Automatisk Planlegging i Oljeindustrien

av

Teis Lindemark

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.

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 oljeplattform for å kunne gjøre problemet så likt som mulig virkeligheten. Vedlikeholdsaktiviteter som skal planlegges/løses genereres i et eget program. Disse, såkalte probleminstanser, blir inputdata i den IBM ILOG Scheduler (Scheduler) implementerte løsningen. Implementasjonen i Scheduler er i denne prosjektoppgaven utvidet med en begrensning som kalles varmebegrensing. Varme i denne sammenheng er ikke å forstå som varme i tradisjonell forstand, men et mål for hvor mye belastning en lokasjon på plattformen 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 kapitlet om metode vil det beskrives hvordan prosjektoppgaven er utført, hvilke metoder som er brukt og hvordan resultatene har blitt evaluert. Under kapitlet om resultater vil det legges frem resultater med forskjellige løsningsstrategier, med kombinasjoner av begrensninger. Løsningene evalueres og til slutt sammenfattes en konklusjon over det arbeidet som er gjort.

1.2 Bakgrunn

I denne prosjektoppgaven 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 her 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-hard problemer. Dette er avgjørelsesproblemer, søkeproblemer og optimaliseringsproblemer [8]. Problemet i prosjektoppgaven 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 teoretisk 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 hovedfunksjoner. 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 gunstige måten å benchmarke en applikasjon på å implementere en løsning for hvert system for så å 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 skal bruke det. Wallace tar også opp vanskeligheten med tidsmekanismer i høynivåspråk. Tidsmekanismer er vanligvis basert på CPU tid som kan variere fra maskinoppsett til maskinoppsett og derfor kan gi unøyaktigheter i benchmarkingen.

2 Motivasjon og målsetting

Prosjektoppaven er motivert av 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 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 da en på mest mulig effektiv måte må benytte seg av de ressursene som er tilgjengelige til enhver tid. Riktig planlegging hvor en tar hensyn til de begrensninger og ytre rammer en er stilt ovenfor, er en viktig faktor for å få utført oppgavene optimalt. Det er også mye penger involvert i oljeindustrien. Det å utføre vedlikeholdsaktiviteter på en ineffektiv 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 utstyr endrer planleggingsforutsetningene.

2.1 Målet med prosjektoppgaven

Målet med prosjektoppgaven er å utvide den eksisterende Scheduler løsningen med en varmebegrensning og evaluere resultatene med og uten varmebegrensning. Mål:

- legge til *varmebegrensning*

- implementere i Scheduler
- minimere *makespan*

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. Denne prosjektoppgaven vil søke å finne ut:

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

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

3 Problemstilling

Problemstillingen i denne prosjektoppgaven tar utgangspunkt i den opprinnelige problemstillingen til Bård Henning Tvedt. Beskrivelsen i det etterfølgende frem til begrensning (6) på side 15, er i hovedsak 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 de blir 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 Schedu-*

ling 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 begrensninger 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 begrensninger. Et eksempel er sikkerhetsbegrensning 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 Notasjon 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-variabler (v)*, *konstanter (c)* og *avledet-variabler (w)*. Et eksempel på en avgjørelses-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 avledete-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 varmebegrensning er en begrensning som viser maksimal kapasitet av hva lokasjonen tåler. Det er også begrensninger når farlig arbeid som tung løfting blir utført, og 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 utførelse av aktiviteter. 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 sier kun at 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. En alternativ ressurs er en ressurs som det er flere av og hvor det ikke spiller en rolle hvilken man bruker så lenge den ikke brukes samtidig av en aktivitet.

Kraner har en lokasjon $c_{loc}(Crane_k) \in Locs$, og hver lokasjon kan bare ha en kran. En kran står på en fast lokasjon og kan ikke flyttes til andre lokasjoner. Kraner kan derimot løfte løfte på hvilken som helst lokasjon. Lokasjonen som en kran er tildelt går ut ifra probleminstansene. På grunn av at tung løfting er farlig arbeid, er kranbruk omgitt med sikkerhetssoner. Disse sikkerhetssonene er satt til både lokasjonen 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. Hver enkelt kran kan brukes til å utføre aktiviteter på alle lokasjoner.

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 $v_{sta}(Act_i) \in [0, c_{hor}(P))$, hvor horisonten som indikerer planleggingens maksimale fullføringstid 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 hvert enkelt medlem av et mannskap og det er ikke mulig å samle ressurser for å redusere varigheten. Den siste ressursen som er tilgjengelig er kraner, men er ikke nødvendig for alle aktivitetene.

I tillegg 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 Begrensninger

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 *kumulativ ressursbegrensing* påføres alle mannskaper for å være sikker på at den totale ressursbruken ikke overstiger tilgjengelig kapasitet. Det er uttrykt 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 den underliggende over-

lappingen uttrykt som to aktiviteter som 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 eneressursbegrensningen blir da:

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

for alle aktiviteter som krever kran.

Sikkerhetsbegrensningene er uttrykt i form av lokasjonen til aktiviteten 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:

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

Når sikkerhetsbegrensningene utelukker bruken av denne kranens lokasjon er gitt ved:

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

Så langt er formler og tekst hentet fra [15]. I det etterfølgende tillegges den nye begrensningen kalt *varmebegrensning*. Dette er en måte å kunne sette en verdi på et mannskap og en kapasitet til 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 varmeverdier ut i fra hva slags arbeid de utfører. Mannskaper som driver arbeid som for eksempel sveising, kan ha en høyere varmeverdi enn et mannskap med elektrikere. Dette fordi sveisere kan ha en høyere varmeverdi da aktivitetene som utføres krever mye av den lokasjonen de er på. De kan for eksempel utgjøre en fare på grunn av farlige gasser, varmeutvikling etc. Varmebe-

grensningen er uttrykt som kumulativ ressursbegrensning og er påført en lokasjon for å sikre at total varmebruk ikke overstiger varmekapasiteten tilgjengelig på lokasjonen. Den er uttrykt ved:

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

Ved en videreutvikling av planleggingsmetodikken skal varmebegrensingen også etterhvert kunne erstatte sikkerhetsbegrensningene på for eksempel kranbruk. Dette ved at lokasjoner med kran får en varmekapasitet, og aktiviteter som bruker kran bruker opp denne varmekapasiten på lokasjonen.

3.5 Målfunksjon

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

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

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

3.6 Probleminstanser

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

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 sirkulariteter på aktivitetsavhengigheter (banalt som for eksempel: aktivitet 2 er avhengig av aktivitet 1, som igjen er avhengig av aktivitet 2) og at det ikke skal være mer enn en kran på en lokasjon, slik at alle 160 instansene er gyldige.

Alle probleminstansene har 10 lokasjoner, som er en reduksjon fra de 25 lokasjonene i de opprinnelige probleminstansene til Bård Henning Tvedt. Dette er gjort fordi med 25 lokasjoner ville det vært så få aktiviteter på hver lokasjon at varmebegrensingen til en lokasjon har liten sannsynlighet for å bli 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. De resterende 70% er følgelig uten avhengigheter eller kranbruk. 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: $c_{heatcap}(Loc_l \in [10, 20]$. Dette settet ble forkastet og dette er begrunnet senere i prosjektoppgaven. Det ble generert et ytterligere 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 dette kapitlet blir det gitt en innføring om begrensningsprogrammering og utfordringer som kommer med begrensningsprogrammering. Det vil også bli gitt eksempler på verktøy som eksisterer idag for å bruke i begrensningsprogrammering samt noen enkle verktøy for enkel planlegging av aktiviteter og ressurser. Til slutt er det en kort beskrivelse av verktøyene som er brukt i denne prosjektetoppgaven.

4.1 Kort om begrensningsprogrammering

Begrensningsprogrammering er et programmeringsparadigme hvor relasjoner mellom variable blir definert i form av begrensninger. Begrensningsprogrammering er en form for deklarativ programmering, som skiller seg fra det mer vanlige

imperative programmeringen[1] ved at løsningen blir til ved å tilfredsstille begrensningene. Det er forskjellige områder innen begrensningsprogrammering som "Constraint Satisfaction Problems", "Constraint Logic Problems" og "Scheduling Problems"[5]. Det mest kjente planleggingsproblemet er "Job Shop Scheduling". I denne oppgaven er det et Scheduling problem, altså planleggingsproblem som omhandles.

