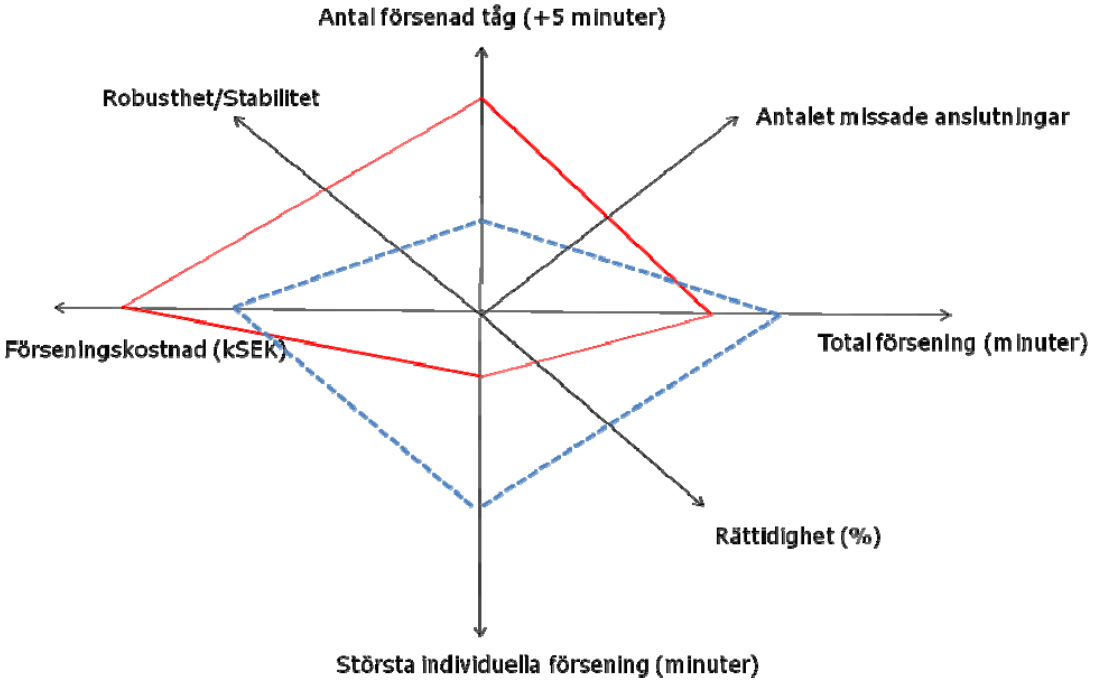
Diskussionen ovan avser hur man säkerställer huruvida ett förslag är trafiktekniskt tillåtet eller inte. Det finns även en gråzon för vad som är lämpligt eller ej. Som exempel kan nämnas att det kan betraktas som olämpligt att byta spår/plattform för vissa pendeltåg av olika skäl och i synnerhet om det sker nära avgångstiden. Det kan röra sig om anslutningar till andra tåg och bussar men också att det bör undvikas för att inte skapa förvirring bland de pendlare som per automatik stiger på tåget som står på en viss plattform vid en viss tidpunkt varje vardagsmorgon av ren vana.

Huruvida ett förslag är lämpligt eller inte beror ju även på vilka antaganden förslaget bygger på såsom gångtidsinformationen och dess tillförlitlighet, hur ”snäva” t ex förbigångar och mötena är planerade och tågens faktiska förmåga att accelerera och

faktiska uppehållstider för resandeutbyten mm. En alltför optimistisk plan bör rimligtvis undvikas om det finns alternativa, mer robusta förslag.

Just egenskapen ”robusthet” eller ”stabilitet” (dvs. förslagen är relevanta även om t ex tågens gångtid är något längre än planerat) är en av de egenskaper vi tror kan vara viktig att studera och beakta. Det finns givetvis en mängd andra viktiga egenskaper och effekter av ett omplaneringsförslag som bör mätas vid en eventuell ranking av alternativa förslag. Idealiskt vore om de olika målen och motsvarande effektmått med omplaneringen skulle vara korrelerade så att om man optimerar mot det ena förbättras även övriga, eller de försämras åtminstone inte. Det är dock ett välkänt faktum att det ibland uppstår konflikter mellan de olika målen. Det kan vara bättre att merförsena ett redan försenat tåg och prioritera rättidiga tåg i ett fall medan i en annan situation kan det vara bättre att bara få flyt i trafiken och få fram tågen även om det blir vissa följdförseningar på flera tåg. Eftersom Trafikverket mäter och redovisar punklighet i form av andelen tåg som anländer till slutstation med en försening på maximal 5 minuter dvs. rättidighet, så är det rimligtvis ett av de mått som bör användas. Samtidigt har Trafikverket under 2012 infört kvalitetsavgifter vilka skulle kunna utgöra någon form av underlag för bedömning av effekterna av olika prioriteringsalternativ. Det är således inte helt uppenbart hur alternativa omplaneringsförslag bör bedömas och rankas.

I Figur 15 nedan illustreras hur två fiktiva alternativa förslag förhåller sig till varandra beroende på olika mått. Som framgår av illustrationen så medför omplaneringsförslaget som representeras av den blå, streckade linjen en bättre rättidighet med färre försenade tåg och brustna anslutningar medan det ger upphov till en totalt större försening för trafiksystemet och de tåg som försenas blir kraftigt försenade.



Figur 15. Illustration över hur alternativa förslag kan värderas och jämföras.

Med detta som utgångspunkt initierades ett examensarbete vid BTH som utfördes under 2011 av två icke-svensktalande magisterstudenter. Syftet med arbetet var att undersöka och analysera vilka målfunktioner och mått som andra forskare inom detta område har använt och hur dessa kan användas för svenska förhållanden. Arbetet, se (Kumar Karthikeyan och Kumar Mani, 2012) resulterade huvudsakligen i en kartläggning av relaterad forskning och hur olika mått och målfunktioner används i andra länder och av andra forskare.

I flera av våra experiment har vi beräknat effekterna av alternativa förslag och studerat hur alternativen strukturellt skiljer sig åt. Nedan ges ett exempel för scenario 20 och i Tabell 5 anges fyra alternativa förslag och vilka tåg som försenas i resp. förslag och hur mycket. I Tabell 6 har vi beräknat de mått som gäller för resp. förslag. Baserat på den primära målfunktion vi använt för algoritmen samt i modellen till CPLEX så är det fjärde förslaget (från CPLEX) mest optimalt men om vi utgår från rättidighet så är det tredje förslaget att föredra. Hur en bedömning och värdering av alternativa förslag skulle gå till i praktiken är mer komplicerat än så men detta är en utgångspunkt för vidare diskussioner och konkretisering.

Tåg	**Alternativa förslag fr. den sekventiella algoritmen***			**Cplex 12.2, 24h**
	Försening för resp. tåg (minuter)			
237	**19**	**19**	**19**	15
80866	3	3	3	3
242	29	29	24	**40**
538	**12**	**12**	**12**	11
539	**28**	**28**	**28**	13
540	20	20	16	**32**
5629	**2**	**2**	**2**	0
8313	**30**	**30**	0	0
8762	**9**	**9**	**9**	8
8764	**25**	15	15	14
8766	0	0	0	**6**
8767	11	11	11	11
8768	32	32	28	**48**
8769	9	9	9	9
8770	**52**	**52**	32	34
8771	**34**	**34**	**34**	19
8773	**29**	**29**	**29**	14
8841	3	3	3	3
8854	0	**3**	0	0
42702	49	49	**50**	35

Tabell 5. *Avser den förbättrade sekventiella algoritmen, se kap.6.7.

Mått	Alternativa förslag fr. den sekventiella algoritmen			Cplex 12.2, 24h
	1	2	3	
Rättidighet (totalt 48 tåg)	68,8	68,8	70,8	68,8
Total försening (min)	396	389	322	315
Total försening +5min (min)	388	378	315	309
Max försening +5min (min)	52	52	50	48
Medel försening +5min (min)	26	25	22	21
Minimi försening +5min (min)	9	9	9	6
Antal försenade tåg (+5min)	15	15	14	15
Antal försenade tåg (+15min)	11	10	9	6

Tabell 6. *Avser den förbättrade sekventiella algoritmen, se kap.6.7.

8 Diskussion och fortsatt arbete

Resultaten från projekten har visat på potentialen att stödja trafikledarnas arbete och komplettera arbetet som utförs inom ramen för STEG, NTL och de projekt som Uppsala Universitet utför. Eftersom algoritmerna ännu inte testats i samverkan med framtida användare och system så är det dock mycket arbete som återstår med att planera för och genomföra en första praktisk utvärdering. I fortsättningen av EOT, som vi benämner FLOAT, avser vi ta detta nästa steg och då delvis genom att utföra fallstudier och experiment i samråd med bl.a. personer som arbetat med dessa frågor i testerna av STEG/CATO på Malmbanan samt med personer verksamma inom NTL-projektet. Dessa personers roll i FLOAT är mycket viktiga eftersom val av testmiljö och plan för genomförande av den praktiska utvärderingen är en nyckelfråga.

Beräkningsmetoderna ska följaktligen också vidareutvecklas och anpassas iterativt i linje med de resultat och rekommendationer som såväl den praktiska som den teoretiska utvärderingen leder fram till. Arbetet inom FLOAT skulle kunna delas upp i fyra olika delar:

A. Kartläggning av behov och omgivning
Denna del utförs i samverkan med framför allt personer som arbetet i STEG-projektet. Centrala frågor är bl. a:

- I vilka huvudsakliga typer av situationer och driftstörningar finns det ett behov av beräkningsstöd?
- Hur bör interaktionen med trafikledarna och omgivande stöd- och informationsystem se ut?
- Vilka frihetsgrader ska beräkningsstödet ha vid framtagandet av förslag på omplaneringslösningar och hur bör detta handlingsutrymme definieras?

- Hur bör man värdera alternativa omplaneringslösningars genomförbarhet och lämplighet samt kommunicera denna bedömning till trafikledarna?

B. Planering och genomförande av praktisk utvärdering samt demonstration
Denna del syftar till att kartlägga hur vi skapar de förutsättningar som krävs för framgångsrikt kunna ”rigga” en praktisk utvärdering i driftnära miljö (”offline” men i realtid). Arbetet har en planeringsfas som innefattar att skapa en utvärderingsgrupp där medarbetare från huvudsakligen Trafikverket och BTH har en tydlig och aktiv roll, att diskutera vad som är en rimlig teststräcka, specificera vad som bör utvärderas, när och på vilket sätt. När planen för utvärderingen är gjord och projektet säkerställt att denna går att genomföra så utförs den och resultaten följs upp.

C. Iterativ metodutveckling och teoretisk utvärdering
Denna del syftar till att vidareutveckla och förbättra beräkninsgmetoderna iterativt och i linje med de resultat och rekommendationer som arbetet i del A) och såväl den praktiska som den teoretiska utvärderingen leder fram till. Metodutvecklingen sker internt på BTH som då säkerställer tillgång till relevant mjuk- och hårdvara för detta arbete.

D. Dokumentering, avrapportering och spridning av resultat
En betydande del av projektet handlar om att dokumentera och presentera hur arbetet löper, vilka resultat och slutsatser projektet når fram till och säkerställer att delleveranser sker till uppdragsgivaren enligt avtal.

En betydande del av arbetet bör ske i samverkan med omgivande projekt och potentiella användare. De studier som genomförts inom projekten FTTS/STEG av Uppsala Universitet där man undersökt vilka former av beslutstöd som trafikledarna i olika situationer och sammanhang har behov av kommer att lägga en grund för det arbete som ska utföras inom del A) beskrivet ovan.

Medverkan av personer med erfarenhet från STEG/CATO/ NTL-projekten skulle också främja arbetet i del B), dvs. i planeringen och utförandet av den praktiska utvärderingen samt i del C) för att diskutera omgivningen ur ett framtida perspektiv, vilka krav man ställer på beräkningstid, visuellt stöd, mm.

Eftersom Trafikverkets arbete i ON-TIME-projektet också avser demonstration och utvärdering av snarlika stödfunktioner så skulle FLOAT –projektet kunna stödja och stödjas av detta arbete i den mån det är lämpligt.

9 Spridning av projektresultat

Projektet har presenterats vid ett flertal publika tillfällen under 2010-2012 däribland vid:

- Transportforum 2011 och 2012
- SOAFs järnvägsseminarium i Stockholm (April 2011)
- Logistikklustermöte i Karlshamn med Trafikutskottets Anders Ygeman (Maj 2011).

- Konferensen MT-ITS2011 i Leuven, Belgien (Juni 2011).
- Projektmöte med NTL-projektet i Göteborg (Oktober 2011).
- INFORMS Annual Conference i Charlotte, USA, (November 2011).
- Konferensen MCC-2011 (4th Swedish Workshop on Multicore Computing) i Linköping (November 2011).
- Konferensen ICORES i Vilamoura, Portugal (Februari 2012).
- Forskningsseminarium om meta-heuristiker i Aalborg, Danmark (Februari 2012).
- Forskningsseminarium arrangerat av BTH under Almedalsveckan (Juli 2012)
- Konferensen Comprail i England (September 2012)
- Forskningsseminarium samt projektpresentation för ARRIVA i Norrköping (Oktober, November 2012)
- Utbildningsutskottets frukostmöte i Riksdagen, Stockholm (November 2012)

Resultat från projektet har också publicerats i följande artiklar och rapporter:

J. Törnquist Krasemann (2012), "Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances", *Transportation Research Part C 20* (2012) pp. 62-78.

H. Grahn and J. Törnquist Krasemann, "A Parallel Re-Scheduling Algorithm for Railway Traffic Disturbance Management --- Initial Results", *Proc. of the 2nd Int'l Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS 2011)*, June 2011, Leuven, Belgium.

A. Kumar Karthikeyan, P. Kumar Mani, *Visual and Analytical Support for Real-time Evaluation of Railway Traffic Re-scheduling Alternatives During Disturbances*, Master thesis, Blekinge Tekniska Högskola, Januari 2012.

S.M.Z. Iqbal, H. Grahn, and J. Törnquist Krasemann (2011), "A Parallel DFS Algorithm for Train Re-scheduling During Traffic Disturbances --- Early Results", *Proc. of the 4th Swedish workshop on Multicore Computing (MCC-11)*, pp. 115--120, November 2011, Linköping, Sweden.

S.M.Z. Iqbal, H. Grahn, J. Törnquist Krasemann (2012), "A parallel heuristic for fast train dispatching during railway traffic disturbances: Early results", *Proceedings of ICORES*, pp.405-414, Feb. 2012.

S.M.Z. Iqbal, H. Grahn, J. Törnquist Krasemann (2012), "A comparative evaluation of re-scheduling strategies for train dispatching during disturbances", *Proceedings of COMPRAIL 2012*, Wessex Institute of Technology, UK, pp. 567-579, Sept.2012.

10 Ekonomisk redovisning

Den ekonomiska redovisningen finns i Bilaga 4.

11 Referenser

Cordeau, J-F., Toth, P., Vigo, D.(1998), "A survey of optimization models for train routing and scheduling", *Transportation Science*, Vol. 32, No. 4, INFORMS, pp. 380-404, 1998.

Emer, J.,Hill, M., Patt, Y., Yi, J., Chiou, D., Sendag, R. (2007), "Single-threaded vs. multithreaded: Where should we focus?" *IEEE Micro*, 27(6), pp. 14–24, 2007.

Grahn, H., Törnquist Krasemann, J., (2011), "A Parallel Re-Scheduling Algorithm for Railway Traffic Disturbance Management --- Initial Results", *Proc. of the 2nd Int'l Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS 2011)*, Leuven, Belgium, Juni 2011.

Grimm, M., Wahlborg, M., *Kapacitetsutnyttjande och kapacitetsbegränsningar hösten 2011*, TRAFIKVERKET, TRV 2012/5114, 2012-01-25. http://www.trafikverket.se/PageFiles/46897/kapacitetsutnyttjande_och_kapacitetsbegransningar_hosten_2011.pdf

Iqbal, S.M.Z., Grahn, H., Törnquist Krasemann, J., (2012a), "A parallel heuristic for fast train dispatching during railway traffic disturbances: Early results", *Proceedings of ICORES 2012*, pp.405-414, Feb. 2012.

Iqbal, S.M.Z. Grahn, H., Törnquist Krasemann, J. (2012b), "A comparative evaluation of re-scheduling strategies for train dispatching during disturbances", *Proceedings of COMPRAIL 2012*, Wessex Institute of Technology, UK, pp. 567-579, Sept.2012.

Isaksson-Lutteman, G. (2012), *Future Traffic Control: Development and deployment of new principles and systems in train traffic control,* Licentiate thesis 2012-001, Uppsala Universitet, Sverige.

Kumar Karthikeyan, A., Kumar Mani, P.(2012), *Visual and Analytical Support for Real-time Evaluation of Railway Traffic Re-scheduling Alternatives During Disturbances*, Master thesis, Blekinge Tekniska Högskola, Januari 2012.

Kroon, L., Romeijn, E., Zwaneveld, P. (1997), "Routing trains through railway stations: complexity issues", *European Journal of Operational Research*, No. 98, pp. 485-498, 1997.

Radtke, A. (2008) "Infrastructure modelling", in: Hansen and Pachl (Eds.): *Railway Timetable & Traffic. Analysis, Modelling, Simulation*, Hamburg: Eurailpress, 2008.

Schachtebeck, M. (2009), *Delay Management in Public Transportation: Capacities, Robustness, and Integration*, PhD thesis, Universitetet i Göttingen, Tyskland.

Törnquist, J. (2005), "Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms", *Proceedings of ATMOS2005*, Palma de

Mallorca, Spain, October 2005, Published within the Dagstuhl Research Online Publication Server (DROPS) http://drops.dagstuhl.de/portals/ATMOS/, 2005.

Törnquist Krasemann, J. (2012), "Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances**",** *Transportation Research Part C (20),* pp. 62-78, 2012.

Bilagor.

Bilaga 1.

J. Törnquist Krasemann, (2012), "Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances**",** *Transportation Research Part C (20),* pp. 62-78, 2012.

S.M.Z. Iqbal, H. Grahn, J. Törnquist Krasemann (2012), "A parallel heuristic for fast train dispatching during railway traffic disturbances: Early results", *Proceedings of ICORES 2012.*

S.M.Z. Iqbal, H. Grahn, J. Törnquist Krasemann (2012), "A comparative evaluation of re-scheduling strategies for train dispatching during disturbances", *Proceedings of COMPRAIL 2012*, Wessex Institute of Technology, UK, 2012.

Bilaga 2.

S. Soltani, *Improving the branch and bound greedy algorithm by introducing a new objective function: Implementation, results and analysis*, Project report, 2012-06-15.

Bilaga 3.

S. Soltani, *Modeling train stations: case study Norrköping*, Project report, 2012-06-14.

Bilaga 4.

Ekonomisk redovisning

Bilaga 1

J. Törnquist Krasemann, (2012), "Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances**",** *Transportation Research Part C (20),* pp. 62-78, 2012.

S.M.Z. Iqbal, H. Grahn, J. Törnquist Krasemann (2012), "A parallel heuristic for fast train dispatching during railway traffic disturbances: Early results", *Proceedings of ICORES 2012.*

S.M.Z. Iqbal, H. Grahn, J. Törnquist Krasemann (2012), "A comparative evaluation of re-scheduling strategies for train dispatching during disturbances", *Proceedings of COMPRAIL 2012*, Wessex Institute of Technology, UK, 2012.

Transportation Research Part C 20 (2012) 62–78

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances

Johanna Törnquist Krasemann *

School of Computing, Blekinge Institute of Technology, Box 214, 374 24 Karlshamn, Sweden

ARTICLE INFO

Article history:
Received 24 September 2009
Received in revised form 15 December 2010
Accepted 16 December 2010

Keywords:
Railway traffic
Scheduling
Timetabling
Slot allocation
Disturbance management
Optimization

ABSTRACT

An attractive and sustainable railway traffic system is characterized by having a high security, high accessibility, high energy performance and offering reliable services with sufficient punctuality. At the same time, the network is to be utilized to a large extent in a cost-effective way. This requires a continuous balance between maintaining a high utilization and sufficiently high robustness to minimize the sensitivity to disturbances. The occurrence of some disturbances can be prevented to some extent but the occurrence of unpredictable events are unavoidable and their consequences then need to be analyzed, minimized and communicated to the affected users. Valuable information necessary to perform a complete consequence analysis of a disturbance and the re-scheduling is however not always available for the traffic managers. With current conditions, it is also not always possible for the traffic managers to take this information into account since he or she needs to act fast without any decision-support assisting in computing an effective re-scheduling solution. In previous research we have designed an optimization-based approach for re-scheduling which seems promising. However, for certain scenarios it is difficult to find good solutions within seconds. Therefore, we have developed a greedy algorithm which effectively delivers good solutions within the permitted time as a complement to the previous approach. To quickly retrieve a feasible solution the algorithm performs a depth-first search using an evaluation function to prioritise when conflicts arise and then branches according to a set of criteria.

1. Introduction

Freight as well as public transportation is a large and important part of our economy and daily life, and railway transportation has a significant share. As an effect of the increasing environmental awareness and desire to decrease emissions, noise pollution and accidents, political aims of increasing the market share of railway freight transportation have been stated, see e.g. (European Commission, 2001). However, in line with the increasing traffic flow and density, the railway networks are facing difficulties with congestion and insufficient reliability. The railway traffic networks in several European countries and regions are partly oversaturated, highly sensitive to even small disturbances and have low average punctuality (Jansson and Jonsson, 2003). For instance, some parts of the Swedish railway network have such a high capacity usage that even a minor incident can propagate and cause large disturbances. During the two most traffic-intensive hours per day, 34% of the entire Swedish railway network is considered saturated (capacity utility of 81–100%) with a low average speed and high sensitivity to disturbances (Banverket, 2009). Statistics shows that 31% of the reported delays in the Swedish network during 2006 were knock-on delays. Furthermore, recent statistics from the UIRR (the International Union of combined Road-Rail

* Tel.: +46 702 667683; fax: +46 457 271 25.
E-mail address: Johanna.Tornquist@bth.se

0968-090X/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trc.2010.12.004

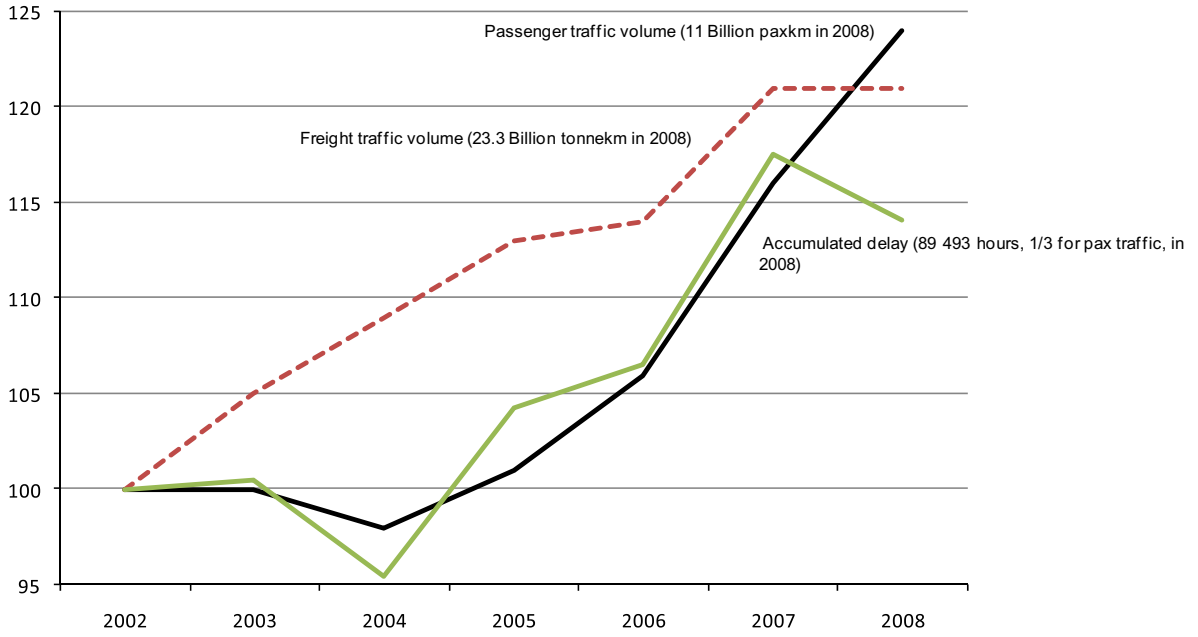


Fig. 1. The growth of the Swedish railway traffic volumes and accumulated delay between year 2002 and 2008 (2002 has index 100). *Source:* (Banverket, 2005–2009).

transport companies) also show that 41% of the cargo trains in the analysed major freight transport corridors arrived 30 min later than scheduled, or more (UIRR, 2008) (see Fig. 1).

The reduction of accumulated delay during 2008 is composed of a delay reduction for the freight traffic from 65,829 h in 2007 to 61,181 h while the delay instead increased for the passenger traffic from 26,377 h to 28,312 h. The corresponding numbers for 2009 are 57,601 and 31,002 h of delay (Banverket, 2009). During 2008 and 2009, several measures have been taken to prevent disturbances from occurring in the first place rather than improving the strategies to limit knock-on delays.

In order for the railway to become and remain an attractive means of transportation, the occurrence of disturbances needs to be limited as well as the consequences of the disturbances that do occur. While the most straightforward way to decrease the effects of disturbances would be to eliminate the risk of primary disturbances arising, it is simply not feasible. Some of the causes can be predicted and prevented from happening, while others cannot. Therefore the ability to deal with the disturbances that do occur can be argued to be as important as eliminating potential causes of primary disturbances. This paper focuses on how to handle disturbances in railway traffic by effective re-scheduling rather than limiting the occurrence of primary disturbances (i.e. initial disturbances).

We will present previous results from a research project at Trafikverket (formerly named Banverket, the Swedish Rail Administration) where focus is on the design of an algorithm for re-scheduling the traffic during or after a disturbance. The driving force is to achieve sufficiently good solutions within a short time. The performance of the algorithm has been evaluated for different types of disturbances in simulated experiments. The simulation experiments have been carried out on one of the densest and central parts of the Swedish railway traffic network and have been based on real data provided by Trafikverket.

2. Railway traffic disturbance management

In this paper, a disturbance is considered to be a situation where the master schedule, or established timetable, has become invalid because one train (or several) is deviating from its schedule and the prerequisites consequently change. The event triggering a disturbance could be a signal malfunction on a track section which temporarily decreases the maximum allowed speed and causes the trains to have an increased running time on that section. It could also be e.g. a no-show of staff resulting in a delayed train departure, or reduced speed of a train set due to partial engine failure or an unannounced increase in train set length and weight, etc.

Handling a disturbance in a railway network and re-scheduling the traffic is typically handled manually by traffic managers that only have very limited access to support systems to make use of relevant information, analyze the consequences of the disturbance as well as the effects of their decision-making. This limitation hampers the possibilities to achieve system-optimal decision-making and to provide the stakeholders (e.g. train operators, cargo owners, commuters) with reliable prognoses about the situation. The time available for decision-making and consequence analysis is also limited and depends on the situation. The aim is thus to find robust and sufficiently good solutions within reasonable time. For illustrative purposes,

let us consider the small-scale example of re-scheduling the traffic in Fig. 2. It shows a graph of the planned railway traffic (train 101, 102 and 103) on the single-tracked line between the stations Enstaberg and Åby with several intermediate stations, and where the time is on the y-axis. The trains belong to three different private transport companies; 101 is a heavy block train, 102 is a long-distance high-speed passenger train while 103 is an express cargo train. Shortly before train 101 leaves Ålberga for Kolmården the track section suffers from a signalling problem (indicated by the thick line below the section) which decreases the maximum speed to 70 km/h. Thus, all trains passing through get an increased running time of

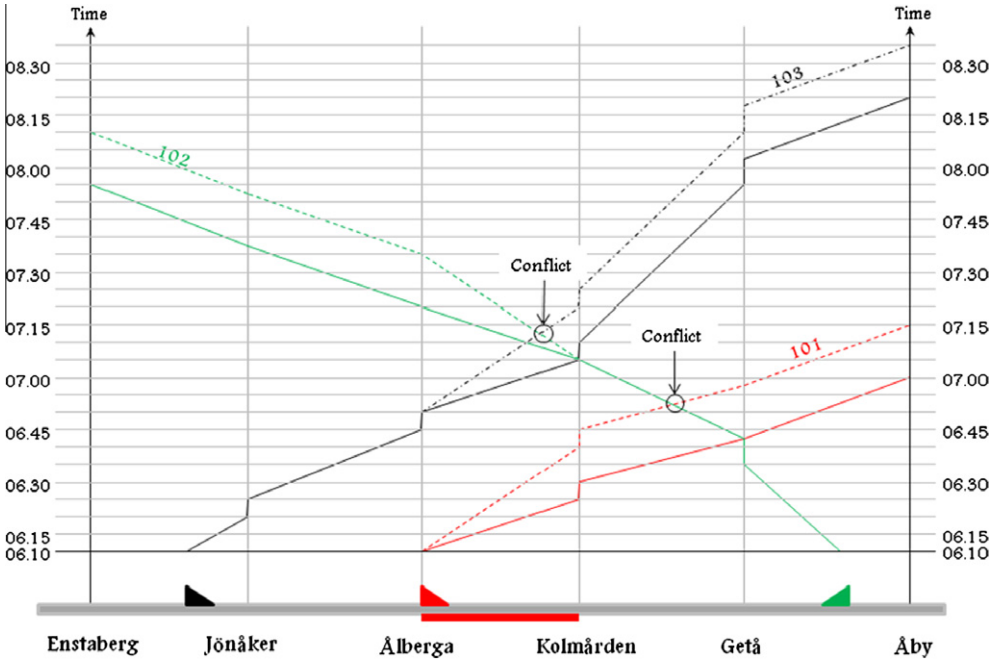


Fig. 2. The initial schedule of the railway traffic and the consequential change of arrival times when a signaling problem occurs generating a revised schedule (indicated by the dotted lines) with two conflicts which need to be resolved into a conflict-free revised schedule.

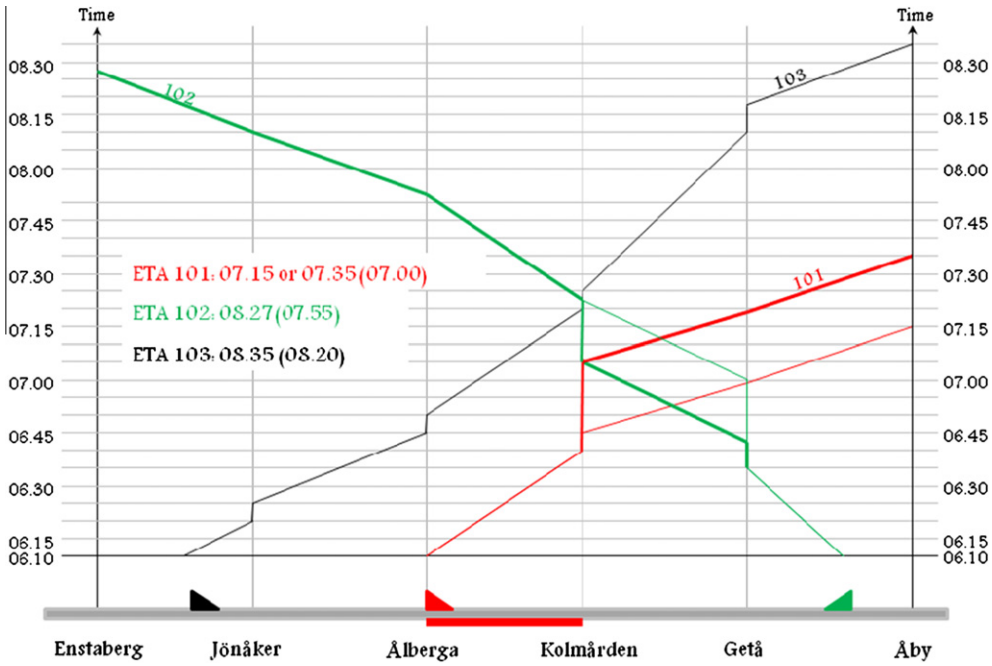


Fig. 3. Two alternative solutions to the disturbance problem illustrated in Fig. 2.

15 min and the revised timetable for each train (the dotted lines) is thus postponed 15 min each. This results in conflicts when the paths of the trains cross on the single-tracked line between stations, and that is forbidden due to safety restrictions. When and how the trains now should meet in order to resolve the conflicts need to be decided by the traffic managers. One policy could be to let the first train that can enter a section be given priority to that section, i.e. First Come, First Served (FCFS) while another could be for example to give highest priority to all long-distance trains.

In Fig. 3, two solutions are illustrated. In the first solution, train 102 and 101 meet at Getå (as in the initial timetable) but that forces train 102 to wait a long time at the station and since the train is a long-distance high-speed train, the corresponding operator does not accept that it will be delayed that much in favour of a cargo train. Instead, the operator wants the traffic manager to arrange for the trains to meet in Kolmården, thereby reducing the waiting time for 102 at Getå significantly. When the traffic manager considers that alternative he discovers that in the end, the waiting time for train 102 over all stations will be the same in both solutions, while in the second solution train 101 will get an additional delay. So, at first it seems like train 102 is disfavoured by the first solution while that solution turns out to be the best one. The initial and updated ETA (Estimated Time of Arrival) for each train is given in Fig. 3.

The small example illustrated by Figs. 2 and 3 describes how difficult it is to achieve robust re-scheduling solutions and to communicate the motivation behind each decision to the operators so that they understand the benefits of co-operating and following the plans made by the traffic managers. Today, the safe strategy is often to keep the important trains rolling and prioritize them to reduce the risk of them becoming further delayed, but it does not always mean that it is the best solution.

3. Related work

The research analyzing effects of disturbances in railway traffic and approaches for disturbance management is extensive. Some studies focus on the primary (i.e. initial) reasons behind the disturbances and preventive counter measures, others focus on the approximation of disturbance propagation effects, while some address the topic of this paper, i.e. approaches for scheduling and re-scheduling railway traffic. Surveys by Assad (1980), Cordeau et al. (1998) and Törnquist (2005) provide an extensive overview of these. More recent re-scheduling approaches are presented by Zhou and Zhong (2007), who propose a branch-and-bound algorithm for railway traffic timetabling in a single-tracked network, and D'Ariano et al. (2007, 2009), who present an iterative algorithm that in real-time resolves conflicts in a pertubated railway traffic timetable. Since the approach suggested by Zhou and Zhong (2007) is developed for a single-tracked network it is not applicable for the problem setting we are facing in Sweden where certain lines have up to four parallel and bi-directional tracks. Furthermore, the approach suggested by D'Ariano et al. is applied to a network where the tracks on the double-tracked lines are devoted to traffic in one special direction (i.e. one track in each direction), while the Swedish double-tracked lines can be used by two trains in the same direction and run in parallel. If and how their approach could be extended to fit the setting this paper deals with, is not clarified in their paper. Another important aspect of the disturbance management problem, which is not explicitly addressed by the previous approaches, is the synchronization of the train services which is a major issue in public transportation; see e.g. Schöbel (2009).

