

Automatisk Planlegging i Oljeindustrien

av

Teis Lindemark

v0.2.1

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 av dette arbeidet. Takk til at dere brukte av deres dyrebare tid for å veilede meg gjennom denne oppgaven.

Ellers må jeg også takke mine foreldre for god støtte gjennom mastergraden, som har støttet meg gjennom gode og mindre gode tider. Når jeg ikke så enden på tunnelen, var det alltid godt å kunne pratet ut og fått nødvenndig støtte. Tilslutt må jeg også takke foreldrene mine for at de kunne hjelpe til med å korrekturlese oppgaven.

Teis Lindemark, 6. Mai 2012

Innhold

| | | |
|----------|--|-----------|
| 1 | Introduksjon | 8 |
| 1.1 | Beskrivelse av kommende kapitler | 8 |
| 1.2 | Bakgrunn | 8 |
| 2 | Motivasjon og målsetting | 9 |
| 2.1 | Målet med prosjektet | 10 |
| 2.2 | Relevans for forskingen | 11 |
| 3 | Problemstilling | 11 |
| 3.1 | Notasjoner og terminologi | 12 |
| 3.2 | Ressurser | 13 |
| 3.3 | Aktiviteter | 13 |
| 3.4 | Begrensinger | 14 |
| 3.5 | Målfunksjon | 16 |
| 3.6 | Probleminstanser | 16 |
| 4 | Begrensningsprogrammering | 17 |
| 4.1 | Kort om begrensningsprogrammering | 17 |
| 4.2 | Utfordringer med begrensningsprogrammering | 17 |
| 4.3 | Begrensningsprogrammeringsverktøy idag | 18 |
| 4.4 | Verktøy brukt i prosjektet | 19 |
| 4.4.1 | Kort om Concert | 19 |
| 4.4.2 | Kort om Solver | 20 |
| 4.4.3 | Kort om Scheduler | 20 |
| 5 | Metode | 21 |
| 5.1 | Forskningsmetoder | 21 |
| 5.1.1 | Implementeringsprosessen | 21 |
| 5.1.2 | Evaluering av prosessen | 22 |
| 5.2 | Evalueringsstrategi | 23 |
| 5.2.1 | Teoretisk øvre grense | 23 |
| 5.2.2 | Teoretisk nedre grense | 23 |
| 6 | Resultater | 24 |

| | | |
|-----------|---|-----------|
| 6.1 | Uten varmebegrensning | 25 |
| 6.2 | Med varmebegrensning | 25 |
| 7 | Evaluering | 26 |
| 7.1 | Probleminstanser | 26 |
| 7.2 | Løsningsstrategier | 27 |
| 7.3 | Resultater | 27 |
| 7.3.1 | 10 lokasjoner | 27 |
| 7.3.2 | 10 lokasjoner mot 25 lokasjoner | 29 |
| 8 | Relatert arbeid | 30 |
| 9 | Fremtidig arbeid | 31 |
| 10 | Konklusjon | 31 |
| 11 | Vedlegg | 32 |

Tabeller

| | | |
|---|--|----|
| 1 | Relativ optimalitets indeks w_{rq} for de forskjellige modellene | 24 |
| 2 | Relativ optimalitets indeks w_{rq} for de forskjellige modellene | 30 |
| 3 | Løsninger med løsningsstrategi 1 med tidsgrense 100 sekunder . . | 36 |
| 4 | Løsninger med løsningsstrategi 2 og tidsgrense på 100 sekunder . | 39 |

Figurer

- 1 Ressursoversikt på Act50Loc10Crew5Crane2 med varmebegrensning 28
- 2 Ressursoversikt på Act50Loc10Crew5Crane2 uten varmebegrensning 28
- 3 Ressursoversikt på Act50Loc10Crew5Crane3 med varmebegrensning 29

Forkortelser

| | |
|--------------------|---|
| AI | Kunstig intelligens (engelsk: artificial intelligence) |
| APO | Automatisk Planlegging i Oljeindustrien |
| Avgjøringsvariable | På engelsk: decision variable |
| Avledningsvariable | 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 | Løsningsstrategi 2 |
| MSPProject | 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 med ”Automatisk Planlegging i Oljeindustrien (APO)” [15]. APO er en planleggingsmetodikk for å løse et problem med planlegging av vedlikeholdsaktiviteter i oljeindustrien. Det blir brukt en fiktiv oljeplattform for å kunne gjøre problemet så likt som mulig virkeligheten. Implementasjonen i IBM ILOG Scheduler (Scheduler) er utvidet med ressurser på det som kalles varmebegrensning. Med bruk av varme er det 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 eksternt program, som kalkulerer teoretisk- øvre grense og nedre grense og sjekker om begrensningene er tilfredstilt.

1.1 Beskrivelse av kommende kapitler

I kapittelet om metode, vil fremgangsmåten for prosjektet bli lagt frem og hvordan løsningene har blitt evaluert. Under kapittelet om eksperimenteringen vil det legges frem løsninger med forskjellige strategier, både med og uten varmebegrensning. Løsningene vil også bli evaluert og til slutt vil det bli sammenfattet en konklusjon over det arbeidet som er gjort.