4.2 Utfordringer med begrensningsprogrammering

Ved å sammenligne ytelsen til begrensnings-logikk programmeringssystemer mot andre verktøy for begrensingsløsningsprogrammering, er det begrensnings-logikk programmeringssystemene som gir den beste ytelsen. Ytelsen er likevel ikke like god som i mer tradisjonelle imperative programmeringsspråk, og spesielt gjelder dette innenfor tallmessige utregninger. Et mulig forbedringspotensiale innenfor begrensningsprogrammeringen vil da være å utvikle en avansert kompilator som sjekker de tilfellene hvor det ikke er behov for begrensningsløsning, og da kompilerer disse på mest mulig effektive måte[4].

Begrensningsprogrammeringssystemer har ofte en svakhet når det kommer til feilsøking (engelsk: debugging). Det finnes ofte ikke tilstrekkelige metoder for å kunne sikre riktigheten av en løsning og muligheten til å sjekke ytelse i programmer. En måte å løse utfordringene med manglende feilsøkingsmuligheter er å ta i 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 oppnådd med metoden. Postforhold er også en boolsk verdi[4].

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 finnes det alternativer som er kommersielle og

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

andre med åpen kildekode. Noen eksempler på skreddersydde programmeringsspråk for begrensningsprogrammering er Prolog og Comet. Sistnevnte er et programmeringspråk for begrensningsprogrammering med lokalt søk og er en kommersiell løsning. Et eksempler på begrensningsprogrammeringsbibliotek er IBM ILOG CP.

Planlegging er ofte tett knyttet opp mot prosjektstyring og prosjektstyringsverktøy finnes det etterhvert veldig mange av [6]. Det finnes programmer installert både lokalt på maskinen og webbaserte tjenester. Fellesnevneren for veldig mange av disse verktøyene, 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 verktøy er ofte gjort 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 [3]. I MSProject er det mulig å legge til begrensninger, men disse kan kun legges til aktiviteters starttidspunkt. Aktiviteter kan tillegges begrensinger ut ifra om aktiviteten skal planlegges tidlig eller sent i prosessen eller på et gitt tidspunkt [1]. Denne prosjektoppgaven har flere ressurser (lokasjoner, mannskaper, kraner, osv.) knyttet til en aktivitet, begrensinger og sikkerhetsbegrensinger. Når problemet blir såpass komplekst, vil ikke et program som MSProject fungere til formålet.

Det finnes systemer som løser de mest kjente og brukte begrensningsprogrammeringsproblemene (for eksempel "Job Shop Scheduling Problem") "Exact JobBOSS Software" er et eksempel på et slikt system. Systemet håndterer planleggings- og produksjonsgjennomføring [2].

4.4 Verktøy brukt i prosjektoppgaven

Etterfølgende verktøy og teknologier utviklet av IBM er brukt for å gjennomføre prosjektoppgaven.

4.4.1 Kort om Concert

Concert er et C++ bibliotek av klasser og funksjoner som gir mulighet til å designe modeller av problemer innen matematisk programmering og innen begrensningsprogrammering, såkalte optimaliseringsproblemer. Det er ikke noe eget programmeringsspråk, men gir muligheter til å bruke datastrukturer og kontrollstrukturer som allerede finnes i C++. Dette gir så gode muligheter til å integrere Concert i allerede eksisterende løsninger og systemer. I implementeringen av løsningen i prosjektoppgaven benyttes enkelte av Concert sine typer og klasser. Alle navn på typer, klasser og funksjoner har prefiksen Ilo.

De enkleste klassene i Concert (eks. IloNumVar og IloConstraint) har en tilhørende klasse med matriser hvor matrisen er instanser av den enkle klassen. Et eksempel 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 et eget programmeringsspråk, men det gir mulighet 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 forklares som følger:

- Steg 1: Problemet må beskrives i programmeringsspråket som brukes.
- Steg 2: Concertklassene brukes for å opprette en modell av problemet. Modellen blir da satt sammen av beslutningsvariable og begrensninger. Beslutningsvariablene er den ukjente informasjonen i problemet som skal løses. Alle beslutningsvariablene har et domene med mulige verdier. Begrensningene setter grensene for kombinasjonene av verdier for beslutningsvariablene.

- Steg 3: Solver brukes for å løse problemet. Dette innebærer å finne verdier for alle beslutningsvariablene samt ikke bryte noen av de definerte begrensningene. Solver ser etter løsninger i et søkeområde, der søkeområdet er alle mulige kombinasjoner av verdier. I denne prosjektoppgaven brukes generelle søkemål i Solver for å tildele alternative ressurser, rangere eneressurser og for å sette tiden på aktiviteter tidligst mulig i planleggingen.
  - IloAssignAlternative, er søkemålet i Solver for å tildele alternative ressurser til aktiviteter.
  - IloRankForward, er søkemålet i Solver som rangerer ressurser av typen eneressurs.
  - IloSetTimesForward, er søkemålet i Solver som tildele starttid til alle aktivitetene tidligst mulig i planleggingen.

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

4.4.3 Kort om Scheduler

Ved utvikling av problemløsings-applikasjoner som krever behandling av ressurser fordelt på tid, brukes IBM ILOG Scheduler. Scheduler er et C++ bibliotek som baserer seg på Solver, og som benytter alle muligheter med objektorientering og begrensningsprogrammering. Scheduler har spesialisert funksjonalitet på å løse problemer innen planlegging og ressurstildeling.

I denne prosjektoppgaven er Scheduler brukt til å definere aktiviteter, kummulative ressurser, eneressurser og alternative ressurser. Aktiviteter er modellert ved bruk av IloActivity, hvor varighet og navn til aktiviteten blir satt som parametere. I og med at alle aktivitetene har en lokasjon, brukes i tillegg setIntParameter for å definere lokasjonen til aktiviteten. Det blir også lagt inn avhengigheter mellom de aktiviteter som er avhengig av hverandre. Kummulative ressurser, eller gjenbrukbare ressurser, blir brukt når det er begrenset med kapasitet på ressursen, ofte representert ved en tidsfunksjon. I Scheduler er kummulative ressurser modellert med IloDiscreteResource. Eneressurser er i Scheduler ressurser med kapasitet en, og kan kun brukes av en aktivitet av gangen. [11]

5 Metode

I denne delen blir verktøy og teknologier i prosjektoppgaven beskrevet. I tillegg vil metodene som er brukt og strategiene for evaluering av løsningene bli beskrevet.

5.1 Metode

I prosjektoppgaven er det eksperimentert med implementasjonen av ressursene og løsningsstrategien, hvilket er en empirisk metode. Det er gjort en kvalitativ evaluering av løsningene, og makespan er evaluert mot teoretisk nedregrense. Det blir i denne delen gjennomgått implementerings- og evalueringsprosessen som er brukt i prosjektoppgaven.

5.1.1 Implementeringsprosessen

I denne prosjektoppgaven er krantildelingen modellert med alternative ressurser. Ved å modellere med alternative ressurser kan krantildelingen endres underveis i søkestrategien ved å bruke søkemålet IloAssignAlternatives.

Det er problematisk å fastslå om en RCPSP løsning med makespan mindre enn en gitt frist er NP-hard. Hvis denne løsningen er NP-hard er det utvidede problemet med varmeressursen også NP-hard. Da er en optimal løsning på selv de enkleste formene av problemet ikke garantert innen polynomisk tid. To forskjellige løsningsstrategier er testet. Begge løsningsstrategiene er implementert i Solver og Scheduler bibliotekene. I begge løsningsstrategiene brukes standard søkemål i Solver, men oppsatt i forskjellig rekkefølge i de ulike løsningsstrategiene.

Varmebegrensningen i prosjektoppgaven er implementert med to metoder som er gjengitt i listing 1 og 2. Mannskapers varme og lokasjoners varmekapasitet blir i første omgang lest inn fra probleminstansene og lagt til i hver sine vektorer. Vektoren med mannskapets varme blir lagt til i samme rekkefølge som andre vektorer som omhandler mannskap. På den måten vil en indeks i varmevektoren kunne brukes på en av de andre vektorene som omhandler mannskap og få tilhørende annen informasjon, for eksempel om mannskapets navn. På samme måte er det

gjort med varmekapasitet på lokasjon i og med at også alle lokasjonene har en varmekapasitet. Varmebegrensingen er definert som diskrete ressurser. Først opprettes det en diskrete ressursmatrise med samme lengde som antall varmekapasiteter. Deretter itereres over denne matrisen og det legges inn instanser av diskrete ressurser hvor varmekapasiteten blir definert.