Depending on the focus, the models used to describe the scheduling and re-scheduling problem differ and have different levels of detail. Radtke (2008) and Gely et al. (2008) distinguish between the different levels of detail in the infrastructure models used for railway traffic network representation, where macroscopic models contain least details and have a more aggregated representation of some resources (e.g. a station is composed of a number of parallel tracks with one platform each), while microscopic includes a lot more detail (e.g. a station is composed of a complex set-up of pieces of tracks separated by switches and signals). When to use which level of detail is not obvious but often decided based on time available for computations and access to input data with sufficient accuracy. The more details included in the model, the more data and computations required and for a real-time scheduling problem there may not be enough time available to collect and process data and to compute the many alternatives that arise as the network is divided into more fine-grained resources. Most re-scheduling approaches do not use a very detailed representation of the station routings as it easily becomes rather complex, see e.g. Kroon et al. (1997) and Billionnet (2003)). For further information on models and methods for railway traffic scheduling and re-scheduling, we refer to Jacobs (2008) and Kroon et al. (2008).

In our previous research, primarily outlined in Törnquist and Persson (2007) and Törnquist (2007), we have designed an optimization-based approach for re-scheduling which seems promising. However, for some disturbance scenarios and a time horizon longer than 60 min it is difficult to find good solutions within seconds. Consequently, the MILP needs to be complemented by a mechanism that independent of disturbance scenario provides a feasible, good-enough solution fast.

Furthermore, the developed MILP formulation and solution method was not initially intended to consider the explicit routing of trains within stations since for most stations the choice of platforms and the routing is straightforward and the dependencies between different routes are more or less negligible. However, our most recent analysis of the infrastructure and train dependencies has indicated that it may be necessary to include more detailed characteristics of the infrastructure in the problem formulation. For example, some stations in the network have a more complex structure because they serve as a junction point for different lines (e.g. Åby). Furthermore, some stations (e.g. Strångsjö) have a more complicated safety system which requires different minimum separation times (from zero to 90 s) between the trains, depending on the sequence of incoming and outgoing trains and which path they take through the station. When the traffic is dense, such as during rush hour, these details have a significant impact even if it may not be clear yet exactly how. Hence, we need to extend our pre-

vious model which will result in an increased number of constraints and variables and consequently the computation time required to solve the problem. For those two reasons, we have developed a greedy algorithm which effectively delivers good solutions within the permitted time as a complement to the previous approach. To quickly retrieve a feasible solution the algorithm performs a depth-first search using an evaluation function to prioritize when conflicts arise and then branches according to a set of criteria.

4. The greedy algorithm

As mentioned, the main motivation for developing an algorithm is to ensure that we quickly (within 30 s) can receive a good-enough feasible solution independent of type of disturbance scenario. Furthermore, to use a tailored scheduling algorithm, in contrast to formulating the problem as a formal optimization model and using commercial solution software, is more flexible and dynamic in supporting the implementation of extended modelling requirements as discussed in Section 3. A disadvantage may, on the other hand, be risking not retrieving the most beneficial solutions.

The problem formulation used here is following the same structure as the one presented in Törnquist and Persson (2007), which divides the railway network resources into *sections*, where a section is either a *line section* or a *station section*. Each section has one to n parallel tracks. Each line section is a sequence of one or several consecutive blocks although these are not explicitly modelled but indicate that trains running in the same direction can use the tracks of a section simultaneously as long as they are separated by sufficient headway (i.e. not occupying the same block). Each section and track allows bi-directional traffic. See illustration in Fig. 4 below.

We denote a train slot, i.e. when a train is planned to occupy a certain section, as an *event* so that the timetable of a train i is a sequence of consecutive events, i.e. an *event list* K_i. Each event k has in its simplest form the parameters $b_k^{initial}$ and $e_k^{initial}$ to indicate its planned begin and end time, d_k to specify the minimum occupation time and h_k to specify if the train has a stop planned thus forcing the train to not leave a station before its planned end time. For trains which occupy the same track t within a station section j, a minimum of Δ_j time units separation time is required. We have used 30 s in our experiments. Trains running in the same direction and occupying the same track of a line section with more than one block are required to be separated by a minimum headway, H, of 3 min. Trains running in the opposite direction, or using the same track of a line section with only one block, need to be separated by zero time units, i.e. the first train needs to leave before the other one enters the section. If one line section j is directly followed by another line section, the trains cannot switch tracks unless it is explicitly stated to be permitted (in our scenarios, the infrastructure does not permit changing tracks between two consecutive line sections). In the case of two or more consecutive line sections, the time separation must be larger than zero to ensure that trains in opposite direction do not 'meet' between any two consecutive line sections. The separation time is then instead a very small number, e.g. 1 second. See constraint (D.12)–(D.20) in Appendix D. The constraints are only relevant to consider when two trains in opposite direction interact at any of these line sections.

The greedy algorithm iteratively searches for the best of all waiting train events to execute next and builds up a tree, referred to as **DT**, of consecutive active or terminated (the event has ended and the train has left the assigned track) events which become nodes in the tree. Each node holds an estimation of the disturbance consequences for the solution so far (a lower bound, LB).

The algorithm has three phases. Phase 1 is the pre-processing phase while Phase 2 is a depth-first search to quickly find a feasible and good-enough solution by building up a first complete branch of the tree. During Phase 3, the algorithm uses the remaining permitted computation time to improve the existing solution by backtracking to potential nodes (where the cost estimation is decreased) in the existing tree and branches to find better solutions. See Appendix A and Figs. 5a, 5b and 5c) for an outline of the algorithm.

In Phase 1, the algorithm first activates the events that were already started when the disturbance occurred and it allocates a start time and a track as initially intended. These events correspond to the green nodes in Fig. 5a) below.

The first waiting event in each train's event list is then put into a *candidate list*, ***NC***, sorted after the maximum value of the minimum starting time of each event k, $b_k^{\min}$ and its planned starting time, $b_k^{initial}$. d_k refers to the minimum running time for event k.

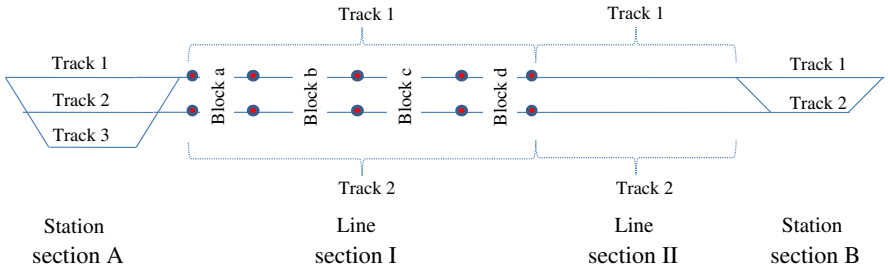


Fig. 4. The network model.

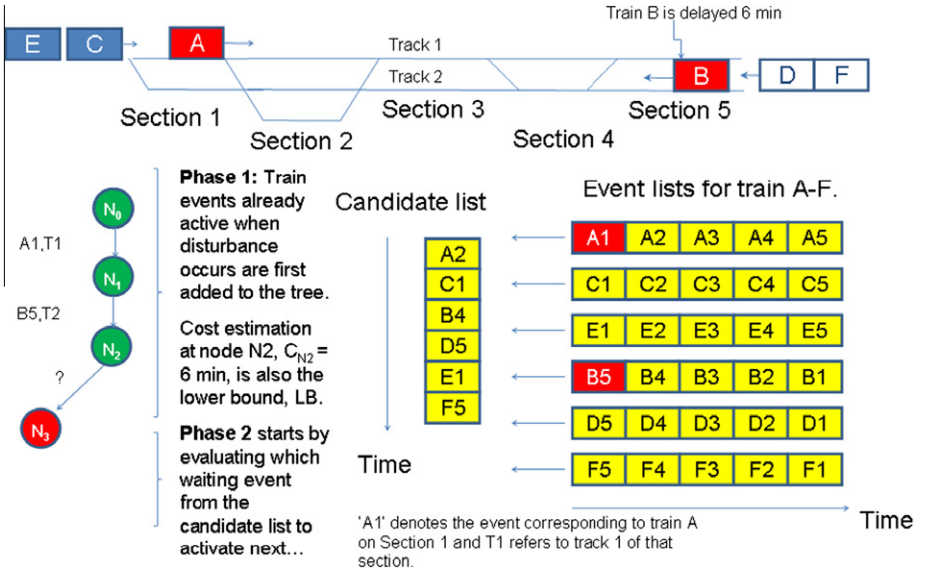


Fig. 5a. Illustration of the algorithmic procedure in Phase 1 and the start-up of Phase 2. The sequence of three green circles (i.e. nodes N_0 to N_2) and the first red circle N_3 represents the start of the decision tree (DT). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

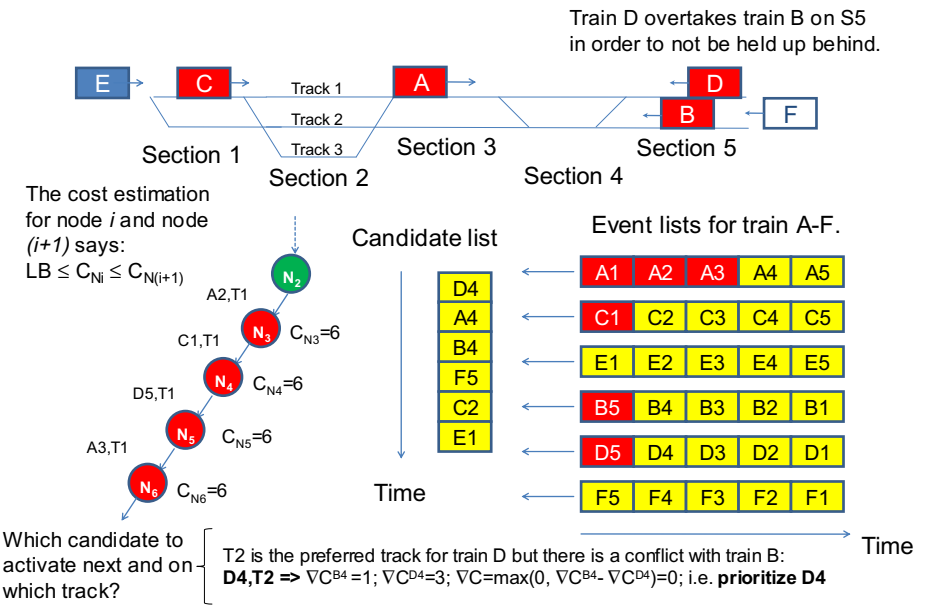


Fig. 5b. Illustration of Phase 2. A conflict of interest between foremost event D4 and B4 is discovered and the consequences of the two alternatives are computed and it shows that D4 should have priority.

In each iteration to select the next event in the candidate list to activate, the candidate with the earliest possible start time, $b_k^{\min}$, is then evaluated first (i.e. event A2 in Fig. 5a). In the evaluation process, the algorithm first checks whether there is any available track, and if the optimal track, t' is free – from the perspective of the event – it is selected. For example, let us assume that train A has a planned stop at track 1 on Section 2 to let passengers off and consequently this track is selected if possible. However, if necessary train A may go to track 2 or 3 just as well.

If the analysis shows that there are tracks possible to use but they are all occupied, the train will have to wait until next round and the algorithm continues to evaluate the next candidate in the list.

If there are no tracks available for other reasons, the train which the event belongs to is considered being deadlocked. That is, if the tracks are;

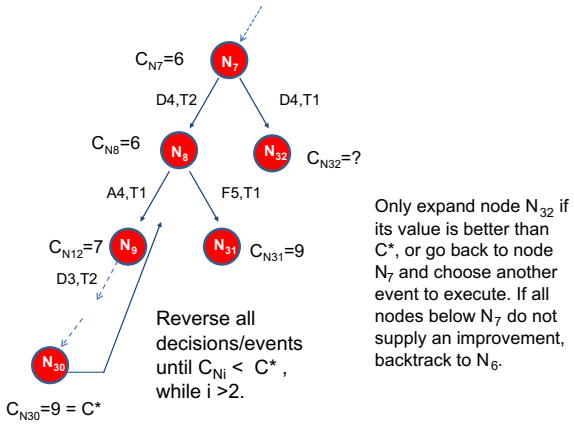


Fig. 5c. Illustration of Phase 3.

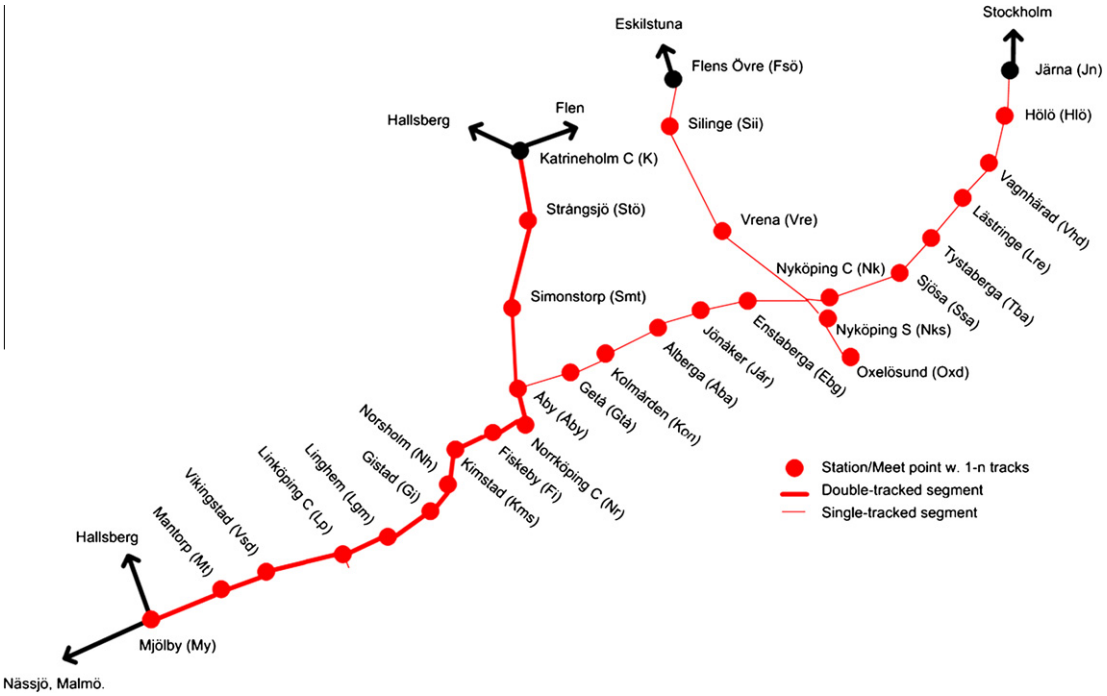


Fig. 6. The railway (sub-)network used for the scenarios and corresponding experiments. Between Ålberga and Kolmården as well as between Åby and Norrköping, there are two consecutive line sections and with no interconnection between the parallel tracks on the double-tracked sections. The fast passenger trains to/from Stockholm run via Katrineholm and Flen and the slower passenger trains run via Nyköping.

(i) physically impossible to reach, or
(ii) blocked, or
(iii) already used in a neighbouring child node (only relevant in Phase 3).

The search for next event to be activated is then interrupted in favour of backtracking to resolve the deadlock.

If there is an available track selected, a deadlock analysis is performed as well as a conflict of interest analysis. The deadlock analysis checks;

(i) whether there are any additional unoccupied tracks on the same section and if not,
(ii) whether the train has any way out of its potential new location if assigned the selected track. The analysis distinguishes between where the other trains are going in relation to that specific train. That is, a track occupied by a train running in the same direction is considered a possible way out.

If the investigated candidate event, k, is scheduled at line section j and this section has only one track which is functioning (i.e. it can be a single-tracked line section or a double-tracked line section where one of the tracks is blocked) the occurrence of possible conflicts of interest is investigated. This is done by checking if there are any events k' also scheduled at section j and who has an earliest possible start time which is smaller than the predicted termination time for event k. That is; $b_{k'}^{\min} \geqslant b_k^{\min}$ and $b_{k'}^{\min} < \left(d_k + b_k^{\min}\right)$

If a possible conflict of interest between two or more events is discovered, the cost (i.e. the consequential additional delay) for the conflict alternatives is computed. That is, let train D in Fig. 5b) be the current evaluated candidate event. If train D is to be allocated a track before train B, it may mean that train B becomes delayed and the *net minimum expected cost increase*, ∇C, is computed taking into account relevant buffer times and current delays. If the value is positive, it indicates that train B short term would suffer more than train D would gain and that prioritizing train B potentially would be wiser. Below is an example:

∇C^k is the predicted resulting delay for event k (event D4 in the example) if it has to wait in favour of event k', while $\nabla C^{k'}$ is the reversed and takes into account that event k' may be delayed already. We know that $b_k^{\min} \leqslant b_{k'}^{\min}$, and if $\left(b_k^{\min} + d_k\right) \leqslant b_{k'}^{initial}$ we let

$\nabla C = 0$. If not, we compute:

$$\nabla C^k = \max(0, (b_{k'}^{\min} + d_{k'})) - \max\left(b_k^{\min}, b_k^{initial}\right)$$

$$\nabla C^{k'} = \max\left(0, \left(b_k^{\min} + d_k - b_{k'}^{initial}\right)\right) - \max\left(0, \left(b_{k'}^{\min} - b_{k'}^{initial}\right)\right)$$

$$\nabla C = \max(0, (\nabla C^{k\prime -} \nabla C^k))$$

In the example, event D4 is able to start before B4 and would become at least 3 min delayed if B4 was to be prioritized, while the reversed situation would cause B4 a delay of only one minute. Therefore, D4 is prioritized. Consecutive effects on a third train are, however, not considered in the analysis.

When the best candidate in the candidate list has been chosen it is *activated* and allocated a track and a start time in line with the traffic conditions. The previous active event of the same train (if it has one) is at the same time closed (i.e. *terminated*) and its allocated resource hence released. That is, when D4 is activated, D5 is terminated and the corresponding track is denoted available.

The new active event is also put as a node onto the tree and the parent node keeps in memory that it has had this particular event allocated to the actual track as a child node. A cost estimation of the solution so far is then computed by summing up the minimum expected delay for each train. That is, the delay so far experienced by each train minus any future buffer available.

When all events have been terminated and stored as nodes in the tree, the first complete solution is found and stored with the corresponding cost for it (the sum of the final delay of all trains reaching their destination within the problem definition). The algorithm enters Phase 3 and uses the remaining computation time to expand the tree by backtracking to a node with a cost estimate which is lower than the value for the best complete solution. See example in Fig. 5c). Note that the backtracking can never go further back than to the last node of the ones activated during Phase 1 (i.e. the green nodes N_0–N_2 can not be revoked). The execution of the nodes below the branching node are reversed, the branching node index is saved and the search continues as before until the algorithm reaches a feasible, improved complete solution, or stops and backtracks one node up again if it encounters a complete or incomplete solution with the same or worse value than the best one found. In order to avoid cycling and repeating to do the same move over and over again, any parent node can only have one specific event allocated to a certain track as a child one time. If the same event is allocated a different track, it is considered to be a different child node.

In the deadlock analysis, the algorithm backtracks and reverses the most recent event of the deadlocked train. If this also leads to that this train becomes deadlocked, the algorithm backtracks one node higher than the previous backtracking.

The application of this algorithm has been implemented in Java using a vector to store the tree (i.e. only the active branch is stored along with the best solution found and other necessary data).

5. Simulation study Set-up

In discussion with Trafikverket, we have chosen to focus on the Norrköping traffic district in our study (see Fig. 6). The sub-network is composed of 28 stations and 15 double-tracked sections and 17 single-tracked sections. All tracks are bi-directional and the backbone of the region is double-tracked (the stretch Katrineholm–Åby–Mjölby) while the connecting lines are single-tracked. The stations have between two and 14 tracks apart from Norsholm which only has one. The line sections between the stations Katrineholm and Simonstorp contain seven consecutive blocks each. The stretch between Åby and Norrköping contains two consecutive line sections as does the stretches between Ålberga and Kolmården and Linköping and Linghem. We have also used a more detailed model of the stations Åby, Simonstorp and Strångsjö (see Appendix B) where only the available routes through the stations can be used whereas for all other stations we assume that all incoming and outgoing line section tracks connect with all station tracks.

We have applied the algorithm on three different categories of disturbances:

Category 1 refers to that a train is coming into the traffic management district with a certain delay or that it suffers from a temporary delay at one section within the district. This happens in practice quite frequently with the long-distance trains running through this traffic district.
Category 2 refers to that a train has a 'permanent' malfunction resulting in increased running times on all line sections it is planned to occupy. This is also a quite frequent problem.
Category 3 refers to an infrastructure failure causing, e.g. a speed reduction on a certain section, which results in increased running times for all trains running through that section. This is a serious problem which often has a significant impact on large parts of the surrounding traffic but it is not as frequent as the other disturbance types.

We have used the traffic for a typical weekday (Thursday the 23rd of April, 2009), which contains the regular amount of passenger and freight traffic and we have induced the disturbances in the afternoon rush hour at dense sections. We have applied the algorithm on a number of scenarios, see Appendix C and Table 1, with a maximum tolerated computation time of 30 s and a time horizon of 90 min. One reason for choosing a time horizon as long as 90 min, is that the time to run the main double-tracked stretch takes about 75 min for a faster passenger train and for certain disturbances of e.g. 20 min for a single delayed train, the delayed train would interact with the relevant trains at the end of its trip instead of the tracks being totally empty.

In attempt to evaluate the optimality of the algorithm, we have also solved the scenarios using solver Cplex version 10 minimizing the total final delay and using the complete formulation proposed in Törnquist and Persson (2007) with some necessary modifications corresponding to the same restrictions as implemented in the algorithm (see Appendix D for the complete model). That is, adding (a) headway conditions for the line sections with more than one block and trains running in the same direction, (b) conditions forbidding changing tracks between two consecutive line sections and (c) routing restrictions in to/out from the stations Strångsjö, Simonstorp and Åby.

Table 1
A selection of evaluation results for a time horizon of 90 min. In the fifth column the maximum difference between the solution values provided by the algorithm and Cplex is presented (since the optimum solution could not be found within 24 h). In some cases is also the minimum {minimum;maximum} value given, when a feasible solution has been found.

Nr	Scenario	# trains/events/ binary variables	Found solutions (min)	Max difference (min)	Optimum (min)/LB (min)
1	Long-distance pax train 538, north-bound, delay 12 min Linköping–Linghem	48/547/8148	24.82; 19.58	9.8	–/9.77
2	Long-distance pax train 538, north-bound, delay 6 min Linköping–Linghem	48/547/8148	12.52; 7.28	3.5	–/3.77
3	Paxtrain 2138, south-bound, delay 12 min Katrineholm–Strångsjö	48/551/8260	19.17; 13.02	3.5	–/9.5
4	Paxtrain 2138, south-bound, delay 6 min Katrineholm–Strångsjö	48/551/8260	13.17; 7.02	3.5	–/3.5
5	Paxtrain 80866 (north-bound), delayed 12 min Linköping–Linghem	49/562/8369	16.9; 15.50	10.3	–/5.25
6	Paxtrain 80866 (north-bound), delayed 6 min Linköping–Linghem	49/562/8369	l.13; 0.88	0.9	0/0
7	Paxtrain 8764 (north-bound), delayed 12 min Mjölby–Mantorp	50/553/8317	9.47; 8.32	0.2	–/8.1
8	Paxtrain 8764 (north-bound), delayed 6 min Mjölby–Mantorp	50/553/8317	4.6; 3.45	1.4	–/2.1
9	Paxtrain 539 (south-bound), delayed 12 min Katrineholm–Strångsjö	50/555/8295	14.48; 13.33	5.7	–/7.63
10	Paxtrain 539 (south-bound), delayed 6 min Katrineholm–Strångsjö	50/555/8295	5.63; 4.48	2.9	–/1.63
11	Paxtrain 538 w. permanent speed reduction causing 50% increased run times on line sections starting at Linköping–Linghem	48/547/8148	25.78; 29.55	3.5	–/17.03
12	Paxtrain 2138 w. permanent speed reduction causing 50% increased run times on line sections starting at Katrineholm–Strångsjö	48/551/8260	17.48; 11.33	3.5	–/7.82
13	Paxtrain 80866 w. permanent speed reduction causing 50% increased run times on line sections starting at Linköping–Linghem	48/563/8321	38.82; 37.42	{1.6; 5.3}	35.8/32.17
14	Paxtrain 8764 w. permanent speed reduction causing 50% increased run times on line sections starting at Mjölby–Mantorp	50/553/8317	26.95; 24.25	{5.7; 6.6}	18.54/17.7
15	Paxtrain 539 w. permanent speed reduction causing 50% increased run times on line sections starting at Katrineholm–Strångsjö	50/555/8295	28.8; 27.65	1.0	26.64/26.64
16	Speed reduction for all trains between Strångsjö and Simonstorp (all trains get a runtime of 27 min, cf. 5–10 min planned runtime) starting w. freight train 43533	46/507/7001	230.83	0.0	–/230.83
17	Speed reduction for all trains between Åby and Simonstorp (all trains get a runtime of 20 min) starting w. train 2138	51/556/8442	129.77; 129.45; 129.42; 129.1	8.9	–/115.17
18	Speed reduction for all trains between Åby and Norrkoping (all trains get a runtime of 8 min) starting w. train 2138	49/551/8154	90.95; 85.42	{6.2; 42.2}	79.27/43.23
19	Speed reduction for all trains between Mjölby and Mantorp (all trains get a runtime of 20 min) starting w. train 8764	50/553/8317	481.38	318.0	–/163.25
20	Speed reduction for all trains between Linköping and Linghem (all trains get a runtime of 15 min) starting w. train 538	48/547/8148	393.48; 382.78; 382.57	266.0	–/116.65

6. Results

The algorithm finds very fast in all scenarios a first feasible, complete solution but is then not always so effective in branching and finding an improvement. Since it is very time and memory consuming to solve certain scenarios using the formulation in Appendix D and the solver Cplex version 10, and sometimes not even possible to find a feasible solution, it has not been possible to perform an optimality check for all scenarios. In Table 1, an overview of the simulated scenarios can be seen. The second column presents the scenarios which also can be seen in the graph in Appendix C. The third column presents information on the size of the scenarios and the number of binary variables is computed below where m_j refers to the number of events on section j that remains to be scheduled. The first term refers to the variables $\gamma_{k\dot{k}}$ while the second term refers to $\lambda_{k\dot{k}}$ and the last, q_{kt} (see notation in Appendix D). The last two only concerns non-single-tracked sections.

$$\sum_{j=1}^{|B|} m_j(m_j-1)/2 + \sum_{j=1:|P_j|>1}^{|B|} m_j(m_j-1)/2 + \sum_{j=1:|P_j|>1}^{|B|} m_j * |P_j| : |P_j| > 1 \tag{6.1}$$

The fourth column presents the cost value of all the solutions found by the algorithm within 30 s and where the highest value refers to the solution found first. It takes less than 1 s to find the first solution for the algorithm and very often it also finds the other solutions (if there are any) within the first few seconds.

The sixth column presents two values, where the first is the best solution found by the solver (if no value is given, a feasible solution was not found within 24 h). The second value is the maximum of (a) the lower bound provided by the solver and (b) the lower bound found by the algorithm. If the solver has found an optimal solution, that is also the lower bound. In the fifth column, the largest possible difference between an optimum and the best solution value provided by the algorithm is presented. This value, however, gives very little information in the cases where the solver has found a solution with a large gap (i.e. large difference between its best found solution and its lower bound) or not found a feasible solution. In a few scenarios, there is also a minimum difference given, i.e. {minimum, maximum} since Cplex has found a feasible solution which is better than the one provided by the algorithm.

Table 2
A selection of evaluation results for a time horizon of 60 min. * Refers to that an optimum could not be found within 24 h by Cplex and hence the lower bound (LB) is given as well as the minimum and maximum difference between the solution values provided by the algorithm and Cplex.

Nr	Scenario	# trains/ events/binary variables	Found solutions (min)	Difference (min)	Optimum (min)
1	Long-distance pax train 538, north-bound, delay 12 min Linköping–Linghem	36/356/3795	16.33	2.1	14.23
2	Long-distance pax train 538, north-bound, delay 6 min Linköping–Linghem	36/356/3795	3.77	0.0	3.77
3	Paxtrain 2138, south-bound, delay 12 min Katrineholm–Strångsjö	34/357/3842	9.50	0.0	9.50
4	Paxtrain 2138, south-bound, delay 6 min Katrineholm–Strångsjö	34/357/3842	3.50	0.0	3.50
5	Paxtrain 80866 (north-bound), delayed 12 min Linköping–Linghem	37/364/3841	19.58	4.1	15.47
6	Paxtrain 80866 (north-bound), delayed 6 min Linköping–Linghem	37/364/3841	4.63	4.1	0.50
7	Paxtrain 8764 (north-bound), delayed 12 min Mjölby–Mantorp	39/357/3813	8.10	0.0	8.10
8	Paxtrain 8764 (north-bound), delayed 6 min Mjölby–Mantorp	39/357/3813	3.23; 2.98	{0.3; 0.9}	**2.93***(**2.1**)
9	Paxtrain 539 (south-bound), delayed 12 min Katrineholm–Strångsjö	41/365/3921	14.92	3.8	11.13
10	Paxtrain 539 (south-bound), delayed 6 min Katrineholm–Strångsjö	41/365/3921	4.27	0.0	4.27
11	Paxtrain 538 w. permanent speed reduction causing 50% increased run times on line sections starting at Linköping–Linghem	36/356/3795	17.03	0.0	17.03
12	Paxtrain 2138 w. permanent speed reduction causing 50% increased run times on line sections starting at Katrineholm–Strångsjö	34/357/3842	7.82	0.0	7.82
13	Paxtrain 80866 w. permanent speed reduction causing 50% increased run times on line sections starting at Linköping–Linghem	38/369/3858	32.08	4.1	27.98
14	Paxtrain 8764 w. permanent speed reduction causing 50% increased run times on line sections starting at Mjölby–Mantorp	39/357/3813	27.28	7.0	20.26
15	Paxtrain 539 w. permanent speed reduction causing 50% increased run times on line sections starting at Katrineholm–Strångsjö	41/365/3921	27.43	0.8	26.64
16	Speed reduction for all trains between Strångsjö and Simonstorp (all trains get a runtime of 27 min, cf. 5–10 min planned runtime) starting w. freight train 43533	34/321/3102	156.80	0.0	156.80
17	Speed reduction for all trains between Åby and Simonstorp (all trains get a runtime of 20 min) starting w. train 2138	40/363/3976	93.02	12.7	80.32
18	Speed reduction for all trains between Åby and Norrkoping (all trains get a runtime of 8 min) starting w. train 2138	39/364/3854	40.87	2.8	38.12
19	Speed reduction for all trains between Mjölby and Mantorp (all trains get a runtime of 20 min) starting w. train 8764	39/357/3813	278.87	{0; 126.2}	**2 78.8 7***(**152.7**)
20	Speed reduction for all trains between Linköping and Linghem (all trains get a runtime of 15 min) starting w. train 538	36/356/3795	155.40	{5.13; 34.8}	**136.8***(**120.62**)

Since it is so difficult to evaluate the level of optimality of the algorithm without optimal solutions or better lower bounds, we have also simulated the same disturbance scenarios but with a shorter time horizon of only 60 min. The results can be seen in Table 2.

In an evaluation of the performance of the algorithm and comparison to the solver, we can first of all see that in most scenarios the solver cannot provide a feasible solution despite a relatively high value on the time limit, i.e. 24 h. When comparing the solutions, we have found that the algorithm and the solver, not surprisingly, have different strategies to find a good solution. Even when they achieve solutions with the same objective value, they have produced different solutions. The solutions provided by the solver contain quite some intermediate delays, i.e. several trains suffer from smaller delays of only a few minutes during their trip but arrive at final destination on time. The strategy used by the algorithm avoids delaying additional trains unless it really pays off and therefore it often delays a smaller number of trains but then delays those more. A visual analysis of some of the solutions provided by the algorithm showed that the solutions seem sound and no obvious, large improvement could be seen. Note also that it is important to understand, from a practical point of view, that the running times and separation times between trains are approximations of what they actually will be in real-time, so that if the solutions differ with a few minutes this is acceptable.

When it comes to the performance of the algorithm with respect to the different categories, we can see that for the first two categories the algorithm performs fairly well. For category three it was difficult to find good values for comparison, but still we can see that the algorithm *may* have some difficulties finding good-enough solutions, see e.g. scenario 19 and 20. However, as described earlier the maximum difference presented in the fifth column in Tables 1 and 2 is really only relevant if there is a feasible solution found by Cplex (i.e. the sixth column contains two values). In the case of scenario 16–17 and 19–20 in Table 1, Cplex can not provide a feasible solution within the time limit and thus only provides a lower bound of the LP relaxation of the problem. The lower bound means that there may exist a solution which is as good as 163.25 (for scenario 19), but probably not in this case. How the disturbances are affecting the trains with respect to e.g. increased running times, can be seen in Appendix C. For scenario 19, we can see that the average running time between Mjölby and Mantorp in the timetable is approximately 5 min and because of the disturbance it is increased to 20 min for all trains passing through during those 90 min (approximately 13 trains). Consequently, each of these trains gets a delay of approximately minimum 15 min and the sum of these delays reaches 195 min and does not include any of the knock-on delays (cf. the lower bound of 163.25).

Since the algorithm focuses on preventing deadlocks from occurring (but is also able to unlock deadlocks) it has not created a deadlock before finding its first solution in our scenarios (i.e. within Phase 2), but later during Phase 3. The reason is that the algorithm is quite conservative when it searches for the first solution, which probably results in that the optimal solution is sometimes not found. That is, in Phase 2 it rather avoids making a decision which may lead to a deadlock than to take the chance and select a candidate event which may generate a lower delay cost estimation.

The ability of the algorithm to handle deadlocks at stations where a line is splitting up, e.g. Åby, could be improved. The reason why deadlocks are more likely to occur for this type of station is that the appropriate track needs to be selected earlier in time than for other, simpler stations.

7. Discussion and future work

In this paper we have presented a greedy algorithm which in short time delivers good re-scheduling solutions to the railway traffic disturbance management problem. We have performed an experimental evaluation and analysis of the performance of the algorithm. The evaluation shows that the algorithm provides good-enough solutions very fast but there are also several possible improvements which can be made and foremost related to its branching strategy.

