Automatisk Planlegging i Oljeindustrien

av

Teis Lindemark

v0.9

AVHANDLING

for en grad av

MASTER I INFORMATIKK

Masteroppgave, Institutt for informatikk

Universitetet i Bergen

Juni 2012

Universitetet i Bergen

Forord

Først ønsker jeg å takke min instituttveileder, professor Marc Bezem, og min veileder hos Epsis, Bård Henning Tvedt for veldig god veiledning gjennom hele prosessen. Takk for at dere brukte av deres dyrebare tid for å veilede meg gjennom denne oppgaven!

Jeg takker også de rundt meg, spesielt mine foreldre, for god støtte gjennom prosessen. De har støttet meg gjennom gode og mindre gode tider. Da jeg ikke så enden på tunnelen, var det alltid godt å få nødvendig støtte.

Til slutt må jeg også takke foreldrene mine for at de kunne hjelpe til med å korrekturlese oppgaven.

Teis Lindemark, 30. Mai 2012

Innhold

Tabeller

Figurer

Forkortelser

AI	Kunstig intelligens (engelsk: artificial inteligence)
APO	Automatisk Planlegging i Oljeindustrien
Avgjørings-variable	På engelsk: decision variable
Avlednings-variable	Derived variable
Concert	IBM ILOG Concert Technology
CP	Begrensningsprogrammering (engelsk: Constraint programming)
Eneressurs	På engelsk: Unary resource
JSSP	Job Shop Scheduling Problem
LS1	Løsningsstrategi 1
LS2	Løsningsstategi 2
MSProject	Microsoft Project 2010
Påstander	Engelsk: Assertions
Scheduler	IBM ILOG Scheduler
Solver	IBM ILOG Solver

1 Introduksjon

Denne oppgaven er en utvidelse av Bård Henning Tvedt sitt arbeid "Automatisk Planlegging i Oljeindustrien (APO)" [15]. APO er en planleggingsmetodikk for planlegging av vedlikeholdsaktiviter i oljeindustrien. Det blir brukt en fiktiv oljeplatform for å kunne gjøre problemet så likt som mulig virkeligheten. Implementasjonen i IBM ILOG Scheduler (Scheduler) er utvidet med en begrensning som kalles varme. Varme i denne sammenheng er ikke å forstå som varme i tradisjonell forstand, men et mål for hvor mye belastning en lokasjon på platformen tåler. Løsningene blir evaluert med et eget (eksternt) program, som kalkulerer teoretisk- øvregrense og nedregrense og sjekker om begrensningene er tilfredstilt.

1.1 Beskrivelse av kommende kapitler

I kapittelet (kapitlet?) om metode vil både fremgangsmåten for prosjektet bli lagt frem og hvordan løsningene har blitt evaluert. Under kapittelet (?) om resultater vil det legges frem løsninger med forskjellige strategier, både med og uten varmebegrensning. Løsningene evalueres og til slutt sammenfattes en konklusjon over det arbeidet som er gjort.

1.2 Bakgrunn

I dette prosjektet er verktøyet som er brukt for å utføre eksperimenter på emnet, Scheduler. Dette er en del av IBM sitt ILOG CP produkt. Scheduler er et C++ bibliotek som gjør det mulig å definere planleggingsbegrensninger i form av ressurser og aktiviteter. Planlegging er en prosess hvor det tildeles ressurser til aktiviteter og tid til de forskjellige aktivitetene slik at det ikke oppstår noen konflikt med begrensningene [16]. Automatisk planlegging er en del av det som kalles kunstig intelligens (AI).

I kompleksitetsteori [7] kaller man problemer som er minst like vanskelige å løse som de vanskeligste problemene i kategorien *NP*, som *NP-hard*. NP er kategorien av problemer som kan løses i polynomisk tid på en ikke-deterministisk

Turing maskin. Det er forskjellige kategorier av NP-harde problemer. Dette er avgjørelsesproblemer, søkeproblemer og optimaliseringsproblemer [8]. Problemet i dette prosjektet(prosjektarbeidet?) er i kategorien optimaliseringsproblem. Fra wikipedia [9] er et optimaliseringsproblem: "Et problem med å finne den beste løsningen ut ifra alle gyldige løsninger".

Taillard [17] har sett på benchmarking i enkle planleggingsproblemer som grunnlag for hvordan man best mulig benchmarker vanskelige problemer. Han har definert vanskelige problemer ved å se på makespan og hvor langt unna makespan er den nedre grensen. Han har tatt utgangspunkt i tre kjente begrensningsproblemer og laget benchmarksett til disse.

Wallace [18] har også sett på benchmarking og tatt for seg positive og negative sider ved dette. Benchmarking har to hovedoppgaver. Den første er å dekke et representativt problem og bredt programmeringsbegrep. Den andre er benchmarking av applikasjoner med "enhetstester". Utfordringen med begge er å opprette en klar målsetting for ytelsen. Teoretisk sett er den mest ideele måten å benchmarke en applikasjon på å implementere en løsning for hvert system og velge den beste. Men dette er ikke alltid mulig i praksis. Måten å gjøre dette på i praksis, er å benchmarke systemer med forhåndsdefinerte problemer og håpe at resultatet kan bli overført til applikasjonen som krever det. Wallace tar også opp vanskeligheten med tidsmekanismer i høynivåspråk. Tidsmekanismer er vanligvis basert på CPU tid som kan variere fra oppsett til oppsett.

2 Motivasjon og målsetting

Prosjektoppaven er motivert av praktisk(generell?) erfaring fra planleggingsprosesser i bedrifter og samfunnet forøvrig. I hverdagen har hver og en av oss mange planleggingsproblemer, alt fra buss- og togtabeller til personlige gjøremål som skal inn i en stram tidsplan. Nærligslivet har i bedriver i varierende grad bemannings- og ressursplanlegging. Det å finne optimale løsninger på planleggingsoppgaver er viktig og ressursbesparende. I denne prosjektoppgaven er det planlegging i oljeindustrien som er i fokus.

Planlegging i oljeindustrien er viktig, for på en mest mulig effektiv måte benyt-

te seg av de ressursene som er tilgjengelige til enhver tid, samtidig som visse begrensinger blir fastsatt med tanke på blandt annet sikkerheten. Det er også mye penger involvert i oljeindustrien. Det å utføre vedlikeholdsaktiviteter på en inneffektiv måte kan for selskapene både bli svært kostbart samt medføre stor risiko dersom sikkerheten ikke er godt nok ivaretatt. Det er derfor svært viktig å ha gode løsninger for planlegging av vedlikeholdsaktiviteter og ressurser. Målet til operatører er bl.a å minimere antallet farlige situasjoner og minimere fare for miljøskader samtidig som de ønsker å maksimere produksjon og dermed økonomisk resultat. På veien for å nå disse kompliserte og utfordrende målene, kan det i planleggingen oppstå konflikter i enkelte situasjoner (overfylte lokasjoner, manglende utstyrskrav, feil i rekkefølgen aktiviteter utføres, etc.).

Spesifikke planleggingssituasjoner i oljeindustrien har forskjellige forutsetninger i forhold til om det er offshore eller onshore. Offshore blir mange aktiviteter utført på et relativt lite lukket område. Onshore blir mange aktiviteter utføres på et relativt stort område. Selv om forutsetningene for disse er ganske forskjellige, så har de flere likhetstrekk:

- størrelse - antall aktiviteter kan være noen hundre til titusener, som gjør planlegging uoversiktelig.
- kompleksitet - et stort antall begrensninger skal tilfredsstilles og gjør planlegging vanskelig.
- dynamikk - avhengigheter som vær, logistikk og defekt på utstyr endrer planleggingsforutsetningene.

2.1 Målet med prosjektet

Målet med prosjektet er å utvide den eksisterende Scheduler løsningen med en varmeressurs og evaluere løsningene med og uten varmebegrensning. Prosjektets mål er da følgende punkter:

- minimere *makespan*
- legge til *varmebegrensning*
- implementere i Scheduler

I den opprinnelige problemstillingen [15] er noen aktiviteter relativt lite begrenset. Dette gjør at løsningsrommet er stort, og traverseringen opp og ned i søketreet tar lang tid. På tross av et antatt stort løsningsrom så sliter den Scheduler implementerte løsningsstrategien med å finne løsninger i mange av probleminstansene. Dette prosjektet vil søke å finne ut:

- Vil flere begrensninger gjøre det lettere å finne en løsning?
- Er det noe spesielt med akkurat disse probleminstansene eller er det implementasjon i Scheduler som er årsaken til at den Scheduler implementerte løsningenstrategien sliter med å finne løsninger i mange av probleminstansene?

2.2 Relevans for forskingen(prosjektoppgave?)

Denne forskningen har som hensikt å undersøker om løsningene blir bedre når et eksistrende planleggingsproblem får lagt til en ekstra begrensning, kalt varmebegrensning. Ved å evaluere løsningene med og uten varmebegrensing vil effekten av tillagt varmebegrensing fremgå. I tillegg fremgår effekten av lokasjon ved å sammenstille resultatene i denne oppgaven med resultatene i [15].

3 Problemstilling

Problemstillingen i denne oppgaven tar utgangspunkt i den opprinnelige problemstillingen til Bård Henning Tvedt . Beskrivelsen i det etterfølgende og frem til det fremkommer i etterfølgende tekst, er en oversettelse av [15] til norsk. Problemet er på en fiktiv oljeplattform inndelt i et sett av lokasjoner. Utstyr som er krevd for vedlikehold er tilfeldig plassert rundt på plattformen og ulike aktiviteter skal planlegges. Aktivitetene blir opprettet med et gitt sett av ressurskrav og mulige avhengigheter til andre aktiviteter. Alle aktiviteter krever et mannskap til å utføre dem og en lokasjon til å bli utført på. I tillegg krever noen aktiviteter kranressurser, fordi tung løfting er involvert. Mannskaps- og kranressurser er knappe, det vil si de har begrenset tilførsel.

Så langt er problemet klassifisert som et *"Resource-Constrained Project Scheduling Problem (RCPSP)"*[10], som kjennetegnes ved:

- Et sett av ressurser med en gitt kapasitet
- Et sett av ikke-forstyrrede aktiviteter som er gitt en prosseseringstid
- Et nettverk av begrensinger mellom aktiviteter
- En mengde av ressurser som er krevd av aktivitetene

Det er en mengde planleggingsproblemer som ikke kan klassifiseres under beskrivelsen av RCPSP, selv om det også er et bredt antall planleggingsproblemer som gjør det. Det er mange tilleggsbegrensninger, typisk i oljeindustrien og andre store industrier, som ikke passer inn i denne klassifiseringen. Siden målet er å generere probleminstanser med begrensninger som finnes i industrien, så må det legges til andre mer komplekse begrensinger. Et eksempel er sikkerhetsbegrensing rundt farlig arbeid, for eksempel kranbruk. I planleggingsløsninger i dag blir informasjon som sikkerhetsbegrensninger lagt til manuelt av de som planlegger aktivitetene på platformen. Ved å definere forutsetninger som aktiverer sikkerhetsbegrensninger, blir resultatet en veldefinert problembeskrivelse. En løsning til et problem $S(P_i)$ er en planlegging hvor aktiviteter er tilegnet en starttid og begrensningene er gjeldende.