Listing 1: Oppretting av varmebegrensing

```
void CpModel::createLocationHeat(EpsModel &dModel)
{
        int nbOfHeatCap = dModel.getHeatcapacity()->size();
        cumulativeLocationsHeat = IloDiscreteResourceArray(env, nbOfHeatCap);

        for (int i = 0; i < nbOfHeatCap; i++)
        {
                cumulativeLocationsHeat[i]
                        = IloDiscreteResource(env,dModel.getHeatcapacity()->at(i));
        }
}
```

I listing 2 tillegnes varmebegrensingen til aktivitetene. Dette gjøres ved å iterere over aktivitetene og legge til varmebegrensning som et krav på alle aktivitetene. Fra matrisen med instansene av varmebegrensningene som diskrete ressurser brukes indeksen $getActLocation() -> at(i)$ for legge til varmebegrensningen til riktig aktivitet som er tilhørende til lokasjonen på den aktuelle aktiviteten. Deretter legges varmen til mannskapet som er brukt på den aktuelle aktiviteten.

Listing 2: Tilegning av varmebegrensing til aktiviteter

```
void CpModel::createCrewHeatConstraints(EpsModel &dModel)
{
        int nbActs = dModel.getActLocation()->size();

        for (int i = 0; i < nbActs; i++)
        {
                model.add(activities[i].requires
                        (cumulativeLocationsHeat[dModel.getActLocation()->at(i)],
                                dModel.getCrewHeat()->at(dModel.getCrewUsed()->at(i))));
        }
}
```

Den første løsningsstrategien (LS1) bruker som første søkemål IloAssignAlternatives. Denne tildeler kraner til aktiviteter som krever kran. Rangering er mulig for alle ressursbegrensningene. Når begrensninger blir rangert, blir aktiviteten ressursen tilhører flyttet foran alle aktivitetene som ikke har blitt rangert. Det andre søkemålet IloRankForward rangerer alle eneressursbegrensningene i modellen. Det tredje søkemålet er IloSetTimesForward. Denne tilegner starttid til aktiviteter i planleggingen. Ved å velge en kranfordeling først, vil denne styre valget av starttid til aktivitetene. Da antall aktiviteter som krever kran er 20%

av alle aktivitetene og tilgjengelig antall kraner er konstant 2 og 3. Ved tildeling først er forventningen da at søketreet blir for stort til at valget av kranfordeling kan gjøres på nytt på en god måte og det kan forventes at løsningene som fremkommer vil være mindre gunstige med hensyn på makespan.

Den andre løsningsstrategien (LS2) tillegner starttid til alle aktivitetene først, ved å bruke IloSetTimesForward. Når alle aktivitetene har fått tildelt en starttid, brukes deretter 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 foran de aktivitetene som ikke har blitt rangert. Med denne løsningsstrategien vil starttidspunktene for aktivitetene settes først og det forventes at dette kan føre til ugyldige løsninger når kranfordelingen blir gjort senere og lokasjonene har fått etablert sikkerhetssoner.

5.1.2 Evaluering av prosessen

Resultatet av eksperimentering med implementasjonen blir evaluert ved å undersøke makespan opp mot teoretisk nedregrense (se 5.2.2) i 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 10 lokasjoner vil også bli evaluert opp mot løsningene med 25 lokasjoner, men da vil gjennomsnitsverdien w_{rq} i formel (13) bli brukt.

Metoden vil bli evaluert ved å bruke probleminstanser som er generert av et eksternt program. I probleminstansene som genereres bestemmes hvor mange av de forskjellige ressursene som skal være med i probleminstansene. I denne prosjektoppgaven er det flere aktiviteter i intervallet 50 - 5000 som implementasjonen blir evaluert på. Denne prosjektoppgavens automatisk genererte probleminstanser er i en kommersiell situasjon selve vedlikeholdsbehovet. Det er et håp at probleminstansene representerer vedlikeholdsbehovet så godt at resultatene i denne prosjektoppgaven er relevant også i kommersiell bruk.

Resultatene blir evaluert av et eksternt program, som sjekker begrensningene som avhengigheter, varme og sikkerhetsbegrensninger. Programmet er utviklet i *Java*

og leser inn en løsning og tilhørende probleminstans. Det blir opprettet en vektor med objekter som aktiviteter, kraner, lokasjoner og mannskaper. På denne måten er all informasjon om de forskjellige objektene samlet, noe som gjør det enkelt å sjekke begrensninger som avhengigheter mellom aktiviteter, varmebegrensning og sikkerhetsbegrensning. I listings 3 er implementasjonen i Java som verifiserer varmebegrensningen vist. Her itereres det over aktivitetene og sjekker mannskapers varme på en aktivitet som er innenfor samme start- og sluttid for aktiviteten (mannskaper som jobber samtidig på en aktivitet) og summen av mannskapenes varme sjekkes opp mot aktivitetens lokasjonens varmekapasitet.

Listing 3: Validering av varmebegrensing

```
public boolean checkHeat() {
        int c_checkHeat = 0;
        int ms = getMakespan();
        for(int t=0;t<ms;t++) {
                int[] heatLoc = new int[locations.size()];
                for(int tmp=0;tmp<heatLoc.length;tmp++) heatLoc[tmp] = 0;
                for(int a = 0;a<activity.size();a++) {
                        int aStart = activity.get(a).getStart();
                        int aEnd = activity.get(a).getStart() + activity.get(a).getDuration();
                        if(aStart <=  t && aEnd > t) {
                                for(int c=0;c<crew.size();c++) {
                                        if(activity.get(a).getCrew().equals(crew.get(c).getName())) {
                                                heatLoc[Integer.parseInt(
                                                        activity.get(a).getLocation().replace("location", ""))-1]
                                                                += crew.get(c).getHeatConsumption();
                                        }
                                }
                        }
                }
                for(int check=0;check<heatLoc.length;check++) {
                        if(heatLoc[check] > locations.get(check).getHeatCapacity()) c_checkHeat+=1;
                }
        }
        if(c_checkHeat == 0) return true;
        else return false;
}
```

En uavhengig verifisering av alle resultatene i prosjektoppgaven er gjort med et program skrevet i Prolog. Dette programmet sjekker alle begrensingene unntatt varmebegrensingen og er utført av Professor Marc Bezem.

5.2 Evalueringsstrategi

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

5.2.1 Teoretisk øvregrense

Teoretisk øvregrense for makespan er:

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

Dette indikerer at i det ugunstigste 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. Mannskapet er optimalisert og teoretisk nedregrense basert på mannskap er:

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

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

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

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.

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

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

Tabell 1: Relativ optimalitets indeks w_{rq} for de forskjellige modellene med tidsgrense på 100 sekunder

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}	$\%^{(1)}$
$LS1\sharp1\sharp3$	1.578	100	1.905	100	1.593	100	1.731	100	1.745	100
$LS1\sharp2\sharp3$	1.672	100	1.961	100	1.711	100	1.766	100	1.810	100
$LS2\sharp1\sharp3$	1.144	84	1.160	27.3	1.157	68	1.042	10.91	1.142	36.88
$LS2\sharp2\sharp3$	1.256	76	1.313	30.91	1.243	72	1.041	18.18	1.232	40
$LS2\sharp1\sharp4$	1.031	100	1.002	100	1.023	100	1.003	100	1.010	100

$^{(1)}$ prosentandel løste probleminstanser
$\sharp 1$ er løsninger uten varmebegrensning
$\sharp 2$ er løsninger med varmebegrensning
$\sharp 3$ er løsninger med sikkerhetsbegrensning
$\sharp 4$ er løsninger uten sikkerhetsbegrensning

Tabell 2: Relativ optimalitets indeks w_{rq} for de forskjellige modellene med tidsgrense på 5 sekunder

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}	$\%^{(1)}$
$LS1\sharp1\sharp3$	1.585	100	1.907	91	1.608	100	1.731	91	1.745	94
$LS1\sharp2\sharp3$	1.674	100	1.963	91	1.718	100	1.767	91	1.808	94
$LS1\sharp1\sharp4$	1.031	100	1.003	91	1.023	100	1.004	91	1.011	94
$LS1\sharp2\sharp4$	1.074	100	1.020	91	1.064	100	1.013	91	1.034	94
$LS2\sharp1\sharp3$	1.155	84	1.160	25	1.323	60	1.060	9	1.151	36
$LS2\sharp2\sharp3$	1.245	84	1.288	24	1.385	60	1.046	15	1.221	37
$LS2\sharp1\sharp4$	1.031	100	1.003	91	1.023	100	1.004	91	1.011	94
$LS2\sharp2\sharp4$	1.074	100	1.020	91	1.064	100	1.013	91	1.034	94

$^{(1)}$ prosentandel løste probleminstanser
$\sharp 1$ er løsninger uten varmebegrensning
$\sharp 2$ er løsninger med varmebegrensning
$\sharp 3$ er løsninger med sikkerhetsbegrensning
$\sharp 4$ er løsninger uten sikkerhetsbegrensning

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 og i tabell 2. Tabell 2 viser at 5 sekunders kjøringer også gir relevante løsninger for samme kjøringer som er gjort i tabell 1. Det er kjørt noen ytterligere modeller i tabell 2.