Apart from making the algorithm select potential branching nodes more effectively and minimizing the amount of computations, we are also working on an implementation and evaluation of the effect of multi-threading. The use of multi-threading means that several branches can be explored simultaneously by use of parallel computing, see e.g. (Emer et al., 2007), which would speed up the search for improvements.

Furthermore, during discussions with the traffic managers we have seen that even further modelling extensions may need to be considered. This includes e.g. modelling the separation of train movements in to and out from certain stations in more detail as well as using a more dynamic way of specifying minimum train movement times depending on how trains enter and leave certain stations. How these extension, if relevant, can be incorporated then need to be considered in more depth.

Finally, the main purpose of this algorithm is to act as a complement to the earlier optimization approach presented in Törnquist and Persson (2007) since for some scenarios and time horizons the previous approach was failing to quickly enough provide a feasible solution. For some of the scenarios considered in this paper, we have also applied the previous approach as well as Strategies 1 and 2 presented in Törnquist and Persson (2007) with the use of Cplex. Even Strategy 1, which simplifies and restricts the solution space significantly, failed to solve some of the scenarios within the time limit of 30 s. What could be interesting to investigate in future work is the possibility and effect of giving the initial solution provided by the greedy heuristic as a start solution to Cplex using our previous approach (if the starting solution is feasible with the restricted formulation) or the full model presented in Appendix D. This could potentially lead to improved solutions or at least generate better estimations of the lower bounds.

Acknowledgements

The research presented has been financially supported by Trafikverket (the Swedish Transport Administration). Furthermore, support and guidance have been provided by several people at Trafikverket – especially Jörgen Hwargård, Thomas Franzén, Hans Dahlberg and Tommy Jonsson – and by Prof. Paul Davidsson, Prof. Håkan Grahn and Dr. Jan A. Persson at Blekinge Institute of Technology, Sweden. Finally, I would like to thank the anonymous reviewers for their effort and significant comments on how to improve this paper.

Appendix A. Outline of the greedy algorithm

A.1. Notation

Overall, the same notation as in Törnquist and Persson (2007) is used.

Let T_0 be the time when the disturbance was inflicted. Let **DT** be the decision tree vector which stores the current branch, i.e. sequence, of active and terminated events and all those events are represented as nodes in **DT**. **DT*** is the vector which stores the best complete solution found so far. s is used to denote the index of node N_s in **DT** and $s = |\mathrm{DT}|$ gives the index of the last activated node (i.e. event). **DT** starts with the nodes corresponding to all the events that were active at T_0 and s^* represents the index of the last of those nodes in **DT**. *LB* refers to the lower bound and C_s refers to the *minimum delay cost* to be expected after the previous events generating nodes N_1–N_s in **DT** have been activated (and possibly terminated). Consequently C_{s^*} provides the LB. Let C^* denote the best solution value with default value -1. Let $\mathrm{DT}_{\max}$ be the max size of **DT**, i.e. the total number of events including the pre-started ones. Let $s_{\mathrm{backtrack}}$ denotes the index of the height in **DT** to which the last backtracking was carried out and $s_{\mathrm{backtrack}} \geqslant s^*$.

Let $Stat_i = \{W = Waiting,\ A = Active,\ T = Terminated\}$ specify the actual status of train i at a certain point while $E_i = \{0; |\mathbf{K}_i| + 1\}$ denote the index of its current active event. $E_i = 0$ refers to $Stat_i = W$ and $E_i = |\mathbf{K}_i| + 1$ refers to $Stat_i = E$. Let $b_k^{\min}$ denote the earliest possible start time of event k. Define **NC** (the candidate list) as the set of the first unstarted event k of each train i. **NC** is also by definition sorted in chronological order w.r.t. $b_k^{\min}$. Let k^* denote the index of the current most high-ranked candidate event found in **NC**, while k^{**} denotes the second most high-ranked event. t^* denotes the chosen track for k^*, and t^{**} the track for $k^{**} \cdot i^*$ denotes the index of an identified deadlocked train.

Let $\boldsymbol{L}'_j$ be the list of the remaining non-active/-terminated events of section j, where $k \in \boldsymbol{L}_j$ and it is by definition sorted in chronological order w.r.t. $b_k^{\min}$. Let P'_k be a subset of tracks P_j, where P'_k only includes tracks which event k is permitted to use (i.e. the event may already have been assigned track t before as a child node from the last active node in **DT**). Let nrT_j represent the actual number of unoccupied tracks at section j excluding any temporarily blocked tracks so that $nrT_j = \{0, 1, \ldots, |P_j|\}$. S_k specifies if event k takes place at a line section ($S_k = 1$) or a station section ($S_k = 0$).

Let $\mathbf{TL}_{j,t}$ be the vector of track t on section j which lists all the executed and active events which have been assigned track t. Let $T_{j,t}^{\min}$ denote the earliest time track t is available for a new event to be activated. The event with index $s = |\mathbf{TL}_{j,t}|$ is the last one which entered the track. Let $e_k^{\min}$ denote the earliest time the active event k is permitted to leave its assigned track. If the corresponding section only contains one block, $e_k^{\min}$ is dependent on the minimum running time and the earliest time the corresponding train can activate its next event. However, on sections with more than one block and where two trains run in the same direction, they can use the same track but separated by some minimum headway (here 3 min) and this needs to be considered both when entering and leaving the track. $e_k^{\min}$ may need to be associated with the closure of another event.

Let *deadlockRisk* be a Boolean variable (takes on the values true/false) specifying if the deadlock analysis resulted in an identified risk of running into a deadlock if the investigated candidate event k is assigned the investigated track t'.

Let t' denote the 'optimal' track at section j selected for the candidate k in focus so that $t' = \{-1, 0, 1, \ldots |P'_k|\}$ which means that if t' is negative there are no tracks possible to choose from and thus it refers to a deadlock. The value '0' refers to that all the tracks which event k would be permitted to use are occupied and the first one becomes available at time $T_j^{\min}$. $T_{j,t}^{\min}$ specifies the earliest time event k may enter the specific track t which becomes relevant when $t' > 0$.

Let *possibleConflictOfInterest* be a Boolean variable specifying if there are any trains which may compete with train i over a track. That is, if there are any trains which would want to start before train i could leave the requested track and no other track would be available either before the other train(s) planned to depart.

Even though the first element in a vector implemented in Java has the index '0', we assume here that the first element in any vector or list has index = 1 and the last element has index = |vector/list size|.

A.2. Phase 1

(1.1) Start event $k \in \boldsymbol{K}_i$, $i \in \boldsymbol{T}$, if it was ongoing at T_0 and assign the values of the corresponding variables; the initial starting time, $x_k^{begin} = b_k^{initial}$, and initial track, $q_{k,t_k^{track}} = 1$. Add event k as a node at the end of **DT**.
(1.2) Update the running times, d_k, of all events k which are directly affected by the inflicted disturbance.
(1.3) For $\forall$ trains $i \in \boldsymbol{T}$, update $Stat_i$.
(1.4) Compute $b_k^{\min}$ for all the non-active/-terminated events $k \in K_i$ of all trains i, LB and C_s: For $\forall\ i \in \boldsymbol{T}$:

If ($Stat_i$=W): Let $b_k^{\min} = \max\left(h_k * b_k^{init}, T_0\right)$, where k = 1 **then**

For $\forall k \in \boldsymbol{K}_i : k > 1\left\{b_{k+1}^{\min} = \max\left(h_{k+1} * b_{k+1}^{init}, \left(b_k^{\min} + d_k\right)\right)\right\}$

Let $C_{s^*} = C_{s^*} + \max\left(0, \left(b_{|t_i|}^{\min} + d_{t_i|} - e_{t_i|}^{init}\right)\right)$

End if

Else if ($Stat_i$ = A): Let $b_k^{\min} = \max\left(h_k * b_k^{init}, x_{k-1}^{start} + d_{k-1}\right)$ where $k = E_{i.} + 1$. **then**

For $\forall k \in \boldsymbol{K}_i : k > (E_i + 1)\left\{b_{k+1}^{\min} = \max\left(h_{k+1} * b_{k+1}^{init}, \left(b_k^{\min} + d_k\right)\right)\right\}$

Let $C_{s^*} = C_{s^*} + \max\left(0, \left(b_{|t_i|}^{\min} + d_{t_i|} - e_{t_i|}^{init}\right)\right)$

End else if

Else (i.e. $Stat_i$ = T) **then** Let $C_{s^*} = C_{s^*} + \max\left(0, \left(x_{|t_i|}^{end} - e_{t_i|}^{init}\right)\right)$ **End else**

Let LB $= C_{s^*}$.
(1.5) Initiate **NC**.

A.3. Phases 2 & 3

While (|**NC**|> 0 and LB != C^* and the time limit is not reached yet) **do**
Let k = 1. Let $k^* = k^{**} = i^* = t^* = t^{**} = -1$.
While (k^* < 0 and i^* < 0 and for $\forall$ k $\in$ **NC**, where k $\in \mathbf{L}_j$ and $\mathbf{K}_i$) **do** Let $t' = -1$.
If ($|P'_k| > 0$ and ($Stat_i = W$ or ($Stat_i = A$ and event $k = E_i$ can be closed))) **then** Select track t' from P'_k .
If(t' < 0) **then** Let $i^* = i$. **end if**
Else if(t' = 0) **then** Let $b_k^{\min} = T_j^{\min}$. For $\forall$ k $\in \boldsymbol{K}_i$: k > (E$_i$ + 1), update $b_k^{\min}$ as in 1.4) **End else if**
Else if $\left(|\boldsymbol{L}'_j \leqslant nrT_j\right)$ **then** Let $t^* = t'$ and $k^* = k$. **End else if**
Else if (deadlockRisk=false) **then**
If (S_k = 1 and nrT_j < 2 and possibleConflictOfInterest=true) **then**
Compute ∇C. **If** ($\nabla C = 0$) **then** Let $t^* = t$ and $k^* = k$. **End if**
Else if (k^{**} < 0 or weight > ∇C) **then** Let $t^{**} = t$ and $k^{**} = k$. Let weight=∇C. **End else if**
End if
End else if
End if
End while

If(i^* > 0) **then** Handle deadlock: Backtrack in **DT**reversing all actions (and deleting corresponding nodes) until the parent node of event $k = |E_{i^*}| - 1$. Update all relevant parameters and variables including adding the reversed events to the corresponding lists $\mathbf{L}'_j$ and removing them from $\mathbf{TL}_{j,t^*}$. **End if**

Else if(k^* < 0 and k^{**} < 0) **then** Handle deadlock: Backtrack in **DT**reversing the last action (and deleting the corresponding node). Update all relevant parameters and variables including adding the reversed event to the corresponding list $\mathbf{L}'_j$ and removing it from $\mathbf{TL}_{j,t^*}$. **End else if**

Else then
If (k^* > 0) **then**
Let $b_{k^*}^{\min} = \max\left(b_{k^*}^{\min}, T_{j,t^*}^{\min}, e_{|E_i|}^{\min}\right) : k^* \in \boldsymbol{K}_i/\boldsymbol{L}_j$. Let $x_{|E_i|}^{end} = x_{k^*}^{begin} = b_{k^*}^{\min}$ and $q_{k^*,t^*} = 1 : t^* \in P'_k$.
Add event k^* to **DT** and $\mathbf{TL}_{j,t^*}$ and remove from $\boldsymbol{L}_j^*$.
If((|Ei| + 1) = |K_i|) **then** $x_{|E_i|+1}^{end} = \max\left(\left(b_{k*}^{\min} + d_{k*}\right), e_{|E_i|}^{\min}\right)$ **End if**
Update $Stat_i$, E_i and any relevant $b_k^{\min}$.
End if

Else if ($k^{**} > 0$) **then**
Let $b^{\min}_{k^{**}} = \max\left(b^{\min}_{k^{**}}, T^{\min}_{j,t^{**}}, e^{\min}_{|E_i|}\right) : k^{**} \in \boldsymbol{K}_i/\boldsymbol{L}_j$. Let $x^{end}_{|E_i|} = x^{begin}_{k^{**}} = b^{\min}_{k^{**}}$ and $q_{k^{**},t^{**}} = 1 : t^{**} \in P'_k$.
Add event k^{**} to **DT** and $\mathbf{TL}_{j,t^{**}}$ and remove from $\boldsymbol{L}'_j$.
If ($(|Ei| + 1) = |K_i|$) **then** $x^{end}_{|E_i|+1} = \max\left(\left(b^{\min}_{k^{**}} + d_{k^{**}}\right), e^{\min}_{|E_i|}\right)$ **End if**
Update $Stat_i$, E_i and any relevant $b^{\min}_k$.
End else if
Compute $C_{|DT|}$ as in 1.4).
If ($|\mathbf{DT}| = DT_{max}$ and $C^* < 0$) **then** Let $C^* = C_{|DT|}$. Backtrack in **DT** to the first encountered node (counting from the end), where $C_s < C^*$ and according to already defined backtracking scheme above. Let $s_{backtrack} = |DT|$ after backtracking. **End if**
Else if($|\mathbf{DT}| = DT_{max}$ and $C^* > 0$ and $C^* > C_{|DT|}$) **then** Let $C^* = C_{|DT|}$. Backtrack in **DT** to the node at index ($s_{backtrack} - 1$) and according to already defined backtracking scheme above.
Let $s_{backtrack} = s_{backtrack} - 1$. **End else if**
Else if ($|\mathbf{DT}| < DT_{max}$ and $C^* > 0$ and $C^* \leqslant C_{|DT|}$) **then** Backtrack in **DT** to the node at index ($s_{backtrack} - 1$) and according to already defined backtracking scheme above. **End else if**
End else
Reinitiate **NC**.
End while

Appendix B. Station configurations

See Fig. 7.

Appendix C. Timetable for the traffic scenarios

See Fig. 8.

Appendix D. Optimization model

D.1. Objective function

$$\text{Minimize} \quad \sum_{i \in T} z_i \tag{D.1}$$

D.2. Train restrictions

$$x^{end}_k = x^{begin}_{k+1} \qquad i \in T, k \in K_i : k \neq n_i \tag{D.2}$$
$$x^{end}_k \geqslant x^{begin}_k + d_k \qquad k \in E \tag{D.3}$$
$$x^{begin}_k \geqslant b^{initial}_k \qquad k \in E : h_k = 1 \tag{D.4}$$
$$x^{begin}_k = b^{static}_k \qquad k \in E : b^{static}_k > 0 \tag{D.5}$$
$$x^{end}_k = e^{static}_k \qquad k \in E : e^{static}_k > 0 \tag{D.6}$$
$$z_i \geqslant x^{begin}_{n_i} - b^{initial}_{n_i} \qquad i \in T \tag{D.7}$$

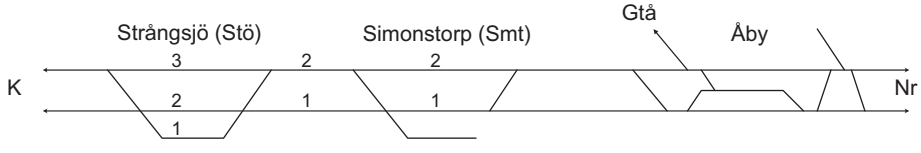


Fig. 7. Illustration of the double-tracked stretch between Katrineholm (K) and Norrköping (Nr) and connects via Åby to the single-tracked stretch to Nyköping and Stockholm. Cf. Fig. 6.

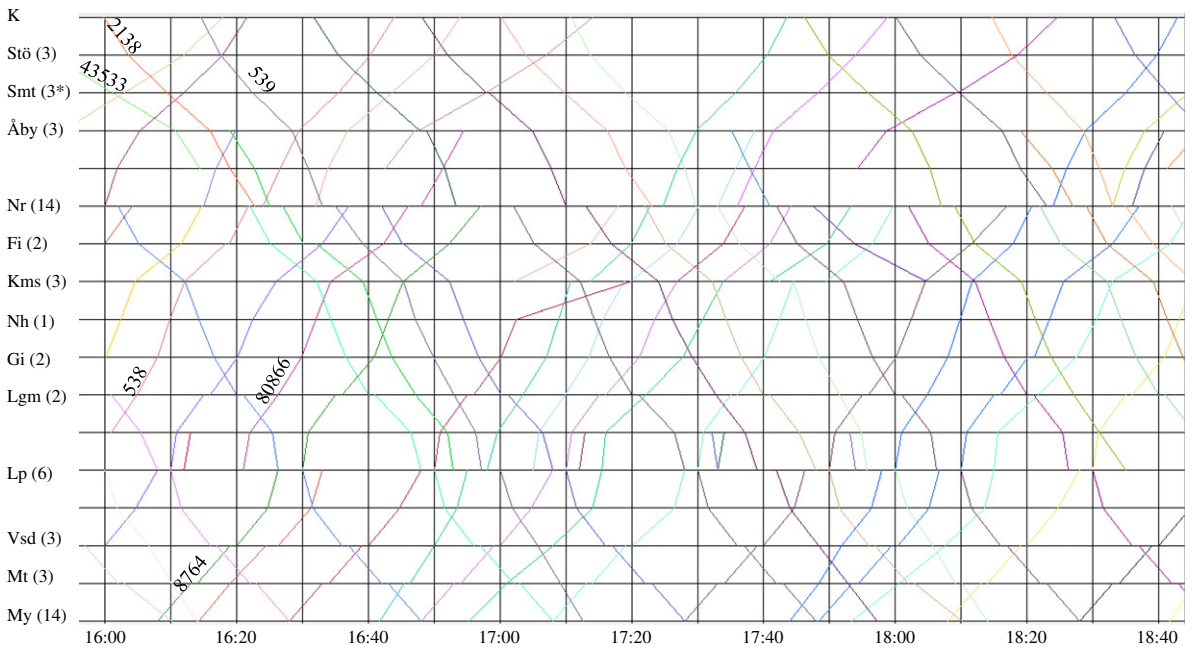


Fig. 8. The timetable for the double-tracked section Katrineholm–Mjölby which was used in the simulation experiments. The traffic on the single-tracked stretch is not as intense. In the parenthesis, after each station acronym on the y-axis, is the number of tracks within the station presented. Smt* refers to that the station Simonstorp has three tracks but the third one is not used. See Fig. 6 for an overview of the line.

D.3. Infrastructure restrictions

$$\sum_{t\in P_j} q_{kt} = 1 \qquad j \in B, k \in L_j : |P_j| > 1 \tag{D.8}$$

$$\sum_{t\in P_j} t^* q_{kt} = r_k^{track} \qquad j \in B, k \in L_j : |P_j| > 1 \& r_k^{fixed} = 1 \ (i.e.\ when\ b_k^{static} > 0 \tag{D.9}$$

Constraint (D.10) is used when a train has two consecutive events where both are scheduled on a line section with no meeting possibility in between.

$$q_{kt} = q_{k+1,t} \qquad i \in T, j \in B, k \in L_j, k\&(k+1) \in K_i, t \in P_j : |P_j| > 1 \& k \neq n_i \& S_k = S_{k+1} = 1 \tag{D.10}$$

$$q_{\hat{k}t} + q_{kt} - 1 \leqslant \lambda_{k\hat{k}} + \gamma_{k\hat{k}} \qquad j \in B, k, \hat{k} \in L_j, t \in P_j : k < \hat{k} \& |P_j| > 1 \tag{D.11}$$

For line sections containing only one block section. Constraint (D.13) is for single-tracked sections and (D.14) for sections with multiple tracks:

$$x_{\hat{k}}^{begin} - x_k^{end} \geqslant \Delta_j \gamma_{k\hat{k}} - M(1 - \gamma_{k\hat{k}}) \qquad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& (dir_k \neq dir_{\hat{k}} \| nrB_j = 1) \tag{D.12}$$

$$x_k^{begin} - x_k^{end} \geqslant \Delta_j (1 - \gamma_{k\hat{k}}) - M\gamma_{k\hat{k}} \qquad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& |P_j| = 1 \& (dir_k \neq dir_{\hat{k}} \| nrB_j = 1) \tag{D.13}$$

$$x_k^{begin} - x_k^{end} \geqslant \Delta_j \lambda_{k\hat{k}} - M(1 - \lambda_{k\hat{k}}) \quad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& |P_j| > 1 \& (dir_k \neq dir_{\hat{k}} \| nrB_j = 1) \tag{D.14}$$

tive blocks within the section and hence where trains in same direction can follow each other if separated by at least H time units:

$$x_{\hat{k}}^{begin} - x_k^{begin} \geqslant H_j \gamma_{k\hat{k}} - M(1 - \gamma_{k\hat{k}}) \quad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& (dir_k = dir_{\hat{k}} \& nrB_j > 1) \tag{D.15}$$

$$x_{\hat{k}}^{end} - x_k^{end} \geqslant H_j \gamma_{k\hat{k}} - M(1 - \gamma_{k\hat{k}}) \qquad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& (dir_k = dir_{\hat{k}} \& nrB_j > 1) \tag{D.16}$$

$$x_k^{begin} - x_{\hat{k}}^{begin} \geqslant H_j (1 - \gamma_{k\hat{k}}) - M\gamma_{k\hat{k}} \quad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& |P| = 1 \& (dir_k = dir_{\hat{k}} \& nrB_j > 1) \tag{D.17}$$

$$x_k^{end} - x_{\hat{k}}^{end} \geqslant H_j (1 - \gamma_{k\hat{k}}) - M\gamma_{k\hat{k}} \qquad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& |P| = 1 \& (dir_k = dir_{\hat{k}} \& nrB_j > 1) \tag{D.18}$$

$$x_k^{begin} - x_{\hat{k}}^{begin} \geqslant H_j \lambda_{k\hat{k}} - M(1 - \lambda_{k\hat{k}}) \quad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& |P_j| > 1 \& (dir_k = dir_{\hat{k}} \& nrB_j > 1) \tag{D.19}$$

$$x_k^{end} - x_{\hat{k}}^{end} \geqslant H_j\lambda_{k\hat{k}} - M(1-\lambda_{k\hat{k}}) \qquad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& |P_j| > 1 \& (dir_k = dir_{\hat{k}} \& nrB_j > 1) \tag{D.20}$$

$$\lambda_{k\hat{k}} + \gamma_{k\hat{k}} \leqslant 1 \quad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& |P_j| > 1 \tag{D.21}$$

$$x_k^{begin}, x_k^{end} \geqslant 0 \qquad k \in E \tag{D.22}$$

$$z_i \geqslant 0 \qquad i \in T \tag{D.23}$$

$$\gamma_{k\hat{k}} \in \{0,1\} \qquad j \in B, k, \hat{k} \in L_j : k < \hat{k} \tag{D.24}$$

$$\lambda_{k\hat{k}} \in \{0,1\} \qquad j \in B, k, \hat{k} \in L_j : k < \hat{k} \& |P_j| > 1 \tag{D.25}$$

$$q_{kt} \in \{0,1\} \qquad j \in B, k \in L_j, t \in P_j : |P_j| > 1 \tag{D.26}$$

The notation is based on the one presented in Törnquist and Persson (2007) but some new constraints, variables and parameters have been added.

T is defined as the set of trains, B is the set of sections, and E is the set of events where an event is a resource request by a certain train for a specific section. We let index i be associated with a train, j with a section and index k with an event. Each event is connected to both a train and a section. Let $K_i \subseteq E$ be the ordered set of events for train i ($i \in T$) and $L_j \subseteq E$ be the ordered set of events of section j ($j \in B$). Events in K_i and L_j are ordered according to the original timetable. We use $(k+1)$ to denote the first proceeding event of event k (in K_i and L_j) and $k < \hat{k}$ to denote that $\hat{k}$ is any event proceeding event k with respect to the order in the sets. Furthermore, let n_i and m_j denote the last event of K_i and L_j, respectively. Each section has a set of parallel tracks $P_j = \{1, \ldots, p_j\}$.

The formulation contains six types of decision variables. The variables x_k^{begin} and x_k^{end} represent the start and end time of event k. z_i represents the delay train i experiences at its final destination within the problem formulation. In addition we have the binary variables:

$$q_{kt} = \begin{cases} 1, \text{if event } k \text{ uses track } t, \text{where } k \in L_j, , t \in P_j.), j \in B. \\ 0, \textit{otherwise}. \end{cases}$$

$$\gamma_{k\hat{k}} = \begin{cases} 1, \text{if event } k \text{ occurs } \textit{before} \text{ event } \hat{k} (\text{as in the initial timetable}), \\ \text{where } k, \hat{k} \in L_j, j \in B : k < \hat{k}. \\ 0, \text{otherwise}. \end{cases}$$

$$\lambda_{k\hat{k}} = \begin{cases} 1, \text{if event } k \text{ is re–scheduled to occur } \textit{after} \text{ event } \hat{k}, \\ \text{where } k, \hat{k} \in L_j, j \in B : k < \hat{k}. \\ 0, \text{otherwise}. \end{cases}$$

S_k specifies if event k takes place at a line section ($S_k = 1$) or a station section ($S_k = 0$).

M is a large constant and H is the headway parameter. The parameter dir_k gives the direction of the train during event k and nrB_j gives the number of consecutive block sections within one and the same line section j.

References

Assad, A., 1980. Models for rail transportation. Transportation Research Part A 14A, 205–220.

Banverket, 2009. Årsredovisning 2008 (in Swedish), Borlänge, Sweden.

Banverket, 2008. Årsredovisning 2007 (in Swedish), Borlänge, Sweden.

Banverket, 2007. Årsredovisning 2006 (in Swedish), Borlänge, Sweden.

Banverket, 2006. Årsredovisning 2005 (in Swedish), Borlänge, Sweden.

Banverket, 2005. Årsredovisning 2004 (in Swedish), Borlänge, Sweden.

Billionnet, A., 2003. Using integer programming to solve the train-platforming problem. Transportation Science 37 (2), 213–222.

Cordeau, J-F., Toth, P., Vigo, D., 1998. A survey of optimization models for train routing and scheduling. Transportation Science 32 (4), 380–404.

D'Ariano, A., Pacciarelli, P., Pranzo, M., 2007. A branch and bound algorithm for scheduling trains on a railway network. European Journal of Operational Research 32 (2), 643–657.

D'Ariano, A., Pranzo, M., 2009. An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances. Networks and Spatial Economics 9, 63–84.

Emer, J., Hill, M., Patt, Y., Yi, J., Chiou, D., Sendag, R., 2007. Single-threaded vs. multithreaded: Where should we focus? IEEE Micro 27 (6), 14–24.

European Commission, 2001. European transport policy: time to decide 2010. White paper.

Gely, L., Dessagne, G., Pesneau, P., Vanderbeck, F., 2008. A multi scalable model based on a connexity graph representation. In: Proceedings of Comprail 2008, Toledo, Spain.

Jansson, G., Jonsson, O., 2003. En pilotstudie av punktligheten i Europeiska godstransporter på järnväg. TFK Rapport 2003:7, Stockholm, Sweden.

Jacobs, J., 2008. Rescheduling. In: Hansen, Pachl (Eds.), Railway Timetable & Traffic, Analysis, Modelling, Simulation. Eurailpress, Hamburg, Germany, pp. 182–191.

Kroon, L., Huisman, D., Maróti, G., 2008. Optimization models for railway timetabling. In: Hansen, Pachl (Eds.), Railway Timetable & Traffic, Analysis, Modelling, Simulation. Eurailpress, Hamburg, Germany, pp. 135–154.

Kroon, L., Romeijn, E., Zwaneveld, P., 1997. Routing trains through railway stations: complexity issues. European Journal of Operational Research 98, 485–498.

Radtke, A., 2008. Infrastructure modelling. In: Hansen, Pachl (Eds.), Railway Timetable & Traffic, Analysis, Modelling, Simulation. Eurailpress, Hamburg, Germany, pp. 43–57.

Schöbel, A., 2009. Capacity constraints in delay management. Journal of Public Transport 1 (2), 135–154.

Törnquist, J., 2005. Computer-based decision support for railway traffic scheduling and dispatching: a review of models and algorithms. In: Proceedings of ATMOS2005, Palma de Mallorca, Spain, October 2005, Published within the Dagstuhl Research Online Publication Server (DROPS) <http://drops.dagstuhl.de/portals/ATMOS/>.

Törnquist, J., Persson, J., 2007. N-tracked railway traffic re-scheduling during disturbances. Transportation Research Part B 41 (3), 342–362.

Törnquist, J., 2007. Railway traffic disturbance management: an experimental analysis of disturbance complexity, management objectives and limitations in planning horizon. Transportation Research Part A 41 (3), 249–266.

UIRR (International Union of combined Road-Rail transport companies), 2008. Punctuality statistics for 1999–2007. <http://www.uirr.com/?action=page&page=52&title=Quality> (accessed 01.09.09).

Zhou, X., Zhong, M., 2007. Single-track train timetabling with guaranteed optimality: branch-and-bound algorithms with enhanced lower bounds. Transportation Research Part B 41 (3), 320–342.

A PARALLEL HEURISTIC FOR FAST TRAIN DISPATCHING DURING RAILWAY TRAFFIC DISTURBANCE — EARLY RESULTS

Syed Muhammad Zeeshan Iqbal, Håkan Grahn, and Johanna Törnquist Krasemann
School of Computing, Blekinge Institute of Technology, SE-371 79 Karlskrona, Sweden
{*Muhammad.Zeeshan.Iqbal,Hakan.Grahn,Johanna.Tornquist*}*@bth.se*

Keywords: Railway traffic: Disturbance management: Optimization: Re-scheduling: Parallel computing: Multiprocessor

Abstract: Railways are an important part of the infrastructure in most countries. As the railway networks become more and more saturated, even small traffic disturbances can propagate and have severe consequences. Therefore, efficient re-scheduling support for the traffic managers is needed. In this paper, the train real-time re-scheduling problem is studied in order to minimize the total delay, subject to a set of safety and operational constraints. We propose a parallel greedy algorithm based on a depth-first branch-and-bound search strategy. A number of comprehensive numerical experiments are conducted to compare the parallel implementation to the sequential implementation of the same algorithm in terms of the *quality of the solution* and the *number of nodes evaluated*. The comparison is based on 20 disturbance scenarios from three different types of disturbances. Our results show that the parallel algorithm; (i) efficiently covers a larger portion of the search space by exchanging information about improvements, and (ii) finds better solutions for more complicated disturbances such as infrastructure problems. Our results show that the parallel implementation significantly improves the solution for 5 out of 20 disturbance scenarios, as compared to the sequential algorithm.

1 INTRODUCTION

Railways are an important part of the infrastructure in most countries. As the railway traffic networks become more and more saturated, even small traffic disturbances can propagate and have severe consequences. Smooth operation of railway systems are also difficult due to different types of unforeseen larger disturbances such as bad weather or infrastructure failures. When disturbances occur, the timetable needs quickly to be re-defined to minimize the delays and the associated penalty costs for operators and infrastructure providers. However, the large number of constraints and complex infrastructure make re-scheduling difficult and time consuming. Therefore, efficient re-scheduling support for the traffic managers is needed.

In Sweden, the railway transport market is deregulated which means that operators and infrastructure providers are two different entities. The Swedish Transport Administration, Trafikverket, is managing the network both in terms of timetabling and traffic management while the operators arrange and run the train services for passengers and freight. The different private operators apply for desirable slots in competition with each other and Trafikverket assigns slots according to predefined market-based routines. The demand for track capacity has increased the past years in Sweden as well as the number of operators (Törnquist and Persson, 2007). As an effect, the network is becoming more and more saturated and vulnerable every year. The Swedish railway industry therefore seeks decision support systems to assist dispatchers in making good re-scheduling and delay management decisions in real time. Since this re-scheduling problem is a difficult problem, solution approaches based on e.g. traditional optimization techniques often require huge amount of memory space and computation time. Especially the computation time is important to reduce since the problem needs to be solved fast in real-time.

The purpose of this paper is to present a fast and effective approach for railway traffic re-scheduling which aims to minimize the delays during a disturbance by the use of heuristics and parallelization techniques. The approach is a parallel depth-first search (DFS) branch-and-bound (B&B) algorithm based on a sequential greedy algorithm proposed by (Törnquist Krasemann, 2010; Grahn and Törnquist Krasemann, 2011). The parallel DFS algorithm has been evaluated experimentally and benchmarked with the sequential greedy version as well as to state-of the art optimization software, Cplex 12.2, for 20 disturbance scenar-

ios. The simulated disturbance scenarios are used to measure the algorithm efficiency in terms of the *quality of the solution* and the *number of nodes explored*. Our results show that the parallel implementation significantly improves the solution for 5 out of 20 disturbance scenarios, as compared to the sequential algorithm.

In the following section, related work is presented. Section 3 outlines the problem domain and its context as well as a description of the sequential greedy algorithm. Sections 4 and 5 present the parallel algorithm and the experiments performed to analyze its performance, respectively. In Section 6, the experimental results are presented and discussed. Finally, Section 7 concludes our study and provides suggestions for future work.

2 RELATED WORK

The railway traffic delay management and re-scheduling problem has been considered an important and difficult problem since quite some time. Comprehensive reviews of related work can be found in, e.g., (Törnquist, 2005; Conte, 2008; Schachtebeck, 2009) and it has been studied from different perspectives such as capacity, robustness, as well as passenger delay and dissatisfaction. Analysis of heuristics and integer solution methods for solving capacitated re-scheduling delay management problems are given in, e.g., (Schachtebeck, 2009).

The capacitated delay management problem (Schachtebeck, 2009) is a special case of the job shop scheduling (JSS) problem, where train trips are jobs which are scheduled on tracks that are considered as resources. A JSS formulation is also proposed in (Liu and Kozan, 2009) where a blocking parallel-machine JSS is used to model the train dispatching.

A branch and bound (B&B) procedure is proposed for a resource-constrained project scheduling formulation by incorporating an exact lower bound rule and a beam search heuristic for a tight upper bound (Zhou and Zhong, 2007). A four step heuristic is proposed in (Lee and Chen, 2009), in which binary integer linear programming is used to accept or reject proposed solutions.

More recently, the delay management problem has been studied by (Corman, 2010), where the complexity of dispatching is discussed, and mathematical models, based on an alternative graph formulation, along with algorithm enhancements are proposed. A similar problem, but with a different problem setting, is studied in (Törnquist and Persson, 2007) where an Mixed-Integer Linear Program (MILP) formulation for dispatching trains during disturbances is proposed and solved using commercial software. The MILP model showed to be too time-consuming to solve using existing commercial solvers for more severe disturbances. Therefore, a greedy depth-first search branch-and-bound algorithm was developed for addressing the re-scheduling problem (Törnquist Krasemann, 2010) and further improved in (Grahn and Törnquist Krasemann, 2011) with a more efficient branching strategy.