3.1 Notasjoner og terminologi

En probleminstans P inneholder aktiviteter som skal gjennomføres, ressurser som er påkrevd for å gjennomføre aktivitene og begrensninger som blant annet er begrensinger mellom aktiviteter og ressursbruk. Det blir skilt mellom forskjellige typer variable som *avgjørelses-variable*, *konstanter* og *avledet-variable*. Et eksempel på en avgjørings-variabel er starttiden til en aktivitet Act_i betegnet som $v_{sta}(Act_i)$. En aktivitets varighet blir betegnet som fast og er derfor en konstant, betegnet som $c_{dur}(Act_i)$. Til slutt er det avledet-variable som for eksempel er en aktivitets sluttid, som er summen av starttiden og varigheten, som er betegnet $w_{end}(Act_i)$. Objekter som aktiviteter og ressurser er skrevet med stor bokstav.

3.2 Ressurser

En *lokasjon* $Loc_l \in Locs = \{Loc_1, \ldots, Loc_n\}$ er stedet hvor aktiviteter blir utført. Lokasjoner blir vist som ressurser, og det er med varmebegrensning en begrensning en maksimal kapasitet av hva lokasjonen tåler. Det er også begrensinger når farlig arbeid som tung løfting blir utført, da er lokasjonen utilgjengelig for alle andre aktiviteter. Når en lokasjon er stengt på grunn av kranbruk sier vi at en sikkerhetsone har blitt opprettet.

Mannskaper er ansvarlige for utførelsene av aktivitetene. Et mannskap er betegnet $Crew_j \in Crews = \{Crew_1, \ldots, Crew_n\}$. Mannskaper er vist som ressurser, og har fått varmebegrensning. Varmen blir brukt til å begrense hvor mange mannskaper som kan være på en lokasjon på samme tid.

En *kran* $Crane_k \in Cranes = \{Crane_1, \ldots, Crane_n\}$ er en potensiell ressurs for aktiviteter. Noen aktiviteter trenger kran og alle probleminstanser har et mindre antall av aktiviteter som krever kranbruk. Kraner er eneressurs som betyr at de kun kan utføre en aktivitet av gangen. En aktivitet som krever kran, spesifiserer ikke en spesifikk kran, men kun sier den trenger kran. En gyldig løsning må derfor tildele en kran til alle aktiviteter som krever kran fra et sett av kraner tilgjengelig, gitt av $v_{crane}(Act_i) \in Cranes$. Dette gjør settet av kraner til en alternativ ressurs.

Kraner har en lokasjon $c_{loc}(Crane_k) \in Locs$, og hver lokasjon kan bare ha en kran. På grunn av at tung løfting er et farlig arbeid, er kranbruk omgitt med sikkerhetssoner. Disse sikkerhetssonene er satt til både loksjonen hvor aktiviteten som krever kranbruk er utført og kranens egen lokasjon. Sikkerhetssonen som blir satt vil derfor variere ut ifra hvilken kran som er tilegnet til aktiviteten.

3.3 Aktiviteter

En *aktivitet* $Act_i \in Acts = \{Act_i, \ldots, Act_n\}$ kommer med en startvariabel, en konstant varighet og ressurskrav. Initielt er domenet til startvariabelen er $v_{sta}(Act_i) \in [0, c_{hor}(P))$, hvor horisonten, indikerer planleggingens maksimale fullføringstid, som er gitt ved $c_{hor}(P) = \sum_i c_{dur}(Act_i)$.

En aktivitet Act_i krever et mannskap $c_{crew}(Act_i) \in Crews$ for å utføre aktiviteten

og en lokasjon $c_{loc}(Act_i) \in Locs$ til å bli utført på. En aktivitet avhenger av et enkelt medlem av et mannskap og det er ikke mulig å samle ressurser for å redusere varigheten. Kraner er den siste ressursen som er tilgjengelig, men er ikke nødvendig for alle aktivitetene.

I tilegg til ressurskravene, kan en aktivitet være avhengig av andre aktiviteter, det betyr at en aktivitet ikke kan starte før en annen aktivitet er ferdig utført.

3.4 Begrensinger

Avhengigheter mellom aktiviteter er vanlig i industrien. En vedlikeholdsaktivitet kan for eksempel være avhengig av både levering av reservedeler og stillasbygging for å sikre tilgang til området hvor vedlikeholdet skal gjøres. Forholdet som viser at aktivitet $Act_{i'}$ avhenger av aktivitet Act_i er uttrykt ved følgende begrensning:

$$w_{end}(Act_i) \leq v_{sta}(Act_{i'}) \tag{1}$$

En *kummulativ ressurs begrensing* påføres alle mannskaper for å være sikkert på at den totale ressursbruken ikke overstiger tilgjengelig kapasitet. Det er utrykt ved:

$$\forall t, j : \sharp\{Act_i | t \in [(v_{sta}(Act_i), w_{end}(Act_i)) \wedge c_{crew}(Act_i) = Crew_j]\} \leq c_{cap}(Crew_j) \tag{2}$$

hvor $c_{cap}(Crew_j)$ er kapasiteten av j's mannskap.

Kraner er unikt induvidielle og er derfor modellert som et sett av eneressurser. Begrensingene tar for seg hvis to aktiviteter er tilegnet den samme kranen, så kan de ikke bli utført samtidig. Vi starter ved å definere de underliggende overlapping uttrykt som to aktiviteter overlapper i tid:

$$overlap(Act_i, Act_{i'}) \equiv \exists t : v_{sta}(Act_i), v_{sta}(Act_{i'}) \leq t < w_{end}(Act_i), w_{end}(Act_{i'}) \tag{3}$$

Den gjensidge uttelukkelsen opprettet av den eneressursbegrensningen blir da:

$$\forall i, i' \neq i : c_{crane}(Act_i) = v_{crane}(Act_{i'}) \rightarrow \neg overlap(Act_i, Act_{i'}) \tag{4}$$

for alle aktiviteter som krever kran.

Sikkerhetsbegrensningene er uttrykt i form av lokasjonen til aktiviten som krever kran og lokasjonen til den valgte kranen. Den første lokasjonen er kjent på forhånd, mens den andre avhenger av hvilken kran som blir brukt. Tilfellet at begrensingene i problemet endrer seg etter hvert som avgjørelser tas, er interessant på grunn av den tillagte kompleksiteten det medfører.

Sikkerhetsbegrensningene utelukker bruken av lokasjonen hvor en aktivitet som krever kran befinner seg:

$$\forall i, i' \neq i : c_{crane}(Act_i) \wedge c_{loc}(Act_i) = c_{loc}(Act_{i'}) \wedge \neg overlap(Act_i, Act_{i'}) \quad (5)$$

når sikkerhetsbegrensingene utelukker bruken av lokasjonen til denne krannen er gitt ved:

$$\forall i, i' \neq i : v_{crane}(Act_i) = Crane_j \wedge c_{loc}(Act_{i'}) = c_{loc}(Crane_j) \rightarrow \neg overlap(Act_i, Act_{i'}) \quad (6)$$

Så langt er formler og tekst hentet fra [15]. I det neste tillegges en ny type begrensning kalt *varmebegrensning*. Med dette, så menes ikke varme i tradisjonell forstand, men som en måte å kunne sette en verdi på et mannskap og en kapasitet på en lokasjon. Det kan være lokasjoner som av forskjellige årsaker (begrenset plass, restriksjoner til lokasjonen, etc.) ikke kan ha ubegrenset med mannskap til å jobbe der samtidig. Forskjellige mannskaper kan også ha forskjellig kapasitet ut ifra hva slags arbeid de utfører. Mannskaper som driver arbeid som sveising, kan for eksempel ha en høyere varmeverdi enn et mannskap med elektrikere. Grunnen til at sveisere kan ha en høyere varmeverdi er fordi det er en aktivitet som gjør det ugunstig å ha for mange på lokasjonen. Det kan være grunner til farlige gasser, eksplosjonsfare osv. De er uttrykt som kummulativ ressursbegrensning og er påført lokasjon for å være sikkert på at total varmebruk ikke overstiger varmekapasiteten tilgjengelig på hver lokasjon. Varmebegrensingen skal også etterhvert kunne erstatte sikkerhetsbegrensningene på for eksempel kranbruk, ved at lokasjoner med kran får en varmekapasitet og aktiviter som bruker kran bruker opp

denne varmekapasiten på lokasjonen. Den er uttrykt ved:

$$\forall t, l : \sum\{c_{heat}(Crew_j) \mid t \in [v_{sta}(Act_i), w_{end}(Act_i)) \wedge c_{crew}(Act_i) = Crew_j \wedge c_{loc}(Act_i) = Loc_l]\} \leq c_{heatcap}(Loc_l) \tag{7}$$

3.5 Målfunksjon

Målet er å minimalisere makespan $w_{ms}(P)$ eller varigheten av planleggingen, som er definert ved:

$$w_{ms}(P) = max_i\{w_{end}(A_i)\} \in [0, c_{hor}(P)] \tag{8}$$

Dette uttrykker at makespanet er likt den siste slutten eller fullføringstiden i settet av aktiviteter.

3.6 Probleminstanser

Problemene er beskrevet ved størrelsen fastsatt av det totale nummeret av aktiviteter, $\#Acts \in \{50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 5000\}$ og kraner, $\#Cranes = [2, 3]$. Det ble generert totalt 5 probleminstanser for hver av de 32 problem størrelsene, som summert opp blir 160 instanser.

Probleminstansene ble tilfeldig generert, ved å tildele mannskaper til aktiviteter, lokasjoner til aktiviteter, lokasjoner til kraner, avhengigheter mellom aktiviteter og aktiviteter som trenger kran. Når instansene ble generert, er det spesifisert at det ikke skal forekomme sirkulasjoner på aktivitetsavhengigheter og at det ikke skal være mer enn en kran på en lokasjon. Så alle 160 instansene er gyldige.