Det kan finnes en annen teoretisk nedregrense da det her er valgt teoretisk nedregrense basert på det mest begrensede mannskap. Men i og med at den er beregnet ut fra den mest begrensede ressursen er det fornuftig benchmark å sammenligne løsningene med denne.

En praktisk forståelse av denne prosjektoppgavens innføring av varmebegrensning er at teoretisk nødvendigvis må være $\geq$ teoretisk nedregrense uten varmebegrensning. Da teoretisk nedregrense i denne prosjektoppgaven er basert på mannskap, vil makespan ved innføring av varmebegrensning se mer ugunstig ut. Om dette er en reell økning av teoretisk nedregrense eller er et resultat av dårlig optimalisering i løsningen pga. økning av kompleksitet, er vanskelig å avgjøre (dette er diskutert annet sted i prosjektoppgaven). Men en positiv praktisk konsekvens er at den planlagte løsningen med varmebegrensning er bedre da lokasjonen ikke får ressurser ut over sin begrensning.

6.1 Uten varmebegrensning

I tabell 1 og 2 er dette benevnt LS1 og 2 ♯1 ♯3 og ♯4 (dvs. med og uten sikkerhetsbegrensing).

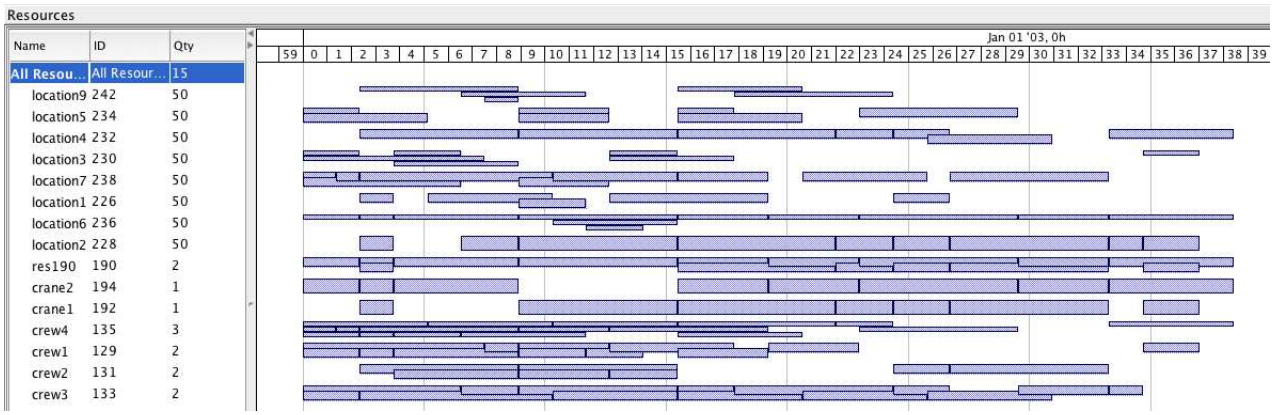
LS1 løser alle probleminstanser og har en gjennomsnittlig makespan på 74.5% høyere enn teoretisk nedregrense. Makespan er for alle $Act(P)$ høye - i intervallet 57.8%-90.5%. Figur 5 viser at lokasjon 2 er fullt utnyttet. Kran 1 er på lokasjon 2 (fra probleminstansene). Ved å bruke søketid på 5 sekunder løser LS1 94% av probleminstansene og har samme gjennomsnittlig makespan som med søketid på 100 sekunder, 74.5%. Makespan er for alle $Act(P)$ høye - i intervallet 58.5%-90.7%.

LS2 løser i gjennomsnitt 36.9% av probleminstansene og har makespan på 14.2% høyere enn teoretisk nedregrense. Makespan for alle $Act(P)$ varierer - i intervallet 4.2%-15.7%. Det laveste/høyeste resultatet er for henholdsvis 3 og 2 kra-

ner $\geq 100(55)$. Dersom sikkerhetsbegrensningen fjernes i dette tilfellet, reduseres makespan til intervall 2.3%-0.3% og alle probleminstanser løses. Figur 1 viser at for det viste tilfellet er lokasjon 6 fullt utnyttet. Det er 2 kraner. Fordelingen av aktiviteter over tidsaksen er jevn. Med å bruke en søketid på 5 sekunder løser LS2 36% av probleminstansene og har en gjennomsnittlig makespan på 15.1%. Makespan for alle $Act(P)$ varierer - i intervallet 6.0%-32.3%.

Løsningene fra tabell 2 med LS1 og LS2 uten sikkerhetsbegrensinger er identiske. Like mange probleminstanser er løst og gjennomsnittlig makespan verdi er også lik.

Tidene som Scheduler bruker for å finne løsninger på probleminstansene er sammenliggnet med å se på tidene for samme antall aktiviteter og henholdsvis 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 de probleminstansene med flest aktiviteter som bruker lengst tid. Tiden for å finne en løsning varierer også stort innenfor probleminstansene med samme antall aktiviteter.



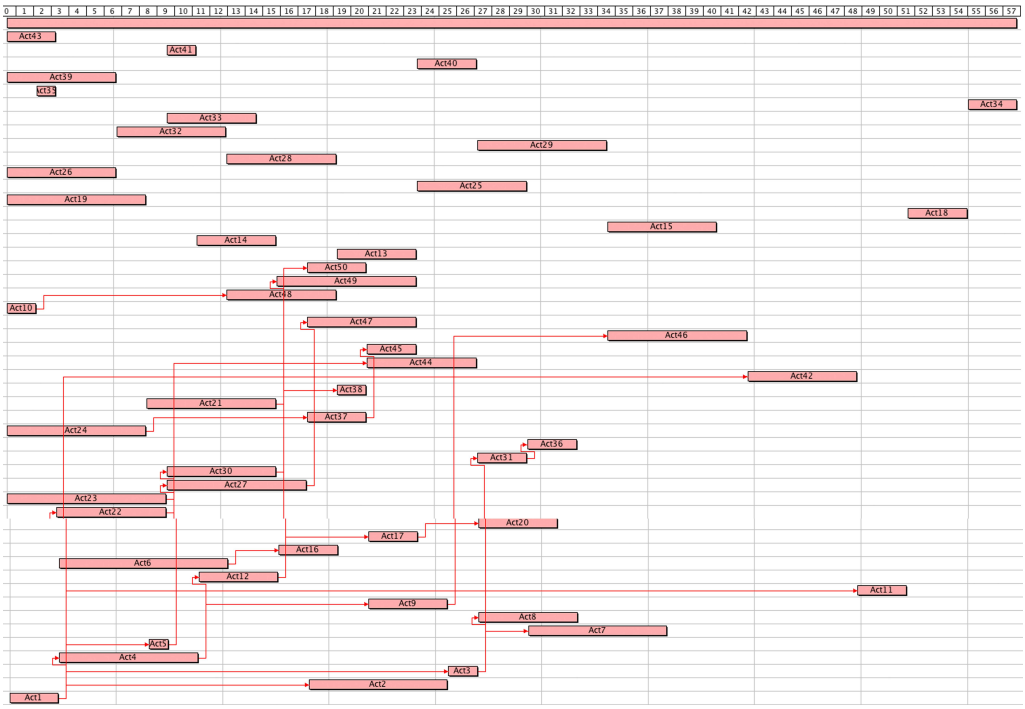
Figur 1: Ressursskjema på Act50Loc10Crew5Crane2, LS2 og ♯1♯3