In (Grama and Kumar, 2002), a survey of parallel search methods in combinatorial optimization problems (COP) in connection to artificial intelligence is presented. The work in (Clausen and Perregaard, 1999) can be considered as an extension of (Grama and Kumar, 2002), while adopting the same searching methods (i.e. DFS or BFS) with B&B. Branching strategies named lazy and eager (i.e. in eager, branching is performed before bound calculation but in lazy vice versa) are introduced with performance results (Clausen and Perregaard, 1999). The search procedures are improved by parallel implementations on multiprocessors in the context of constraint programming (Perron, 2004). The advantages and disadvantages of central or distributed together with mixed control schemes for implementation of parallel B&B are discussed (Shinano et al., 1997). Further, a parallel search engine has been devised using different time limits (Perron, 2004).

The focus of this paper is different from related research in other countries since the complete Swedish railway network permits bi-directional traffic. Furthermore, on double-tracked line sections track swapping and using both tracks for traffic in one direction is a commonly used traffic management strategy when re-scheduling the traffic during disturbances. These properties complicate the problem and make it harder to solve. Furthermore, the application of parallelization has not been previously addressed to solve the real-time railway re-scheduling problem.

3 PROBLEM DESCRIPTION AND SEQUENTIAL GREEDY ALGORITHM

3.1 Railway network representation

The railway network consists of *station* and *line* sections, *tracks*, *blocks*, and *events*. Each station and line section can have one or more parallel tracks. All tracks are bi-directional, i.e., the track can be used

for traffic in both directions depending on the schedule. A train uses exactly one track on a station or line section, but which specific track to use is often only predefined for events on stations where the corresponding train has a passenger stop. The track allocation is therefore a part of the re-scheduling problem. Each track is composed of one or several *blocks* connected serially and separated by signals. Each block can hold only at most one train at a time due to the safety restriction imposed by *line blocking*. A track composed of two blocks can in theory hold two trains in the same direction, but not two trains in opposite direction due to the lack of a meeting point. Each train has an individual, fixed route (i.e. the sequence of sections to occupy) which is represented as a sequence of train *events* to execute. A train event is when a certain train occupies a certain section. A train event has certain static properties such as minimum running time, advertised start and end times (i.e. it can not depart earlier than the advertised departure time) and recommended track at stations with passenger connections. The event has also some dynamic properties, e.g., track allocation and start and end times on the section.

3.2 Problem specification

In the train re-scheduling problem we have a disturbance in the railway traffic network forcing us to modify the predefined timetable in line with certain objective(s) and constraints. We have a set of n trains, $T = \{t_1, t_2, \ldots, t_n\}$ on a set of m sections, $S = \{s_1, s_2, \ldots, s_m\}$ where each section $s_j \in \{station, line\}$ have a number of tracks $p \in \{1, \ldots, p_j\}$. A station is called *symmetric* if the choice of track to occupy has no, or negligible, effect on the result. Each train i has a set of events, K_i and the set of *all* train events is denoted as $K = \{K_1, K_2, \ldots, K_n\}$ and its cardinality is: $C = \sum_{i=1}^{|T|} |K_i|$. Each train event k has a predefined start time t_k^{start} and end time t_k^{end} in line with the timetable and which needs to be modified based on the minimum running time d_k. It also belongs to a specific section $s_k \in \{s_1, s_2, \ldots, s_m\}$. Each event is executed on exactly one track of its section.

In Figure 1, 9 trains are shown. Train A has 7 events and each event is associated with a section (e.g., A1 at section 1). There are totally 7 sections, where sections 1, 3, 5, and 7 are stations and sections 2, 4, and 6 are line sections.

The objective of the re-scheduling procedure is to minimize the sum of the final delay suffered by each train at its final destination within the problem instance. The *quality of the solution* is thus given by this objective value, where a lower value indicates an

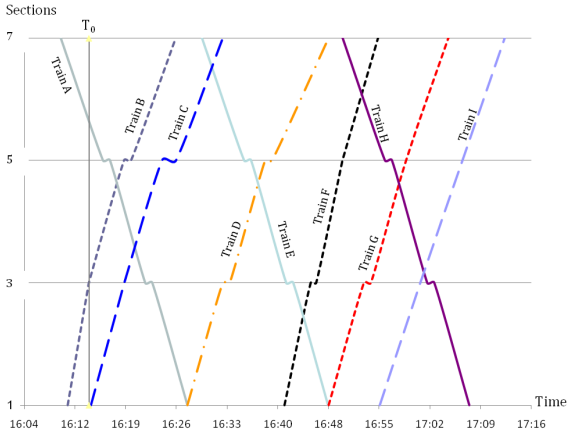


Figure 1: Illustration of railway traffic on a double-tracked line with four stations and three line sections. The time stamp T_0 indicates the time when Train C just has left section 1 and experiences an engine failure. The itinerary of Train C will then look different than from the planned one.

improvement. The *optimal solution* is the one found by the optimization software (Cplex 12.2 in our case). The search space explored is quantified by the *number of nodes* visited.

3.3 A sequential greedy algorithm

The main objective of the sequential greedy algorithm is to quickly find a feasible solution, and therefore it performs a depth-first search. It uses an evaluation function to prioritize when conflicts arise and branches according to a set of criteria. When a first feasible solution has been found the algorithm continues to search for improvements if the time limit permits it. In our experiments we have set the time limit to 30 seconds. A detailed description of algorithm with pseudo code and examples is given in (Törnquist Krasemann, 2010).

A search tree is built iteratively by selecting the earliest event of each train, collecting them into a sorted candidate list, assessing which event to execute first and executing the selected event. An event represents a train movement, i.e., a train running on a certain section with a start time, a minimum running time, a preferred track to occupy, and an end time. Each node in the search tree represents an active or terminated event (i.e.the train has left the assigned track and either started the next event or reached its final destination). For each node, we compute an optimistic estimation of the minimum cost, CV, for the final solution given the partial solution that the branch corresponds to. The further down in the tree a node is located, the more exact the estimation becomes and consequently nodes at maximum depth holds a cost

estimation value which corresponds to the objective value of that solution.

The tree building process is divided into three phases: (i) pre-processing, (ii) depth-first search, and (iii) solution improvement using backtracking and branching on potential nodes. In the pre-processing phase, all events which were active at the disturbance time T_0 (see Figure 1) are executed by allocating a start time and a track. A lower bound, LB, is defined and assigned the value of CV for the end node in the pre-processing phase tree. A sorted next candidate list (i.e., sorted w.r.t the earliest possible starting time of the event), denoted NC, holds the first waiting event of each train. In the second phase, feasible (i.e., deadlock free, without conflicts, etc.) candidate events are executed and removed from the candidate list one by one. The next candidate list is updated accordingly by adding the next waiting event of the train that just executed an event (if it has any left to execute) and is then re-sorted. The third phase starts as soon as the first feasible solution has been found. It aims to improve the best solution found so far by backtracking to a potential node, where the estimated cost, CV, is lower than the current best solution, and explores another branch from there. The improvement process continues until the time limit is reached or a feasible solution with an objective value equal to LB is found.

4 PARALLEL DEPTH-FIRST SEARCH BRANCH AND BOUND ALGORITHM

Our parallel algorithm is based on the sequential greedy algorithm described in Section 3.3, and where the B&B procedure is improved by sharing improved solutions among workers using a synchronized white board. We use a master-slave parallelization strategy. Initially, only the master is active and the workers (slaves) are waiting to get the initial unexplored subspaces. Using the notations in Table 1, we outline the parallel algorithm starting with the master thread.

4.1 The master process

Let *NC* and *PS* be empty, and the disturbance occurs at time T_0. As in the sequential algorithm, identify the events that are active at T_0, execute them, and put them into *PS*. Populate the *NC* with the next event to execute of each train, sorted w.r.t the earliest starting time, and compute the theoretical lower bound. Determine the values of T_c and W where $W = T_c$ in these experiments. A unique copy of the problem along with ET_{Limit}, C_i and *PS* are sent to each worker. Looking at the example in Figure 1, *PS* contains A6, B4 and C2 (i.e., train A associated to section 6 as event A6 etc.), while *NC* consists of A5, B5, C3, D1, E7, F1, G1, H7, and I1. One candidate event each is assigned to the workers to start with. Figure 2 gives an overview of how the parallelization is applied at different depths of the search tree.

Table 1: Notation used in the parallel search.

Symbol		Definition
NC	=	$C_1, C_2, ..., C_n$ where NC is the candidate list
PS	=	partial solution branch
T_0	=	the time when the disturbance occurs
ET_{Limit}	=	execution time limit (30 sec in our experiments)
C_i	=	candidate index to start with
T_c	=	total number of candidates
BV_w	=	branching value
GBV	=	global best value communicated via the white board
CV_w	=	cost estimation value of the current node
W	=	total number of workers
w	=	worker index
S(w)	=	solution branch found by worker w

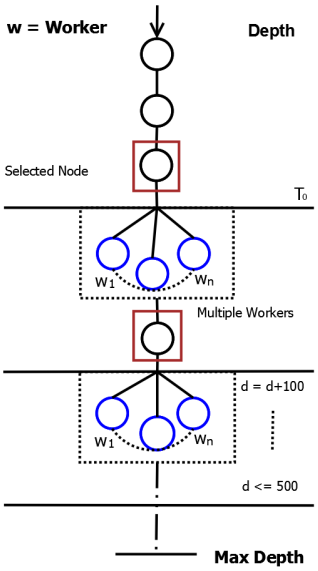


Figure 2: Parallelization at different depths in the search tree.

4.2 The worker process

The outline of the worker process is as follows:
Candidate selection: First execute the candidate C_i, determine the new *NC* and get a suitable candidate based on the depth-first search node selection rule.

Stopping criteria: If the bounds in terms of execution time limit ET_{Limit} is exceeded or the lower bound is reached, then terminate and output the best result. If the candidate to execute, i.e., C_i, is not suitable, then stop execution and return.
Read white board: Read the white board for availability of improved solutions found by any other worker; if available, then update BV_w in line with GBV.
B&B process: When the value of CV_w is greater than or equal to the value of GBV, discard the node, backtrack and try other alternatives, and discard branches from symmetric stations (see Section 3.2).
Feasible solution: If all of the train events are terminated, then update the white board if the new solution found is better than the previously best solution. With an updated value of BV_w, backtrack and start branching from a node with a value less than BV_w.
Deadlock handling: In case of no track availability, backtrack to detect where the wrong decision possibly was made, revoke this decision (i.e. the execution of a certain event) and start branching from there.
Results: After termination, send back the solution S(w) to the master.

5 EXPERIMENTAL SETUP

Our objective is to investigate a number of important aspects through a series of numerical experiments: (1) How does the parallelization point (i.e. at what depth in the tree to start the parallel search) and, (2) the number of workers affect the performance of the parallel algorithm. Given the results from this analysis, indicating a suitable combination of parallelization depth and number of workers, we also investigate 3) how the algorithm performs in comparison to solutions found by the sequential algorithm and the commercial solver Cplex 12.2. The performance analysis is based on the metrics described in Section 3.2. In addition, we investigate how many improved solutions that are found as compare to the sequential algorithm.

In our experiments we consider a dense traffic area of Sweden that consists of both single- and double-tracked line sections as shown in Figure 3. All sections are bi-directional and several of them have multiple blocks. The 28 stations have 2 to 14 tracks each except Norsholm with only one track. The stations Åby, Strångsjö, and Simonstorp are modelled in more detail by defining all the forbidden paths into and out of the stations explicitly, see Appendix B in (Törnquist Krasemann, 2010)

For a systematic and comprehensive assessment of the performance of the parallel algorithm, we have constructed 20 realistic disturbance scenarios. The scenarios are presented in Table 2, and are slightly modified from the scenarios in (Törnquist Krasemann, 2010). The few minor modifications made are done to avoid that the event list of a train ends with an event in the middle of a set of consecutive line sections, e.g., as between the stations Åby and Norrköping. A small number of additional events are therefore included in the scenarios used in this paper. For a 90 minute long time horizon, the third column in Table 2 shows the total number of trains be scheduled, the total number of events, and the number of binary variables required for the corresponding MILP formulation solved by Cplex. The disturbance scenarios cover three types of disturbances:

1. Scenarios 1-10 have initially a temporary single source of delay, e.g., a train comes into the traffic

Table 2: Description of the 20 disturbance scenarios.

No.	Scenario description	#trains/events/ binary variables
1	Long-distance pax train 538, north-bound, delay 12 minutes Linköping-Linghem.	50/549/8214
2	Long-distance pax train 538, north-bound, delay 6 minutes Linköping-Linghem.	50/549/8214
3	Pax train 2138, south-bound, delay 12 minutes Katrineholm-Strångsjö.	50/553/8326
4	Pax train 2138, south-bound, delay 6 minutes Katrineholm-Strångsjö.	50/553/8326
5	Pax train 80866 (north-bound), delayed 12 minutes Linköping-Linghem.	51/565/8430
6	Pax train 80866 (north-bound), delayed 6 minutes Linköping-Linghem.	51/565/8430
7	Pax train 8764 (north-bound), delayed 12 minutes Mjölby-Mantorp.	52/556/8425
8	Pax train 8764 (north-bound), delayed 6 minutes Mjölby-Mantorp.	52/556/8425
9	Pax train 539 (south-bound), delayed 12 minutes Katrineholm-Strångsjö.	52/558/8369
10	Pax train 539 (south-bound), delayed 6 minutes Katrineholm-Strångsjö.	52/558/8369
11	Pax train 538 w. permanent speed reduction causing 50% increased run times on line sections starting at Linköping-Linghem	50/549/8214
12	Pax train 2138 w. permanent speed reduction causing 50% increased run times on line sections starting at Katrineholm-Strångsjö.	50/553/8326
13	Pax train 80866 w. permanent speed reduction causing 50% increased run times on line sections starting at Linköping-Linghem.	50/566/8382
14	Pax train 8764 w. permanent speed reduction causing 50% increased run times on line sections starting at Mjölby-Mantorp.	52/556/8425
15	Pax train 539 w. permanent speed reduction causing 50% increased run times on line sections starting at Katrineholm-Strångsjö.	52/558/8369
16	Speed reduction for all trains between Strångsjö and Simonstorp (all trains get a runtime of 27 min, cf. 5-10 min planned runtime) starting w. freight train 43533.	48/509/7059
17	Speed reduction for all trains between Åby and Simonstorp (all trains get a runtime of 20 min) starting w. train 2138.	53/558/8516
18	Speed reduction for all trains between Åby and Norrköping (all trains get a runtime of 8 min) starting w. train 2138.	51/554/8224
19	Speed reduction for all trains between Mjölby and Mantorp (all trains get a runtime of 20 min) starting w. train 8764.	52/556/8224
20	Speed reduction for all trains between Linköping and Linghem (all trains get a runtime of 15 min) starting w. train 538.	50/549/8214

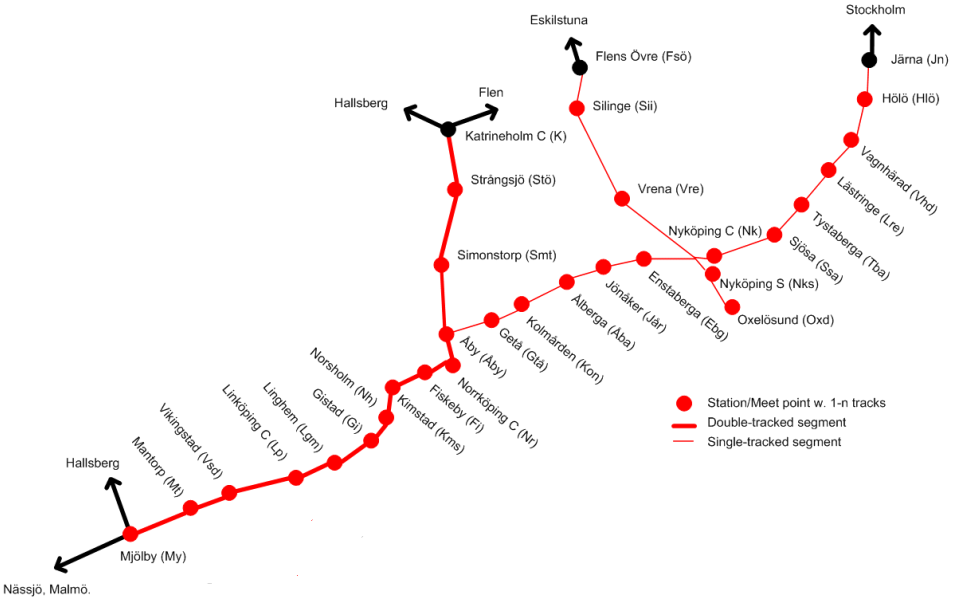


Figure 3: The traffic area in Sweden that are used in the study. It has in total 28 stations, all line sections are bi-directional, wide lines indicate double-tracked sections, and thin lines single-tracked.

management district with a certain delay, or it suffers from a temporary delay at one section within the district.

2. In scenarios 11-15, a train has a 'permanent' malfunction resulting in increased running times on all line sections it is planned to occupy.
3. In scenarios 16-20, the disturbance is an infrastructure failure causing, e.g., a speed reduction on a certain section, which results in increased running times for all trains running through that section.

The sequential and parallel algorithms are implemented in Java using the multithreaded API with JDK 1.6, and all experiments are conducted on a server running Ubuntu 10.04 and equipped with two quad-core processors (Intel Xeon E5335, 2.0 GHz) and 16 GB main memory. The execution time limit ET_{Limit} is set to 30 seconds. Cplex (version 12.2) was run on a AMD Opteron(tm) 285 quad-core processor and in parallel, deterministic mode with 4 threads and given a time limit of 24 hours. We also set the time limit for Cplex to 30 seconds, but it did not manage to provide any feasible solution in all 20 disturbance scenarios within this time.

6 EXPERIMENTAL RESULTS

In this section, the results from the experimental evaluation of the parallel algorithm are presented. First, we analyze how the parallelization affects the performance and at which parallelization depths solution improvements are found. Secondly, a statistical analysis is made for the second performance metric, i.e., the number of visited nodes. This part of the evaluation focuses on one very complex disturbance scenario, i.e., scenario 20 in Table 2.

Finally, we compare the parallel algorithm with the sequential algorithm and the optimal solutions found by Cplex for all 20 disturbance scenarios.

Our evaluation metrics are the *solution quality*, i.e., the sum of the final delay of all trains, and the *number of nodes explored*, see Section 3.2 .

6.1 Analysis of Parallelization Depths and Number of Workers

In the evaluation of how the number of workers and the parallelization depth affect the solution quality, we selected six different parallelization depths: at T_0, 100, 200, 300, 400, and 500 nodes as shown in Figure 2. For each depth, we use 4 different sets of workers containing 8, 16, 32, and the maximum number of available candidates at the particular depth in the search tree. Table 3, shows the experimental results for all combinations of the number of workers and the depth for disturbance scenario 20. In Table 3, the different solutions are characterized by the worker id (i.e. which worker found the solution), the associated solution value, and the time to find that solution. The minimum time to find an initial feasible solution and

Table 3: Experimental results for scenario 20 using a time horizon of 90 minutes and 30 seconds execution time, for different number of workers and parallelization depths.

Depth	Nodes visited	Solutions (Worker id, cost, time)	Nodes visited	Solutions (Worker id, cost, time)
	8 Workers		16 Workers	
T_0	8 573 135	(2, 27208, 0.24), (3, 27186, 0.49), (0, 23609, 0.52), (0, 23587, 0.83)	7 841 418	(5, 27208, 0.56), (11, 27186, 0.84), (1, 23609, 0.91), (1, 23587, 1.22)
100	8 048 136	(7, 27186, 0.51), (0, 23609, 0.60), (5, 23587, 0.64)	8 381 525	(12, 27208, 0.14), (8, 27186, 0.56), (5, 23587, 0.77)
200	7 334 836	(3, 27186, 0.16), (2, 23587, 0.23)	8 440 333	(1, 27208, 0.18), (15, 27186, 0.24), (8, 23587, 0.41)
300	4 699 518	(6, 27208, 0.06), (5, 27186, 0.14), (7, 23587, 0.19)	7 816 092	(12, 27208, 0.07), (11, 27186, 0.34), (9, 23587, 0.47)
400	4 730 399	(5, 27186, 0.10), (4, 23587, 0.15)	7 790 739	(11, 27208, 0.07), (8, 27186, 0.21), (5, 23587, 0.30)
500	1 916 351	(6, 27186, 0.07), (7, 23587, 0.12)	1 986 419	(7, 27186, 0.09), (6, 23587, 0.16)
	32 Workers		Maximum number of Workers	
T_0	6 817 627	(5, 27208, 1.27), (26, 26765, 1.10), (29, 23609, 1.65), (26, 23166, 2.06), **(26, 23144, 2.52)**	5 722 240	(19, 27208, 1.03), (32, 26765, 1.14), (31, 23609, 2.34), (26, 23166, 2.56), **(26, 23144, 3.70)**
100	7 065 428	(2, 27208, 0.34), (14, 27186, 1.45), (7, 23609, 1.59), (1, 23587, 1.69)	6 668 677	(9, 27208, 0.33), (6, 27186, 1.43), (5, 23587, 1.83)
200	7 418 218	(1, 27208, 0.25), (28, 27186, 0.57), (3, 23587, 0.83)	7 160 066	(4, 27208, 0.17), (7, 27186, 0.70), (18, 23587, 0.72)
300	6 626 197	(6, 27208, 0.24), (6, 27208, 0.24), (12, 23587, 0.77)	7 044 556	(14, 27208, 0.11), (20, 27186, 0.52), (8, 23587, 0.73)
400	7 547 346	(8, 27186, 0.48), (5, 23587, 0.69)	7 704 260	(6, 27208, 0.09), (8, 27186, 0.46), (10, 23587, 0.59)
500	2 007 587	(6, 27186, 0.10), (7, 23587, 0.16)	1 911 910	(6, 27186, 0.07), (7, 23587, 0.13)

maximum time to find an improved solution is 0.06 sec and 3.70 sec, respectively.

6.1.1 Number of Workers

We start by analyzing the effect of the number of workers, focusing on the results for parallelization depth T_0. At depth T_0 (i.e., the time when the disturbance occurs), the number of trains to re-schedule (i.e. the size of the candidate list) is as large as possible. For scenario 20, the maximum number of candidate events is 50, as can be seen in the third column in Table 2. Thus, we can potentially explore 50 candidates (branches/subtrees) in parallel at depth T_0. Looking at the solutions found by selecting T_0 in Table 3, we observe that both more (5 solutions) and better solutions (a total delay of 23144 s) are found when we use the maximum number of workers (i.e. 50) and 32 workers, as compared to the solutions found when using only 8 or 16 workers (4 solutions and a total delay of 23587). This is also indicated by the fact that it is worker 26 (which evaluates candidate event 26 in the initial next candidate list) that finds the best solution. Therefore, we conclude for this type of disturbance scenario that the parallel algorithm should explore as many candidates as possible concurrently when the parallel phase starts.

6.1.2 Parallelization Depth

When evaluating at which depth in the search tree it is most beneficial to start the parallel phase, we focus on the case with the maximum number of workers in Table 3. From the results, we can observe two important things. First, the best solution (23144) is found when the parallel execution starts at depth T_0. Further, it is only when the parallel search starts as high up in the tree as possible, i.e., at T_0, that we find this best solution. Second, looking at the number of nodes visited, we find that if we start the parallel search too far down in the search tree, in this case at depth 500, the number of nodes explored decreases drastically. For this type of disturbance scenario, we can conclude that the parallel search should start as high up in the tree as possible.

One important aspect, which affects the performance of the B&B procedure in the algorithm significantly, is how the cost estimate, CV_w at intermediary nodes in the tree reflects the effect of each decision and the resulting complete solution. That is, does the optimistic delay estimation provide enough information to guide the branching procedure well, so that the pruning and selection of nodes to explore are effective. As shown in Figure 4, the cost increment trend is very similar for all solutions up to depth 500 approximately, after that they start to diverge sig-

Table 4: Statistical Metrics

Nodes Visited				
No. of Workers	Mean	Standard Deviation (σ)	95% CI	95% C I half size
8	8584243	90090	7896.58	0.09
16	7749170	146961	12881.46	0.17
32	7148119	243361	21331.14	0.30
52	6148055	515094	45149.14	0.73

nificantly. The higher divergence is found deep in the tree. The implication of this is that it becomes harder to perform efficient pruning of non-promising branches early, i.e., high up in the search tree. Consequently, the efficiency of the algorithm (the sequential as well as the parallell version) could potentially be increased by computing additional information about the "goodness" of the partial solutions and use this in the B&B procedure.

6.2 Statistical Analysis

A measurement issue is associated with the second metric (i.e., the number of nodes explored), because it is not deterministic when the number of workers is higher than the number of available CPU cores due to the underlying platform scheduling policy. To cope with this validity threat, a statistical analysis is made. Table 4 shows the results for the number of nodes visited for different number of workers, averaged over 500 repeated experiments with standard deviation σ and 95% confidence interval. The mean value shows that the number of visited nodes decreases and the corresponding standard deviation increases as the number of workers increases. The meaning of the results is that when the number of workers are equal to the number of cores, then the maximum number of nodes are visited with less standard deviation σ.

6.3 Comparative Evaluation

In Table 5, we present the results from a comparative evaluation between the sequential and parallel algorithm, along with a comparison with the optimal solutions found be Cplex. Note that the algorithms are only executed 30 seconds, while Cplex are executed 24 hours in order to find the best solution. Cplex did not manage to provide any feasible solution within 30 seconds. The experimental results in terms of number of nodes visited by the algorithms, are presented in the second and third column. The fourth and fifth column show the solutions (total delay of the trains) found by the sequential and parallel implementations. The optimal solutions found by Cplex are given in the sixth column. The difference in solution quality, i.e. the delay difference given in time units, between the best solution found by the sequential as well as the parallel algorithm, respectively, as compared to the optimal solution provided by Cplex is presented in the last two columns in Table 5. Rather than computing the percentage of improvement by the parallel algorithm and the optimal gap, we find that the reduction given in time units provides a more practical viewpoint of an improvement. That is, a large percentage delay reduction is irrelevant to spend time on if the best found solution value is already as low as 4-5 minutes (e.g. scenario 10), while a small percentage improvement is highly relevant if the best solution found is as large as in scenario 20, as an example.

In Table 5, starting with the quality of the solution, we observe that the parallel algorithm finds better solutions than the sequential algorithm in disturbance scenarios 1, 5, 9, 17, and 20 (shown in bold). For example, the parallel algorithm finds a solution with a final delay of 701 seconds in scenario 5, while the best solution found by the sequential algorithm has a total delay of 930 seconds. Comparing the best parallel solutions with the optimal solutions found by Cplex, we observe that in most cases the solutions found by the parallel algorithm are close to optimal.

The other aspect we compare is how large part of the search space the sequential and the parallel algorithms explore. We measure this by counting the number of nodes visited by each of the algorithms. By comparing column 2 and 3 in Table 5, we observe that the parallel algorithm explores between 4.6-6.4 times more nodes than the sequential algorithm.

7 CONCLUSIONS AND FUTURE WORK

This study aims to solve the railway re-scheduling problem efficiently by proposing a parallel algorithm. The parallel algorithm successfully improves the solutions in disturbance scenarios 1, 5, 9, 17, and 20 as compared to the sequential counterpart. By considering different candidates concurrently at specified depths, a number of alternatives are evaluated. Our results indicate that the parallel algorithm explores significantly more nodes of the search space, approximately 5-6.3 times as many as the sequential version on an 8-core machine. Further, the parallel algorithm successfully improves the found re-scheduling solutions for complicated disturbances, and offers superior performance with limited computational cost.

From the experimental results, we can see that all solutions were found within 5 seconds, which shows

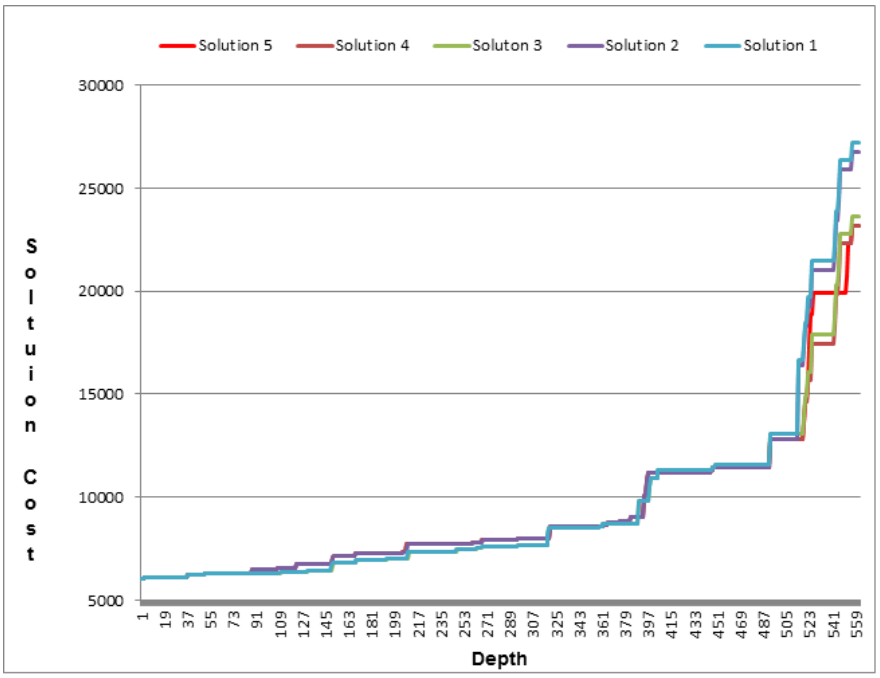


Figure 4: The value of the cost function, i.e., the total delay for all trains, at different levels in the search tree for the five solutions to disturbance scenario 20.

that the algorithm and its parallelized version are very fast and effective for the considered problem. It also indicates that the behavior of the algorithm may need to be modified, and the workers may need to adopt different, complementary search schemes in such scenarios when the disturbance is more complex to solve (e.g., scenario 20). There are a number of potential modifications of the approach that may lead to further improvements and which we plan to investigate further:

i) Improve the ranking of the candidates.

ii) Compute better lower bounds and estimates of the final solution value at intermediary nodes.

iii) Find a correlation between the intermediary node values, the corresponding node depth and solution value to enhance early promising exploration and pruning.

Furthermore, not all workers find a feasible solution since some of them may continuously get interrupted by new information making them prune its current branch and expand on another node further up the tree. Perhaps it would be reasonable to assign different search behavior to some of the workers. Based on additional information as outlined above, certain workers could apply a slightly more random and experimental search strategy (c.f. Simulated Annealing). The information communicated between and used by the workers is at this point kept at a minimum. Some additional information about not only progress but also unsuccessful moves should be communicated (e.g., the use of a tabu list as in Tabu Search).

ACKNOWLEDGMENT

We would like to thank Trafikverket(the Swedish Transport Administration), formerly known as Banverket, for providing financial support for the project EOT (Effektiv operativ Omplanering av Tåglägen vid Driftstörningar). We would also like to thank Prof. Markus Fiedler at Blekinge Institute of Technology for useful discussions.

REFERENCES

Clausen, J. and Perregaard, M. (1999). On the best search strategy in parallel branch-and-bound: Best-First search versus lazy Depth-First search. *Annals of Operations Research*, 90:1–17.

Conte, C. (2008). *Identifying dependencies among delays*. PhD thesis, Niedersächsische Staats-und Universitätsbibliothek Göttingen, Germany.

Corman, F. (2010). *Real-time Railway Traffic Management: Dispatching in complex, large and busy railway networks*. Ph.D. thesis, Technische Universiteit Delft, The Netherlands. 90-5584-133-1.

Table 5: Experimental results for all scenarios using a time horizon of 90 minutes and 30 sec. execution time.

No.	Nodes Visited		Found solutions (s)			Difference (s)	
	Sequential Algorithm	Parallel Algorithm	Sequential Algorithm	Parallel Algorithm	Cplex version 12.2 in 24h	Sequential Algorithm	Parallel Algorithm
1	1 439 990	9 052 530	1489, 1175	1489, 1486, 1175, **1172**	855	320	317
2	1 384 481	8 726 846	751, 437	751, 714, 628, 437	226	211	211
3	1 407 388	7 488 996	1150, 781	1150, 1087, 781	570	211	211
4	1 404 736	8 300 651	790, 421	790, 727, 421	210	211	211
5	1 429 323	7 549 277	1188, 930	1188, 793, **701**	686	244	15
6	1 387 105	8 554 538	68, 53	68, 53	30	23	23
7	1 339 729	8 050 851	568, 499	568, 499	486	13	13
8	1 438 316	8 469 438	276, 207	276, 207	176	31	31
9	1 314 606	7 606 163	869, 800	869, 813, 800, **744**	731	69	13
10	1 335 729	7 563 889	338, 269	338, 269	256	13	13
11	1 382 442	9 125 263	1547, 1233	1955, 1930, 1815, 1429, 1233	1022	211	211
12	1 422 193	9 001 247	1049, 680	6856, 1457, 1355, 876, 871, 680	469	211	211
13	1 406 908	7 408 835	2503, 2245	3279, 2711, 2613, 2401, 2360, 2245	2230.5	14.5	14.5
14	1 419 492	8 473 574	1627, 1519	1783, 1731, 1709, 1519	1112.5	406.5	406.5
15	1 330 892	7 511 906	1728, 1659	1728, 1659	1598.5	60.5	60.5
16	1 328 808	8 476 918	13850	13850	13850	0	0
17	1 349 033	7 527 636	7109, 7088	7109, 7088, **7069**	7038	50	31
18	1 359 288	7 299 110	23940, 18692, 18672, 14679, 14419, 4494, 4295	23940, 18692, 18672, 14679, 14419, 4494, 4295	4130	165	165
19	1 216 437	5 554 610	28883	28883	28740	143	143
20	1 244 374	5 722 240	27208, 27186, 23609, 23587	27208, 26765, 23609, 23166, **23144**	18971	4616	4173

Grahn, H. and Törnquist Krasemann, J. (2011). A parallel re-scheduling algorithm for railway traffic disturbance management — initial results. In *Proc. of the 2nd Int'l Conference on Models and Technologies for Intelligent Transportation Systems*, pages XX–YY.

Grama, A. and Kumar, V. (2002). State of the art in parallel search techniques for discrete optimization problems. *IEEE Trans. on Knowledge and Data Engineering*, 11(1):28–35.

Lee, Y. and Chen, C.-Y. (2009). A heuristic for the train pathing and timetabling problem. *Transportation Research Part B: Methodological*, 43(8-9):837 – 851.

Liu, S. Q. and Kozan, E. (2009). Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers & Operations Research*, 36(10):2840 – 2852.

Perron, L. (2004). Search procedures and parallelism in constraint programming. In *Principles and Practice of Constraint Programming (CP'99)*, pages 346–361.