Alle problemene har 10 lokasjoner, som er redusert fra 25 i de opprinnelige probleminstansene til Bård Henning Tvedt . Med aktiviteter fordelt utover 25 lokasjoner, så ville det vært så få aktiviteter på hver lokasjon at varmebegrensingen til en lokasjon aldri ville blitt oversteget. Det er 4 forskjellige mannskaper med kapasitet $c_{cap}(Crew_j) \in [2, 3]$ tatt fra en uniform fordeling. Domenet for aktivitenes startvariabel er generelt $v_{sta}(Act_i) = [0, c_{hor}(P)]$ og de konstante varighetene $c_{dur}(Act_i)$ er tilfeldig tatt fra en uniform fordeling i området $[1, 6]$ tidssteg. Omtrent 20% av aktivitetene er tilfeldig valgt til å bruke kran

og omtrent 10% av aktivitetene er begrenset ved avhengighet til en annen aktivitet. I det første settet med probleminstanser ble mannskapers varme tilfeldig generert i området: $c_{heat}(Crew_j) \in [5, 15]$ og lokasjoners varmekapasitet ble tilfeldig generert i området: $c_{heatcap}(Loc_l \in [10, 20]$. Det ble generert et nytt sett med probleminstanser, hvor mannskapers varme er tilfeldig generert i området: $c_{heat}(Crew_j) \in [1, 5]$, mens lokasjoners varmekapasitet er tilfeldig generert i området: $c_{heatcap}(Loc_l) \in [6, 10]$.

4 Begrensningsprogrammering

I denne delen blir det gitt en innføring om begrensningsprogrammering og utfordringer som kommer med begrensningsprogrammering. Det vil også bli sett litt på verktøy som eksisterer idag for å bruke begrensningsprogrammering samt noen enkle verktøy for enkel planlegging av aktiviteter og ressurser. Tilslutt i denne delen er det en kort beskrivelse av verktøyene som er brukt i dette prosjektet.

4.1 Kort om begrensningsprogrammering

Begrensningsprogrammering er en programmeringsparadigme hvor relasjoner mellom variable blir satt i form av begrensninger. Begrensninger er en form for deklarativ programmering, som skiller seg fra den mer vanlige imperativ programmeringsspråk[1] ved at løsningen blir til ved å tilfredsstille begrensningene. Det er forskjellige områder i begrensningsprogrammering som "Constraint Satisfaction problems"og planleggingsproblemer. Det mest kjente planleggingsproblemet er "Job Shop Scheduling"[5].

4.2 Utfordringer med begrensningsprogrammering

Systemer for begrensings-logikk programmeringssystemer som sammenliggnes med begrensningsløsningssystemer, er ofte ytelsen bedre i begrensings-logikk programmeringssystemer. Ytelsen er likevel ikke like god som mer tradisjonelle imperative

[1]Imperativ programmeringsspråk har sekvenser med instanser som blir utført.

programmeringsspråk, og spesielt gjelder det innen tallmessige utregninger. For å løse dette, er det mulig å utvikle en avansert kompilator som sjekker de tilfellene hvor det ikke er behov for begrensingsløning, og da kompilerer disse på mest mulig effektive måte [1].

Begrensningsprogrammeringssystemer har ofte en svakhet når det kommer til feilsøking (engelsk: debugging). Uten tilstrekkelige måter å kunne være sikkert på riktigheten og muligheter å sjekke ytelse i programmer. En måte å løse utfordringene med manglende feilsøkingsmuligheter er å bruke påstander . Ved bruk av påstander kan det opprettes pre- og postforhold. Preforhold skal resultere i en boolsk verdi (sann eller falsk) og metoden blir ikke utført hvis ikke preforholdet er sant. Postforhold til en metode beskriver hva som skal være oppnåd med metoden. Postforold er også en boolsk verdi. Både pre- og postforhold blir skrevet for å "evaluereen metode, enten før metoden blir utført eller hva metoden har oppnåd. Påstander kan bli sjekket ved kompilering eller når programmet kjører. Det er også mulig å genere påstander av kompilatoren, som utvikleren kan sjekke om det eksisterer høynivåfeil [1].

4.3 Begrensingsprogrammeringsverktøy idag

Det finnes idag flere forskjellige verktøy for begrensningsprogrammering, både i form av egne programmeringsspråk som er skreddersydd for begrensningsprogrammering og biblioteker til godt kjente programmeringsspråk som Java og C++. I begge disse kategoriene så finnes det løsninger som er kommersielle og med åpen kildekode. Noen eksempler på egne programmeringsspråk for begrensningsprogrammering er Prolog og Comet. Sistnevnte er et programmeringspråk for begrensningsprogrammering med lokalt søk og er en kommersiell løsning. Eksempler på begrensningsprogrammeringsbiblotek så er det IBM ILOG CP.

Planlegging er ofte tett knyttet opp mot prosjektstyring og prosjektstyringsverktøy finnes det veldig mange av etterhvert [6]. Både programmer du har lokalt på maskinen og også webbaserte tjenester. Fellesnevneren for veldig mange av disse tjeneste, enten de er lokalt eller webbaserte er at de skal hjelpe til med alt fra prosjektplanlegging, dokumentdeling, oversikt over oppgaver, oversikt over frister, møteplanlegging og mye mer. Denne typen programvare er ofte gjort ganske enkle

å bruke. I samme kategori er det også noen programmer som gjør det mulig å definere aktiviteter (ofte kalt oppgaver) og knytte ressurser til aktivitetene.

Et eksempel på et slikt program er Microsoft Project (MSProject). Dette programmet har et grafisk grensesnitt som andre programmer i Office-pakken til Microsoft og er et enkelt program å bruke. MSProject gir muligheter til manuell og automatisk planlegging[4]. I MSProject er det mulig å legge til begrensninger, men er begrenset til aktiviteters starttidspunkt. Aktiviteter kan tillegges begrensinger ut ifra om aktiviteten skal planlegges tidlig eller sent i prosessen eller på et gitt tidspunkt [3]. Dette prosjektet har flere ressurser (lokasjoner, mannskaper, kraner, osv.), begrensinger og sikkerhetsbegrensinger. Når problemet blir såpass komplekst, vil ikke et enkelt program som MSProject fungere til formålet.

Til noen av de mest kjente og brukte begrensingsprogrammeringsprobleme (for eksempel "Job Shop Scheduling Problem") finnes det noen systemer som løser de. "Exact JobBOSS Softwareer et eksempel på et slikt system, hvor det er mulig å utføre planleggings og produksjons gjennomføring. Dette systemet har funksjonalitet å opprette aktiviteter, planlegge aktiviteter, materialkravplanlegging osv [2].

4.4 Verktøy brukt i prosjektet

De følgende verktøyene og teknologiene utviklet av IBM var brukt for å gjennomføre formålet med prosjektet.

4.4.1 Kort om Concert

Concert er et C++ bibliotek med funksjoner som gir mulighet til å designe modeller av problemer innen matematisk programmering og innen begrensningsprogrammering. Det er ikke noe eget programmeringsspråk, gir muligheter til å bruke datastrukturer og kontrollstrukturer som allerede finnes i C++. Igjen så gir det gode muligheter til å integrere Concert i allerede eksisterende løsninger og systemer. Alle navn på typer, klasser og funksjoner har prefiksen Ilo.

De enkleste klassene (eks. IloNumVar og IloConstraint) i Concert har også tilhørende en klasse med matriser hvor matrisen er instanser av den enkle klassen. Et ek-

sempel på det er IloConstraintArray som er instanser av klassen IloConstraint [13].

4.4.2 Kort om Solver

IBM ILOG Solver er et C++ bibliotek utviklet for å løse komplekse kombinatoriske problemer innen forskjellige områder. Eksempler på anvendelsesområder kan være produksjonsplanlegging, ressurstildeling, timeplanplanlegging, personellplanlegging, osv. Solver er basert på Concert. Som i Concert, så er heller ikke Solver noe eget programmeringsspråk, som gir mulighetene til å bruke egenskapene til C++.

Solver bruker begrensningsprogrammering for å finne løsninger til optimaliseringsproblemer. Det å finne løsninger med Solver er basert på tre steg: beskrive, modellere og løse. De tre stegene nærmere forklart følger:

Først må problemet beskrives i programmeringsspråket som brukes.

Det andre steget er å bruke Concertklassene for å opprette en modell av problemet. Modellen blir da satt sammen av besluttningsvariable og begrensninger. Besluttningsvariablene er den ukjente informasjonen i problemet som skal løses. Alle besluttningsvariablene har et domene med mulige verdier. Begrensningene setter grensene for kombinasjonene av verdier for de besluttingsvariablene.

Det siste steget er å bruke Solver for å løse problemet. Det inneholder å finne verdier for alle besluttingsvariablene samt ikke bryte noen av de definerte begrensningene og dermed enten maksimere eller minimere målet, hvis det er et mål inkludert i modellen. Solver ser etter løsninger i et søkeområdet. Søkeområdet er alle mulige kombinasjoners av verdier. I dette prosjektet brukes det generelle søkemål i Solver for å tildele alternative ressurser, rangere eneressurser og for å sette tiden på aktiviteter lengst mulig frem i løsningen.

- IloAssignAlternative, er søkemålet i Solver for å tildele alternative ressurser til aktiviteter.
- IloRankForward, dette søkemålet i Solver rangerer ressurser av typen eneressurs.

- IloSetTimesForward, med dette søkemålet vil Solver tildele starttid til alle aktivitetene først.

En kombinasjon av disse søkemålene vil kunne gi forskjellige løsninger.[12]

4.4.3 Kort om Scheduler

[11] IBM ILOG Scheduler hjelper med å utvikle problemløsnings-applikasjoner som krever behandling av ressurser fordelt på tid. Scheduler er et C++ bibliotek som baserer seg på Solver, og som Solver, så gir det alle mulighetene med objektorientering og begrensningsprogrammering. Scheduler har spesifisert funksjonalitet på å løse problemer innen planlegging og ressurstildeling.

I dette prosjektet er det i Scheduler brukt aktiviteter, kummulative ressurser, eneressurser og alternative ressurser. Aktiviteter er modellert ved bruk av IloActivity, hvor varigheten til aktiviteten blir satt som en parameter og navnet på aktiviteten blir også lagt til. I og med at alle aktivitetene har en lokasjon, brukes setIntParameter på aktiviteten. Det blir også lagt inn avhengigheter mellom aktiviteter som er avhengig av hverandre. Kummulative ressurser blir brukt når det er begrenset med kapasitet på ressursen, de er også ofte kjent som gjennbrukbare ressurser som er representert ved en tidsfunksjon. I Scheduler er kummulative ressurser modellert med IloDiscreteResource. Da blir det opprettet en IloDiscreteResourceArray hvor de kummulative ressursene blir plassert og så blir den matrisen iterert over og ressursene blir lagt til aktivitetene. Eneressurser er i Scheduler ressurser med kapasitet en, de kan kun brukes av en aktivitet på samme tid.

5 Metode

I denne delen, blir verktøy og teknologier som er brukt i prosjektet beskrevet. I tilegg vil forskingsmetoder som er brukt bli beskrevet og beskrivelse av strategiene for evaluering av løsningene.

5.1 Forskningsmetoder

Forskningsmetoden som er brukt i prosjektet er å eksperimentere med implementasjonen av ressursene og løsningsstrategien. Samt å gjøre en kvalitativ vurdering av løsningene og vurdere makespan mot teoretisk- øvre og nedregrense som blir forklart under. Det blir i denne delen gått gjennom implementeringsprosessen og evalueringsprosessen som er brukt i prosjektet.