1.2 Bakgrunn

I dette prosjektet, verktøyene som er brukt for å utføre eksperimenter på emnet er Scheduler som er endel 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 aktiviteter slik at det ikke oppstår noen konflikt med begrensningene [16]. Automatisk planlegging er endel av det som kalles kunstig intelligens (AI) . **Skriv litt mer om sammenheng AI og prosjektet**

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 ma-

skin. Det er forskjellige kategorier av NP-harde problemer, avgjøresproblemer, søkeproblemer og optimaliseringsproblemer [8]. Problemet i dette prosjektet 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 og sett på hvordan best mulig benchmarke vanskelige problemer. De har definert vanskelige problemer med å se på makespan og hvor langt unna den er den nedre grensen. De har tatt utgangspunkt i tre kjente begrensingsproblemer og laget benchmarksett til disse. Taillard har definert nedregrensen (C_{max}^*) i JSSP som større enn eller lik maksimale mellom den minste tiden krevd av maskinene og den minste tiden krevd av hver enkel jobb. Wallace [18] har også sett på benchmarking og tatt for seg positive og negative sider ved benchmarking. Benchmarking har to hovedoppgaver, hvor det første er å dekke et representativt problem og bredt programmerings begrep. Den andre oppgaven er benchmarking av applikasjoner med ”unit tester”. Utfordringen med begge er å opprette en klar målsetting for ytelsen. Teoretisk sett er den mest ideelle måten å benchmarke en applikasjon å implementere en løsning for hvert system og velge den beste, men dette er ikke alltid mulig i praksis. Måten da å gjøre det på i praksis er å benchmarke systemer med forhåndsdefinerte problemer og håpe resultatet kan bli overført til applikasjonen som krever det. Wallace tar også opp vanskeligheten med tidsmekanismer i høynivåspråk, da dette ofte er basert på CPU tid.

2 Motivasjon og målsetting

Forskningen er motivert av praktisk erfaring fra planleggingsprosesser i bedrifter og samfunnet forøvrig. Det å finne optimale løsninger på planleggingsoppgaver er viktig og ressurs besparende. Det er ikke bare i oljeindustrien som det fokuseres på i denne oppgaven hvor planleggingsløsninger er aktuelt, men generelt bemannings- og ressursplanlegging som hele næringslivet i større eller mindre grad er berørt av. I menneskers personlige hverdagen er det også mange planleggingsproblemer, alt fra buss- og togtabeller til personlige gjøremål som skal inn i en stram tidsplan.

Planlegging i oljeindustrien er viktig for på en mest mulig effektiv måte benytte

seg av de ressursene som er tilgjengelige til enhver tid, samtidig som visse begrensinger blir fastsatt med tanke på blandt annet sikkerheten. Det er mye penger involvert i olje- og gassindustrien og det å utføre vedlikeholdsaktiviteter på en ineffektiv måte kan det for selskapene både bli svært kostbart og også medføre risiko hvis sikkerheten ikke er godt nok ivaretatt. Det er derfor viktig å ha gode løsninger for å ta seg av planleggingen av vedlikeholdsaktivitetene og ressursene. Operatører innen olje- og gassektorens mål er å minimere antallet farlige situasjoner, minimere miljømessige skade og maksimere produksjon og dermed økonomisk resultat. På veien for å nå disse kompliserte og utfordrende målene, kan det i enkelte situasjoner oppstå konflikter (overfylte lokasjoner, manglende utstyrskrav, feil i rekkefølgen aktiviteter utføres, etc.).

Spesifikke planleggingssituasjoner i oljeindustrien har forskjellige forutsetninger i forhold til om plattformen er offshore eller på land. Offshore kan mange aktiviteter bli utført på et relativt lite lukket område, mens områder på land kan ha mange aktiviteter utført på et relativt stort område. Selv om forutsetningene for disse to typene plattformer er ganske forskjellige, så har de flere likhetstrekk som:

- størrelse - antall aktiviteter kan være noen hunder til titusener, for å gjøre planleggingen interaktiv.
- kompleksitet - et stort antall av begrensningene som skal bli gjennomført, gjør en mulig planlegging vanskelig.
- dynamikk - avhengigheter som vær, logistikk og utstyr som feiler kan avbryte planleggingen.

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 ILOG Scheduler

I den opprinnelige problemstillingen vil noen aktiviteter være 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 instansene eller er det implementasjon i Scheduler som er årsaken til at Scheduler implementerte løsningsstrategien sliter med å finne løsninger i mange av probleminstansene?

2.2 Relevans for forskingen

Denne forskningen er ment for å undersøke om løsningene blir bedre når et eksisterende problem får lagt til en ekstra begrensning, kalt varmebegrensning. Ved å evaluere løsningene med og uten varmebegrensning vil det si noe om løsningene når søketreet er stort i forhold til når søketreet er litt mindre etter å ha lagt til den ekstra begrensningen.