Schachtebeck, M. (2009). *Delay Management in Public Transportation: Capacities, Robustness, and Integration*. PhD thesis, Niedersächsische Staats-und Universitätsbibliothek Göttingen, Germany.

Shinano, Y., Harada, K., and Hirabayashi, R. (1997). Control schemes in a generalized utility for parallel branch-and-bound algorithms. In *Proc. of the 11th Int'l Parallel Processing Symp.*, page 621.

Törnquist, J. (2005). Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. In *5th Workshop on Algorithmic Methods and Models for Optimization of Railways*.

Törnquist, J. and Persson, J. A. (2007). N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B: Methodological*, 41(3):342–362.

Törnquist Krasemann, J. (2010). Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. *Transportation Research Part C: Emerging Technologies*, In Press, Corrected Proof.

Zhou, X. and Zhong, M. (2007). Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds. *Transportation Research Part B: Methodological*, 41(3):320 – 341.

A Comparative Evaluation of Re-scheduling Strategies for Train Dispatching during Disturbances

S.M.Z. Iqbal, H. Grahn, and J. Törnquist Krasemann
School of Computing, Blekinge Institute of Technology,
SE-371 79 Karlskrona, Sweden

Abstract

Railway traffic disturbances occur and train dispatchers make re-scheduling decisions in order to reduce the delays. In order to support the dispatchers, good re-scheduling strategies are required that could reduce the delays. We propose and evaluate re-scheduling strategies based on: (i) earliest start time, (ii) earliest track release time, (iii) smallest buffer time, and (iv) shortest section runtime. A comparative evaluation is done for a busy part of the Swedish railway network. Our results indicate that strategies based on earliest start time and earliest track release time have the best average performance.

Keywords: Railway traffic, Disturbance management, Optimization

1 Introduction

Today's railway networks are becoming more and more saturated, and even small single disturbances can propagate and have severe consequences. In case of a disturbance, the trains need to be re-scheduled in order to minimize the delays in the railway network. Dispatchers make re-scheduling decision by, e.g., changing tracks, modifying train orders, and changing departure times. The re-scheduling decisions required depend on the disturbance type, and when and where it occurred.

We have addressed the re-scheduling problem by proposing an optimization-based approach [1], implementing a greedy depth-first search branch-and-bound algorithm [2], and implementing a parallel version of this algorithm [3]. These studies have shown that the performance, i.e., the ability to find a good re-scheduling solution, is very dependent on which train event that is selected for execution.

Therefore, the search algorithm needs good guidance when selecting the most promising candidates to re-scheduling at each step in the algorithm. In this study, we focus on different strategies to guide the search for re-scheduling solutions in the greedy algorithm. We propose and evaluate re-scheduling strategies based on: (i) earliest start time, (ii) earliest track release time, (iii) smallest buffer time, and (iv) shortest section runtime.

The solution quality of the proposed strategies is evaluated using experiments with data from a busy part of the Swedish railway network. The time horizon for re-scheduling is 90 minutes and we study 100 disturbance scenarios in three different categories, i.e., the initial source of delay is (1) a single train is delayed, (2) a single train has a permanent speed reduction, and (3) all trains are delayed on a particular section. All consecutive delays are also correctly modeled.

The results show that the strategies that prioritize earliest start time first and the earliest track release time first have the best average performance. However, we have found that the strategies complement each other, since none of the evaluated strategies is superior in all cases. Further, the strategies are sensitive to how large the initial disturbance is.

2 Related Work

Train re-scheduling during railway traffic disturbances is an important problem. An extensive survey is done in [4], where published work is differentiated based on, e.g., infrastructure representation and various traffic properties.

A Mixed-Integer Linear Program (MILP) model is proposed for train re-scheduling of N-tracked railway traffic during disturbances [1]. In [5], the same model as proposed in [1] is used along with two solution methods: (i) right-shift re-scheduling to produce the initial feasible solution and (ii) local search to limit the search.

The train scheduling problem can be formulated as a job shop scheduling problem, as in [6, 7], where train trips are jobs which are scheduled on tracks that are considered as resources. Two studies [8, 6] addressed the problem from the perspectives of capacity, robustness, and dependencies. The heuristics and integer solution methods along with analysis are given in [6]. A variable speed dispatching system is proposed in [7] to control railway traffic by considering acceleration and deceleration time in the model. Furthermore, the work in [7] extends [9] with detailed microscopic and comprehensive models to fulfill additional requirements.

Studies of the computational complexity of disturbance handling in large railway networks are done in [10] and [11]. An experimental study of optimization (i.e. Minimize delay cost) and myopic based policies (i.e First Come First Served etc.) concludes that both complement each other [10]. The performance of each one is dependent on the region and disturbance type. A performance evaluation of centralized and distributed strategies for dispatching trains is given in [11].

A greedy depth-first search branch-and-bound algorithm is proposed in [2] to handle the re-scheduling problem. It generates a feasible solution within 30 sec in most cases. Recently, we have presented a parallel search heuristic [3] based on the greedy algorithm in [2] to reduce the delays.

3 Proposed Re-scheduling Strategies

The greedy algorithm [2] searches for a solution by trying to schedule all train events in some order, with the goal of minimizing the final delay for all trains at their destination. A train event, e, represents a train movement on a line section or a train stop at a station. In each search step, i.e., scheduling the next event from a candidate list, the algorithm prioritizes the most promising event. Our strategies outlined below select the most promising event based on different objectives.

The greedy algorithm [2] works in three phases when searching for solutions: (i) pre-processing, (ii) depth-first search to find a first feasible solution, and (iii) backtracking and improvement search. The pre-processing phase executes the events that were active at the disturbance time T_0. After that, a lower bound is calculated and a candidate list (NC, sorted according to one of the proposed strategies) is constructed. NC contains the next event to execute for each train. In the second phase, feasible events from the candidate list are executed one by one. When an event has been executed, the candidate list is updated with the next event of the train and re-sorted. The process goes on until a first feasible solution is found. The algorithm then searches for improved solutions in the third phase using backtracking and branch-and-bound until the time limit is reached.

Strategy s_0: s_0 is the strategy implemented in [2] and gives precedence to events that have the earliest start time. It has shown effective to find a first feasible solution. The candidate list, NC, with events is sorted with respect to the following condition: $t_{e'}^{min\_start} < t_{e''}^{min\_start}$, where e' and e'' represent train events in NC and $t_e^{min\_start}$ is the *minimum start time* for an event e. When $t_{e'}^{min\_start} = t_{e''}^{min\_start}$, then $t_{e'}^{min\_start} + t_{e'}^{runtime} < t_{e''}^{min\_start} + t_{e''}^{runtime}$ is used as a second criteria. $t_e^{runtime}$ represents the *section run time* of train event e.

Strategy s_1: The motivation behind s_1, is that strategy s_0 does not consider track release time. A train with long section run time may delay other trains significantly. Strategy s_1 tries to minimize the delay caused by late track release times by sorting NC according to $t_{e'}^{release} < t_{e''}^{release}$. We divide s_1 into two sub-strategies: $s_{1\alpha}$ and $s_{1\beta}$.

Strategy $s_{1\alpha}$ calculates the track release time as $t_e^{release} = t_e^{min\_start} + t_e^{stop} - t_e^{start}$, where t_e^{start} and t_e^{stop} are planned start and stop times, respectively, of event e. If a set of events have the same $t_e^{release}$, we calculate the release time as $t_e^{release} = t_e^{min\_start} + t_e^{runtime}$ as a second sorting criteria.

A railway network has both *station* and *line* sections. Therefore, we introduce strategy $s_{1\beta}$. The track release time is different for events on a *station* as compared to on a *line* section. If an event has a planned stop at a station, then we consider its t_e^{stop} otherwise $t_e^{runtime}$. For both types of sections, the release time is calculated by condition (1), where s_e is the section type for event e.

$$t_e^{release} = \begin{cases} t_e^{stop} & \text{if } s_e = station \text{ and } planned\_stop = true \\ & \quad \text{and } t_e^{deviation} < t_e^{buffer} \\ t_e^{min\_start} + t_e^{runtime} & \text{otherwise} \end{cases} \tag{1}$$

Strategy s_2: A timetable is designed with buffer times to absorb minor delays, where $t_e^{buffer} = t_e^{stop} - t_e^{start} - t_e^{runtime}$. Strategy s_2 seeks to take advantage of the buffer times and also tries to ensure that the buffer times are fully utilized, thereby aiming at minimizing the delay. The comparison between two events is done based on the condition: $t_{e'}^{min\_start} + t_{e'}^{buffer} < t_{e''}^{min\_start} + t_{e''}^{buffer}$. It behaves as strategy s_0 for events with no buffer time.

Strategy s_3: The strategy s_0 uses the $t_e^{runtime}$ partially when two events have the same start time. Therefore, we introduce a strategy that is based on the *minimum section runtime* (i.e. the minimum time required by each train to use the section) and it is expected to perform well to minimize the delay. It gives precedence to event e' over event e'' as follows: $t_{e'}^{min\_start} + t_{e'}^{runtime} < t_{e''}^{min\_start} + t_{e''}^{runtime}$. Strategy s_3 behaves similarly to s_0 for larger delays.

4 Experimental Methodology

In our experimental evaluation we will address how well the proposed strategies perform regarding re-scheduling the trains. The following main questions have been addressed: (i) For how many disturbance scenarios do the re-scheduling strategies find solutions? (ii) How well do the strategies perform relative to each other in terms of total delay at the final destination for all trains? (iii) How sensitive are the strategies to delay variations and the complexity of the disturbance scenarios?

4.1 Input Data and Setup

We have considered a dense traffic area of Sweden as shown in Figure 1. The 28 stations have 2 to 14 tracks each except Norsholm with only one track. All stations are modeled in detail, including forbidden paths into and out of the stations, see Appendix B in [2]

We use 20 disturbance scenario sets divided into three categories in our evaluation, where each set has five delay variations. In total, we use 100 disturbance scenarios. The sets of disturbance scenarios are described in Table 1. Category 1 has six sets of scenarios where a train has an initial temporary single source of delay, e.g., a train suffers from a temporary delay at one section within the district. Category 2 has seven sets of scenarios where a train has a permanent malfunction resulting in increased running times on all line sections it is planned to occupy. Category 3 has seven sets of scenarios where the disturbance is an infrastructure failure causing, e.g., a speed reduction on a certain section which results in increased running times for all trains running through that section. In all scenarios and experiments we correctly model all consecutive delays that occur.

The greedy algorithm with the evaluated strategies is implemented in Java with JDK 1.6. All experiments are conducted on a server running Ubuntu 10.04 and equipped with two quad-core processors (Intel Xeon E5335, 2.0 GHz) and 16 GB main memory. We have set an execution time limit of 30 seconds for the algorithm.

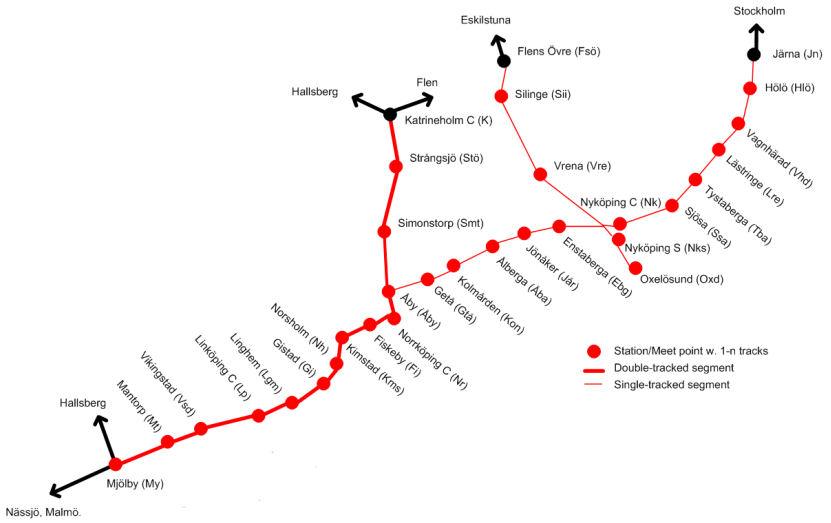


Figure 1: The traffic area in Sweden that are used in the study. It has in total 28 stations, all line sections are bi-directional, wide lines indicate double-tracked sections, and thin lines single-tracked.

4.2 Performance Metrics

The main performance metric is the total final delay for all trains at their destination. The performance of each strategy, i.e., $s_{1\alpha}$, $s_{1\beta}$, s_2, and s_3, is compared to the base strategy s_0 [2]. We start by comparing for how many scenarios do the different strategies find any solutions, and if their best solutions are better or worse than the best solution for s_0. We measure the relative performance $RP_i^{s_x}$ of each strategy s_x for each scenario i, see Eqn (2), where $C_i^{s_0}$ and $C_i^{s_x}$ denote the final solution value (i.e., the total delay at destination) found by strategy s_0 and s_x, respectively. $RP_i^{s_x} > 1$ means that strategy s_x is better than s_0, and $RP_i^{s_x} < 1$ means that strategy s_x is worse than s_0 for scenario i.

We also calculate the relative improvement or degradation $\mu_i^{s_x}$ for each strategy s_x compared to the base strategy s_0 for scenario set S_i in percentage, see Eqn (2). Our objective is to evaluate the average relative improvement or degradation $\bar{\mu}^{s_x}$ for each strategy s_x. The average is taken over all scenarios, N, where solutions are available.

$$RP_i^{s_x} = \frac{C_i^{s_0}}{C_i^{s_x}}, \quad \mu_i^{s_x} = \left(\frac{C_i^{s_0} - C_i^{s_x}}{C_i^{s_0}}\right) \times 100, \quad \bar{\mu}^{s_x} = \frac{\sum_{i=1}^{N} \mu_i^{s_x}}{N} \tag{2}$$

5 Experimental Results

5.1 Number of Scenarios Where Solutions are Found by the Strategies

The overall performance of the different re-scheduling strategies for the three categories of disturbance scenarios is shown in Figure 2. It shows the number of

Table 1: Description of the disturbance scenarios divided into three categories. Each set of scenarios has five different delay variations. These initial delays generate consecutive delays that we also model.

Set	Description
Category 1 - Delay variation 6, 9, 12, 15, and 25 minutes initial delay for a specific train	
S_1	Pax train 8762, north-bound, delay Vikingstad-Linköping, having 16 events
S_2	Pax train 538, north-bound, delay Linköping-Linghem, having 20 events
S_3	Pax train 2138, south-bound, delay Katrineholm-Strångsjö, having 9 events
S_4	Pax train 539, south-bound, delay Katrineholm-Strångsjö, having 30 events
S_5	Pax train 8765, south-bound, delay Linköping-Linghem, having 10 events
S_6	Pax train 8764, north-bound, delay Mjölby-Mantorp, having 20 events
Category 2 - Delay variation 50%, 75%, 100%, 200%, 300% increased running times for a specific train	
S_7	Pax train 8767 w. permanent speed reduction causing increased run times on line sections starting at Linghem-Gistad, having 13 events
S_8	Pax train 8765 w. permanent speed reduction causing increased run time on line sections starting at Linköping-Linghem, having 10 events
S_9	Pax train 538 w. permanent speed reduction causing increased run times on line sections starting at Linköping-Linghem, having 20 events
S_{10}	Pax train 2138 w. permanent speed reduction causing increased run times on line sections starting at Katrineholm-Strångsjö, having 9 events
S_{11}	Pax train 80866 w. permanent speed reduction causing increased run times on line sections starting at Linköping-Linghem, having 35 events
S_{12}	Pax train 8769 w. permanent speed reduction causing increased run times on line sections starting at Fiskeby-Norrköping, having 20 events
S_{13}	Pax train 539 w. permanent speed reduction causing increased run times on line sections starting at Katrineholm-Strångsjö, having 30 events
Category 3 - Delay variation 7, 11, 15, 20, and 28 minutes running time for all trains on a particular section	
S_{14}	Speed reduction for all trains (i.e. 16) between Linghem-Gistad starting w. train 8767
S_{15}	Speed reduction for all trains (i.e. 17) between Linköping-Linghem starting w. train 80866
S_{16}	Speed reduction for all trains (i.e. 19) between Fiskeby-Norrköping starting w. train 8769
S_{17}	Speed reduction for all trains (i.e. 18) between Åby and Norrköping starting w. train 2138
S_{18}	Speed reduction for all trains (i.e. 14) between Vikingstad and Linköping starting w. train 8762
S_{19}	Speed reduction for all trains (i.e. 14) between Mantorp and Vikingstad starting w. train 8764
S_{20}	Speed reduction for all trains (i.e. 16) between Linköping and Linghem starting w. train 538

scenarios where solutions are found, and now many of these solutions are better or worse than s_0. The horizontal dotted line shows the number of scenarios (i.e. 83) where the base strategy s_0 found a solution. Strategy s_0 finds a solution for

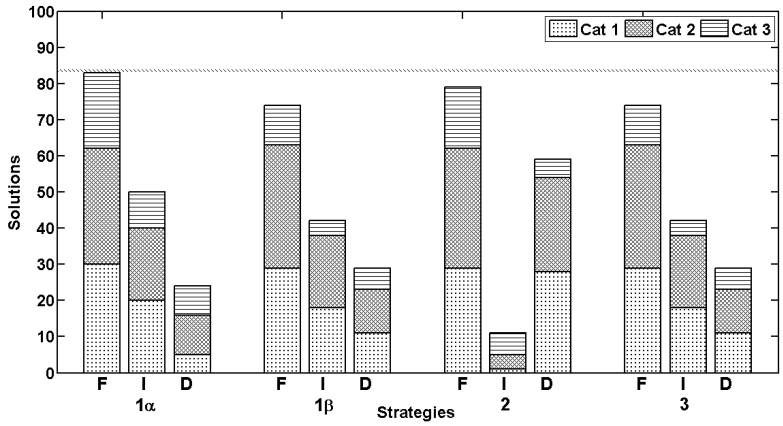


Figure 2: Number of scenarios for with solutions are found for each strategy and for each disturbance category.

all scenarios of category 1, and it finds a solution in 34 and 19 out of 35 scenarios in category 2 and 3, respectively. The labels on the x-axis represent found (F), improved (I), and degraded (D) solutions, respectively, for each strategy.

As Figure 2 shows, strategy $s_{1\alpha}$ is generally better in terms of number of scenarios where improved and degraded solutions are found in each category. The number of improvements is large for disturbance scenarios of category 1 and category 2 because these are less complex and hence, less risk of deadlock. However, strategy $s_{1\alpha}$ finds only few improvements (i.e. 10 of 21 found solutions) for category 3 scenarios, since they are more complex where all trains on a particular section are delayed.

The performance of $s_{1\beta}$ and s_3 does not differ for any category of scenarios as shown in Figure 2. Strategy $s_{1\beta}$ behaves like s_3 when the event deviation is larger than the buffer time. Strategy s_2 finds improved solutions in only a few scenarios (i.e. 11 of 79 found solutions) because it, e.g., prioritizes events that have small buffer times but large occupation times.

The strategies are not able to find any solution on average for 23 scenarios out of 100, and most of them belong to category 3. The algorithm goes into a cycle, which means that there is no event in the candidate list that is suitable to execute.

The results show that $s_{1\alpha}$ finds 50 improved and 24 degraded solutions of 83 found solutions. Hence, we conclude that the performance of $s_{1\alpha}$ in terms of the number of improvements is better as compared to the other strategies.

5.2 Relative Improvement or Degradation

We want to analyze the performance of strategies in terms of $\bar{\mu}^{s_x}$, i.e., average relative improvement or degradation. The value of $\bar{\mu}^{s_x}$ is given in Table 2 for each

Table 2: Average relative improvement or degradation in percent, $\bar{\mu}^{s_x}$, over the base strategy s_0 for scenarios where solutions are found.

Strategy	Overall	Category 1	Category 2	Category 3
$s_{1\alpha}$	8%	8%	10.49%	2.06%
$s_{1\beta}$	1.65%	-2.24%	6.28%	-1.94%
s_2	-36.51%	-75.81%	-18.41%	-3.54%
s_3	1.81%	-2.29%	6.67%	-1.94%

strategy s_x. The columns show both $\bar{\mu}^{s_x}$ for all scenarios and for the scenarios in each category. For each scenario, two strategies are comparable if both find a solution.

Starting with the overall results, we find that strategy $s_{1\alpha}$ has an overall positive effect on $\bar{\mu}^{s_x}$ and is on average 8% better than s_0. We also observe that strategies $s_{1\beta}$ and s_3 are sligthly better than s_0. Finally, strategy s_2 behaves significantly worse (-36.51% worse) than s_0. We conclude that track release time seems to be a good criteria to prioritize when selecting train events to schedule.

Comparing how the strategies behave for each category, we find that the strategies perform the best for disturbances of category 2, i.e., when a single train has a permanent malfunction causing it to run on slower speed. Strategies $s_{1\alpha}$, $s_{1\beta}$, and s_3 all have better performance than s_0 for category 2 disturbances. For category 3 disturbances, we find that the different strategies have similar performance on average, just $\pm 4\%$ as compared to s_0. Finally, the results also show that strategy s_2 has much worse performance than the other strategies for disturbance categories 1 and 2, i.e., when a single train causes the initial disturbance.

5.3 Sensitivity Analysis

In the previous sections, we have evaluated the overall average performance of the different strategies. In order better understand the strategies' behavior and performance we need to do a more detailed analysis of their sensitivity to disturbance category, scenarios, and delay variation. The relative performance $RP_i^{s_x}$ of the different re-scheduling strategies for each scenario is shown in Figure 3, where the base strategy s_0 is the reference line (i.e., the value 1). The x-axis represents the different scenario sets, $S_1, \ldots, S_{20}$, each with five different delay variations.

For the category 1 scenarios, the benefit of the strategies $s_{1\alpha}$, $s_{1\beta}$, and s_3 over s_0 decreases as the delay variation increases. For example, $RP_i^{s_x} \approx 1.4$ for scenario 1 and then $RP_i^{s_x}$ decreases up to scenario 5. A similar behavior is observed also for scenario sets $S_2 \ldots S_6$. For strategy s_2 we observe that it generally has significantly lower performance the s_0. We also observe that the performance of s_2 becomes more similar to the performance of s_0 when the delay increases. The behavior is explained as follows: s_2 considers buffer times when prioritizing among events.

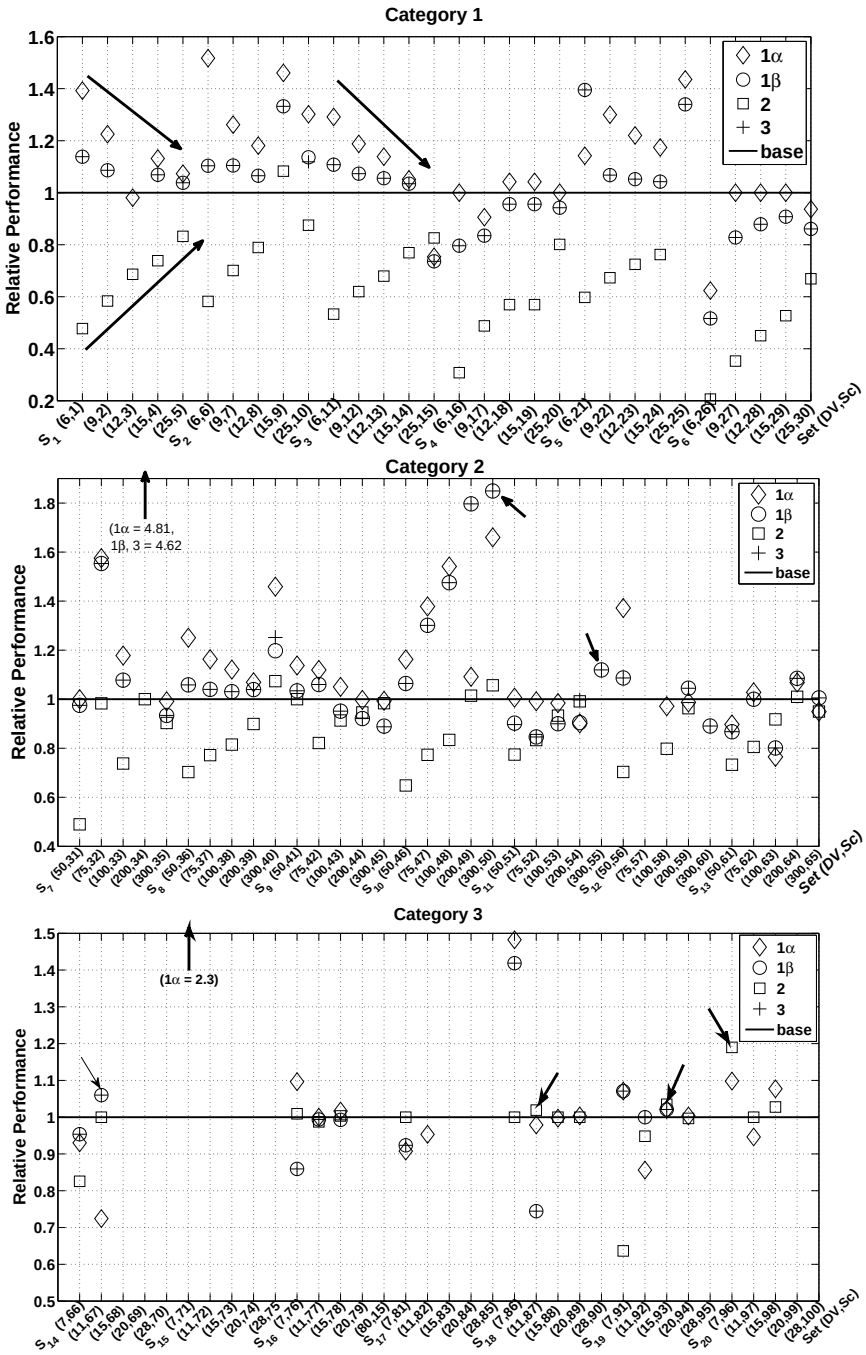


Figure 3: Relative performance, $RP_i^{s_x}$, of the re-scheduling strategies for scenarios of three categories.

When the initial delay increases so will the consecutive delays. As a result, the buffer times will be utilized fully, and strategy s_2 behaves more as s_0. It is interesting to note that the strategies are more sensitive in scenarios of set S_4 (where the delayed train has 30 events) as compared to S_3 (where the delayed train has 9 events). A train with a large number of events may interact with more trains, which causes more conflicts, and then the probability of finding an improved solution decreases. Based on the above observations, we conclude that the performance of strategies in scenarios of category 1 are sensitive to delay variations and the number of events a delayed train has.

For category 2 scenarios, we observe a similar behavior for scenario sets S_7, S_8, and S_9 as for the category 1 scenarios, i.e., as the delays increase the more similar performance have all the strategies. Further, we observe that $RP_i^{s_x} < 1$ for most of the strategies in set S_{11} (the delayed train has 35 events) and in set S_{13} (the delayed train has 30 events). On the other hand, $RP_i^{s_x} > 1$ is observed in most scenarios in set S_8 (the delayed train has 10 events) and S_9 (the delayed train has 20 events) as compared to S_{11}, and these four sets have initial delays on the same section. This indicates that trains with a large number of events interact with a larger number of other trains, thus increasing the complexity of the scenario. Our observations indicate that the performance of the strategies is sensitive to delay variations as well as to the complexity of the disturbance scenario.

For the scenarios in category 3, solutions are found to only few of the scenarios by any strategy. The reason is that all trains on the disturbed section are delayed with different delay variations which contributes to the scenario complexity. Thus, the algorithm either does not find a solution within the execution time limit or ends up in some deadlock situation that it cannot solve. The solutions found are generally for relatively short delay variations. This indicates that the performance of the strategies is more delay sensitive in the scenarios of category 3 as compared to category 1 and 2.

5.4 Summary and Discussion

Some of the found solutions in category 2 and 3 are interesting to analyze more, and they are indicated by arrows in Figure 3. Table 3 lists the solutions found by the different strategies for each selected scenario, where bold means an improvement over the base strategy s_0.

We observe in scenario 34 that strategy $s_{1\alpha}$ significantly reduces the total final delay for all trains as compared to s_0. $s_{1\alpha}$ prioritizes events with early track release times. This is important in order to reduce the consecutive delays for trains with a large number of events (i.e., they may interact with many other trains). Further, on a block based section, i.e., a line section with several consecutive blocks, the scheduling of a train with a large section runtime may increase the consecutive delay for succeeding trains with small section runtimes. Strategy s_3 successfully handles this situation in scenarios 49 and 50. The same solution is found by strategies $s_{1\beta}$ and s_3 in scenario 55 and 67. This is due to the fact that when the event deviation is larger than the buffer time then both strategies behave similarly.

Table 3: Selective experimental results for a 90 minutes time horizon and a 30 seconds execution time limit (bold means an improvement over s_0).

Scenario	s_0	s_1		s_2	s_3
		$s_{1\alpha}$	$s_{1\beta}$		
34	20209	**4150**	4376	20209	4376
49	4962	4546	**2762**	4891	**2762**
50	8252	4971	**4462**	7810	**4462**
55	16666	-	**14884**	-	**14884**
67	6627	9151	**6252**	6627	**6252**
71	2772	**1207**	-	-	-
87	4833	4938	6491	**4742**	6491
93	13212	12941	12941	**12765**	12941
96	1310	1193	-	**1101**	-

Strategy s_2 reduces the delay in a situation (i.e. in scenario 96) when a train with large buffer times is delayed in favor of trains with small buffer times when their earliest start time is the same. In scenario 71 and 87, the delay on consecutive line sections is reduced. A train must occupy the same track as it used on the previous line section. If the track release time is larger than the time on a second line section then it contributes to the delay. Branching on different tracks may be helpful in delay reduction, and this advantage has been observed in scenario 93.

6 Conclusions

In this study we have evaluated the performance of re-scheduling strategies for train dispatching during railway traffic disturbances. The initial source of delay is of three types: (1) a single train has a temporary delay, (2) a single train has a permanent reduced speed on all line sections, and (3) all trains are delay on a particular section. In total, 100 different disturbance scenarios are evaluated.

Our results show that strategies based on earliest start time and earliest track release time have the best average performance. The earliest track release time strategy has on average 8% better performance than the earliest start time strategy. The strategy based on shortest buffer time has the worst average performance, on average 36% worse than the earliest start time strategy. Our results also show that disturbance scenarios where all trains are affected, e.g., an infrastructure problem on a particular section, are difficult for the strategies to handle. In only 19 cases out of these 35 scenarios were feasible re-scheduling solutions found.

Our variation analysis shows that the different strategies are sensitive to the number of events for a delayed train, i.e., a train with many events may cause consecutive delays for many other trains, and the size of the initial delay, i.e.,

the longer initial delay the more similar is the performance of the strategies. The quality of solution also depends on scheduling of good candidates on consecutive line sections in scenarios of category 2 and 3 for consecutive delay reduction.

The analysis also shows that no strategy is superior to the others for all scenarios. Therefore, a combined approach can a be a promising alternative. In our future work, we will combine the proposed strategies in the design of parallel algorithms, where different workers can use different re-scheduling strategies when searching for good re-scheduling solutions.

References

[1] Törnquist, J. & Persson, J.A., N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B: Methodological*, **41(3)**, pp. 342–362, 2007.

[2] Törnquist Krasemann, J., Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. *Transportation Research Part C: Emerging Technologies*, **20(1)**, pp. 62–78, 2012.

[3] Iqbal, S.M.Z., Grahn, H. & Törnquist Krasemann, J., A parallel heuristic for fast train dispatching during railway traffic disturbance – early results. *ICORES-2012: 1st Int'l Conf. on Operations Research and Enterprise Systems*, pp. 405–414, 2012.

[4] Törnquist, J., Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. *5th Workshop on Algorithmic Methods and Models for Optimization of Railways*, 2005.

[5] Acuna-Agost, R., Michelon, P., Feillet, D. & Gueye, S., A mip-based local search method for the railway rescheduling problem. *Networks*, **57(1)**, pp. 69–86, 2011.

[6] Schachtebeck, M., *Delay Management in Public Transportation: Capacities, Robustness, and Integration*. Ph.D. thesis, Niedersächsische Staats-und Universitätsbibliothek Göttingen, Germany, 2009.

[7] D'Ariano, A., *Improving Real-Time Train Dispatching: Models, Algorithms and Applications*. Ph.D. thesis, Technische Universiteit Delft, The Netherlands, 2008.

[8] Conte, C., *Identifying dependencies among delays*. Ph.D. thesis, Niedersächsische Staats-und Universitätsbibliothek Göttingen, Germany, 2008.

[9] Corman, F., *Real-time Railway Traffic Management: Dispatching in complex, large and busy railway networks*. Ph.D. thesis, Technische Universiteit Delft, The Netherlands, 2010. 90-5584-133-1.

[10] Törnquist Krasemann, J., Dynamic railway traffic management during disturbances: Focus on the complexity imposed by deregulation. *ITSWC : Intelligent Transportation System World Congress*, 2008.

[11] Corman, F., D'Ariano, A., Hansen, I., Pacciarelli, D. & Pranzo, M., Dispatching trains during seriously disrupted traffic situations. *2011 IEEE Int'l Conf. on Networking, Sensing and Control (ICNSC)*, pp. 323 –328, 2011.

Bilaga 2

S. Soltani, *Improving the branch and bound greedy algorithm by introducing a new objective function: Implementation, results and analysis*, Project report, 2012-06-15.

Improving the branch and bound greedy algorithm by introducing a new objective function, implementation, results and analysis

Sara Soltani

2012-06-15

This work has been carried out as part of the work in the project titled "Efficient real-time re-scheduling of trains during disturbances (EOT)". This project is financed by Trafikverket (The Swedish Transport Administration).

1 Introduction

The branch and bound algorithm from the paper [1], designed to solve the re-scheduling problem of the railway traffic during disturbances is greedy, the most appropriate train event is chosen to execute first.

The cost at each node in the branch and bound search tree is the optimistic assessment of the total final delay and the total number of trains, based on the current partial solution. "A cost estimation of the solution so far is then computed by summing up the minimum expected delay for each train. That is, the delay so far experienced by each train minus any future buffer available" ([1]). In the search tree the cost of a leaf (the end node) is the total final delay for this feasible solution found.

It is of our interest to investigate how to improve the "pricing/cost assessment" of the intermediate nodes in the search tree which the greedy algorithm creates. As this branch and bound algorithm only uses a crude estimation of the objective function value and for many nodes it becomes the same so it is hard to tell which nodes that are good to backtrack to and explore further and which to dismiss. Moreover the cost function tends to change more frequently deep into the search tree. The reason could be that when a train is primarily delayed the effect on other trains increases during time. This fact has been stated in the article [2]: "Using the cost function (i.e., the total delay for all trains) we examine the performance of different solutions. The cost increment trend is same for all solutions up to a deep depth (approximately 500 in our case scenarios), after that they start to diverge. The higher divergence is found deeper in the tree. The implication of that it is difficult to do efficient pruning of non-promising branches early, i.e., high up in the search tree. Thus other cost functions should be considered in future work."