5.1.1 Implementeringsprosessen

Listing 1: Opprinnelig kranfordeling

```
IloRandom rnd = IloRandom(env, (IloInt)time(NULL));
int nbActivities = dModel.getActNames()->size();
for (int i = 0; i < nbActivities; i++)
{
        if (dModel.getRequireCrane()->at(i))
        {
                IloInt cIndex = rnd.getInt(nbCranes);
                model.add(altConstraint.select(cranes[cIndex]));
                altConstraint.setSelected(cranes[cIndex]);
                index=index+1;
        }
}
```

Måten kraner blir tildelt til aktiviteter som krever kran er endret fra løsningen til Bård Henning Tvedt . Kraner er fortsatt modellert som eneressurser, men tildelingen av kraner i den opprinnelige løsningen til Bård Henning Tvedt tildelte kraner tilfeldig ved å bruke IloRandom (listing 1). Denne metoden plukket en tildeldig kran fra tilgjengelige kraner i problemstillingen og tildelte den til ressursen. Da varierte løsningene på hvor god den tilfeldig tildelingen av kran var.

Listing 2: Endret kranfordeling

```
int nbActivities = dModel.getActNames()->size();

for (int i = 0; i < nbActivities; i++)
```

```
{
        if (dModel.getRequireCrane()->at(i))
        {
                IloAltResConstraint altConstraint
                        = activities[i].requires(altCranes);
                model.add(altConstraint);
                model.add(activities[i].requires(cumulativeCrane));
        }
}
```

I dette prosjektet er kranbegrensningen modellert med alternative ressurser (listing 2). Ved å modellere kranbegrensningen med alternative ressurser kan krantildelingen endre underveis i søkestrategien, ved å bruke søkemålet IloAssignAlternatives.

Problemet med å finne ut om en RCPSP løsning i tillegg til sikkerhetsbegrensningene med makespan mindre enn en gitt frist er NP-hard. Dette betyr at det utvidede problemet med varmeressursen også må være NP-hard og derfor en optimal løsning med til og med de enkleste formene av problemet er ikke garantert innen polynomisk tid. To forskjellige løsningsstrategier er testet. Begge løsningsstrategiene er implementert i Solver og Scheduler biblotekene. I begge løsningsstrategiene brukes standard søkemål i Solver.

Den første løsningsstrategien (LS1) bruker IloAssignAlternatives, som blir brukt til å tildele kraner til aktivitetene ved å tilegne en mulig ressurs til en alternativ ressurs, her kran. Det å rangering er mulig for alle ressursbegrensingene. Når begrensninger blir rankert, blir aktiviteten ressursen tilhører flyttet først av alle aktivitetene som ikke har blitt rangert. Søkemålet som kommer nå er da IloRankForward, som rangerer alle ressursenebegrensningene av typen eneressurs i modellen. Tilslutt blir IloSetTimesForward brukt, denne tillegner starttid til aktiviteter i planleggingen. Ved å velge en kranfordeling først, så løses resten av problemet rundt kranfordelingen. Da blir søketreet for stort til at valget av kranfordeling kan gjøres på nytt på en god måte. Det fører til en forventning at disse løsningene vil være noe dårligere når det gjelder makespan.

Den andre løsningsstrategien (LS2) tillegner starttid til alle aktivitetene først, ved å bruke IloSetTimesForward. Når alle aktivitetene har fått tildelt en starttid brukes IloAssignAlternatived for å fordele kran til aktiviteter som krever det. Tilslutt brukes IloRankForward som rangerer eneressurser og når en ressurs blir rangert, kan aktiviteten til denne ressursen bli rangert først av de aktivitetene som ikke har blitt rangert. Med denne løsningsstrategien vil starttidspunktene for aktivitetene settes først, det kan føre til ugyldige løsninger når kranfordelingen blir valg på grunn av sikkerhetssonene.

5.1.2 Evaluering av prosessen

Prosessen med eksperimenteringen av implementasjonen blir evaluert ved å undersøke makespan opp mot teoretisk øvregrense (se 5.2.1) og teoretisk nedregrense (se 5.2.2) i både løsningene uten varmebegrensning og med varmebegrensning. Løsningene fra prosessen med og uten varmebegrensningen vil også bli evaluert opp mot hverandre. Dette innebærer bruk av kvantitative metoder. Løsningene med med 10 lokasjoner vil også bli evaluert opp mot løsningene med 25 lokasjoner, men da vil gjennomsnitsverdien 13 bli brukt.

Forskningsmetoden vil bli evaluert ved å bruke genererte probleminstanser, som er generert av et eksternt program. Probleminstansene som genereres kan bestemmes hvor mange av de forskjellige ressursene som skal være med i probleminstansene. I dette prosjektet er det et sprang på 50 - 5000 aktiviteter som implementasjon blir evaluert på. I dette prosjektet er det generert forhåndsdefinerte problemer (her: probleminstanser) og håper resultatet kan blir overført til applikasjoner som krever denne forskningen.

Løsningene blir evaluert av et eksternt program, som sjekker begrensningene som avhengigheter, varme og sikkerhetsbegrensninger. Det programmet er utviklet i *Java* og leser inn en løsning og tilhørende probleminstans. Det blir opprettet en vektor med objekter som aktiviteter, kraner og mannskaper. På den måten er all informasjon om de forskjellige objektene samlet, som gjør det enkelt å sjekke begrensninger som avhangigheter, varme og sikkerhetsbegrensning. Løsningene har i tillegg varifisert løsningene for alle begrensninger unntatt varmebegrensningen med et eget program.

5.2 Evalueringsstrategi

For å evaluere kvaliteten på løsningene, er teoretisk- øvregrense og nedregrense for makespan blir kalkulert.

5.2.1 Teoretisk øvregrense

Teoretisk øvregrense for makespan er

$$c_{ub,ms}(P) = c_{hor}(P) = \sum_i c_{dur}(Act_i) \tag{9}$$

som indikerer at i det verste tilfelle blir alle aktivitetene utført etter hverandre, en om gangen.

5.2.2 Teoretisk nedregrense

En teoretisk nedregrense er kalkulert basert på ressurstilgjengeligheten for den mest begrensede mannskapet. Resultatløsningen blir kanskje ikke gyldig for hele problemet, men er komprimert tett sammen for mannskapet og utnytte hver eneste mannskapstime.

$$c_{load}(Crew_j) = \sum_{c_{crew}(Act_i)=Crew_j} c_{dur}(Act_i) \tag{10}$$

$$c_{reload}(Crew_j) = \frac{c_{load}(Crew_j)}{c_{cap}(Crew_j)} \tag{11}$$

$$c_{lb,ms}(P) = max_j\{c_{reload}(Crew_j)\} \tag{12}$$

6 Resultater

Eksperimenteringen er utført på en Macbook Air med 1.8 GHz Intel Core i7 prosessor og 4 GB 1333 MHz DDR3 minne. Det er totalt sett 160 benchmarksett som implementasjonene er evaluert på. Hvor mange det ble funnet en løsning på,

varierte avhengig av om tilleggsressursene var med eller ikke og tidsgrensen som var satt.

Tabell 1: Relativ optimalitets indeks w_{rq} for de forskjellige modellene

Modell	**2 kraner**				**3 kraner**				**Alle**	
$\sharp Act(\sharp P)$	$< 100(25)$		$\geq 100(55)$		$< 100(25)$		$\geq 100(55)$		(160)	
Modell	w_{rq}	$\%^{(1)}$	w_{rq}	$\%^{(1)}$	w_{rq}	$\%^{(1)}$	w_{rq}	$\%^{(1)}$	w_{rq}	$\%^{(1)}$
$LS1\sharp1^{(2)}$	1.578	100	1.905	100	1.593	100	1.731	100	1.745	100
$LS1\sharp2^{(3)}$	1.672	100	1.961	100	1.711	100	1.766	100	1.810	100
$LS2\sharp1^{(2)}$	1.144	84	1.160	27.3	1.60	68	1.042	10.91	1.142	36.88
$LS2\sharp2^{(3)}$	1.256	76	1.313	30.91	1.243	72	1.041	18.18	1.232	40

$^{(1)}$ prosentandel løste probleminstanser $^{(2)}$ $\sharp1$ er løsninger uten varmebegrensning $^{(3)}$ $\sharp2$ er løsninger med varmebegrensning

En måling av relativ kvalitet er brukt for å evaluere resultatene fra forskjellige strategier. Den avledede variabelen w_{rq} er gitt ved (13).

$$w_{rq} = \frac{1}{|P_{sol}|} \sum_{P \in P_{sol}} \frac{w_{ms}(P)}{c_{lb,ms}(P)} \tag{13}$$

P_{sol} er det settet med probleminstanser som er løst ved hver enkelt løsningsstrategi. Verdiene av P_{sol} varierer fra løsningsstrategi til løsningsstrategi og disse gjennomsnittene er derfor ikke helt sammenlignbare, men de vil gi en indikasjon på kvaliteten på løsningen. w_{rq} skiller ikke på strategier som feiler med å finne løsninger, men hvor robust løsningen er vil antallet løste probleminstanser indikere. Resultatene er summert opp i tabell 1.

Teoretisk nedregrense er basert på mannskaper som er den mest begrensende ressursen. Det er ikke sikkert det finnes gyldige løsninger er ikke sikkert finnes, men i og med den er beregnet ut fra den mest begrensede ressursen er det fornufig benchmark å sammenligne løsningene med denne.

6.1 Uten varmebegrensning

Løsningene som er kommentert her er fra det andre genererte benchmarksettet, den første løsningsstrategien uten varmebegrensing finner løsninger på alle 160 benchmarksettene. Makespan på er nærmest teoretisk nedregrense med to kraner

og under 100 aktiviteter, hvor det i gjennomsnitt er 57.8% over teoretisk nedregrense. Lengst unna teoretisk nedregrense er banchmarksettene med 2 kraner og over 100 aktiviteter, hvor makespan ligger i gjennomsnitt 90.5% over teoretisk nedregrense. Alle benchmarksettene med den første løsningsstrategien er i gjennomsnittet 74.5% over teoretisk nedregrense.

Ut ifra figur 4 er det lokasjon som er den mest begrensede ressursen. På lokasjon 2 er kapasiteten helt brukt opp gjennom hele makespan og på lokasjon 2 finnes også kran 1.