6.2 Med varmebegrensning

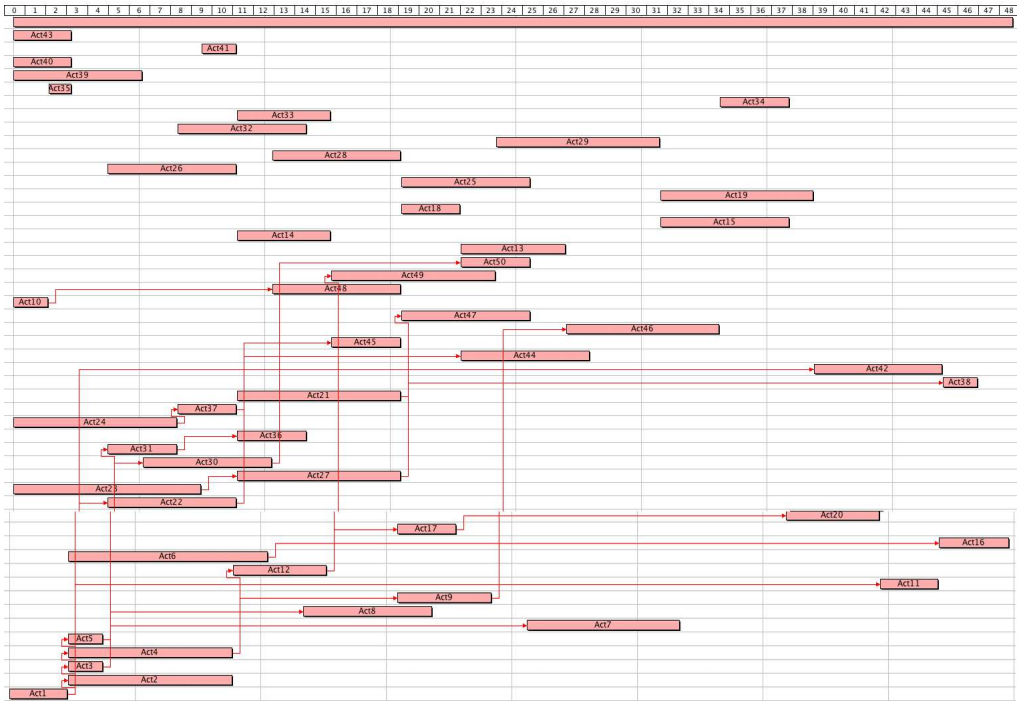
I tabell 1 og 2 er dette benevnt LS1 og 2 ♯2♯3 (dvs. med sikkerhetsbegrensing).

LS1 løser alle probleminstanser og har en gjennomsnittlig makespan på 81.0%. Makespan er for alle $Act(P)$ høye - i intervallet 67.2%-96.1%. Figur 4 viser at for dette tilfellet er lokasjon 2 fullt utnyttet. Kran 1 er på lokasjon 2 (fra probleminstansene). Det er to kraner. Figur 6 viser at for det viste tilfellet er mannskap fullt utnyttet. Det er 3 kraner. Figur 2 med resultat for det viste tilfellet er vist som eksempel. Diagramet viser mange aktiviteter tidlig i planleggingen og få aktiviteter mot slutten. Tidsaksen viser en makespan på 57. Ved å benytte søketid på 5 sekunder løser LS1 94% av probleminstansene og har en gjennomsnittlig makespan på 80.8%. Makespan for alle $Act(P)$ høye - i intervallet 67.4%-96.3%.

LS2 løser i gjennomsnitt 40% av probleminstansene og har makespan på 23.2%. Makespan for alle $Act(P)$ er lave - i intervallet 4.1%-31.3%. Det laveste resultatet er for 3 kraner og $\geq$ 100(55). Figur 3 med resultat for det viste tilfellet vises som eksempel. Diagrammet viser noe bedre fordeling av aktiviteter enn figur 2. Tidsaksen viser en makespan på 48. Ved å benytte søketid på 5 sekunder løser LS2 37% av probleminstansene med en gjennomsnittlig makespan på 22.1%. Makespan for alle $Act(P)$ er lave - i intervallet 4.6%-38.5%.



Figur 2: Ganttskjema på Act50Loc10Crew5Crane2 med varmebegrensning, LS1 og ♯2♯3



Figur 3: Ganttskjema på Act50Loc10Crew5Crane2 med varmebegrensning, LS2 og ♯2♯3

Løsningene fra tabell 2 med LS1 og LS2 uten sikkerhetsbegrensinger er identiske. Like mange probleminstanser er løst og gjennomsnittlig makespan verdi er også lik.

Tidene som Scheduler bruker for å finne løsninger på probleminstansene er sammenliggnet med å se på tidene for samme antall aktiviteter og henholdsvis 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 de probleminstansene med flest aktiviteter som bruker lengst tid. Tiden for å finne en løsning varierer også stort innenfor probleminstansene med samme antall aktiviteter.

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.

Generelle forutsetninger for probleminstansene er: Med 10 lokasjoner vil vil det i snitt være 5-500 aktiviteter pr. lokasjon. Varmekapasitet er en gitt verdi på hver lokasjon. Det er å forvente at:

- jo flere aktiviteter, desto større blir effekten av varmekapasiteten på lokasjonen.
- jo høyere aktivitet (større behov for kran), desto større begrensende effekt vil kranressursene få.

7.1 Probleminstanser

Probleminstansene har fra 50-5000 aktiviteter, mens antall lokasjoner og mannskaper i de genererte probleminstansene er henholdsvis 10 og 5.

Det ble først 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 varmebegrensningen ble tatt med. Det ble funnet løsninger på noen av probleminstansene under 100 aktiviteter, mens over 100 aktiviteter var det kun en løsning. Det er grunn til å tro dette var tilfeldig, ut i fra at varme og varmekapasitet blir tilfeldig plassert innenfor et gitt område. Erfaringene fra dette er kun tatt til etterretning og resultatene er ikke interessante her pga. få løsninger.

Probleminstansene hvor varme og varmekapasitet ikke er overlappende, ga flere løsninger og er brukt videre gjennom prosjektoppgaven.

7.2 Løsningsstrategier

Tabell 1 viser et ganske klart bilde av de sterke og svake sidene til de to løsningsstrategiene som er brukt. LS1 løser alle probleminstanser, mens LS2 gir betydelig bedre makespan.

Ser vi samtidig på ressursskjemaene for $LS1\sharp1\sharp3$, figur 5, og $LS2\sharp1\sharp3$, figur 1, for gitte tilfeller, ser vi følgende:

- $LS1\sharp1\sharp3$ har få aktiviteter mot enden av tidsaksen. Det er lengere makespan.
- $LS2\sharp1\sharp3$ har jevnt med aktiviteter langs tidsaksen. Det er kortere makespan.

I henhold til tidligere skisserte forventninger har dette mest sannsynlig sin årsak i at LS1 produserer flere løsninger fordi kraner med sine begrensninger er på plass tidlig i søkeprosessen. LS2 produserer bedre løsninger fordi det tildeles startverdier til aktiviteter uten å betrakte kranfordelingen så nøye. Men antallet løsninger blir gjerne redusert fordi søkeprosessen ikke klarer å komme ut av konfliktene som oppstår når man tildeler kran senere.

Modellene uten sikkerhetsbegrensinger ga identiske løsninger. Dette er muligens fordi når sikkerhetsbegrensning på kran fjernes fungerer LS1 og LS2 likt.