Examples on potential "quantitative indicators" could be how many trains that are delayed, how the average delay changes, what the actual delay is, etc.

The aim of the work presented in this report is to improve the cost assessment of the intermediate nodes in the search tree, enhance local search by looking ahead for each possible local choice.

2 Backgrounds: A multi objective branch and bound

Looking through various literature with the key word branch and bound and combinatorial optimization an interesting expression for a meta-heuristic greedy algorithm called pilot method came up. The term pilot method is generally used to define looking ahead for each possible choice building a feasible solution of a combinatorial optimization problem. It can be interpreted as creating an upper bound at each node using a sub-heuristic, visiting child nodes whose cost, fall within a cost

bound to be determined at each step - i.e., search more intelligently. The pilot method is a parallel procedure to the branch and bound, with the same procedure as the branch and bound rule except that at each node the sub-heuristic here bounds each subproblem with an upper bound ([3]). If we intend to use this nice idea in our branch and bound algorithm we will need a sub-heuristic at each node to give an upper bounds together with the existing lower bound that the classical branch and bound algorithm calculates.

Another interesting term appear frequently in the literature is the multi objective branch and bound (MOBB) (e.g., [4]). The multi objective branch and bound seems to be a suitable choice as our aim is to simultaneously optimizing two or more objectives with subject to certain constraints.

In the branch and bound process we intend to evaluate each (partial) solution x in the set $\mathcal{X}$ of feasible (partial) solutions by p objective functions $f(x) = (f_1(x), \ldots, f_p(x))$.

Definition 1. *An element u in the set $\mathbb{R}^p_+$ is said to be dominated for some $v \in \mathbb{R}^p_+$ if $\{\forall i \in 1, \ldots, p, v_i < u_i\}$, and non-dominated if no such v exists.*

Definition 2. *Within a set $\mathcal{Y} \in \mathbb{R}^p_+$, a Pareto Front is a curve of non dominated integer elements.*

In multi-objective integer linear programming one has to find the Pareto Front of $\mathcal{X}$, denoted by $\mathcal{X}^*$, i.e., one looks for each point in $\mathcal{Y}^*$, where $\mathcal{Y}^* = f(\mathcal{X}^*)$ (i.e., set of optimal Pareto Front). Therefore instead of a single incumbent as in the single objective branch and bound, the set **UB** of the best (non-dominated) solutions found so far is saved during the multi objective branch and bound.

In our single objective B&B there are 2 criterion to dismiss a node: fathom by in-feasibility and fathom by worst/equal objective. The main idea in multi-objective branch and bound is the separation of the reachable and improving solutions where the reachable solutions are the ones driven from the partial solution of the current node in the B&B tree, and the improving solutions are the solutions that are not dominated by **UB**. Precisely, a node can be discarded if a separation hyperplane in the objective space can be defined between the set of feasible solutions in the subtree and the set of points corresponding to the potential Pareto optimal solutions.

Definition 3. *The vector u^I such that $u^I_i = \min_{u \in \mathcal{Y}} u_i$, for $i = 1, \ldots, p$, is called the ideal point of $\mathcal{Y}$.*

Definition 4. *The vector u^N such that $u^N_i = \max_{u \in \mathcal{Y}} u_i$, for $i = 1, \ldots, p$, is called the nadir point of $\mathcal{Y}$.*

Definition 5. *The vector w^{SN} such that $w^{SN}_i = \max(u_i, v_i)$, $\forall u, v \in \mathcal{Y}$ and $i = 1, \ldots, p$, is called a semi-nadir point of $\mathcal{Y}$.*

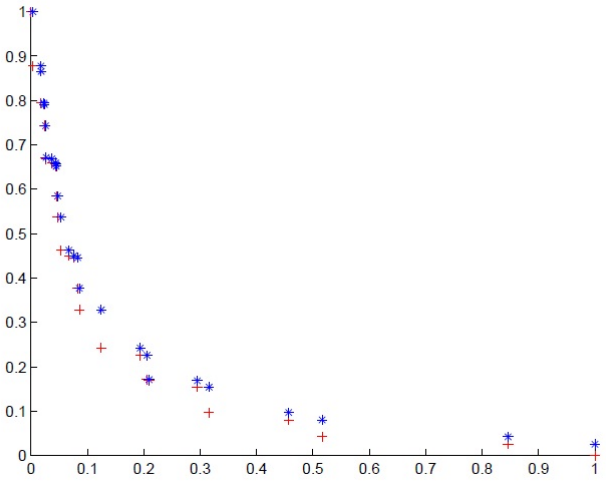
The multi objective branch and bound (MOBB method) is identical to the original version in the branching part but differs in the bounding part. The role of bounding is of great importance, as at each node of the search, some computations are unavoidable to prove that all solutions enumerated in the subtree of the current node are unable to improve the current **UB**.

Initialization: We need to start with some good feasible solutions i.e., initialization of the set **UB**. A feasible solutions can be computed with the greedy algorithm(e.g., see [1]), evaluated with the weighted sum of the objective functions. The first feasible solution and its neighbours given by our greedy algorithm can initialize the **UB**. Semi-nadir points of the set **UB** then need to be calculated.

Bounding and dismiss a node: Semi-nadir points creates an upper bound to the space of potential feasible solutions. A (partial) solution is compared to the current worst semi-nadir cost of the best feasible solutions, (can be computed by weighted sum of objectives) and only if this solution is worst than the current worst semi-nadir point then it can be fathomed (a separation hyperplane exists).

Updating: Pareto Front does not exists in the single branch and bound algorithm. So while backtracking in the search tree two main tasks should be done: (1) decide if a (partial) solution is worse than the worst semi-nadir points; (2) add one integer point to an existing Pareto Front, delete the dominating solutions in the Pareto Front, and update the semi-nadir point list. (i.e., A new feasible solution x is found at some node, if it is not dominated by any $u \in$ **UB** its cost $f(x)$

Figure 1: Pareto front and its semi-nadir point list, + points are integer points and $*$ points are semi-nadir points. Reference [5].



is included in **UB**. All the costs $u \in$ **UB** such that u is dominating $f(x)$ are removed from **UB** once $f(x)$ is inserted.)

We do not explicitly use the procedure described above but rather try to introduce a new cost function with the intention that this new objective function gives us a better indication of the partial solutions in the tree and by using the basic idea behind the multi objective branch and bound try to minimize both the original cost function introduced in [1] and this new cost function.

3 Minimizing future conflicts

In the trains timetable there is usage of two trains of the same section at the same time, but these overlap of time for two or more trains at the same section is feasible as this section has a couple or several tracks. A delay of a train may result into infeasible conflicts of infrastructural resource requests for trains. In the short report [6], an idea to minimize the conflicts of the resources in the non-periodic train timetabling problem for a long horizon time has been presented. The definition from [6] describes the concept of conflict nicely.

Definition 6. *If two trains are scheduled in a way such that both need the same resource(track or other entity of the network) at the same time, they are said to be in conflict. The number of seconds they are in conflict is called the number of conflict seconds.*

Let us assume we are at the beginning of the time when at least one train is delayed. We know:

1. How many trains are delayed and how much the delay is, in seconds.

2. We have a start time and end time of each train resource request; initial start time for trains that are not delayed and we have an approximation of minimum possible start time and end time of train events for those trains that are already delayed. So we can compute how many conflicts a train which is late according to its timetable will cause and how much (in seconds).

3. We have a primarily order of trains according to the scheduled timetable, we have build the candidate list according to these orders.

4. We have number of train events for each train and number of section events for each section.

5. We know the section capacities, so

$$\frac{\text{Total number of section events for section } j}{\text{capacity of } j}, \quad j \in B; \tag{1}$$

can estimate how much a section is dangerous in terms of spreading conflicts, we will not explicitly use this data.[1]

We decide the followings during finding the feasible solutions through the branch and bound strategy implemented in the greedy algorithm:

1. We will decide allocation of the tracks to trains in the process.
2. We will assign new start time and end time for each event of each train if necessary.
3. We may change the order of trains.

When two or more trains request capacity in a way that cannot be satisfied simultaneously, we would use the term of conflict. We wish to minimize the conflicts that delay cause. An idea for the cost function is to minimize the total amount of time during which the resource is over-allocated.

let $\beta_{k,\tilde{k}}$ be the binary variable and $= 1$ if a train event $k \in K_i$ for the train $i \in T$ have a request for the same section at the same time as the train event $\tilde{k} \in K_{\tilde{k}}$ for train $\tilde{i} \in T$, (overlap in time of usage) and $= 0$ otherwise.

Let $NrOfConT_i$ be the number of conflicts for train $i \in T$ and $NrOfConS_j$ be the number of conflicts for section $j \in B$, with the initial values 0. Let $w_{k,\tilde{k}}$ measures the total amount of time in which the delayed train i has a request to use the same resource as the train $\tilde{i}$. Please note that $\tilde{i}$ can be a delayed train as well. Let b_k^{init} and e_k^{init} be the initial planned start time of event k of train i on section j and let b_k^{min} be the minimum possible start time of event k. d_k specifies the minimum time of the operation time of event k and $secIndex_{i,k}$ gives the section index of event k in the list of train events K_i of train i.

Algorithm 1 computes the number of conflicts and amount of conflict in seconds for each request of the same resources at the same time of train $i \in T$ which is delayed and another train $\tilde{i}$ in the list of trains T.

Algorithm 1 can be repeated at each node in the branch and bound process which could give us an ahead look of future conflicts. The idea is at each node the following computations should be done. Let $\delta_i = 1$ if train $i \in T$ is delayed and $= 0$ otherwise.

How much in seconds all delayed trains have caused conflict:

$$\sum_{\substack{k \in K_{\tilde{i}} \\ \tilde{i} \in T \\ i \neq \tilde{i}}} \delta_i \beta_{k,\tilde{k}} w_{k,\tilde{k}}, \quad i \in T; \tag{2}$$

Estimation of future conflicts (can present the risk of spreading delay by one delayed train or seriousness of a train delay on a particular section):

$$\sum_{k \in K_i} \frac{NrOfConS_{secIndex_{i,k}}}{capacity_{secIndex_{i,k}}}, \tag{3}$$

for $i \in T$ where i is delayed and k is an active or a waiting event. The equation (3) equals $risk_i$ in the Algorithm 1.

How much the total expected conflict is, can be answered with the total actual conflicts and an estimation of the influence of those conflicts. So now we can introduce our new cost function:

- Minimize total amount of conflicts,

$$Minimize \sum_{\substack{k \in K_i \\ i \in T}} \sum_{\substack{k \in K_{\tilde{i}} \\ \tilde{i} \in T \\ i \neq \tilde{i}}} \delta_i \beta_{k,\tilde{k}} w_{k,\tilde{k}}, \tag{4}$$

[1] The standard UIC 406 methods can be used for calculating the capacity consumption on the railway lines for the time horizon of our problem.

Algorithm 1 (Computing amount and number of train conflicts and risk of conflicts at each node)

Step 1: For $i \in T$ and $i = 1, \ldots, NrT$, where T is the set of trains and NrT is the total number of trains, set $NrOfConT_i = 0$, $NrActiveDelayedT = 0$ and $risk_i = 0$;
Step 2: for $i = 1, \ldots, NrT$, and $\forall\ k \in K_i$, $\forall\ \tilde{i} \in T$ and $\forall\ k \in K_{\tilde{i}}$, such that $i \neq \tilde{i}$: Let $w_{k,\tilde{k}} = 0$;
If $i \in T$ is delayed and has an active event,
Let $\delta_i = 1$ and $NrActiveDelayedT = NrActiveDelayedT + 1$;
Do:

1. For all sections $j \in B$ and $j = 1, \ldots, NrS$, where B is the set of sections and NrS is the total number of sections, let $NrofConS_j = 0$.

2. If event $k \in K_i$ is active then for $t = k, \ldots, NrofTE_i$, where $NrofTE_i$ is the total number of events for train i, Do:
   for $j = secIndex_{i,t}$, and $\tilde{k} \in L'_j$, and $\tilde{k} \notin K_{\{1,\ldots,i\}}$, where L'_j is the list of the remaining waiting events of section j,

   (a) If $b_t^{min} \leq max(b_{\tilde{k}}^{init}, b_{\tilde{k}}^{min})$ and $b_t^{min} + d_t \geq max(b_{\tilde{k}}^{init}, b_{\tilde{k}}^{min})$,
   let $w_{t,\tilde{k}} = b_t^{min} + d_t - max(b_{\tilde{k}}^{init}, b_{\tilde{k}}^{min})$;

   (b) If $b_t^{min} \geq max(b_{\tilde{k}}^{init}, b_{\tilde{k}}^{min})$ and $b_t^{min} \leq max(e_{\tilde{k}}^{init}, b_{\tilde{k}}^{min} + d_{\tilde{k}})$,
   let: $w_{t,\tilde{k}} = max(e_{\tilde{k}}^{init}, b_{\tilde{k}}^{min} + d_{\tilde{k}}) - b_t^{min}$;

   If $w_{t,\tilde{k}} \neq 0$; then:
   Let $\beta_{t,\tilde{k}} = 1$;
   Let $NrOfConT_i = NrOfConT_i + 1$;
   Let $NrOfConS_{secIndex_{i,t}} = NrOfConS_{secIndex_{i,t}} + 1$;
   When t has been compared with all the waiting events $\tilde{k}$ on j then:
   $risk_i = risk_i + \frac{NrOfConS_{secIndex_{i,t}}}{capacity_{secIndex_{i,t}}}$;
   let $t = t + 1$;

3. For $i \in T$ delayed: print $risk_i$ -The risk that delayed train i cause in terms of causing secondary delays to other trains or resulting in more deviation of its original timetable.
   Print $NrOfConT_i$ -Total number of conflicts that the delayed train i has.
   Print $NrActiveDelayedT$ -Number of delayed trains that are active.

Step 3: Compute the total amount of conflict:

$$\sum_{\substack{k \in K_i \\ i \in T}} \sum_{\substack{k \in K_{\tilde{i}} \\ \tilde{i} \in T \\ i \neq \tilde{i}}} \delta_i \beta_{k,\tilde{k}} w_{k,\tilde{k}},$$

Compute the total risk:

$$\sum_{i \in T} \delta_i risk_i,$$

- Minimize total expected conflicts or the estimated future conflicts,

$$Minimize \sum_{\substack{k\in K_i \\ i\in T}} \delta_i \frac{NrOfConS_{secIndex_{i,k}}}{capacity_{secIndex_{i,k}}} = Minimize \sum_{i\in T} \delta_i risk_i, \tag{5}$$

- We will combine the two cost functions (4) and (5) into a new objective function as:

$$Minimize \sum_{\substack{k\in K_i \\ i\in T}} \delta_i risk_i \sum_{\substack{k\in K_{\tilde{i}} \\ \tilde{i}\in T \\ i\neq\tilde{i}}} \beta_{k,\tilde{k}} w_{k,\tilde{k}}, \tag{6}$$

  The equation (6) (we can call it, minimizing the risk of future conflicts) minimizes the amount of conflict with the weight of risk of spreading more delays into the timetable. This new objective function makes sense as the amount of future conflicts plays a role in increasing the total delay when the ratio of the number of conflicts on a section to the capacity of that section is higher.

One can: Minimize(weighted total amount of conflicts + total delay) as we should also check and minimize the total delay of trains as well. We would like the new timetable to be optimal in sense of deviation from the original timetable as little as possible. Note that with less conflict we will have more robustness in the timetable. It is of importance how to define the weights in this case so that both would be in the same scale, to achieve this first we propose the scale of $\frac{1}{10}$.

Minimum total expected delay is monotonic and non decreasing. Total amount of future conflicts may decrease or increase, if a new train get a positive deviation from the original time table when executing a new event at one node in the search tree; But, in total the conflict values will decrease in which when a complete feasible solution is in hand we only have the total delay and the conflict values will be zero. This is because, the number of conflicts or the amount of conflicts in seconds would be zero when all events of a delayed train are executed. Since there is a difference in the property of these two functions which is one decreases and another increases, it dose not seems a good idea to simply add these functions and define a new cost function.

Other suggestions for the objective functions are:

- Minimizing the changes of average delay: Minimum expected delay for each train divided by the total number of trains delayed.

- Minimizing the changes of average conflict for each train: Sum of Conflicts for each active delayed train divided by the number of active delayed trains at each node.

In addition to all the statements above we have to mention that using the multi objective theory seems more accurate, as well as more suitable, so that one compute several cost function values and do not lose the essence of minimizing total delay and as much less divergence as possible to the original time table with respect to the trains arrival a their final destinations.

This is how we suggest to proceed: Let us take the two objectives, scaled estimated future conflicts weighted by risks (6) and the minimum expected delay computed at each node by the greedy algorithm. Assume that at each node we have a pair of these two values $x = (x_1, x_2)$, we compute the Euclidean norm of this pair as follows.

$$\|x\| = \sqrt{x_1^2 + x_2^2} \tag{7}$$

This euclidean norm can substitute the previous node value (total expected delay) and help us in the backtracking phase of our branch and bound process. Note that the maximum norm (infinite norm) of x is computed by $\|x\|_\infty = Max(|x_1|, |x_2|)$.

At the same depth when comparing two siblings nodes x and y in which $\|x\|_\infty \geq \|y\|_\infty$, node x can be cut off if and only if for $x = (x_1, x_2)$ and $y = (y_1, y_2)$, $x_1 \geq y_1$ and $x_2 \geq y_2$. One node with larger conflicts value comparing to another node at the same tree depth and with the same minimum expected delay can be prioritize to branch first in the tree because as larger conflicts

value in the future of train events, more causing secondary delays and we may end up with a lower total final delay.

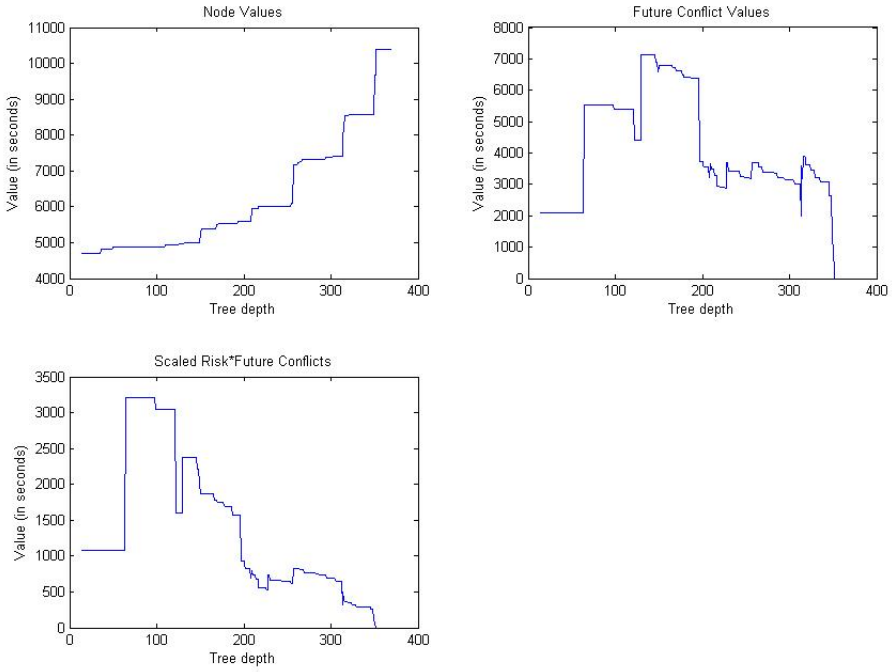
Suggestion to improve choice of executing event at one critical point: In the article [1] the conflict of interest of two trains in occupying one resource has been handled by computing their potential delay on that section when one train is prioritized over another. The estimation of future conflicts at each step could be really helpful when we want to prioritize one train over another and choose to delay one train more or delay a train which has been on time up to the time of this decision in favor of an already delayed train. Because we not only look at the conflict of events at the current section but also the two trains future events on other sections.

4 Experimental evaluation

4.1 Step1: Test the new values

We have computationally test all the values at each node to obtain the first feasible solution in the depth first search branch and bound algorithm. The reason is that for now we wish to observe how the node values changes when going down into the tree. The example is chosen from the delay scenario 20 with a one hour time horizon.

Figure 2: Graphs of values: minimum expected delay and formulations (4) and (6), computed at each node



Figures 2 and 3 shows the values computed at each node with the formulations (4) and (6), these values oscillates more at higher nodes in the tree as more trains became active and the number of delayed trains increases.

The graphs for average conflict and delay values at each node are illustrated in Figure 4.

The Figure 5 shows that summing the minimum expected delay and the scaled future conflicts weighted by risk values could cause unstable oscillations. The Maximum norm of the two cost functions values at each node of the tree can be seen in the plot in Figure 6.

The Euclidean norm computed for the two cost functions minimum expected delay and scaled future conflicts weighted by risk values is illustrated in the Figure 7.

Figure 3: 3D Graphs of values in formulations (4) and (6), computed at each node

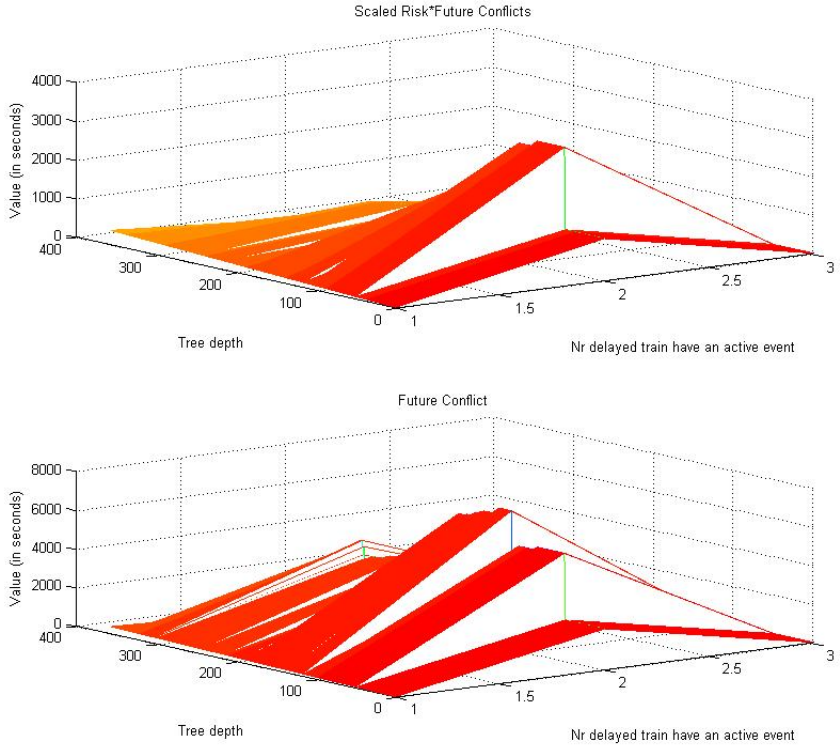


Figure 4: Graphs of average delay and average future conflicts computed at each node

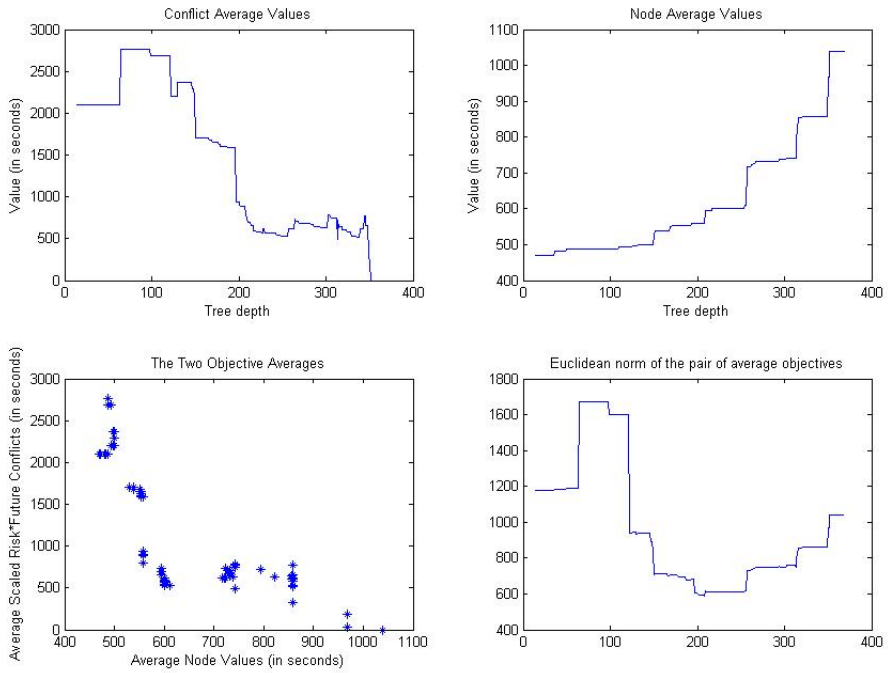


Figure 5: Graph of the sum of two values: minimum expected delay and risk of conflicts formula (6), computed at each node

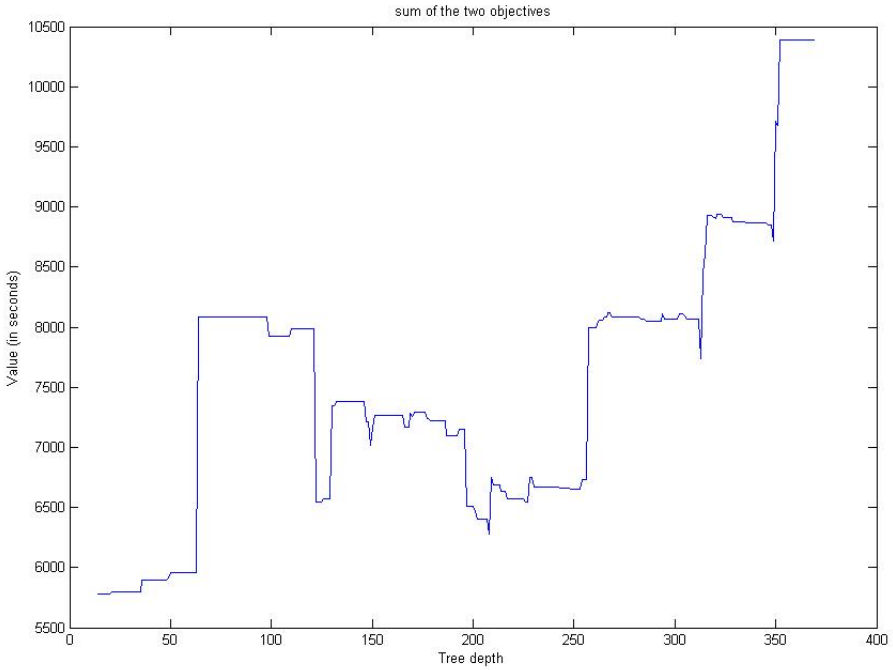
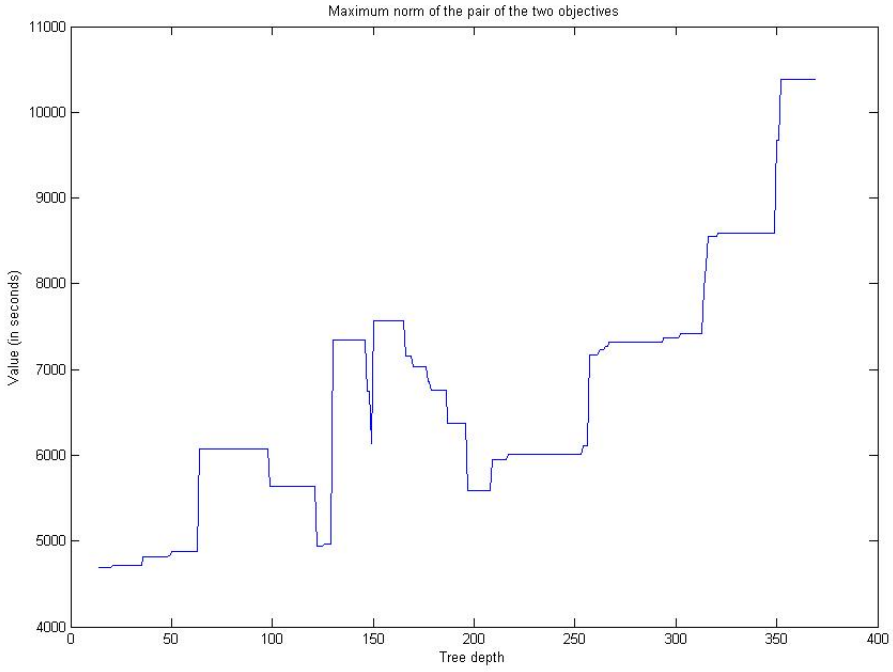


Figure 6: Graph of the maximum norm of two values: minimum expected delay and formulation (6), computed at each node



4.2 Step2: Solutions to different scenarios

In section 3 we have developed a new cost function for the branch and bound algorithm that solves the re-scheduling problem of the railway traffic during disturbances, it has been defined as follows.

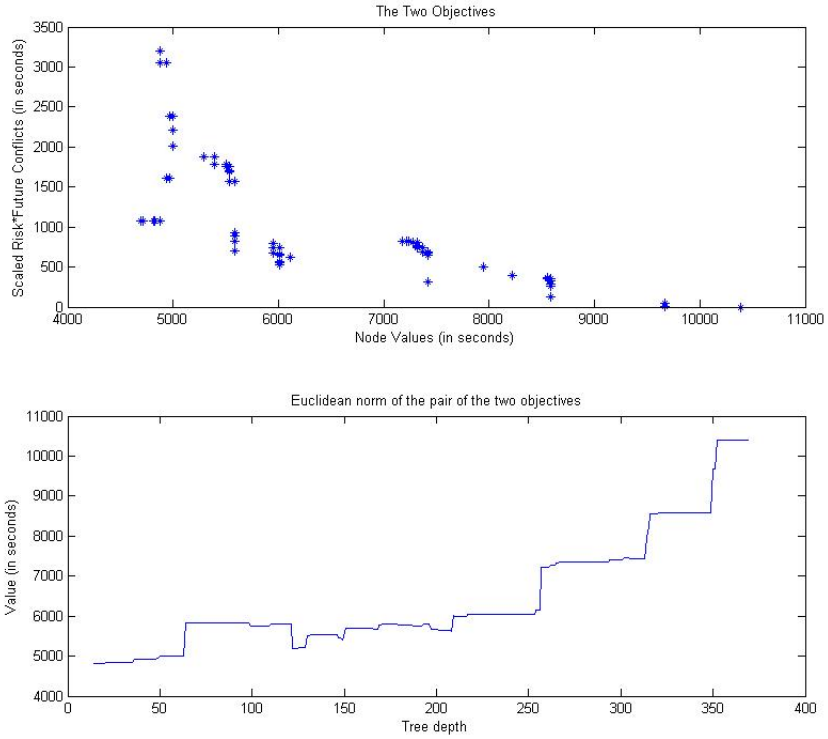
In the original algorithm node values are computed by the sum of minimum expected delay for each delayed train (for details of the computation of minimum expected delay please see Appendix A in [1]). The new suggested cost estimation at each node (the partial solutions) has been defined as the Euclidean norm of the pair of the original node values and weighted risks multiplied by the future conflict values of each delayed train. I.e. we consider two different values, one is the minimum expected delay for each train and other is the seriousness of the conflicts which delayed trains cause; then to minimized both, we consider a two dimensional space which each coordinate represent one of these values, so we compute the distance of this pair to the origin, i.e. the Euclidean norm. In the search tree at each node both values are computed and then the Euclidean norm of this values is assigned to the nodes as the new estimation for the node values. i,e,:

$$Minimize \ \|(\text{Minimum expected delay}, K \sum_{\substack{k \in K_i \\ i \in T}} \delta_i risk_i \sum_{\substack{k \in K_{\tilde{i}} \\ \tilde{i} \in T \\ i \neq \tilde{i}}} \beta_{k,\tilde{k}} w_{k,\tilde{k}})\|_2. \tag{8}$$

Where $\beta_{k,\tilde{k}}$ is the binary variable so that $= 1$ if a train event $k \in K_i$ for the train $i \in T$ have a request for the same section at the same time as the train event $\tilde{k} \in K_{\tilde{k}}$ for train $\tilde{i} \in T$, (overlap in time of usage) and $= 0$ otherwise and $w_{k,\tilde{k}}$ is the total amount of time of conflict in seconds. The constant value K is an arbitrary weight given to the value of risk of conflicts. Also $\delta_i = 1$ if train $i \in T$ is delayed and $= 0$ otherwise. The value $risk_i$ can be computed from the equation 3.

Here we would like to investigate how these new cost values effect on the branch and bound in the backtracking phase and especially on the improvement of the solutions; therefore, in the branch and bound algorithm the original node values (minimum expected delay) is replaced by the

Figure 7: Graph of the Euclidean norm of two values: minimum expected delay and formulation (6), computed at each node



new node values given by the equation 8.

We simply replace the old node values (minimum expected delay) with the new values (8), so the new values are assigned to each node at each step of the search process and perform the backtracking with this new values. Different scenarios refer to 3 different categories of train delays. Category 1 refer to a situation when a train suffers from a temporary delay at one section; trains which suffer from a permanent delay on all their line sections fall into category 2 and the cases when there is an infrastructure failure on one section and cause all the trains passing that section have a positive deviation from their original timetable are called category 3. Situations as the category 3 is less frequent but make serious impact on the scheduled timetable. Scenarios 5, 15, 17, 19 and 20 has been chosen from different categories to represent different examples of when a train is delayed. Scenario 5 is chosen from category 1 and scenario 15 from category 2, scenarios 17, 19 and 20 are chosen from category 3.

Table 1: Comparison between the original and new cost values (8) in the branch and bound process computation time 30 seconds and different weights $K = 0.1, 0.001$

Scenario	Time	Max depth	1st Feasible Sol	Original Final delay	New Final delay $K = 0.1$	New Final delay $K = 0.001$
				Best Sol & Nr d-Trains	Best Sol & Nr d-Trains	Best Sol & Nr d-Trains
05	90 m	575	1174	686-3	1174-1	1174-1
15	90 m	570	1680	1611-1	1680-2	1680-2
17	90 m	570	7338	7265-10	7338-10	7306-10
19	90 m	567	28883	28740-14	28740-14	28740-14
20	90 m	560	21922	21247-18	20752-20	20268-20*
20	60 m	369	10386	9281-10	9281-10	9281-10

Delayed trains in the Table 1 are referred to those trains that have arrived at their final destinations with more than 1 minute deviation from their planed arrival time at their final destination.