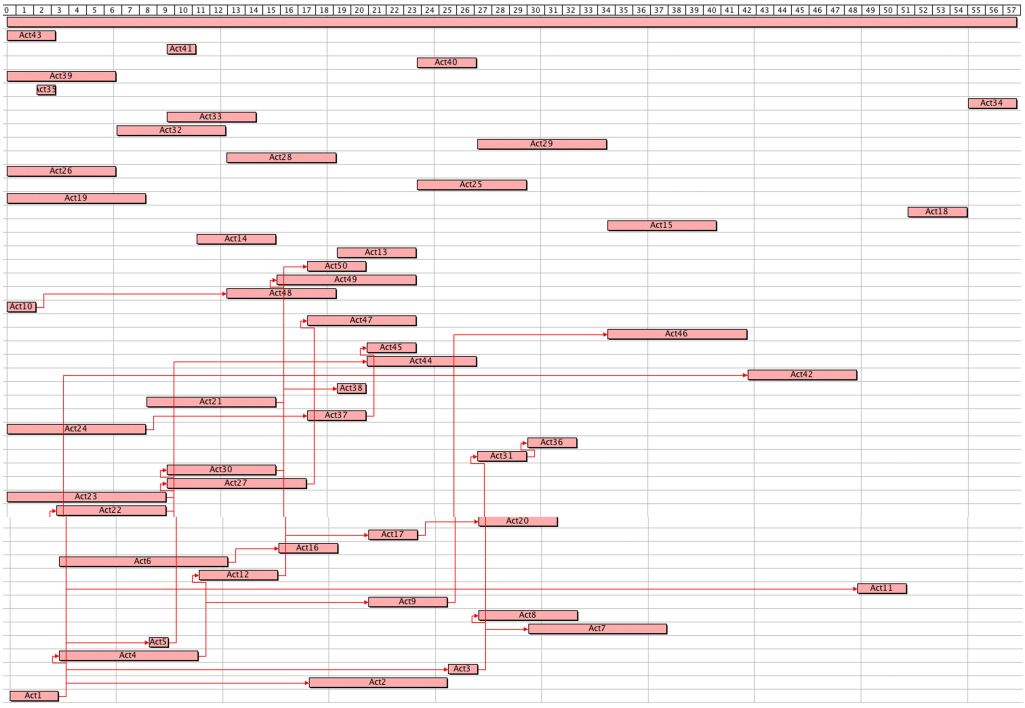
De beste løsningene uten varmebegrensning er med den andre løsningsstrategien, hvor det i gjennomsnitt er 4.2% over teoretisk nedregrense med 3 kraner og over 100 aktiviteter. Da finner den kun løsning på 10.91% av probleminstansene. Med 3 kraner og under 100 aktiviteter er de dårligste løsningene med den andre løsningsstrategien, hvor det i gjennomsnitt er 60% over teoretisk nedregrense. Den finner løsninger på 68% av probleminstansene. Totalt sett med den andre løsningstrategien, er makespan 14.2% over teoretisk nedregrense og den finner løsninger på 36.88% av probleminstansene.

Tidene som Scheduler bruker for å finne løsninger på probleminstansene er sammenliggnet med å se på tidene for samme antall aktiviteter og se på de med to og tre kraner. Det er ingen klare skiller om probleminstansene inneholder to eller tre kraner. Tidene for å finne løsninger varierer fra 0 til over 100 sekunder og det er ikke tilfelle de probleminstansene med flest aktiviteter som bruker lengst tid. Tiden på å finne en løsning variere også stort innenfor probleminstansene av et antall aktiviteter.

6.2 Med varmebegrensning

I den første probleminstansen var det kun noen løsninger under 100 aktiviteter. Probleminstansene over 100 aktiviteter hadde ingen løsninger, annet enn et av probleminstansene på 900 aktiviteter.

Med det andre genererte probleminstansene ble det flere løsninger, med den første løsningsstrategien med varmebegrensning gir løsninger på alle probleminstansene, men makespan er ganske langt unna teoretisk nedregrense. Med denne

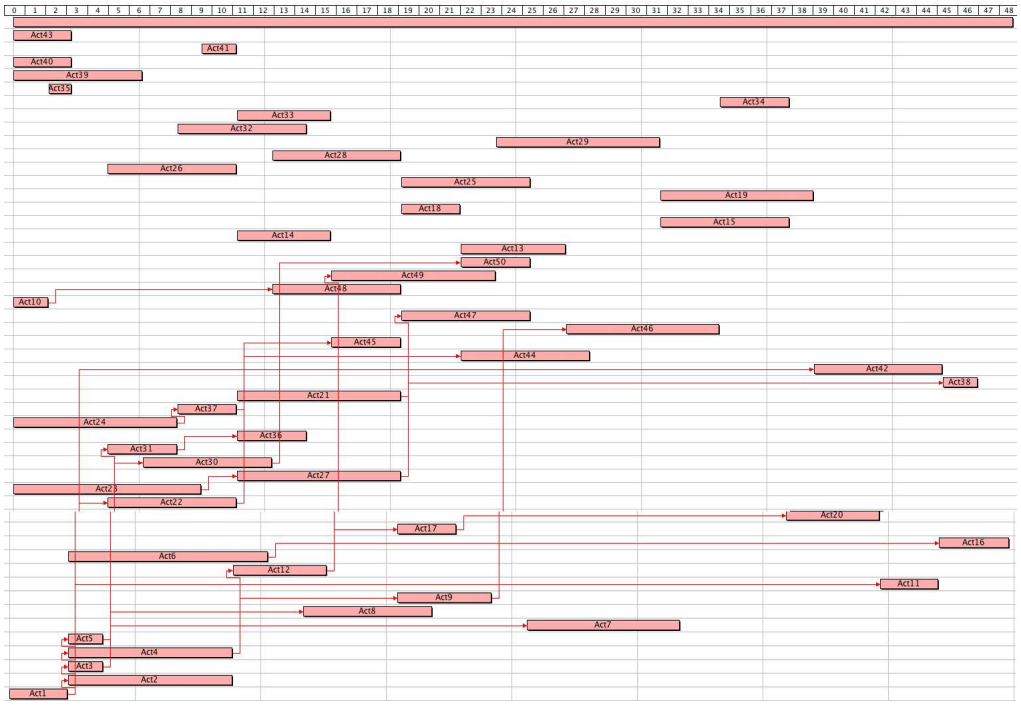


Figur 1: Gantskjema på Act50Loc10Crew5Crane2 med varmebegrensning

løsningstratteien, er det probleminstansene med to kraner og under 100 aktiviteter, hvor gjennomsnittsverdien er nærmest den teoretiske nedregrensen med 67.2% over. Probleminstansene med to kraner og over 100 aktiviteter er i gjennomsnitt 96.1% over teoretisk nedregrense. Totalt sett for den første løsningstrategien er gjennomsnittsverdien 81.0% over teoretisk nedregrense. Figur 1 viser en av løsningene med 50 aktiviteter og 2 kraner.

Ut ifra figur 3 er lokasjon tydelig den mest begrensende ressursen. Grunnen til dette er fodi den er helt fyllt opp uten at det er noen ledig kapasitet. Lokasjonen som er helt fyllt opp er lokasjon 2 og på denne lokasjonen befinner også kran 1 seg. Disse to figurene viser ressursene på probleminstansene med 50 aktiviteter og 2 kraner, med og uten varmebegrensning. På figur 5 hvor det er 3 kraner er ikke lokasjon den mest begrensende, men mannskap er her den mest begrensende.

Med den andre løsningsstrategien finner den bedre løsninger, men generelt sett færre av probleminstansene blir løst. Den beste løsningen her har gjenomsnittet 4.1% over teoretisk nedregrense. Den dårligste løsningen ligger på 31.3% over



Figur 2: Gantskjema på Act50Loc10Crew5Crane2 med varmebegrensning

teoretisk nedregrense. Der har 18.18% og 30.91% av probleminstansene blitt løst. Totalt sett med den andre løsningstrategien ligger det 23.2% over teoretisk nedregrense og 40% av probleminstansene er løst med denne løsningsstrategien. Figur 2 viser en av løsningene med 50 aktiviteter og 2 kraner.

Tidene Scheduler bruker for å finne løsninger på probleminstansene er sammenliggnet med å se på de aktivitene som har samme antall aktiviteter og se de opp mot hverandre med to og tre kraner. Det er ikke noe klart mønster i utførelsestidene som kan si noe om Scheduler bruker lengere eller kortere tid med to eller tre kraner. Tidene varierer fra 0 sekunder til flere hundre sekunder. Det er ikke nødvenndigvis de probleminstansene med flest aktiviteter som bruker lengst tid. Innenfor et antall aktiviteter kan de forskjellige probleminstansene variere fra 0 til over 100 sekunder.

7 Evaluering

I denne delen blir probleminstanser, løsningsstrategier og løsninger evaluert. Denne delen blir avsluttet med å evaluere løsningene funnet i dette prosjektet opp mot løsningene funnet i artikkelen til Bård Henning Tvedt .

7.1 Probleminstanser

Probleminstansene har fra 50-5000 aktiviteter, mens antall lokasjoner og mannskaper i de generert probleminstansene er henholdsvis 10 og 5. Det ble eksperimentert litt med mannskapets varme og lokasjoners varmekapsitet og i det første settet med probleminstanser var det overlapp mellom mannskapers varme og en lokasjons varmekapasitet. Dette førte til veldig få løsninger når varmekapasiteten ble tatt med. Det ble funnet løsninger på noen av probleminstansene under 100 aktiviteter, mens over 100 aktiviteter var det en løsning. Det er grunn til å tro dette var tilfeldig, ut ifra at varme og varmekapasitet blir tilfeldig plassert innenfor et gitt område. Probleminstansene hvor varme og varmekapasitet ikke er overlappende har mange flere løsninger og med den første løsningsstrateien er det også løsning på alle probleminstansene, uavhengig om de er kjørt med eller uten varmebegrensing. I løsningene er det probleminstansene med under 100 aktiviteter som har makespan nærmest teoretisk nedregrense, uanhengig av løsningsstrategi. Ut ifra det Taillard [17] kom frem til er da probleminstansene over 100 aktiviteter vanskeligere å løse i og med at makespan er lenger unna teoretisk nedregrense.

7.2 Løsningsstrategier

Ut ifra tabell 1 er det et ganske klart bilde av de sterke og svake sidene til de to løsningsstrategiene som er brukt. Den ene løsningsstrategien finner flest løsninger, mens den andre løsningsstrategien finner løsninger som er nærmest teoretisk nedregrense.

Den første løsningsstrategien (LS1) finner løsninger på alle de 160 benchmarksettene, men løsningene er i gjennomsnitt 74.5% og 81.0% over teoretisk nedre-

grense. Det er løsninger uten varmebegrensningen som er nærmest den teoretisk nedregrensen, både med 2 og 3 kraner.

Den andre løsningsstrategien (LS2) finner endel færre løsninger, henholdsvis 36.88% og 40%. Løsningene er vesentlig nærmere den teoretiske nedregrensen, hvor 4.1% over teoretisk nedregrense på det nærmeste og 60%.