7.3 Resultater

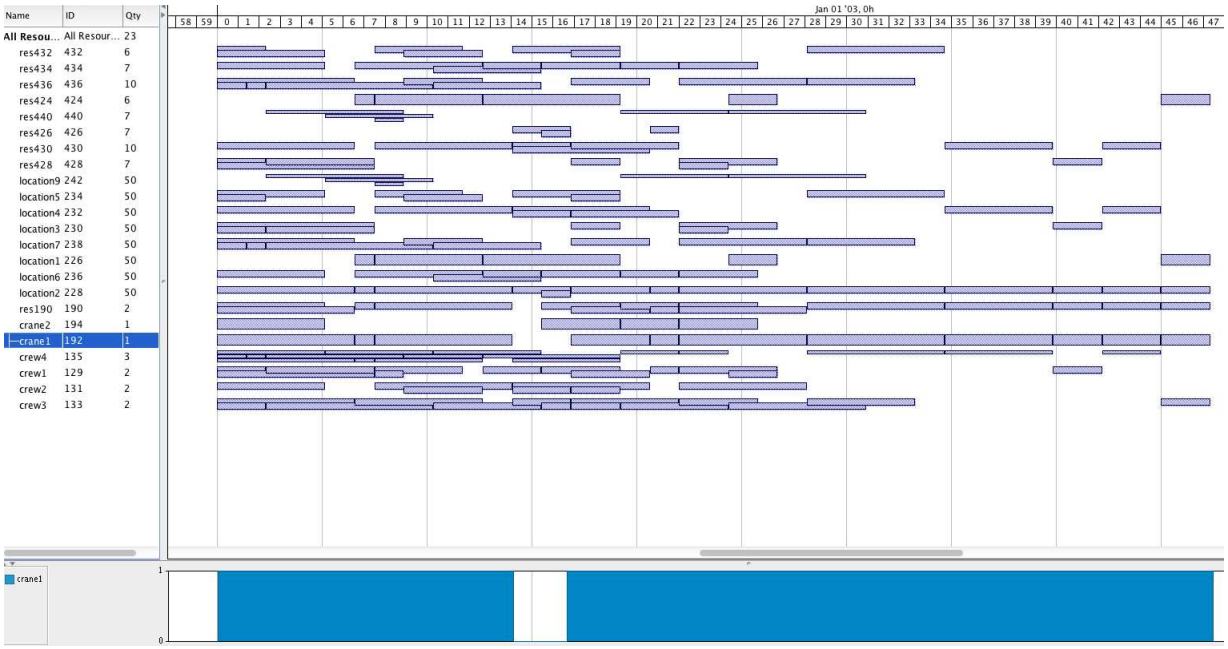
I denne prosjektoppgaven er en eksisterende løsning utvidet med en ressurs kalt varmeressurs. For å evaluere resultatene fra den utvidete løsningen, evalueres den utvidete løsningen med 10 lokasjoner opp mot den opprinnelige løsningen med 25 lokasjoner. Hensikten med å sammenligne resultatene fra den opprinnelige løsningen med 25 lokasjoner og 10 lokasjoner er å analysere effekten av antall lokasjoner.

Evalueringen innebærer også å analysere effekten av varmebegrensningen, som er gjort ved å sammenligne resultatene med 10 lokasjoner med og uten varmebegrensning. Evalueringen av resultatene baserer seg på gjennomsnittsverdiene i tabell 1 og 2.

7.3.1 10 Lokasjoner

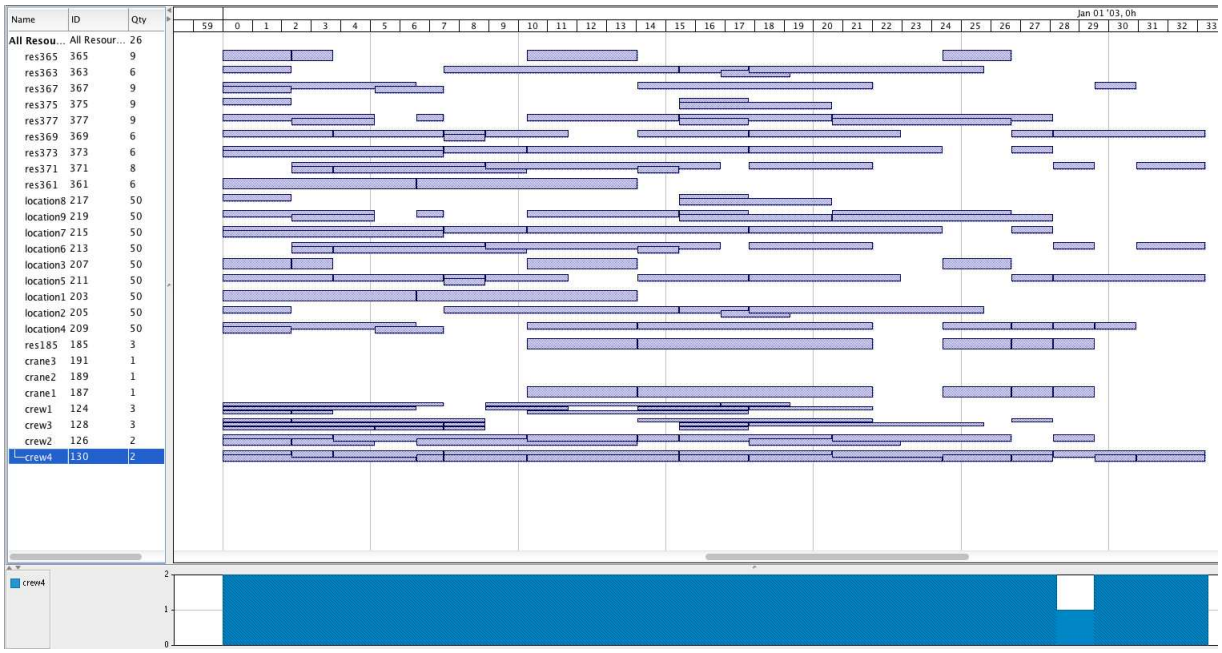
Løsningene er i ganske god overensstemmelse med forventningene i kapitel 5.1.1. Med LS1 hvor kranfordelingen ble valgt først var gjennomsnittlig makespan dårligere enn med LS2.

Figur 4 og 6 er ressursdiagrammer med varmebegrensing. Til venstre i disse bildene er ressursene listet opp. De 8 øverst i denne listen er varmeressursene. Deretter følger lokasjons- og kranressursene, hvor det også er 8 lokasjonsressurser. I og med

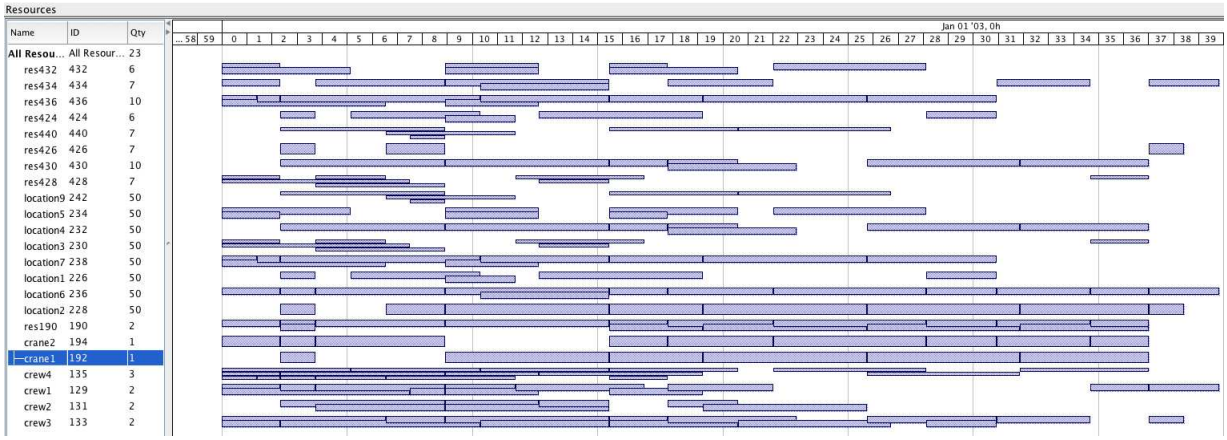


Figur 4: Ressursoversikt på Act50Loc10Crew5Crane2 med varmebegrensning, LS1, ♯2♯3

Figur 5: Ressursoversikt på Act50Loc10Crew5Crane2 uten varmebegrensning, LS1, ♯1♯3



Figur 6: Ressursoversikt på Act50Loc10Crew5Crane3 med varmebegrensning, LS1 og ♯2♯3



Figur 7: Ressursoversikt på Act50Loc10Crew5Crane2 med varmebegrensning, LS2 og ♯2♯3

at varmeressursen er knyttet til lokasjon, er det samme antall varmeressurser som lokasjonsressurser. I og med at varmebegrensningen er knyttet til en aktivitet og lokasjonene er knyttet til en aktivitet, er det aktiviteter i disse figurene som har samme varighet under varmebegrensningen og på lokasjonsressursen. Dette er tydelig til høyre på figurene hvor det ikke er så mange aktiviteter.

Figur 5 er en gitt situasjon uten varmebegrensning og figur 4 er den samme med varmebegrensning. Begge figurene har lokasjon 2 fullt utnyttet. Kran 1 er på lokasjonen og er den reelle begrensningen. Innføring av varmebegrensningen gir i dette tilfellet med kran som begrensning lite utslag i henhold til tabell 1 (liten økning i makespan). Generelt er det forventet en økning i makespan når varmebegrensing innføres. Denne økningen antas ikke i sin helhet å kun komme fra varmebegrensningen, men sannsynligvis også være et resultat at det blir vanskeligere å løse problemet.

Figur 6 kontra figur 5 er en gitt situasjon og økning fra to til tre kraner. Det er samme betrakning rundt utnyttelse, men i figur 6 endres dette. Her er det nå mannskap som over tid bruker hele kapasiteten.