Table 1 shows the comparison between solution to the branch and bound algorithm with new values and original ones for scenarios 5, 15, 17, 19 and 20 in a 90 minute planning time horizon and scenario 20 with 1 hour time horizon with different weights in the cost function. Obviously with $K = 0.1$ using the new values for the backtracking phase fails to improve the first feasible solution in scenario 5, 15 and 17. For scenario 19 and 20 with 90 minute time horizon and scenario 20 with 1 hour time horizon the best solution found is the same as the original version. Only using the new values in the backtracking phase seems to be beneficial for the example of scenario 20 with 90 minutes time horizon, the solution is improved and the optimal gap is smaller. A complete computations for all scenarios can be seen in the Appendix.

Note that scenario 20 with 90 minutes time horizon is a hard example of our NP complete combinatorial problem. The computation times are set to 30 seconds for the solutions reported in Table 1.

We can say that computation of the conflicts and risks give a good overview for the delayed trains in the short time window of future but so many trains start or finish their journeys and the number of delayed trains and the sum of amount of conflict changes that may be make these values not consistently reliable in the search. It can be suggested that in our depth first strategy search tree with the new values the backtracking can be performed in different divisions of the tree where the number of active delayed trains is constant and then the best solution for that specific divisions of the tree is found then we choose that branch to continue.

4.2.1 Weights in the cost function

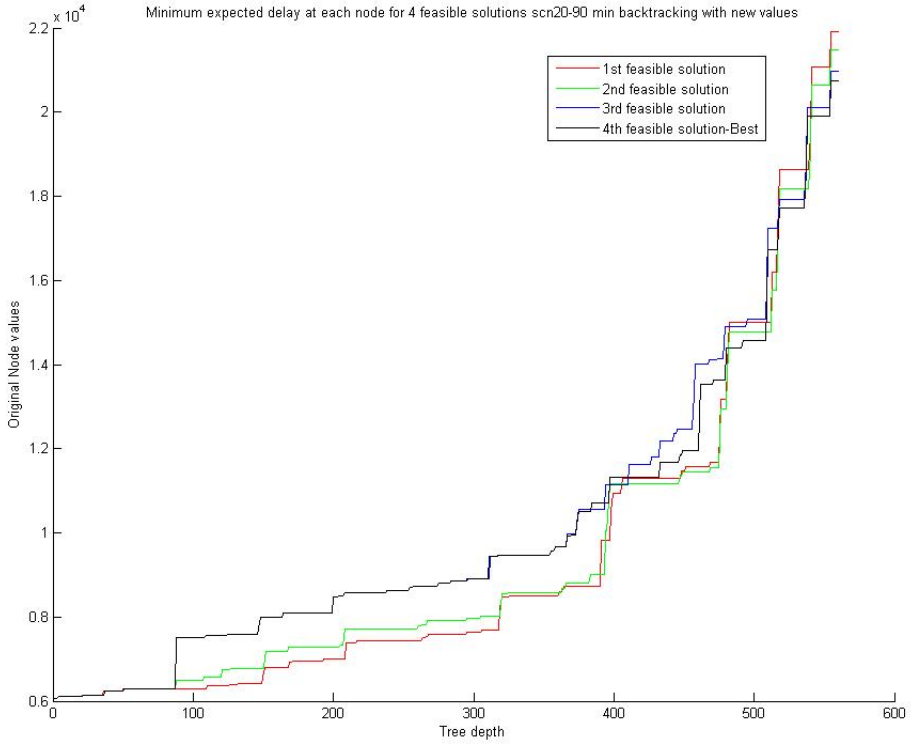
In column 6 Table 1, when considering the future conflict values for trains and summing them up, we have scaled this values by $\frac{1}{10}$. This choice seems natural as conflict values are in seconds, the same as the sum of the minimum expected delay, although they may be very larger. The conflicts are multiplied by the risks which are in our time horizons, 1 digit numbers or a small 2 digit number. But it is of our interest to investigate the affects of weight of conflict values in the new cost function. The answer to the question "should the higher weights be given to the

minimum expected delay or the conflicts" is that conflict values have a decreasing essence and their rate of change is high when drawing the graph of conflict values versus depth of the search tree; minimum expected delay is on the other hand, monotonous and non-decreasing. Therefore minimum expected delay is weighted higher as in 8 it has been weighted 1 as the conflicts of risks has been weighted a small value and less than 1. Let us weight the risks multiplied by the future conflict for delayed trains very small, say 0.001. This scale is chosen as the the risks multiplied by the future conflict for delayed trains could be as big as 4 digit number which at the end drops to zero. This weight can be interpreted as a meaningful perturbation to the original node values.

The solution in column 7 Table 1 shows an even better solution for scenario 20 with 90 minute time horizon, the smaller weight also resulted in finding a better feasible solution for scenario 17 although other solutions in column 7 in Table 1 is as same as column 6.

Figures 8-10 shows the feasible solutions found for scenario 20 in a 90 minute time horizon with different weights. For all the feasible solutions which has been found by the algorithm, the node values at each node is plotted in the same graphs. Figure 10 with the 0.001 as the risks and conflicts weight, comparing to Figure 9 shows an smoother values as the new node values (norm values).

Figure 8: Feasible solutions of Scenario 20 with weight 0.1, minimum expected final delay versus tree depth, backtracking with the new cost function



In total a smaller weight in the objective function 8 has resulted into better improvements in the final solution, comparing columns 6 and 7 in the Table 1. Also, as seen in Figures 9 and 10, a smaller weight (K in equation 8) gives a smoother behavior in sense of changes in the node values as the depth of the tree increases.

We have chosen scenario 15 and scenario 20 to investigate more since using the new cost function in the backtracking phase has failed to improve the first feasible solution in scenario 15 but in scenario 20 an improved solution has been obtained. In other words we want to see what happens in the backtracking phase for these two scenarios.

Figure 9: Feasible solutions of Scenario 20 with weight 0.1, New node values (Eq. 8) versus tree depth, backtracking with the new cost function

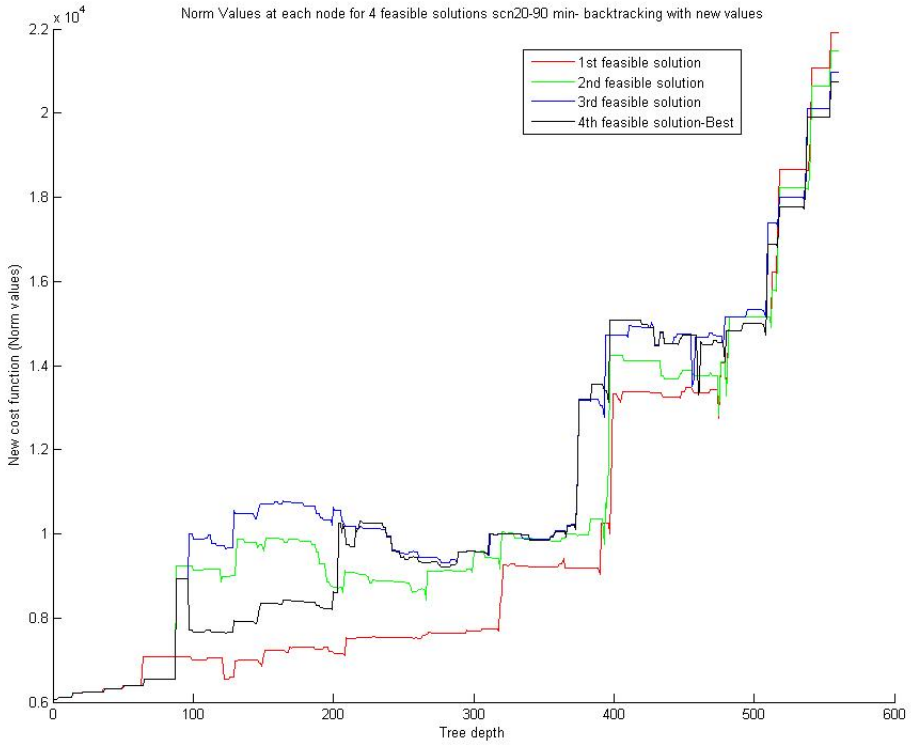
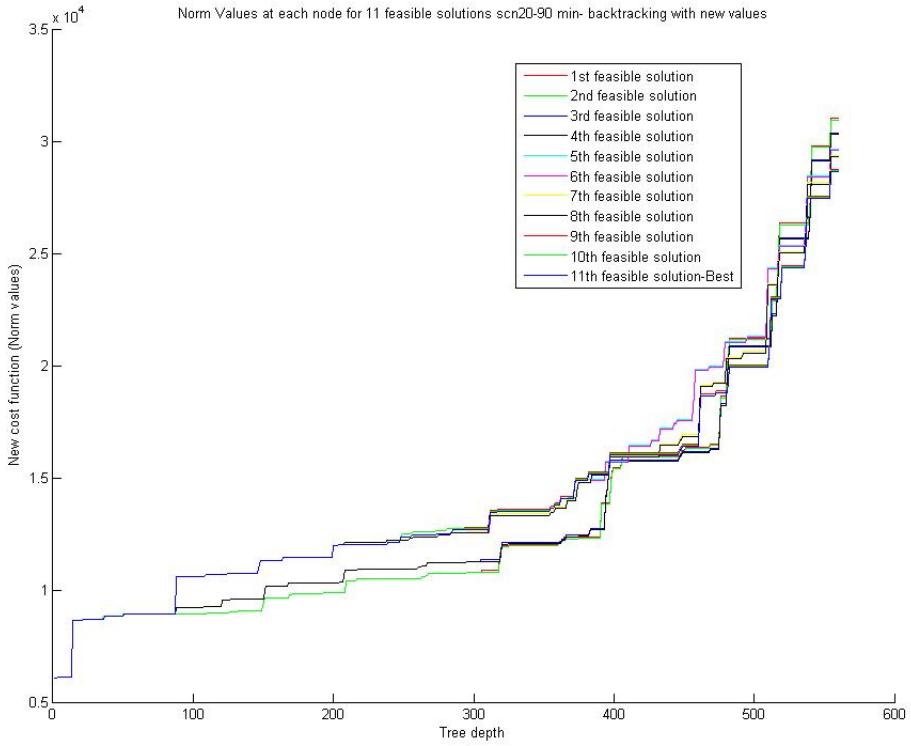


Figure 10: Feasible solutions of Scenario 20 with weight 0.001, New node values (Eq. 8) versus tree depth, backtracking with the new cost function



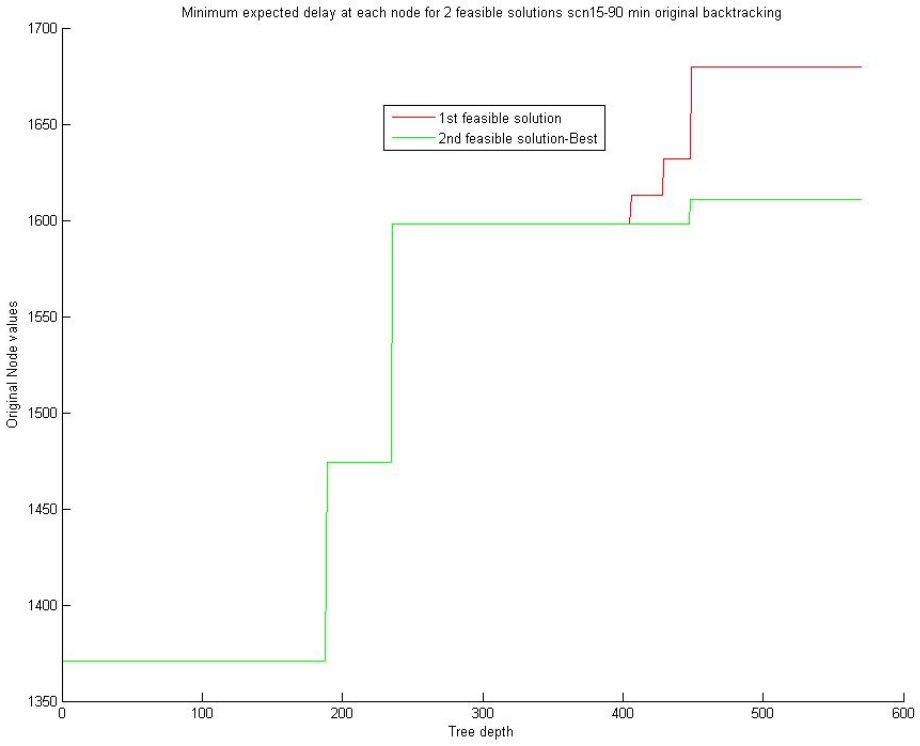
4.3 Scenario 15

In this section we would like to investigate and find an answer to the question "Why using the new cost values at each node in the backtracking phase fails to improve the first feasible solution in scenario 5 or 15 where only 1 or 2 trains are delayed". Figures 11 and 12 shows the minimum expected delay and the new values (norm of minimum expected delay and conflict values) for the 2 different feasible solutions obtained by backtracking to the depths with lower original value (i.e. minimum expected delay).

It can be seen from Figures 11 that with only one train originally delayed the minimum expected delay does not change dramatically, while the depth of the tree increases. The first feasible solution correspond to a large deviation to the train schedule which suffers from a permanent delay and a secondary delay to another train. Backtracking results in a solution with only one train delay, that is the originally delayed train. The conflict values are computed at first for this train only which is decreasing with a considerable slop, the secondary delay happens deep in the tree and the amount of future conflicts is small for this train. Therefore not surprisingly the conflict values are dominating the values of minimum expected delays as they are large, Figure 12 illustrate this very well. Changing the weights has seem to have no influence on this fact[1]. This is the main reason of the failure of the backtracking for scenario 15, e.g. compare Figures 11 and 12.

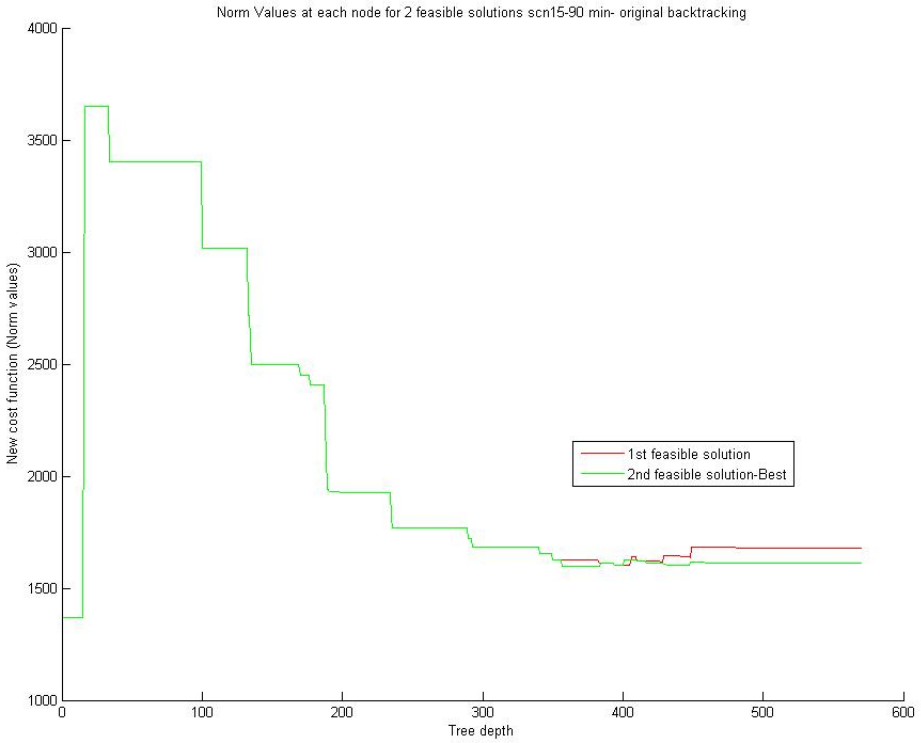
Let's remind that in a branch and bound algorithm for a minimization problem with the depth-first search strategy, the search is optimal when the node costs are a non-decreasing function of the depth of the node, e.g. scenario 20, with new values and scenario 15 with original values. One could at this point suggest that hybridizing the search with a best first search strategy instead of using the pure depth first search strategy would be beneficial.

Figure 11: Feasible solutions of Scenario 15 with weight 0.1, Estimated final delay versus tree depth, backtracking with the original cost function



[1]The author also failed to get an improvement with backtracking using the minus of the node values or trying to bound the starting point of the backtracking to an specific range of tree depth.

Figure 12: Feasible solutions of Scenario 15 with weight 0.1, New node values versus tree depth, backtracking with the original cost function



4.4 Scenario 20

In the section 4.2, a better solution in term of the optimality gap compared to the original version of the branch and bound algorithm has been found for the scenario 20 with 90 minute time horizon. Therefore, it is of our interest to follow the backtracking procedure in both search trees with node values of minimum expected delay and the new node values where the conflict values and risks has been considered together with the minimum expected delay (from equation 8).

For scenario 20 in the original search tree after the first feasible solution has been found with backtracking to the node with value lower than the value for the best complete solution, 3 other feasible solutions with improved final solutions has been found and in the search tree with new node values after total 30 seconds computation time 11 feasible solutions has been found, ie., the backtracking phase has been carried out using the values computed by the equation 8. The details of the backtracking phase when an improved solution has been found, for the two search trees, can be seen in Tables 2 and 3.

Table 2: Scenario 20, improving of solutions in the backtracking phase in the original algorithm, total number of feasible solutions found is 4.

Nr	Final delay	Branch Index	Minimum expected delay
1	21922	N/A	N/A
2	21880	304	7639
3	21289	85	6299
4	21247	303	7790

Table 3: Scenario 20, improving of solutions in the backtracking phase in the branch and bound algorithm with the new proposed values from equation 8, total number of feasible solutions found is 11.

Nr	Final delay	Branch Index	Node Values from Eq. 8
1	21922	N/A	N/A
2	21880	304	7639.0057
3	21479	13	6114.0096
4	21437	303	7980.1743
5	20961	86	6299.0245
6	20919	294	8864.0720
7	20752	282	8805.0534
8	20710	293	8864.0136
9	20332	206	8512.1995
10	20290	292	9008.0768
11	20268	247	8734.0890

The 4 feasible solutions found with the original algorithm has been plotted in Figure 13. Figures 13 and 10 can be compared together which show the two node values versus tree depth for all the feasible solutions obtained by the original algorithm and the algorithm with new node values.

A fact which can be seen from Figure 10 is that we can not conclude that from two sibling nodes at a certain depth the one which has less cost than the other one, can lead to a better solution at the leaf of the tree. E.g., the first feasible solution for this example of scenario 20 has the value 6299.0008 at depth 88 with the final solution of 21922 total delay in seconds, and the best solution has the value 7508.0016 at the depth of 88 but the total final delay of the best solution is 20268 seconds.

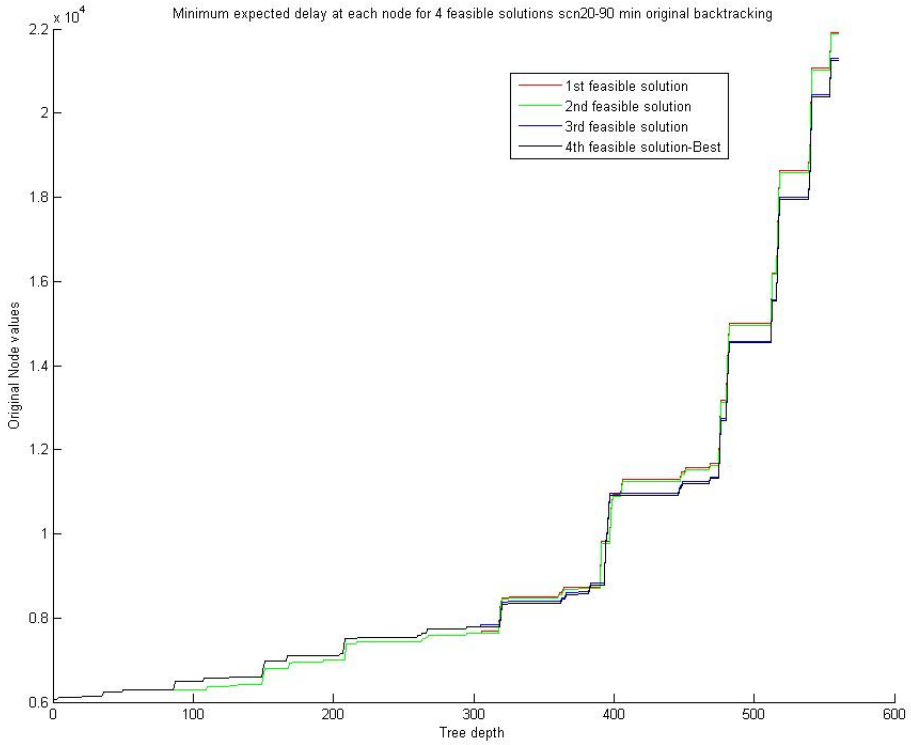
4.5 Step3: Possible conflict of interest

The analysis of conflict of interest for two trains competing over one infrastructure resource is already taken care by the model. If a candidate event is scheduled at a section with one available track, then it is checked if there are any other event also scheduled at that section, who has an earliest possible start time which is smaller that the predicted termination time of our candidate event. When this conflict of interest is detected, the potential consecutive delay is computed for both trains at that section. If prioritizing of executing the first event on this section result in a greater delay time for the other train, i.e., the second train suffer more-if executed later on this section- than the first train gain when executed first, then it is logical to give priority to the second train to occupy this section although the first train had a smaller start time on this section. For details on how to compute the consequential additional delay in the case of conflict of interest please see [1].

In the original branch and bound algorithm the consecutive delay is only computed for the first candidate event and the first next event which a conflict of interest is detected with it; and does not consider how many waiting events do those trains have, and on which sections. The idea is to compute the future amount of time which these two trains have conflict with other trains and on which sections, also dose these sections have more than one available tracks? I.e., we want to compute the conflict and risks for their waiting events on the future planned sections. This conflicts are computed ignoring the fact if the train is already delayed or it is on time according to the initial timetable.

Previously the priority was given in case of interest of conflict between two trains to a train which would have less total delay if executed first on the next section. We keep this logic when the time of future conflicts of this train multiply the risk of the sections it has waiting events on, is less than the other train. Now also in a case when a train would have a greater predicted delay on that section if it has to wait for another train to pass the section but has less future conflicts and risk on it's remaining sections then we give priority to it.

Figure 13: Feasible solutions of Scenario 20, node values versus tree depth, backtracking with the Original node values(minimum expected delay)



The results of such experiment is given in Table 4. The computations are done with the new cost function and backtracking according to these values with weights 0.001 and the computation time is 30 seconds (See section 4.2.1).

Table 4: Solutions with new priority when conflict of interest for two train occur, computation time 1 min and weights 0.001.

Scenario	Time	Lower bound	Solution			
			1st Feasible Sol	Nr of Feasible Sol	Best Sol	Nr d-Trains
5	90	201	1321	2	1174	1
15	90	1371	1827	2	1680	2
17	90	6534	7466	2	7338	10
19	90	9270	28817	2	28770	14
20	90	6114	21317	4	20978	17
20	60	4694	10386	6	8516	10

The results in Table 4 shows a different first feasible solution than the solutions of the original version of branch and bound. In scenarios 5, 15, 17 we have a worse first feasible solution and in scenario 19 and 20 a better first feasible solution with one and half hour of planning time horizon. In scenario 20 with 1 hour time horizon the first feasible solution is the same. Scenario 5 and 15 give the same solution as in Table 1 but a worse solution for scenarios 17 and 19 has been found in 30 seconds computation time. Although for scenario 20 with 90 minute time horizon a worse solution in term of total final delay is found (comparing 20978 s and 20268 s) but the total number of delayed trains is reduced from 20 to 17. Solving the scenario 20 with 1 hour time horizon give us a better feasible solution with 8516 seconds delay in total for 10 trains comparing to 9281 seconds for 10 trains.

The results of this experiment for all the scenarios has been reported in the Table 7. No improvements in the sense of the optimality gap in the final solutions has been seen for any scenario for a 90 minute planning time horizon. It can be suggested that also a test for smaller time horizons for different scenarios and another test in the original branch and bound algorithm with the strategy as described in this section in prioritizing the trains differently when a conflict of interest occurs can be done.

4.6 Step4: A different cost function – Is it an asset to use average values

In solving the rescheduling problem, certainly one objective is to have fewer number of trains delayed as well as minimizing the total final delay of all train, therefore it has been of interest how to use the information of the total number of trains delayed in the backtracking process. It has been suggested that minimizing the changes of average delay could be used as an indicator for the backtracking phase, i.e., minimum expected delay for each train divided by the total number of trains delayed. In this section then we have investigated the solution behavior using the average delays. Table 5 shows the results for the examples from scenarios 5, 15, 17, 19 and 20 with a 90 minute time horizon.

Table 5: Solutions with Minimizing the changes of average delay in the branch and bound algorithm with computation time 30 sec, Time horizon for all scenarios is 90 minutes.

Scenario	Nr of Feasible Sol	Best Sol. Average	Best Sol. Final delay	Best Sol. Nr d-Trains
5	8	281.5	1689	6
15	1	840	1680	2
17	4	630.92	7571	12
19	12	1711.29	29092	17
20	7	1037.6	20752	20

It can be easily seen from Table 5 that using the average values as the node values in the backtracking phase and searching for an improved solutions results in finding worse solutions in terms of total final delay, as the solutions having a larger number of delayed trains gives a smaller average value and would be accepted as an improved solution. The author suggests that total number of trains delayed or the average delay can be added as new dimensions to the bi-objective function which we already had defined, and using Euclidean norm of all these values, although the computational tests run on these examples lead into the same solutions as in Table 1.

5 Discussions and conclusions

In this report a new cost function for the greedy branch and bound algorithm from [1] has been developed (equation 8) and some tests have been done to investigate the behavior of the algorithm with the new cost function. The results of the solutions to different scenarios has been presented in Table 6 in the Appendix. An improved solution has been only observed for the Scenario 20 in the 90 minute planning time horizon.

The weights in the cost function 8 as seen in section 4.2 plays a role in the behavior of the algorithm; therefore, parameter tuning and sensitivity analysis should be considered in the future work.

For improving the backtracking phase one can divide the search tree into parts where the number of active delayed trains is constant and perform the backtracking in different parts of the search tree and find the best branch for that divisions of the tree and then continue that branch to obtain the final solution.

As mentioned in sections 4.4 and 3 we can not conclude that from two sibling nodes at a certain depth the one which has less cost than the other one, can lead to a better final solution. In the branch and bound algorithm in hand, only a depth first search is carried out prioritizing branching from a node with a value less than the best solution found so far. Two suggestions can be made at this point one is to hybridizing the search with a best first search strategy instead of using the pure depth first search strategy and secondly one can use the idea in the Backgrounds section 2 where a multi-objective branch and bound procedure with a saved pool of feasible siblings at the depths when the child node values differ from their parent values has been described.

Appendix

The Table 6 shows the computations for examples from all the scenarios $1 - 20$ with the settings from Section 4.2.1, i.e, the original values has been replaced by the new node values. The trains arrived at their final destination with less than 60 seconds positive deviation from the original timetable are assumed to be on time.

†: Note that a train with 31 seconds deviation from the timetable is not considered as being delayed by Trafikverket.

The complete computations for all scenarios $(1 - 20)$ from Section 4.5 is reported in Table 7.

Table 6: Comparison between the original and new cost values in the branch and bound process computation time 30 seconds and weights 0.001, Scenarios with 90 minute time horizon

Scenario	Max depth	Lower bound	Original Final solution		New Final solution	
			1st Fea Sol & Nr Fea Sol	Best Sol & Nr d-Trn	1st Fea Sol & Nr Fea Sol	Best Sol & Nr d-Trn
01	560	586	$2395-6$	$1285-2$	$2395-3$	$1599-3$
02	560	226	$751-2$	$437-2$	$751-2$	$437-2$
03	558	570	$1150-2$	$781-2$	$1150-2$	$781-2$
04	558	210	$790-2$	$421-2$	$790-2$	$421-2$
05	575	201	$1174-4$	$686-3$	$1174-1$	$1174-1$
06	575	0	$46-2$	$31-0\dagger$	$46-2$	$31-0$
07	567	486	$568-2$	$499-1$	$568-2$	$499-1$
08	567	126	$401-2$	$332-1$	$401-2$	$332-1$
09	570	458	$869-2$	$800-1$	$869-1$	$869-2$
10	570	98	$338-2$	$269-2$	$338-1$	$338-3$
11	560	1022	$1547-2$	$1233-2$	$1547-2$	$1233-2$
12	558	469	$1049-2$	$680-2$	$1049-2$	$680-2$
13	576	1669	$2246-2$	$2231-3$	$2246-1$	$2246-3$
14	567	969	$1815-2$	$1653-5$	$1815-1$	$1815-6$
15	570	1371	$1680-2$	$1611-1$	$1680-1$	$1680-2$
16	508	13776	$13850-1$	$13850-12$	$13850-1$	$13850-12$
17	570	6534	$7338-2$	$7265-10$	$7338-2$	$7306-10$
18	565	1997	$4376-4$	$4100-14$	$4376-2$	$4211-15$
19	567	9270	$28883-3$	$28740-14$	$28883-3$	$28740-14$
20	560	6114	$21922-4$	$21247-18$	$21922-11$	$20752-20*$

Table 7: Solutions with new priority when conflict of interest for two trains occur, computation time 30 seconds and weights 0.001, backtracking to the new node values, all scenarios 90 minute time horizon.

Scenario	Solution			
	1st Feasible Sol	Nr of Feasible Sol	Best Sol	Nr d-Trains
01	2395	3	1599	3
02	751	2	437	2
03	1150	2	781	2
04	790	2	421	2
05	1321	2	1174	1
06	193	3	31	0
07	715	3	499	1
08	548	3	332	1
09	1016	2	869	2
10	485	2	338	3
11	1547	2	1233	2
12	1049	2	680	2
13	2393	2	2246	3
14	4134	4	1815	6
15	1827	2	1680	2
16	13850	1	13850	12
17	7466	2	7338	10
18	4604	2	4376	15
19	28817	2	28770	14
20	21317	4	20978	17

References

[1] Johanna Törnquist Krasemann, *Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances.* Transportation Research, Part C 20(2012), Pages 62–78.

[2] Syed Muhammad Zeeshan Iqbal, Håkan Grahn and Johanna Törnquist Krasemann, *A parallel heuristic for fast train dispatching during railway traffic disturbances-Early results.* ICORES 2012 Proceedings of the 1st International Conference on Operations Research and Enterprise Systems, Vilamoura, Algarve, Portugal, 4-6 February, 2012, Pages 405-414.

[3] Stefan Voß, Anders Fink, *Looking ahead with pilot method*, Annals of Operations Research, Number 136, Pages 285-302, 2005 Springer Science.

[4] Francis Sourd, Olivier Spanjaard, *A Multiobjective Branch-and-Bound Framework: Application to the Biobjective Spanning Tree Problem*, INFORMS Journal on Computing, Summer 2008, Volum. 20, Number 3, Pages 472-484.

[5] Wei Zhang, *Parallel Multi-Objective Branch and Bound*, Master thesis, Technical University of Denmark, Department of Informatics and Mathematical Modelling, Kongens Lyngby Denmark 2008, IMM-M.Sc-2008-39.

[6] Malin Forsgren, Martin Aronsson, Per Kreuger, Hans Dahlberg, *The Maraca - A tool for minimizing resource conflicts in a non-periodic railway timetable*, Project report, Trafikverket (Swedish Transport Administration), 2010.

Bilaga 3

S. Soltani, *Modeling train stations: case study Norrköping*, Project report, 2012-06-14.

Modeling train stations: case study Norrköping

Sara Soltani

June 14, 2012

Abstract

This report addresses the mathematical modeling of the problem, finding a conflict free train routing through complicated stations, assumptions and constraints. A simple example is then solved by this model and illustrated.

This work has been carried out as part of the work in the project titled "Efficient real-time re-scheduling of trains during disturbances (EOT)". This project is financed by Trafikverket (The Swedish Transport Administration).

1 Problem description and assumptions

When a disturbances happens in the railway network, the trains need to be re-scheduled and as a result, the re-allocation of tracks is unavoidable. A previously developed mathematical model of re-scheduling train timetables during disturbances is able to generate conflict free train schedules on stations and tracks by considering a minimum separation time (see Appendix D in [1]). In this macro model the structure of stations is simplified into sets of one to multiple parallel tracks. Each station is consists of a number of platforms and a large number of track sections while the conflicts of train paths could occur at the stations due to crossovers of tracks. For instance, in Figure 1 of the station Norrköping, if a trains arrives at an entering point from Fiskeby and have a request for using the platform 7, since any track section can only be claimed by one train at a time, this would cause a conflict with another train coming from the alternative entering point of Fiskeby at the same time having a request for platform 9. This kind of conflict has not been taken care of, by the macro model. We may need a more detailed model of the station to avoid such conflicts.

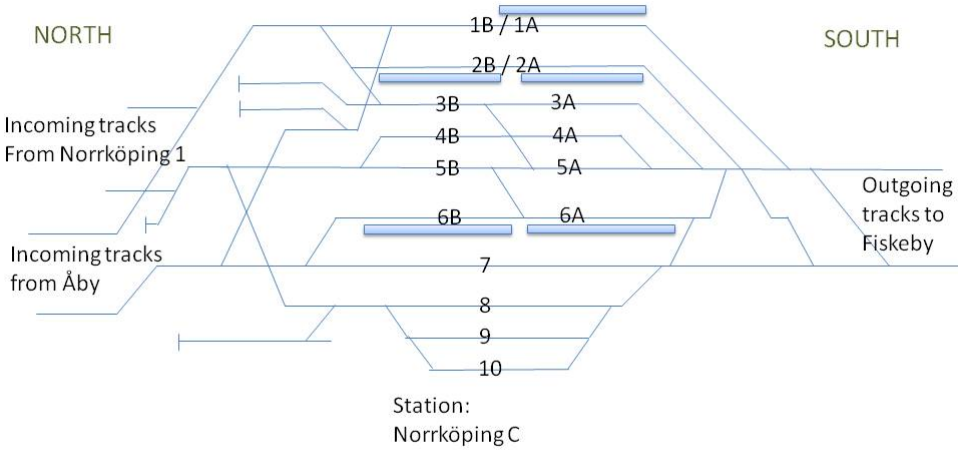
Identifying all routes is necessary to model allowed paths for each train. A route is a sequence of track sections linking an entering point to a platform and from that platform to an exiting point. A route is a one unique path through the station connecting an entering point to an exiting point. In general each entering point can be considered as an exiting point and vice versa according to the direction of the travel. Routes are generally defined as a triple of incoming track, station track/platform and outgoing track (triple of $\{x_1,\ x_2,\ x_3\}$ where x_1, x_2 and x_3 is a number between 1 and the station/line section capacity). Note that only relevant routes should be considered in the model.