7.3 Resultater

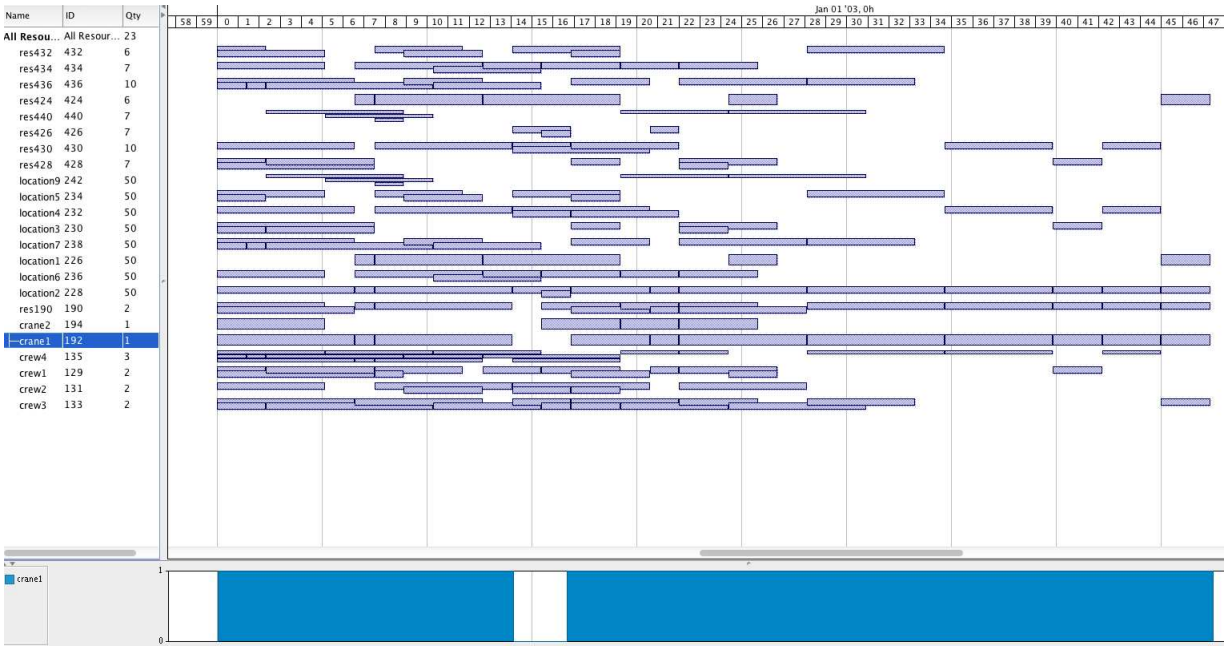
I dette prosjektet er en eksisterende løsning utvidet med en ressurs kalt varmeressurs. For å evaluere resultatene fra den utvidete løsningen, kommer det en evaluering av den eksisterende løsningen med 25 lokasjoner sett opp mot den utvidede løsningen med 10 lokasjoner. Evalueringen innebærer også å analysere effekten av varmebegrensingen, som er gjort ved å sammenligne resultatene med 10 lokasjoner med og uten varmebegrensing. Evalueringen av resultatene baserer seg på gjennomsnittsverdiene i tabell 1. En tilsvarende oppsummering av løsningene eksisterer også for 25 lokasjoner [15]. Hensikten med å sammenligne resultatene fra den opprinnelige løsningen med 25 lokasjoner og 10 lokasjoner er å analysere effekten av antall lokasjoner.

7.3.1 10 lokasjoner

Løsningene er i ganske god overomstemmelse med forventningene i seksjon 5.1.1. Med løsningsstrategi 1 hvor kranfordelingen ble valgt først var gjennomsnittlig makespan dårligere enn med løsningsstrategi 2.

I figur 3 er en av probleminstansene med to kraner kjørt med varmebegrensing. Ut ifra denne grafiske fremstilling er det kran som er den mest begrensende ressursen. På figur 4 er det også kran som er den mest begrensende ressursen. Når ser på figur 5 hvor det er tre kraner, så er det ikke lengere kran som er den mest begrensende ressursen. Her er det mannskap som over tid bruker hele kapasiteten.

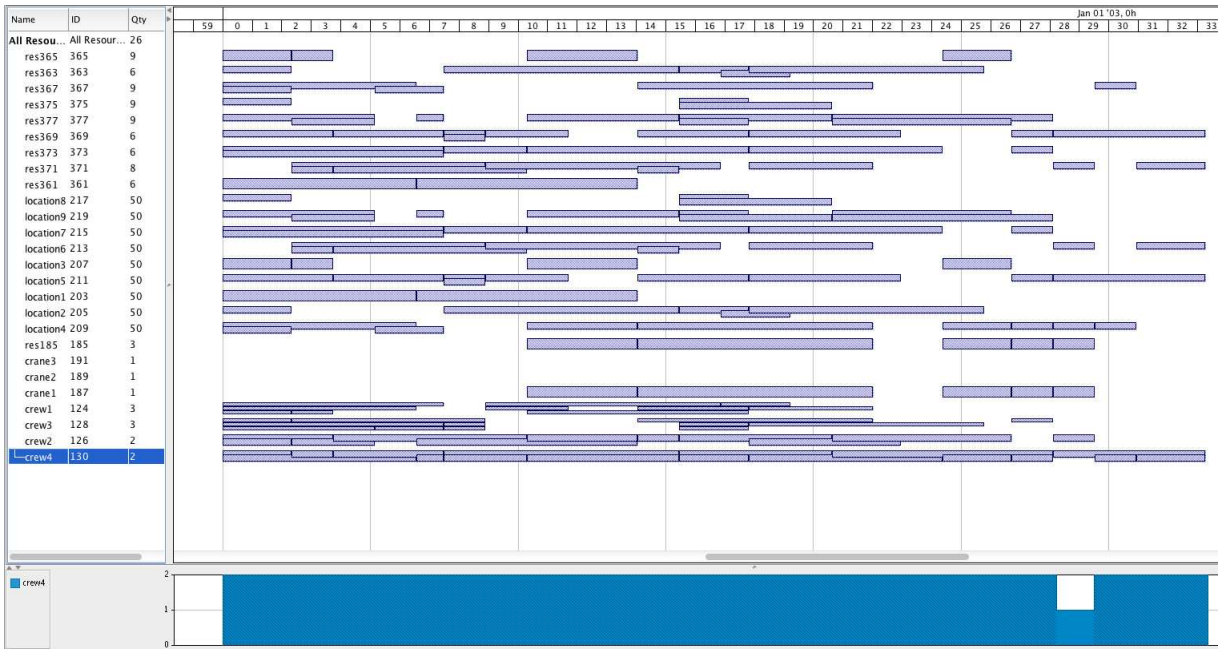
$$c_{load}(Crane_k) = \sum_{c_{Crane}(Act_i)=Crane_k} c_{dur}(Act_i) \tag{14}$$



Figur 3: Ressursoversikt på Act50Loc10Crew5Crane2 med varmebegrensning

Figur 4: Ressursoversikt på Act50Loc10Crew5Crane2 uten varmebegrensning

Ved å sammenligne om verdier fra (14) er mindre eller større enn (11), gir det et uttryk for hvor sterk eller svak kranressursen er. Hvis (14) er mindre vil det si at kranressursen er svak. Ut ifra dette er det en signifikant forskjell på to og tre



Figur 5: Ressursoversikt på Act50Loc10Crew5Crane3 med varmebegrensning

kraner. Ved to kraner er kranressursen veldig sterke, mens kranressursen ikke er fullt så sterke med tre kraner.

Når sammenligner LS1 løses alle probleminstanser både med og uten varme. Den gjennomsnittlige makespanen er litt bedre uten varmebegrensning enn med varmebegrensing, men forskjellen er ikke så stor. Totalt sett er den 74.5% over teoretisk nedregrense uten varme, mens den 81.0% over teoretisk nedregrense med varme. Tiden det tar og løse probleminstansene er også veldig lik uten noe spesielt mønster både med og uten varmebegrensning. Det er derfor litt bedre løsninger uten varmebegrensning.

Sammenliggnes LS2 så løser verken programmet med eller uten varme alle probleminstansene. Her løser programmet uten varme flest probleminstanser når antall aktiviteter er under 100, men forskjellen er ikke stor. Totalt sett løser programmet med varmebegrensning 40% av probleminstansene, mens uten varmebegrensing løser 36.88% av probleminstansne. Selve løsningene i forhold til teoretisk nedregrense så er den gjennomsnittlige makespanen bedre uten varmebegrensing, hvor den ligger 14.2% over og med varmebegrensing ligger den 23.2% over.

Det var forventet at LS1 skulle ha dårligere makespan enn LS2. I LS1 tildeles kra-

ner først i søket i Solver, mens i LS2 settes starttiden til aktivitetene først.

Den teoretiske nedregrensen som er benyttet i dette prosjektet er basert på mannskap, men med 2 kraner er det tydelig lokasjon som er den mest begrensende ressursen. Ved å se litt nærmere på lokasjonen som er mest begrenset finner man at det er samme lokasjon som kran 1 er lokaslisert på. Fra målingene av kranstyrke er det funnet ut at probleminstansene med 2 kraner er der hvor kran er sterkest og det er kran 1 som er mest begrenset i forhold til de to kranene som er i probleminstansene. Det at lokasjonen hvor kran 1 da er lokalisert blir den som er mest begrenset er ganske naturlig. Når det er sett på figur 5 så er ikke lokasjon den mest begrensende lengere, men her er det mannskap som er den mest begrensende ressursen. Når det ble sjekket kranstyrken på probleminstansene med 3 kraner var disse vesentlig svakere enn probleminstansene med 2 kraner. Dette underbygger grunnen til at lokasjon er den mest begrensende ressursen i probleminstansene med 2 kraner.

7.3.2 10 lokasjoner mot 25 lokasjoner

For å sammenligne løsningene i dette prosjektet med 10 lokasjoner, med løsningene til Bård Henning Tvedt med 25 lokasjoner, tabell 2 hentet fra [15]. Begge løsningene

Tabell 2: Relativ optimalitets indeks w_{rq} for de forskjellige modellene

Modell	**2 kraner**				**3 kraner**				**Alle**	
$\sharp Act(\sharp P)$	$< 100(25)$		$> 100(23)$		$< 100(28)$		$> 100(23)$		(99)	
Modell	w_{rq}	$\%^{(2)}$	w_{rq}	$\%^{(2)}$	w_{rq}	$\%^{(2)}$	w_{rq}	$\%^{(2)}$	w_{rq}	$\%^{(2)}$
Greedy	1.311	100	1.524	100	1.276	100	1.393	100	1.370	100
Inferred	1.068	100	1.063	35	1.017	32	1.000	4	1.055	43
Under$^{(1)}$	1.032	100	1.014	100	1.015	100	1.002	96	1.016	99
Over$\sharp$1	1.143	100	1.076	100	1.176	100	1.037	91	1.114	98
Over$\sharp$2	1.104	100	1.047	96	1.068	96	1.005	65	1.063	90

$^{(1)}$ Denne modellen garanterer ikke gyldige løsninger. $^{(2)}$ prosentandel løste instanser

inneholder modeller som klarer å løse alle probleminstansene. I Greedy modellen løses alle probleminstansene, men løsningene har en gjennomsnittlig makespan godt over teoretisk nedregrense. Det samme gjelder løsningene i dette prosjektet, hvor det er løsninger på alle probleminstansene når kranfordelingen blir valgt

først (LS1). Her er også det løsninger hvor gjennomsnittlig makespan ligger over teoretisk nedregrense. Med Greedy ligger det totalt 37% over teoretisk nedregrense, mens med løsningsstrategi 1 i dette prosjektet ligger det henholdsvis 74.5% og 81.0% over teoretisk nedregrense.