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

Ved å sammenligne om verdier fra (14) er mindre eller større enn (11), gir det et uttrykk 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 kraner med LS1. Ved to kraner er kranressursen veldig sterk på kran 1, mens kranressursen ikke er fullt så sterke med tre kraner. Figur 4 viser den skjeve fordelingen mellom kran 1 og kran 2. Denne figuren viser tydelig at kran 1 er ganske godt utnyttet, mens kran 2 er mindre utnyttet. De tre aktivitetene som er helt til høyre i figur 4 krever alle kran og er tildelt kran 1. Med LS2 er det en jevnere kranfordeling som også vises i figur 7. Med en jevnere utnyttelse av kranene som i LS2 vil også makespan kunne bli bedre. Fra resultatene og målingene gjort med (14) opp mot (11) og ved å studere figur 4 og 7, virker det som å tildele kraner først i søkemålet i Solver så klarer man ikke å utnytte alle kranressursene som er tilgjengelig. Men når starttidene til aktivitetene blir tildelt først virker det som tildelingen av kraner klarer å utnytte kranressursene som er tilgjengelige på en bedre måte. En mulig forklaring kan være at ved å tildele kraner først, så vil søkerommet fortsatt være såpass stort at Solver ikke finner andre mulige kraner. Det er mulig at søkerommet ved å tildele starttider til aktiviteter først vil gjøre søkerommet mindre, så det å tildele kraner etter tildeling av starttider gjør Solver i stand til å finne beste mulige kranen til aktiviteter som krever det.

Det ble tidligere fremsatt forventninger til varmebegrensningen og kraners effekt på løsningene når antall aktiviteter øker. Dette ser i all hovedsak ut til å stemme ut ifra resultatene som fremkommer i tabell 1 og 2. At det er noen unntak, kan sannsynligvis forklares ved lav løsningsprosent, slik at optimale løsninger ikke kan etableres.

Tiden det tar å løse probleminstansene er også veldig lik uten noe spesielt mønster både med og uten varmebegrensning. Det er derfor litt bedre løsninger uten varmebegrensning.

Ved å ta utgangspunkt i LS2 uten varmebegrensing og uten sikkerhetsbegrensning på kran fra tabell 1 er makespan totalt sett 1.0% over teoretisk nedregrense med tidsgrense på 100 sekunder. Det er også løsninger på alle probleminstanser. Denne løsningen er veldig lite begrenset, da det ikke er noen grense for hvor mange mannskaper som kan være på en lokasjon og heller ingen begrensning på gjennomføring av aktiviteter ved kranbruk. Når sikkerhetsbegrensningene blir lagt til øker makespan med 13.2% og det er ikke løsninger på mer enn 36.88% av probleminstansene. Dette virker fornuftig med tanke på at sikkerhetsbegrensningene vil opprette sikkerhetssoner som vil stenge lokasjonen til kranen og stenger lokasjonen aktiviteten som bruker kranen blir utført på. Ytterligere øker makespan med 9% og det blir funnet løsninger på 40% av probleminstansene da varmebegrensningen blir lagt til. Dette viser at det kan være en tendens at innføringen av varmebegrensningen bidrar til at Scheduler finner flere løsninger. Ved å se på tabell 2 kan det se ut som tendensen også er gjeldene med en søketid på 5 sekunder. Her er løsningsprosentene for henholdsvis LS2♯1♯3 og LS2♯2♯3 36% og 37%. Når varmebegrensningen blir lagt til, begrenses antall mannskaper som kan jobbe på lokasjonene samtidig. Det kan se ut som Scheduler sliter med sikkerhetsbegrensningene på kran.

7.3.2 10 lokasjoner mot 25 lokasjoner

For å sammenligne løsningene i denne prosjektoppgaven med 10 lokasjoner, mot løsningene til Bård Henning Tvedt med 25 lokasjoner, er tabell 3 hentet fra [15]. Det benyttes henholdsvis $LS2\sharp1\sharp3$ og Inferred i sammenlikningen. Det er stort samsvar mellom både makespan og løsningsgrad. Makespan er bra og varierer

Tabell 3: 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

lite. Løsningsgrad reduseres svært mye med mange aktiviteter, både for to og tre kraner. I begge modellene tildeles tid til aktiviteter før kranressurser. Som tidligere også beskrevet om LS2 antas tendensen å skyldes at kranressurser skal tildeles i et økende antall allerede disponerte aktiviteter. I Inferred modellen i tabell 3 er makespan litt bedre enn i LS2$\sharp$1$\sharp$3 i tabell 1. Dette kan muligens forklares med reduksjonen av antall lokasjoner fra 25 i tabell 3 til 10 i tabell 1. Det blir brukt samme antall kraner, 2 og 3, som med sikkerhetssonene aktivert vil låse en mye større andel av lokasjonene i løsningene med 10 lokasjoner enn løsningene med 25 lokasjoner. Med henholdsvis 2 kraner i bruk og aktiviteter som ikke utføres på samme lokasjon som kranen befinner seg, vil det da med sikkerhetssoner bli låst 4 lokasjoner. I løsningene med 10 lokasjoner vil da 40% av lokasjonene være låst, mens med 25 lokasjoner vil bare 16% av lokasjonene være låst pga. sikkerhetssonene.

8 Relatert arbeid

Nuijten og Pape har brukt ILOG Scheduler til å løse "Job Shop Scheduling" problemet, 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 planleggingsproblemer 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 begrensninger og domener [14].

9 Fremtidig arbeid

- Erstatte sikkerhetsbegrensinger med varmebegrensning
- Innføre varmebegrensning også på kranressurser
- Utvikle egne målsetninger i Solver.

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 varmebegrensning. 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 varmebegrensning og deretter med varmebegrensning ligger løsningene henholdsvis 14.2% og 23.2% over teoretisk nedregrense.

Det kan være hensiksmessig å vurdere hvilken løsningsstrategi som skal brukes avhengig av om det er ønskelig med å ha høy løsningsgrad eller bedre makespan.

Løsningene med varmebegrensning ga med begge løsningsstrategiene noe høyere makespan enn uten varmebegrensning. Dette kan være både en reell økning av nedre grense eller være en økning av at problemet er mer komplekst å løse, eller en kombinasjon av begge. 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 4: 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 5: 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 6: 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 7: Løsninger med løsningsstrategi 2 og tidsgrense på 100 sekunder, over 100 aktiviteter

Benchmark	Nedregrense	Øvregrense	Uten varme Makespan
Act50Loc10Crew5Crane2_1	24	164	25
Act50Loc10Crew5Crane2_2	15	164	19
Act50Loc10Crew5Crane2_3	19	174	20
Act50Loc10Crew5Crane2_4	32	189	33
Act50Loc10Crew5Crane2_5	30	174	31
Act60Loc10Crew5Crane2_1	28	193	29
Act60Loc10Crew5Crane2_2	24	235	25
Act60Loc10Crew5Crane2_3	25	196	26
Act60Loc10Crew5Crane2_4	25	188	25
Act60Loc10Crew5Crane2_5	31	222	31

Act70Loc10Crew5Crane2_1	25	242	26
Act70Loc10Crew5Crane2_2	38	226	39
Act70Loc10Crew5Crane2_3	34	244	35
Act70Loc10Crew5Crane2_4	38	245	38
Act70Loc10Crew5Crane2_5	36	233	36
Act80Loc10Crew5Crane2_1	39	260	39
Act80Loc10Crew5Crane2_2	37	279	38
Act80Loc10Crew5Crane2_3	30	295	30
Act80Loc10Crew5Crane2_4	48	290	49
Act80Loc10Crew5Crane2_5	37	287	38
Act90Loc10Crew5Crane2_1	55	310	56
Act90Loc10Crew5Crane2_2	42	321	43
Act90Loc10Crew5Crane2_3	57	331	57
Act90Loc10Crew5Crane2_4	35	283	36
Act90Loc10Crew5Crane2_5	43	324	43
Act50Loc10Crew5Crane3_1	25	161	26
Act50Loc10Crew5Crane3_2	21	159	22
Act50Loc10Crew5Crane3_3	25	161	25
Act50Loc10Crew5Crane3_4	33	173	34
Act50Loc10Crew5Crane3_5	20	188	21
Act60Loc10Crew5Crane3_1	25	204	25
Act60Loc10Crew5Crane3_2	27	206	28
Act60Loc10Crew5Crane3_3	38	226	39
Act60Loc10Crew5Crane3_4	21	223	21
Act60Loc10Crew5Crane3_5	27	185	28
Act70Loc10Crew5Crane3_1	31	263	32
Act70Loc10Crew5Crane3_2	39	257	40
Act70Loc10Crew5Crane3_3	26	263	27
Act70Loc10Crew5Crane3_4	32	259	33
Act70Loc10Crew5Crane3_5	38	246	38
Act80Loc10Crew5Crane3_1	34	273	34
Act80Loc10Crew5Crane3_2	30	313	31
Act80Loc10Crew5Crane3_3	49	280	50
Act80Loc10Crew5Crane3_4	37	250	38
Act80Loc10Crew5Crane3_5	30	275	31
Act90Loc10Crew5Crane3_1	45	327	45
Act90Loc10Crew5Crane3_2	34	354	35
Act90Loc10Crew5Crane3_3	47	322	48
Act90Loc10Crew5Crane3_4	39	301	40
Act90Loc10Crew5Crane3_5	29	296	29