If two routes are physically separated then there is no risk of conflict (the model already capture this) but in case of infrastructural conflict, i.e., if two different routes in the station have crossover of track sections or they share one or few track sections in the station entrances then the safety rules are not fulfilled if two trains using those track resources of that routes at the station at the same time. Then obviously those two train events at the station need to be separated so that the safety rules are satisfied.

The problem statement is that is it possible to route a set of trains with a given timetable through a specified train station such that no pair of trains is conflicting.

To solve this problem the main idea is to obtain a complete microscopic model of a railway station, model the train routing problem through a station as an integer linear programing subproblem to the the macroscopic train scheduling problem. I.e., it is assumed

Figure 1: The layout of the train station Norrköping C, Sweden



that the train scheduling problem of the macroscopic railway infrastructure is solved and the arrival times and departure times of the set of trains passing that particular station and the allocated tracks at entrances and exits of the station to the next infrastructure block are known.

Each station is unique and should be modeled as a unique integer linear programing problem. Since it is necessary to fulfill the requirements and demands of the train timetable all trains have to be routed through the station.

The first objective is to find a feasible conflict free routing of all trains in our given time. We intend to maximize the preferences of the trains for certain platforms or routes. I.e., giving different weights to different routes based on the dispatch preferences-ideally this is already captured in the macro model. Another objective could for instance, be minimizing the total time spent for each train at the station.

In this model we assume that the allocation of platforms to trains are given by the macroscopic model and conflict free routes for each train through the station should be obtained.

Our case study is the station Norrköping, Sweden, which has been illustrated in Figure 1. We consider all the trains passing by Norrköping. It is assumed that the station has 14 platforms and 3 parking areas. The parking areas are only used for non-passenger trains shunting down at Norrköping or have their first event at this station. The trains who have been shunted down in these parking areas, later, change their train number and direction and start a new travel. The length of those trains using the parking areas should be considered as well. In the simplified model we do not explicitly consider parking areas. We have simplified our model as follows: There are only 4 sections in the routing sub-problem and each train have at most 3 events (1 station section-Norrköping- and 3 line sections).
$FI - NR$: South stretch Fiskeby-Norrköping, $capacity = 2$;
NR: Station Norrköping C, $capacity = 14$;

$NR1-NR$: North stretch Norrköping1-Norrköping, $capacity = 1$;
$NR2-NR$: North stretch Åby-Norrköping, $capacity = 2$;

It is assumed that the events at these sections are ordered in time according to the new allocated start time solved with the macro model. Direction of each train should be known. It is assumed that in our example the clearing time for a resource at the station is given and is 30 seconds.

2 The mathematical model

In the mathematical model of the routing problem all trains passing Norrköping are considered. Let i and j and k represent indexes for a train, a railway section and a train event respectively. In the model 4 sections ($NR2-NR$, $NR1-NR$, NR and $FI-NR$) are included. The arrival and departure times and the used tracks for each train at $NR2-NR$, $NR1-NR$, NR and $FI-NR$ are known.

Let R be the set of routes at the station Norrköping. Then we define the binary variable $y_{i,k,r} = 1$, for $r \in R$ if event k of train i takes the route r at Norrköping, and 0 otherwise. Then clearly:

$$\sum_{r \in R} y_{i,k,r} = 1, \ k \in L_j \text{ and } k \in K_i, \tag{1}$$

where j is Norrköping. This equations ensure that train i is assigned exactly one route at the station.

Let us define the matrix $A = (a_{r,\tilde{r}})$ such that $a_{r,\tilde{r}} = 1$ if routes r and $\tilde{r}$ have infrastructural conflicts and $a_{r,\tilde{r}} = 0$ otherwise, where $r \neq \tilde{r}$.

If two trains are entering and exiting the station from the same tracks and use a different platform, then they have been already separated in time in the macro model by the λ and γ binary variables. If they use different tracks outside the station and different platforms in the station, the routes they use through the station, may have infrastructural conflicts and therefore they need to be separated in time. Let us define the parameter:

$$u_{k,\tilde{k}} = \begin{cases} 1, & \text{If } k, \tilde{k} \in L_j, k < \tilde{k} \text{ have conflict in time,} \\ 0, & \text{otherwise,} \end{cases}$$

where j is Norrköping. In other words $u_{k,\tilde{k}} = 1$, where $k \in K_i$ and $\tilde{k} \in K_{\tilde{i}}$, means that two trains i and $\tilde{i}$ appear in the station at the same time.

Note that the section events are ordered according to the new start time given by the solution of the macro model. Therefore if two train events have time conflicts over the routes with infrastructural conflicts, first train goes first (i.e., we keep the order).

Assigning the value 1 to parameters $u_{k,\tilde{k}}$ is done by the following inquiry. Let the start time and the end time for the train i at the station Norrköping (event $k \in K_i$) obtained by solving the macro model are $x_{i,k}^{start}$ and $x_{i,k}^{end}$; then:

$$u_{k,\tilde{k}} = 1, \text{ If } k, \tilde{k} \in L_j, k < \tilde{k}, \ k \in K_i, \ \tilde{k} \in K_{\tilde{i}}, \text{ and} \tag{2}$$

$$x_{\tilde{i},\tilde{k}}^{start} \geq x_{i,k}^{start} \ \& \ x_{i,k}^{end} \geq x_{\tilde{i},\tilde{k}}^{start};$$

where j is the station Norrköping. In addition we have the binary variable:

$$\mu_{k,\tilde{k}} = \begin{cases} 1, & \text{If event } k \text{ is in time conflict with event } \tilde{k} \\ & \text{and need to be separated, } k, \tilde{k} \in L_j, k < \tilde{k}, \\ 0, & \text{otherwise,} \end{cases}$$

where j is Norrköping. We define the constraint

$$u_{k,\tilde{k}} \leq \mu_{k,\tilde{k}}. \tag{3}$$

The constraint (3) is defined to imply that if two events are both in time conflict and infrastructural conflict, then event k will be separated in time from the event $\tilde{k}$ by the constraints stated below; and if two events are not in time conflict then they do are already fulfill the safety conditions and the separation constraints will be true.

Define the continuous time variables $u_{i,k}^{start}$ and $u_{i,k}^{end}$ which specifies the start time and the end time of event k of train i at station Norrköping; then we will have the time constraints:

$$u_{i,k}^{start} \geq x_{i,k}^{start},\ i \in T,\ k \in L_j, \tag{4}$$

$$u_{i,k}^{end} - u_{i,k}^{start} \geq d_{i,k}, \tag{5}$$

Where j is the station Norrköping and $d_{i,k}$ is the minimum time required for train i to path the station.

We define three objective functions as follows:

1. Maximize route preference: Some routes have different importance for different trains, for instance, the fast long distance passenger $X2000$ train 519 from Stockholm to Malmö has a preference to use the platform $1B/1A$ and the local passenger train 2140 from Norrköping C towards Åby and then to Katrineholm C to is preferred to stop at platform $2B$ to exchange the passengers. The slow cargo trains are preferred to use tracks 8, 9 or 10 to stop at the station, where there are no actual platform which is needed for passenger trains (see the Figure 1). The shorter routes for passenger trains through the station are preferred.

$$\text{Maximize} \sum_{i,k,r} w_{i,r} y_{i,k,r}, \quad i \in T,\ k \in K_i,\ r \in R; \tag{6}$$

   where $w_{i,r}$ is the weights given to each route for different trains; i.e., preferred routes have higher weights.

2. Minimize travel time through the station: scheduled passenger stops at station Norrköping for passenger trains should be fulfilled while unnecessary stops should be avoided.

$$\text{Minimize} \sum_{i\ \in T} (u_{i,k}^{end} - u_{i,k}^{start}), \quad k \in K_i. \tag{7}$$

3. Minimize discrepancy: As in the equations (4) and (5) we do not explicitly use the $u_{i,k}^{end}$ variables, we may end up with new start and end times for events in the station which are very different from the solution in the macro model. To avoid this, we can minimize the discrepancy from the start and end times given by the macro model.

$$\text{Minimize} \sum_{i \in T} |u_{i,k}^{start} - x_{i,k}^{start}| + |u_{i,k}^{end} - x_{i,k}^{end}|, \quad k \in K_i. \tag{8}$$

   Therefore, we will mainly use this later objective.

Of great importance are constraints defining infrastructural conflicts of routes. There are routes in conflict at entrances. There are conflicts of routes with platforms, as one route may pass a platform to get to another one. These kind of constraints in general can be presented as follows:

$$y_{i,k,r} + y_{\tilde{i},\tilde{k},\tilde{r}} \leq 1 + a_{r,\tilde{r}} u_{k,\tilde{k}} + (1 - \mu_{k,\tilde{k}}); \tag{9}$$

$$\text{for all } i, \tilde{i} \in T,\ k \in K_i,\ \tilde{k} \in K_{\tilde{i}},\ r, \tilde{r} \in R.$$

The constraints below ensure that one event on a route at Norrköping must end and a required separation time must elapsed until next event may start at another route with infrastructural conflict with the first route. Note that the simultaneous usage of the same routes has been taken care of by the macro model. Here Δ is the minimum required separation time for two trains and M is a large constant.

Considering the direction of travel, there are three different situations of train event conflicts at the station which are needed to be taken care of, in the separation constraints.

- Two trains in the opposite direction, which they meet.
- Two trains in the same direction, who follow each other.
- Two trains in the same direction, and one train overtake the other one, (The train enters later than the other one but leaves earlier).

Two trains in the opposite direction: The start time and the end time of trains with opposite direction should be separated in time, but it is assumed that if a train ends its journey in Norrköping then it is not necessary to separate it's end time with the start time of another train at Norrköping. Also if a train starts its journey at the station, it's start time do not need to be separated from the end time of another train, and obviously if two trains with opposite directions one ends its journey and another starts its journey to/from Norrköping they do not need to be separated. In this situation we have assumed there are no infrastructural conflicts.

$$u^{start}_{\tilde{i},\tilde{k}} - u^{end}_{i,k} \geq \Delta(1-\mu_{k,\tilde{k}}) - M\mu_{k,\tilde{k}}, \tag{10}$$

$$\text{where } k,\tilde{k} \in L_j,\ k \in K_i,\ \tilde{k} \in K_{\tilde{i}},\ dir_k \neq dir_{\tilde{k}},$$

$$secEvent_{j,k} \neq t\_nrE[i]\ \&\ secEvent_{j,\tilde{k}} \neq 1,\ j = \text{Norrköping};$$

$$u^{end}_{i,k} - u^{start}_{\tilde{i},\tilde{k}} \geq \Delta\mu_{k,\tilde{k}} - M(1-\mu_{k,\tilde{k}}), \tag{11}$$

$$\text{where } k,\tilde{k} \in L_j,\ k \in K_i,\ \tilde{k} \in K_{\tilde{i}},\ dir_k = dir_{\tilde{k}},$$

$$secEvent_{j,k} \neq t\_nrE[i]\ \&\ secEvent_{j,\tilde{k}} \neq 1,\ j = \text{Norrköping};$$

where $secEvent_{j,k}$ is the index of the train event k for $k \in K_i$, and $secEvent_{j,\tilde{k}}$ is the index of the train event $\tilde{k}$ for $\tilde{k} \in K_{\tilde{i}}$ at the station. t_nrE is the total number of train events for train $i \in T$.

Two trains in the same direction: The situation in which two trains coming from the same direction using the routes with infrastructural conflicts is handled with the start times and end times separated with a clearing time. So one is entering waiting at the station, other enters after the clearing time and they leave with a safe separation time of their end times. It is assumed that if two trains have the same direction and one of the trains has it's first event at the station then their start times do not need to be separated. Also if two trains with the same direction and one of the trains has it's last event at the station then obviously their end times do not need to be separated.

Separation of the start times:

$$u^{start}_{\tilde{i},\tilde{k}} - u^{start}_{i,k} \geq \Delta\mu_{k,\tilde{k}} - M(1-\mu_{k,\tilde{k}}), \tag{12}$$

$$\text{where } k,\tilde{k} \in L_j,\ k \in K_i,\ \tilde{k} \in K_{\tilde{i}},\ dir_k = dir_{\tilde{k}},$$

$$secEvent_{j,k} \neq 1\ \&\ secEvent_{j,\tilde{k}} \neq 1,\ j = \text{Norrköping};$$

$$u^{start}_{i,k} - u^{start}_{\tilde{i},\tilde{k}} \geq \Delta(1-\mu_{k,\tilde{k}}) - M\mu_{k,\tilde{k}}, \tag{13}$$

$$\text{where } k, \tilde{k} \in L_j,\ k \in K_i,\ \tilde{k} \in K_{\tilde{i}},\ dir_k = dir_{\tilde{k}},$$
$$secEvent_{j,k} \neq 1\ \&\ secEvent_{j,\tilde{k}} \neq 1,\ j = \text{Norrköping};$$

Two trains in the same direction, who follow each other:

$$u^{end}_{\tilde{i},\tilde{k}} - u^{end}_{i,k} \geq \Delta\mu_{k,\tilde{k}} - M(1 - \mu_{k,\tilde{k}}), \tag{14}$$
$$\text{where } k, \tilde{k} \in L_j,\ k \in K_i,\ \tilde{k} \in K_{\tilde{i}},\ dir_k = dir_{\tilde{k}},\ x^{end}_{\tilde{i},\tilde{k}} > x^{end}_{i,k}$$
$$\&\ secEvent_{j,s} \neq t\_nrE[i]$$
$$\&\ secEvent_{j,\tilde{k}} \neq t\_nrE[\tilde{i}],\ j = \text{Norrköping};$$

$$u^{end}_{i,k} - u^{end}_{\tilde{i},\tilde{k}} \geq \Delta(1 - \mu_{k,\tilde{k}}) - M\mu_{k,\tilde{k}}, \tag{15}$$
$$\text{where } k, \tilde{k} \in L_j,\ k \in K_i,\ \tilde{k} \in K_{\tilde{i}},\ dir_k = dir_{\tilde{k}},\ x^{end}_{\tilde{i},\tilde{k}} > x^{end}_{i,k}$$
$$\&\ secEvent_{j,s} \neq t\_nrE[i]$$
$$\&\ secEvent_{j,\tilde{k}} \neq t\_nrE[\tilde{i}],\ j = \text{Norrköping};$$

Two trains in the same direction, where one train overtake the other one,: In this case the two trains end times are compared and the train which has a shorter waiting time at the station leaves earlier.

$$u^{end}_{\tilde{i},\tilde{k}} - u^{end}_{i,k} \geq \Delta(1 - \mu_{k,\tilde{k}}) - M\mu_{k,\tilde{k}}, \tag{16}$$
$$\text{where } k, \tilde{k} \in L_j,\ k \in K_i,\ \tilde{k} \in K_{\tilde{i}},\ dir_k = dir_{\tilde{k}},\ x^{end}_{\tilde{i},\tilde{k}} \leq x^{end}_{i,k}$$
$$\&\ secEvent_{j,s} \neq t\_nrE[secTrain_{j,s}]$$
$$\&\ secEvent_{j,\tilde{k}} \neq t\_nrE[\tilde{i}],\ j = \text{Norrköping};$$

$$u^{end}_{i,k} - u^{end}_{\tilde{i},\tilde{k}} \geq \Delta\mu_{k,\tilde{k}} - M(1 - \mu_{k,\tilde{k}}), \tag{17}$$
$$\text{where } k, \tilde{k} \in L_j,\ k \in K_i,\ \tilde{k} \in K_{\tilde{i}},\ dir_k = dir_{\tilde{k}},\ x^{end}_{\tilde{i},\tilde{k}} \leq x^{end}_{i,k}$$
$$\&\ secEvent_{j,s} \neq t\_nrE[i]$$
$$\&\ secEvent_{j,\tilde{k}} \neq t\_nrE[\tilde{i}],\ j = \text{Norrköping};$$

Another factor that need to be considered is modeling the feasible physical train paths (routes) at the station, i.e., defining the start, the station platform and the end points of each route. For example trains coming to Norrköping from $NR1 - NR$ use a set of specific routes different from routes coming from $NR2 - NR$. Tracks 1 and 2 enter/exit $NR2 - NR$ and track 1 enter/exit $NR1 - NR$. If a train is assigned to a specific platform, it is only allowed to use routes that paths that platform. In these kind of constraints the direction of trains are of importance.

Let R^*_i be the set of feasible routes for train i in the set of trains T. In general the constraints giving the true value to feasible routes for each train can be stated as follows.

$$\sum_{r \in R^*_i} y_{i,k,r} = 1, \quad \text{where } r \in R,\ \text{for } i \in T,\ k \in K_i; \tag{18}$$

$$\sum_{r \notin R^*_i} y_{i,k,r} = 0, \quad \text{where } r \in R,\ \text{for } i \in T,\ k \in K_i. \tag{19}$$

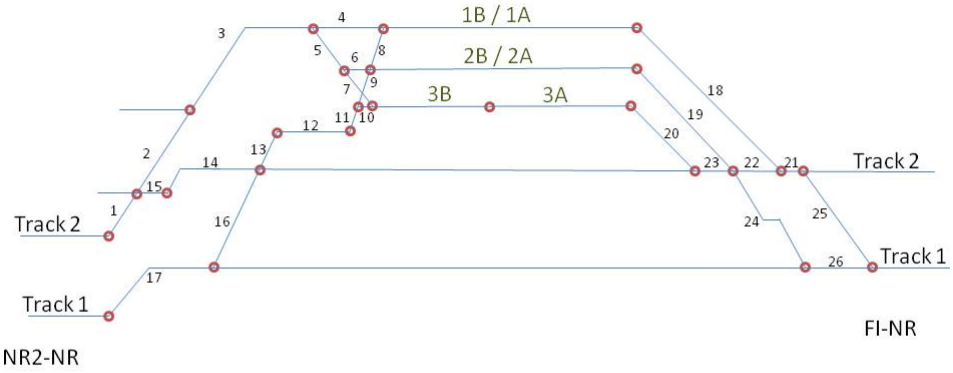
The constrains which in details, give the feasible routes for trains using specific tracks at the station Norrköping can be found in the Appendix.

3 Identifying routes

In our mathematical model we have assumed that the allocation of platforms are already given by the macroscopic model and we only need to find a conflict free route for each train. To capture the details of the station it is necessary to identify all the possible relevant routes through the station.

This is how we identify all routes in details with resources: Each segment (resource) at the station is numbered uniquely. Figure 2 shows a numbering to each track segment for a part of the station Norrköping. The numbering here is only for illustrative purposes and it is not the same numbering as in the real situation of train traffic dispatching in Sweden.

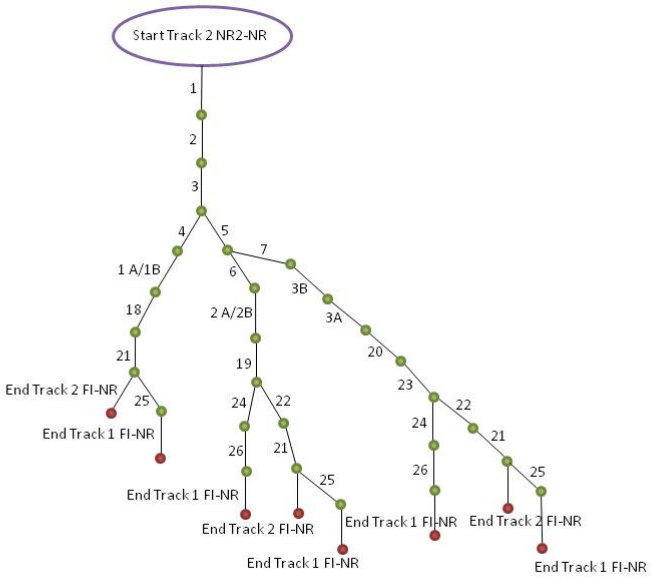
Figure 2: Numbered track sections for a part of Norrköping C



To model all routes we use a graph model which can be called resource tree, it represents a decision of choosing the next track segment with different alternatives at crossovers of tracks. Each train at the station's entering point can choose different routes to enter a specific platform. The edges represents the specific track and have the same name and thus a node groups routes that have a common sub-route. The reason of using this resource tree is that it is easer to later on, identify all the conflicts. Each starting point of the entrances or the exits to the platforms can be the starting point of a graph. This way a complete list of all routes with given track numbers can be written. All routes in the same graph have crossovers and conflicts. All graphs should be compared to each other to identify all the remaining conflicts. The Figure 3 shows a graph with the starting point of track 2 of stretch $NR2 - NR$ with the numbering given in the Figure 2. The graph in Figure 3 gives 8 different routes through the station.

Then we code the routes according to which that are in conflict due to usage of the same platform/station track and identify the routes that have the same way in and out and which then are separated by the λ and γ constraints from the macro model. Hence the physical conflicts are already handled by the λ and γ. Now we only need to investigate which

Figure 3: Resource tree: the starting point of track 2 of $NR2 - NR$, track numbers from Figure 2



routes that are conflicting in time and if the conflict is on the north (N) end of the station (exit/enter to $FI - NR$), or the south (S) end (exit/enter to $NR1 - NR$ or $NR2 - NR$) or both (B).

In the simplified model 207 possible relevant routes has been identified which can be used by trains to path station Norrköping. It is assumed that all the routes are bi-directional. For more details see Appendix B.

4 Example

Suppose that we have a 10 minutes window of time at Norrköping, during this time 4 trains enter and exit the station.

- North-bound train 2140 incoming from UPP-spåret (track 1) to platform $2B$ and then wait for 3 minutes, then out to UPP-spåret towards Åby.
- Southbound train 519 incoming from NED-spåret (track 2) to platform 1b and then wait for 1 minute, then out on NEDs-påret towards Fiskeby.
- North-bound cargo train 49500 coming from NED-spåret (to platform 9) and then out to UPP-spåret towards Åby.
- North-bound train 500 incoming from UPP-spåret (track 1) to platform 7 and then wait for 2 minutes, then out to UPP-spåret towards Åby.

And let's assume that the start and end times for each train given by the macro model are $60 - 260$, $75 - 135$, $410 - 600$ and $420 - 540$ respectively. The solution given by the proposed mathematical model is illustrated in the Figures 4-6.

Figure 4: Solution times given by the micro model

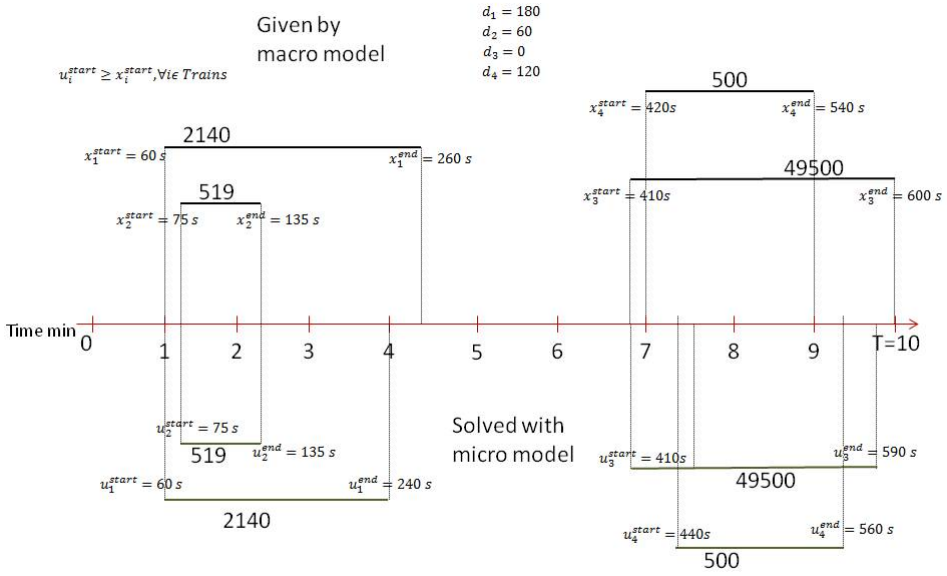


Figure 5: Routes assigned to trains 1 and 2 by the micro model

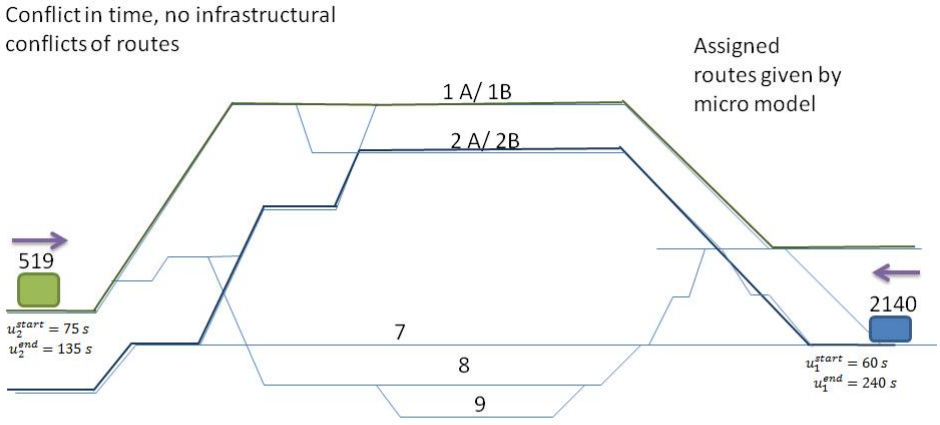
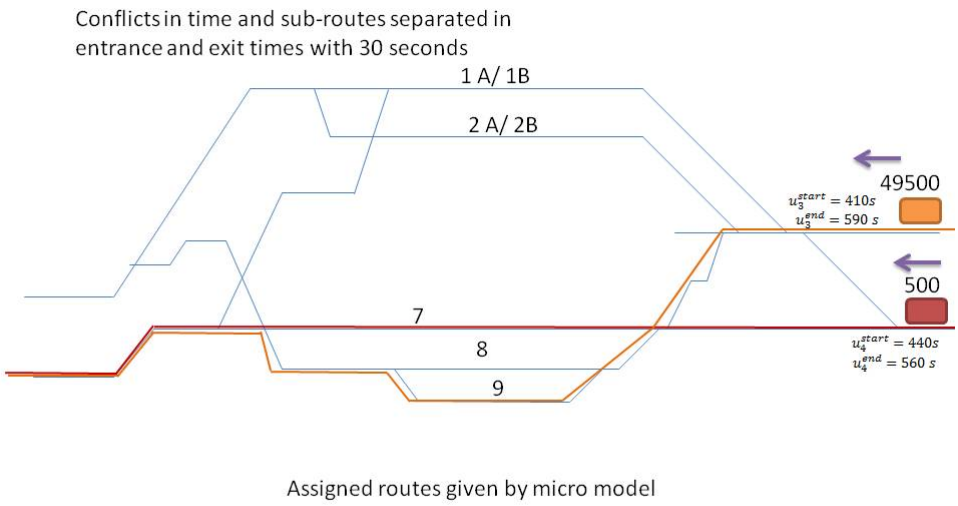


Figure 6: Routes assigned to trains 3 and 4 by the micro model



5 Further discussions

In our mathematical model of the station Norrköping some minor infrastructural details are ignored, as it is assumed the stretch $NR1 - NR$ has only one track or the parking areas are not in use. It is important to get real data and information from train dispatchers to capture necessary details and learn about route preferences for specific trains and implement them in the model.

We have assumed that the platforms in the station are already given by the macro model, but in reality some trains have no stop at the station and they can be assigned any route to pass the station (preferably the shortest path). If the trains are already given a platform to path, there is not so much liberty to maximize the preferred routes in the objective function 1. Therefore the choice of station track or platform should be imported from the macro model only if the train is very long, or has a passenger stop at a pre-scheduled passenger platform. Also the length of trains has not been explicitly considered in our model.

The reader should note that in our model, when considering the conditions of a train event being the first or the last event of that train no difference has been made to know that if they are the actual first or last event or they are, as a result of our considered window of time.

Appendix A

Let $q_{i,k,t}$ be the parameter that specifies the tracks used by train events in the sections $NR1-NR$ or $NR2-NR$ and $FI-NR$ and the platform for NR. Then the feasible routes for each train in the station Norrköping is given by:

For trains coming from $NR2-NR$ entering NR using track 1

$$\sum_{r \in R^1_{NR2-NR}} y_{i,k,r} = q_{i,k-1,1}, \tag{20}$$

$$k, (k-1) \in K_i,\ k-1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = NR2-NR,\ j = NR;$$

where R^1_{NR2-NR} is the set of routes starting/ending to track 1 stretch $NR2-NR$.

For trains coming from $NR2-NR$ entering NR using track 2

$$\sum_{r \in R^2_{NR2-NR}} y_{i,k,r} = q_{i,k-1,2}, \tag{21}$$

$$k, (k-1) \in K_i,\ k-1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = NR2-NR,\ j = NR;$$

where R^2_{NR2-NR} is the set of routes starting/ending to track 2 stretch $NR2-NR$.

For trains exiting from $NR1-NR$ entering NR using track 1

$$\sum_{r \in R^1_{NR1-NR}} y_{i,k,r} = q_{i,k-1,1}, \tag{22}$$

$$k, (k-1) \in K_i,\ k-1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = NR1-NR,\ j = NR;$$

where R^1_{NR1-NR} is the set of routes starting/ending to track 1 stretch $NR1-NR$.

For trains exiting to $FI-NR$ from NR using track 1

$$\sum_{r \in R^1_{FI-NR}} y_{i,k,r} = q_{i,k+1,1}, \tag{23}$$

$$k, (k-1) \in K_i,\ k+1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = FI-NR,\ j = NR;$$

where R^1_{FI-NR} is the set of routes starting/ending to track 1 stretch $FI-NR$.

For trains exiting to $FI-NR$ from NR using track 2

$$\sum_{r \in R^2_{FI-NR}} y_{i,k,r} = q_{i,k+1,2}, \tag{24}$$

$$k, (k-1) \in K_i,\ k+1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = FI-NR,\ j = NR;$$

where R^2_{FI-NR} is the set of routes starting/ending to track 2 stretch $FI-NR$.

For trains using platform p at the station NR

$$\sum_{r \in R^p_{NR}} y_{i,k,r} = q_{i,k,p}, \tag{25}$$

$$k \in K_i,\ k \in L_j,\ j = NR;$$

where R^p_{NR} is the set of routes pathing platform p.

For trains exiting to $NR2-NR$ from NR using track 1

$$\sum_{r \in R^1_{NR2-NR}} y_{i,k,r} = q_{i,k+1,1}, \tag{26}$$

$$k, (k-1) \in K_i,\ k+1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = NR2 - NR,\ j = NR;$$

where R^1_{NR2-NR} is the set of routes starting/ending to track 1 stretch $NR2 - NR$.

For trains exiting to $NR2 - NR$ from NR using track 2

$$\sum_{r \in R^2_{NR2-NR}} y_{i,k,r} = q_{i,k+1,2}, \tag{27}$$

$$k, (k-1) \in K_i,\ k+1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = NR2 - NR,\ j = NR;$$

where R^2_{NR2-NR} is the set of routes starting/ending to track 2 stretch $NR2 - NR$.

For trains exiting to $NR1 - NR$ from NR using track 1

$$\sum_{r \in R^1_{NR1-NR}} y_{i,k,r} = q_{i,k+1,1}, \tag{28}$$

$$k, (k-1) \in K_i,\ k+1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = NR1 - NR,\ j = NR;$$

where R^1_{NR1-NR} is the set of routes starting/ending to track 1 stretch $NR1 - NR$.

For trains coming from $FI - NR$ to NR using track 1

$$\sum_{r \in R^1_{FI-NR}} y_{i,k,r} = q_{i,k-1,1}, \tag{29}$$

$$k, (k-1) \in K_i,\ k-1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = FI - NR,\ j = NR;$$

where R^1_{FI-NR} is the set of routes starting/ending to track 1 stretch $FI - NR$.

For trains coming from $FI - NR$ to NR using track 2

$$\sum_{r \in R^2_{FI-NR}} y_{i,k,r} = q_{i,k-1,2}, \tag{30}$$

$$k, (k-1) \in K_i,\ k-1 \in L_{\tilde{j}},\ k \in L_j,\ \tilde{j} = FI - NR,\ j = NR;$$

where R^2_{FI-NR} is the set of routes starting/ending to track 2 stretch $FI - NR$.

Appendix B

In the excel file titled "Routes" accompanying this report, all the identified routes through the station has been listed. It is stated in Section 3 that 207 routes has been identified for the simplified Norrköping where there are 1 entrance track from Norrköping1 and there are 14 platforms and no parking areas. The excel file "Routes" has listed 270 identified routes. Routes 1 to 207 in this file are the paths through the stations from 1 entrance track of Norrköping1 and 2 entrance tracks from Åby, to 2 entrance tracks toward Fiskeby. Routes 208–228 are the routes through the station from Norrköping1 if we consider an alternative entrance track (see Figure 1). Routes 228–270 are the identified routes for the 3 parking areas in the station. These routes are not paths through the station but rather a path for a non passenger train to start the journey or end a journey from a parking area.

To identify the routes the entrance direction is assumed to be from Åby/Norrköping1 and the exit direction is to Fiskeby. In practice the direction does not play any role as the routes are assumed to be bidirectional. Routes can be defined as a triple of incoming track, station track/platform and outgoing track. Here to demonstrate better a certain path (the route) through the complicated station Norrköping, 4 different middle points (The station numbered safety points) has been considered at the entrances of the station. So a certain route is defined by the entrance track from Åby/Norrköping1, 4 middle points for a train to

get to a certain platform, the platform number, and 4 middle points from the platform to the exit point toward Fiskeby and at last the outgoing track to Fiskeby. This numberings are according to the real numbering system found in "Trafikbilder".

The routes has been colored according to which that are in conflict due to usage of the same platform/station track ,i.e., routes through the same platform have the same color (The physical conflicts are already handled by the λ and γ variables in the macro model for the routes using the same platforms).

This excel file "Routes" can then be used to look up the exact path for a certain train, when the route number has been printed in the "Norrköping.txt" file for each train, as the solution of the micro model of the station Norrköping.

The intention for the excel file "Route conflicts table" is to build up a complete matrix of all the conflicts of all the identified routes. The table in this excel file is a matrix of the size 270 by 270, any route should be compared by any other route to obtain a complete table. Conflicts of the two routes that have the same way in and out of the station, are separated by the λ and γ constraints in the macro mathematical model and are denoted by λ_2. The conflicts of the routes using the same platforms are denoted by λ_1. The routes that are conflicting in time and where the conflict is on the entrance of Åby/Norrköping1 is denoted by (N), and the conflict at the end of station toward Fiskeby is denoted by (S) and both (B) is used in case of conflicts at both ends.

References

[1] Johanna Törnquist Krasemann, *Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances.* Transportation Research, Part C 20(2012), Pages 62–78.