I løsningene til Bård Henning Tvedt er det den underbegrensede modellen som gir de beste løsningene med en gjennomsnittlig makespan 1.6% over teoretisk nedregrense. Denne modellen kan sees i sammenheng med løsningene i dette prosjektet som er løst uten varmebegrensing. Løsningene uten varmebegrensing i dette prosjektet gir bedre løsninger med begge løsningsstrategiene. Med løsningsstrategi 1 er den gjennomsnittlige makespanen 74.5% over teoretisk nedregrense, mens for løsningstrategi 2 er den 14.2% over teoretisk nedregrense.

8 Relatert arbeid

Nuijten og Pape har brukt ILOG Scheduler til å løse "Job Shop Schedulingproblemet og det er et problem som er NP-hard. Når problemer blir løst med Scheduler blir alle starttidene til aktivitetene, i en mulig løsning, som ikke er bevist om er en gyldig løsning tatt vare på i søkefasen. Begrensingsbasert planlegging blir ofte brukt for å redusere beregningstiden som trengs. Begrensingene reduserer domenet av variable. I industrien er ofte planleggingsproblemmer uført i dynamiske miljøer, som vil øke behovet for reaktive planleggingsalgoritmer. En mulighet for å bruke begrensningsprogrammering på en reaktiv måte er å overføre handlinger og deler av planleggingen som den er nå til begrensinger og domener. [14]

9 Fremtidig arbeid

- Erstatte sikkerhetsbegrensinger med varmebegrensning
- Ha varmebegrensning også også på kran?
- Skrive egne søkemål

10 Konklusjon

Målet med denne masteroppgaven var å se på utfordringer innen automatisk vedlikeholdsplanlegging i oljeindustrien. Det er prøvd ut forskjellige løsningsstrategier samt lagt til en varmebegrensing. Alt for å ta hånd om vedlikeholdsplanlegging, i olje og gassindustrien som har høye krav høye krav til planlegging, kvalitetssikring og gjennomføring.

Den første løsningsstrategien (LS1), som tildelte kraner først, løste alle probleminstansene uavhengig av størrelse og kompleksitet. For implementasjonen uten varmebegrensing og deretter med varmebegrensning ligger løsningene henholdsvis 74.5% og 81.0% over teoretisk nedregrense.

Den andre løsningsstrategien (LS2), som tildeler starttidspunkt til alle aktivitene først, løste ikke alle probleminstansene, men har de beste løsningene i forhold til teoretisk nedregrense. For implementasjonen uten varmebegrensing og deretter med varmebegrensning ligger løsningene henholdsvis 14.2% og 23.2% over teoretisk nedregrense.

Løsningene med varmebegrensning var med begge løsningsstrategiene noe dårligere enn uten varmebegrensning. Resultatene viser at løsningsstrategien har større betydning enn varmebegrensningen når det gjelder hvor mange probleminstanser som blir løst. Det ser ut som det å legge til varmebegrensing ikke forbedrer makespan og det må tas beslutninger på om det er best mulig løsninger eller flest mulig løsninger som er ønskelig når det skal utvikles en appliksjon med Scheduler.

11 Vedlegg

Benchmark	Nedregrense	Øvregrense	Med varme Makespan	Uten varme Makespan
Act50Loc10Crew5Crane2_1	24	164	37	37
Act50Loc10Crew5Crane2_2	15	164	24	24
Act50Loc10Crew5Crane2_3	19	174	42	36
Act50Loc10Crew5Crane2_4	32	189	33	33
Act50Loc10Crew5Crane2_5	30	174	49	48
Act60Loc10Crew5Crane2_1	28	193	29	29
Act60Loc10Crew5Crane2_2	24	235	42	36
Act60Loc10Crew5Crane2_3	25	196	41	39
Act60Loc10Crew5Crane2_4	25	188	36	33
Act60Loc10Crew5Crane2_5	31	222	58	57
Act70Loc10Crew5Crane2_1	25	242	44	39
Act70Loc10Crew5Crane2_2	38	226	39	39
Act70Loc10Crew5Crane2_3	34	244	61	59
Act70Loc10Crew5Crane2_4	38	245	44	39
Act70Loc10Crew5Crane2_5	36	233	42	44
Act80Loc10Crew5Crane2_1	39	260	70	54
Act80Loc10Crew5Crane2_2	37	279	83	84
Act80Loc10Crew5Crane2_3	30	295	76	72
Act80Loc10Crew5Crane2_4	48	290	78	74
Act80Loc10Crew5Crane2_5	37	287	75	63
Act90Loc10Crew5Crane2_1	55	310	82	77
Act90Loc10Crew5Crane2_2	42	321	72	72
Act90Loc10Crew5Crane2_3	57	331	92	80
Act90Loc10Crew5Crane2_4	35	283	83	83
Act90Loc10Crew5Crane2_5	43	324	74	76
Act50Loc10Crew5Crane3_1	25	161	26	26
Act50Loc10Crew5Crane3_2	21	159	53	47
Act50Loc10Crew5Crane3_3	25	161	33	25
Act50Loc10Crew5Crane3_4	33	173	34	34
Act50Loc10Crew5Crane3_5	20	188	35	28

Act60Loc10Crew5Crane3_1	25	204	45	45
Act60Loc10Crew5Crane3_2	27	206	32	32
Act60Loc10Crew5Crane3_3	38	226	55	55
Act60Loc10Crew5Crane3_4	21	223	59	58
Act60Loc10Crew5Crane3_5	27	185	58	41
Act70Loc10Crew5Crane3_1	31	263	55	48
Act70Loc10Crew5Crane3_2	39	257	97	83
Act70Loc10Crew5Crane3_3	26	263	63	60
Act70Loc10Crew5Crane3_4	32	259	51	49
Act70Loc10Crew5Crane3_5	38	246	39	39
Act80Loc10Crew5Crane3_1	34	273	44	38
Act80Loc10Crew5Crane3_2	30	313	48	48
Act80Loc10Crew5Crane3_3	49	280	64	64
Act80Loc10Crew5Crane3_4	37	250	54	54
Act80Loc10Crew5Crane3_5	30	275	58	59
Act90Loc10Crew5Crane3_1	45	327	83	78
Act90Loc10Crew5Crane3_2	34	354	64	61
Act90Loc10Crew5Crane3_3	47	322	71	71
Act90Loc10Crew5Crane3_4	39	301	58	60
Act90Loc10Crew5Crane3_5	29	296	61	53

Tabell 3: Løsninger med løsningsstrategi 1 med tidsgrense 100 sekunder, under 100 aktiviteter

Benchmark	Nedregrense	Øvregrense	Med varme Makespan	Uten varme Makespan
Act100Loc10Crew5Crane2_1	54	327	72	71
Act100Loc10Crew5Crane2_2	51	340	107	107
Act100Loc10Crew5Crane2_3	59	357	85	86
Act100Loc10Crew5Crane2_4	33	363	112	108
Act100Loc10Crew5Crane2_5	32	347	88	78
Act200Loc10Crew5Crane2_1	103	751	194	194
Act200Loc10Crew5Crane2_2	82	695	200	205

Act200Loc10Crew5Crane2_3	97	699	167	164
Act200Loc10Crew5Crane2_4	105	703	171	184
Act200Loc10Crew5Crane2_5	92	688	153	152
Act300Loc10Crew5Crane2_1	132	1034	261	261
Act300Loc10Crew5Crane2_2	128	1009	258	256
Act300Loc10Crew5Crane2_3	144	1028	281	259
Act300Loc10Crew5Crane2_4	141	1062	251	238
Act300Loc10Crew5Crane2_5	171	1069	305	276
Act400Loc10Crew5Crane2_1	123	1307	295	302
Act400Loc10Crew5Crane2_2	235	1464	403	413
Act400Loc10Crew5Crane2_3	188	1445	373	344
Act400Loc10Crew5Crane2_4	175	1385	340	305
Act400Loc10Crew5Crane2_5	179	1412	289	263
Act500Loc10Crew5Crane2_1	222	1743	381	371
Act500Loc10Crew5Crane2_2	246	1750	531	504
Act500Loc10Crew5Crane2_3	197	1665	445	445
Act500Loc10Crew5Crane2_4	189	1763	454	453
Act500Loc10Crew5Crane2_5	256	1828	442	442
Act600Loc10Crew5Crane2_1	263	2064	553	498
Act600Loc10Crew5Crane2_2	270	2061	552	544
Act600Loc10Crew5Crane2_3	264	2100	582	538
Act600Loc10Crew5Crane2_4	271	2033	533	527
Act600Loc10Crew5Crane2_5	272	2084	535	496
Act700Loc10Crew5Crane2_1	313	2356	633	611
Act700Loc10Crew5Crane2_2	333	2472	676	669
Act700Loc10Crew5Crane2_3	328	2428	590	556
Act700Loc10Crew5Crane2_4	358	2448	574	553
Act700Loc10Crew5Crane2_5	297	2346	601	570
Act800Loc10Crew5Crane2_1	371	2835	722	689
Act800Loc10Crew5Crane2_2	368	2826	688	684
Act800Loc10Crew5Crane2_3	368	2750	683	664
Act800Loc10Crew5Crane2_4	364	2809	663	670
Act800Loc10Crew5Crane2_5	379	2931	693	693
Act900Loc10Crew5Crane2_1	368	3209	700	676

Act900Loc10Crew5Crane2_2	399	3101	752	741
Act900Loc10Crew5Crane2_3	426	3204	866	855
Act900Loc10Crew5Crane2_4	364	3071	768	760
Act900Loc10Crew5Crane2_5	400	3139	737	717
Act1000Loc10Crew5Crane2_1	422	3452	836	807
Act1000Loc10Crew5Crane2_2	484	3612	939	944
Act1000Loc10Crew5Crane2_3	417	3463	803	792
Act1000Loc10Crew5Crane2_4	494	3422	828	780
Act1000Loc10Crew5Crane2_5	446	3484	830	802
Act5000Loc10Crew5Crane2_1	2154	17413	4494	4458
Act5000Loc10Crew5Crane2_2	2182	17508	4293	4031
Act5000Loc10Crew5Crane2_3	2089	17557	4264	4113
Act5000Loc10Crew5Crane2_4	2262	17155	3948	3873
Act5000Loc10Crew5Crane2_5	2291	17629	4626	4468
Act100Loc10Crew5Crane3_1	38	350	93	90
Act100Loc10Crew5Crane3_2	34	345	76	76
Act100Loc10Crew5Crane3_3	56	372	101	101
Act100Loc10Crew5Crane3_4	46	365	93	90
Act100Loc10Crew5Crane3_5	53	326	53	53
Act200Loc10Crew5Crane3_1	66	671	164	147
Act200Loc10Crew5Crane3_2	100	751	193	176
Act200Loc10Crew5Crane3_3	102	710	168	176
Act200Loc10Crew5Crane3_4	103	699	168	168
Act200Loc10Crew5Crane3_5	88	669	110	101
Act300Loc10Crew5Crane3_1	134	1057	240	241
Act300Loc10Crew5Crane3_2	150	1035	209	209
Act300Loc10Crew5Crane3_3	140	1046	264	251
Act300Loc10Crew5Crane3_4	134	1058	209	218
Act300Loc10Crew5Crane3_5	122	1029	321	285
Act400Loc10Crew5Crane3_1	185	1363	359	361
Act400Loc10Crew5Crane3_2	177	1424	377	359
Act400Loc10Crew5Crane3_3	173	1345	325	326
Act400Loc10Crew5Crane3_4	211	1370	281	277
Act400Loc10Crew5Crane3_5	183	1395	326	337