Tabell 8: Løsninger med løsningsstrategi 2 og tidsgrense på 100 sekunder, under 100 aktiviteter. Ingen sikkerhetsbegrensinger og uten varme

Benchmark	Nedregrense	Øvregrense	Uten varme Makespan
Act100Loc10Crew5Crane2_1	54	327	55
Act100Loc10Crew5Crane2_2	51	340	51
Act100Loc10Crew5Crane2_3	59	357	60
Act100Loc10Crew5Crane2_4	33	363	34
Act100Loc10Crew5Crane2_5	32	347	32
Act200Loc10Crew5Crane2_1	103	751	103
Act200Loc10Crew5Crane2_2	82	695	82
Act200Loc10Crew5Crane2_3	97	699	97
Act200Loc10Crew5Crane2_4	105	703	106
Act200Loc10Crew5Crane2_5	92	688	92
Act300Loc10Crew5Crane2_1	132	1034	132
Act300Loc10Crew5Crane2_2	128	1009	129
Act300Loc10Crew5Crane2_3	144	1028	144
Act300Loc10Crew5Crane2_4	141	1062	142
Act300Loc10Crew5Crane2_5	171	1069	172
Act400Loc10Crew5Crane2_1	123	1307	123
Act400Loc10Crew5Crane2_2	235	1464	236
Act400Loc10Crew5Crane2_3	188	1445	189
Act400Loc10Crew5Crane2_4	175	1385	175
Act400Loc10Crew5Crane2_5	179	1412	180
Act500Loc10Crew5Crane2_1	222	1743	222

Act500Loc10Crew5Crane2_2	246	1750	246
Act500Loc10Crew5Crane2_3	197	1665	197
Act500Loc10Crew5Crane2_4	189	1763	189
Act500Loc10Crew5Crane2_5	256	1828	257
Act600Loc10Crew5Crane2_1	263	2064	264
Act600Loc10Crew5Crane2_2	270	2061	271
Act600Loc10Crew5Crane2_3	264	2100	265
Act600Loc10Crew5Crane2_4	271	2033	271
Act600Loc10Crew5Crane2_5	272	2084	272
Act700Loc10Crew5Crane2_1	313	2356	314
Act700Loc10Crew5Crane2_2	333	2472	333
Act700Loc10Crew5Crane2_3	328	2428	329
Act700Loc10Crew5Crane2_4	358	2448	359
Act700Loc10Crew5Crane2_5	297	2346	298
Act800Loc10Crew5Crane2_1	371	2835	371
Act800Loc10Crew5Crane2_2	368	2826	368
Act800Loc10Crew5Crane2_3	368	2750	368
Act800Loc10Crew5Crane2_4	364	2809	365
Act800Loc10Crew5Crane2_5	379	2931	380
Act900Loc10Crew5Crane2_1	368	3209	369
Act900Loc10Crew5Crane2_2	399	3101	399
Act900Loc10Crew5Crane2_3	426	3204	426
Act900Loc10Crew5Crane2_4	364	3071	365
Act900Loc10Crew5Crane2_5	400	3139	401
Act1000Loc10Crew5Crane2_1	422	3452	423
Act1000Loc10Crew5Crane2_2	484	3612	485
Act1000Loc10Crew5Crane2_3	417	3463	417
Act1000Loc10Crew5Crane2_4	494	3422	494
Act1000Loc10Crew5Crane2_5	446	3484	446
Act5000Loc10Crew5Crane2_1	2154	17413	2155
Act5000Loc10Crew5Crane2_2	2182	17508	2182
Act5000Loc10Crew5Crane2_3	2089	17557	2089
Act5000Loc10Crew5Crane2_4	2262	17155	2263
Act5000Loc10Crew5Crane2_5	2291	17629	2291
Act100Loc10Crew5Crane3_1	38	350	39
Act100Loc10Crew5Crane3_2	34	345	35
Act100Loc10Crew5Crane3_3	56	372	57
Act100Loc10Crew5Crane3_4	46	365	46
Act100Loc10Crew5Crane3_5	53	326	53
Act200Loc10Crew5Crane3_1	66	671	67
Act200Loc10Crew5Crane3_2	100	751	101
Act200Loc10Crew5Crane3_3	102	710	103
Act200Loc10Crew5Crane3_4	103	699	104
Act200Loc10Crew5Crane3_5	88	669	88
Act300Loc10Crew5Crane3_1	134	1057	135
Act300Loc10Crew5Crane3_2	150	1035	150
Act300Loc10Crew5Crane3_3	140	1046	140
Act300Loc10Crew5Crane3_4	134	1058	135
Act300Loc10Crew5Crane3_5	122	1029	123
Act400Loc10Crew5Crane3_1	185	1363	185
Act400Loc10Crew5Crane3_2	177	1424	178
Act400Loc10Crew5Crane3_3	173	1345	174
Act400Loc10Crew5Crane3_4	211	1370	211
Act400Loc10Crew5Crane3_5	183	1395	184
Act500Loc10Crew5Crane3_1	181	1693	182
Act500Loc10Crew5Crane3_2	227	1719	227
Act500Loc10Crew5Crane3_3	217	1738	217
Act500Loc10Crew5Crane3_4	230	1705	231
Act500Loc10Crew5Crane3_5	205	1762	206
Act600Loc10Crew5Crane3_1	280	2095	280
Act600Loc10Crew5Crane3_2	280	2088	280
Act600Loc10Crew5Crane3_3	307	2167	307
Act600Loc10Crew5Crane3_4	258	2053	258
Act600Loc10Crew5Crane3_5	261	2043	261
Act700Loc10Crew5Crane3_1	329	2424	329
Act700Loc10Crew5Crane3_2	338	2516	338
Act700Loc10Crew5Crane3_3	340	2415	340
Act700Loc10Crew5Crane3_4	333	2511	333
Act700Loc10Crew5Crane3_5	304	2386	304
Act800Loc10Crew5Crane3_1	374	2830	375
Act800Loc10Crew5Crane3_2	375	2781	376

Act800Loc10Crew5Crane3_3	406	2949	406
Act800Loc10Crew5Crane3_4	389	2862	390
Act800Loc10Crew5Crane3_5	364	2782	365
Act900Loc10Crew5Crane3_1	433	3156	434
Act900Loc10Crew5Crane3_2	420	3167	420
Act900Loc10Crew5Crane3_3	424	3230	424
Act900Loc10Crew5Crane3_4	418	3169	418
Act900Loc10Crew5Crane3_5	453	3111	453
Act1000Loc10Crew5Crane3_1	456	3534	457
Act1000Loc10Crew5Crane3_2	399	3538	400
Act1000Loc10Crew5Crane3_3	439	3545	439
Act1000Loc10Crew5Crane3_4	477	3639	477
Act1000Loc10Crew5Crane3_5	458	3462	458
Act5000Loc10Crew5Crane3_1	2162	17347	2163
Act5000Loc10Crew5Crane3_2	2190	17682	2190
Act5000Loc10Crew5Crane3_3	2097	17551	2098
Act5000Loc10Crew5Crane3_4	2203	17539	2204
Act5000Loc10Crew5Crane3_5	2292	17536	2292

Tabell 9: Løsninger med løsningsstrategi 2 og tidsgrense på 100 sekunder, over 100 aktiviteter. Ingen sikkerhetsbegrensinger og uten varme

Referanser

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

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

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

[4] Some challenges for constraint programming. http://clip.dia.fi.upm.es/herme/con.html.

[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. *Melbourne School of Engineering*.

[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. Project Management anf Scheduling.

[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.