Act500Loc10Crew5Crane3_1	181	1693	361	362
Act500Loc10Crew5Crane3_2	227	1719	367	367
Act500Loc10Crew5Crane3_3	217	1738	430	427
Act500Loc10Crew5Crane3_4	230	1705	415	394
Act500Loc10Crew5Crane3_5	205	1762	353	354
Act600Loc10Crew5Crane3_1	280	2095	475	439
Act600Loc10Crew5Crane3_2	280	2088	504	495
Act600Loc10Crew5Crane3_3	307	2167	451	432
Act600Loc10Crew5Crane3_4	258	2053	413	413
Act600Loc10Crew5Crane3_5	261	2043	437	423
Act700Loc10Crew5Crane3_1	329	2424	541	555
Act700Loc10Crew5Crane3_2	338	2516	604	590
Act700Loc10Crew5Crane3_3	340	2415	574	537
Act700Loc10Crew5Crane3_4	333	2511	573	558
Act700Loc10Crew5Crane3_5	304	2386	519	519
Act800Loc10Crew5Crane3_1	374	2830	568	599
Act800Loc10Crew5Crane3_2	375	2781	581	575
Act800Loc10Crew5Crane3_3	406	2949	656	661
Act800Loc10Crew5Crane3_4	389	2862	643	643
Act800Loc10Crew5Crane3_5	364	2782	632	598
Act900Loc10Crew5Crane3_1	433	3156	749	711
Act900Loc10Crew5Crane3_2	420	3167	675	661
Act900Loc10Crew5Crane3_3	424	3230	732	696
Act900Loc10Crew5Crane3_4	418	3169	721	721
Act900Loc10Crew5Crane3_5	453	3111	665	674
Act1000Loc10Crew5Crane3_1	456	3534	872	849
Act1000Loc10Crew5Crane3_2	399	3538	730	707
Act1000Loc10Crew5Crane3_3	439	3545	741	749
Act1000Loc10Crew5Crane3_4	477	3639	955	903
Act1000Loc10Crew5Crane3_5	458	3462	831	827
Act5000Loc10Crew5Crane3_1	2162	17347	3793	3745
Act5000Loc10Crew5Crane3_2	2190	17682	3819	3772
Act5000Loc10Crew5Crane3_3	2097	17551	3939	3964
Act5000Loc10Crew5Crane3_4	2203	17539	3845	3799

Act5000Loc10Crew5Crane3_5	2292	17536	4046	4017

Tabell 4: Løsninger med løsningsstrategi 1 med tidsgrense 100 sekunder, over 100 aktiviteter

Benchmark	Nedregrense	Øvregrense	Med varme Makespan	Uten varme Makespan
Act50Loc10Crew5Crane2_1	24	164	31	30
Act50Loc10Crew5Crane2_2	15	164	19	19
Act50Loc10Crew5Crane2_3	19	174	33	28
Act50Loc10Crew5Crane2_4	32	189	33	33
Act50Loc10Crew5Crane2_5	30	174	35	34
Act60Loc10Crew5Crane2_1	28	193	29	29
Act60Loc10Crew5Crane2_2	24	235	41	31
Act60Loc10Crew5Crane2_3	25	196	27	26
Act60Loc10Crew5Crane2_4	25	188	26	25
Act60Loc10Crew5Crane2_5	31	222	45	43
Act70Loc10Crew5Crane2_1	25	242	32	30
Act70Loc10Crew5Crane2_2	38	226	39	39
Act70Loc10Crew5Crane2_3	34	244	42	35
Act70Loc10Crew5Crane2_4	38	245	38	38
Act70Loc10Crew5Crane2_5	36	233	40	36
Act80Loc10Crew5Crane2_1	39	260	57	41
Act80Loc10Crew5Crane2_2	37	279	43	43
Act80Loc10Crew5Crane2_3	30	295	48	45
Act80Loc10Crew5Crane2_4	48	290	52	52
Act90Loc10Crew5Crane2_3	57	331	-	60
Act90Loc10Crew5Crane2_5	43	324	-	43
Act50Loc10Crew5Crane3_1	25	161	26	26
Act50Loc10Crew5Crane3_3	25	161	37	28
Act50Loc10Crew5Crane3_4	33	173	34	34
Act50Loc10Crew5Crane3_5	20	188	34	21
Act60Loc10Crew5Crane3_1	25	204	25	25

Act60Loc10Crew5Crane3_2	27	206	36	34
Act60Loc10Crew5Crane3_4	21	223	32	32
Act60Loc10Crew5Crane3_5	27	185	30	30
Act70Loc10Crew5Crane3_1	31	263	41	41
Act70Loc10Crew5Crane3_3	26	263	38	35
Act70Loc10Crew5Crane3_4	32	259	37	37
Act70Loc10Crew5Crane3_5	38	246	38	38
Act80Loc10Crew5Crane3_1	34	273	34	-
Act80Loc10Crew5Crane3_2	30	313	36	36
Act80Loc10Crew5Crane3_3	49	280	50	50
Act90Loc10Crew5Crane3_2	34	354	48	44
Act90Loc10Crew5Crane3_4	39	301	48	45
Act90Loc10Crew5Crane3_5	29	296	39	31

Tabell 5: Løsninger med løsningsstrategi 2 og tidsgrense på 100 sekunder, under 100 aktiviteter

Benchmark	Nedregrense	Øvregrense	Med varme Makespan	Uten varme Makespan
Act100Loc10Crew5Crane2_1	54	327	55	55
Act100Loc10Crew5Crane2_2	51	340	62	62
Act100Loc10Crew5Crane2_3	59	357	60	60
Act100Loc10Crew5Crane2_5	32	347	70	56
Act200Loc10Crew5Crane2_1	103	751	125	125
Act200Loc10Crew5Crane2_2	82	695	124	117
Act200Loc10Crew5Crane2_3	97	699	134	-
Act200Loc10Crew5Crane2_5	92	688	106	-
Act300Loc10Crew5Crane2_3	144	1028	177	167
Act300Loc10Crew5Crane2_4	141	1062	193	164
Act300Loc10Crew5Crane2_5	171	1069	235	172
Act400Loc10Crew5Crane2_1	123	1307	215	164
Act400Loc10Crew5Crane2_2	235	1464	246	236
Act400Loc10Crew5Crane2_3	188	1445	255	-

Act400Loc10Crew5Crane2_4	175	1385	-	189
Act400Loc10Crew5Crane2_5	179	1412	-	180
Act500Loc10Crew5Crane2_1	222	1743	-	222
Act500Loc10Crew5Crane2_5	256	1828	257	257
Act600Loc10Crew5Crane2_2	270	2061	347	-
Act600Loc10Crew5Crane2_4	271	2033	327	-
Act100Loc10Crew5Crane3_2	34	345	43	41
Act100Loc10Crew5Crane3_3	56	372	58	58
Act100Loc10Crew5Crane3_5	53	326	53	53
Act200Loc10Crew5Crane3_3	102	710	103	103
Act300Loc10Crew5Crane3_2	150	1035	-	150
Act300Loc10Crew5Crane3_3	140	1046	140	-
Act300Loc10Crew5Crane3_4	134	1058	139	-
Act400Loc10Crew5Crane3_4	211	1370	-	211
Act600Loc10Crew5Crane3_3	307	2167	307	-
Act600Loc10Crew5Crane3_5	261	2043	267	-
Act800Loc10Crew5Crane3_3	406	2949	406	-
Act900Loc10Crew5Crane3_1	433	3156	450	-

Tabell 6: Løsninger med løsningsstrategi 2 og tidsgrense på 100 sekunder, over 100 aktiviteter

Referanser

[1]

[2] http://www.softwareadvice.com/manufacturing/exact-jobboss-profile/.

[3] Definition of microsoft project constraints. http://support.microsoft.com/kb/74978.

[4] Manual vs. autoscheduled tasks in microsoft project 2010. http://www.techrepublic.com/blog/tech-manager/manual-vs-autoscheduled-tasks-in-microsoft-project-2010/5591.

[5] Wikipedia. http://en.wikipedia.org/wiki/Constraint_programming.

[6] Wikipedia: Comparison of project-management software. http://en.wikipedia.org/wiki/List_of_project_management_software.

[7] Wikipedia: Computational complexity theory. http://en.wikipedia.org/wiki/Computational_complexity_theory.

[8] Wikipedia: Np-hard. http://en.wikipedia.org/wiki/NP-hard.

[9] Wikipedia: Optimization problem. http://en.wikipedia.org/wiki/Optimization_problem.

[10] Peter J. Stuckey Andreas Schutt, Thibaut Feydy and Mark G. Wallace. Solving the resource constrained project scheduling problem with generalized precedences by lazy clause generation.

[11] IBM. *IBM ILOG Scheduler V6.8*. IBM.

[12] IBM. *IBM ILOG Solver V6.8*. IBM.

[13] ILOG. *ILOG Concert Technology 2.0 Reference Manual*. ILOG.

[14] Wim Nuijten and Claude Le Pape. Constraint-based job shop scheduling with ILOG SCHEDULER. *Journal of Heuristics*, 3(4):271–286, March 1998.

[15] Bård Henning Tvedt og Marc Bezem. Modeling safety constraints in oil and gas maintenance scheduling. *Universitetet i Bergen*, 2011.

[16] Claude Le Pape. Implementation of resource constraints in ilog schedule: A library for the development of constraint-based scheduling systems. *Intelligent Systems Engineering*, 3:55–66, 1994.

[17] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278 – 285, 1993. ¡ce:title¿Project Management anf Scheduling¡/ce:title¿.

[18] Mark Wallace, Joachim Schimpf, Kish Shen, and Warwick Harvey. On benchmarking constraint logic programming platforms. response to fernandez and hill's "a comparative study of eight constraint programming languages over the boolean and finite domains". *Constraints*, 9(1):5–34, January 2004.