3 Problemstilling

Problemstillingen til denne oppgaven tar utgangspunkt i den opprinnelige problemstillingen til Bård Henning Tvedt . Beskrivelsen følger i første omgang [15], som jeg har oversatt norsk. Problemet er på en innbilt 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 muligens 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. Mannskap- og kranressurser er knappe, som betyr at de er 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 prosesseringstid
- 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, selvom det også er et bredt antall planleggingsproblemer som gjør det. Det er mange tilleggsbegrensinger, typisk i oljeindustrien og andre store industrier, som ikke passer inn i denne klassifiseringen. Siden målet er å generere probleminstanser med begrensinger 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 aktivitetene og begrensninger som blandt annet er begrensinger mellom aktiviteter og ressursbruk. Det blir skillt mellom forskjellige typer variable som *avgjørelsesvariable*, *konstanter* og *avledetvariable*. Et eksempel på en avgjøringsvariabel 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 $c_{dur}(Act_i)$. Tilslutt så er det avledetvariable 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 en stor bokstav.

3.2 Ressurser

En *lokasjon* $Loc_l \in Locs = \{Loc_1, \dots, Loc_n\}$ er stedet hvor aktiviteter blir utført. Selv om lokasjoner blir vist som ressurser, så er det ikke noen begrensinger på hvor mange aktiviteter som kan bli utført samtidig på en lokasjon. Det er 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 $Crew_j \in Crews = \{Crew_1, \dots, Crew_n\}$.

En *kran* $Crane_k \in Cranes = \{Crane_1, \dots, Crane_n\}$ 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 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.

3.3 Aktiviteter

En *aktivitet* $Act_i \in Acts = \{Act_1, \dots, Act_n\}$ 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

reduere varigheten. Kraner er den siste ressursen som er tilgjengelig, 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 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:

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

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

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

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:

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

Den gjensidige uttelukkelsen opprettet av den monopolatoriskeressursbegrensningen blir da:

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

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 begrensningene 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 sikkerhetsbegrensningene utelukker bruken av lokasjonen til denne kranen 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. Varmebegrensningen skal også etterhvert kunne erstatte sikkerhetsbegrensningene på for eksempel kranbruk, ved at lokasjoner med kran får en varmekapasitet og aktiviteter som bruker kran bruker opp denne varmekapasiten på lokasjonen. Den er uttrykt ved:

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

3.5 Målfunksjon

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

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

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 gene-

rert 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 programmeringsspråk¹ ved at løsningen blir til ved å tilfredsstille begrensningene. Det er forskjellige områder i begrensningsprogrammering som Constraint Satisfaction problems² og planleggingsproblemer. Det mest kjente planleggingsproblemet er "Job Shop Scheduling" [5].

4.2 Utfordringer med begrensningsprogrammering

Systemer for begrensings-logikk programmeringssystemer som sammenlignes med begrensningsløsningssystemer, er ofte ytelsen bedre i begrensings-logikk programmeringssystemer. Ytelsen er likevel ikke like god som mer tradisjonelle imperative 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 begrensingsløsning, og da kompilerer disse på mest mulig effektive måte [1]

¹Imperativ programmeringsspråk har sekvenser med instanser som blir utført.

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 feilsøkingmuligheter 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 oppnådd med metoden. Postforhold er også en boolsk verdi. Både pre- og postforhold blir skrevet for å ”evaluere” en metode, enten før metoden blir utført eller hva metoden har oppnådd. 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 Begrensningsprogrammeringsverktøy 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 programmeringsspråk for begrensningsprogrammering med lokalt søk og er en kommersiell løsning. Eksempler på begrensningsprogrammeringsbibliotek 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 tjenestene, 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. Dette programmet har et grafisk grensesnitt som andre programmer i Office-pakken til Microsoft og er

et enkelt program å bruke. Microsoft Project 2010 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 begrensninger 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.), begrensninger og sikkerhetsbegrensninger. 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 begrensingsprogrammering. 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 eksempel på det er IloConstraintArray som er instanser av klassen IloConstraint [13].

Concert gjør det mulig å lage en modell av optimaliseringsproblemer uavhengig av algoritmene som er brukt for å løse det. Det tilbyr en utvidelse modelerings lag

tatt fra flere forskjellige algoritmer som er klare til å brukes ut av boksen. Dette modeleringslaget gjør det mulig å endre modellen uten å skrive om applikasjonen [12].

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 begrensingsprogrammering 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 besluttningsvariablene.

Det siste steget er å bruke Solver for å løse problemet. Det inneholder å finne verdier for alle besluttningsvariablene 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 kombinasjoner av verdier.[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 begrensingsprogrammering. Scheduler har spesifisert funksjonalitet på å løse problemer innen planlegging og ressurstildeling.

I Scheduler er det blandt annet kummulative ressurser, eneressurser, aktiviteter og alternative ressurser.

5 Metode

I denne delen, blir verktøy og teknologier som er brukt i prosjektet beskrevet. I tillegg vil forskningsmetoder 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.

5.1.1 Implementeringsprosessen

Måten kraner blir tildelt til aktiviteter som krever kran er endret fra løsningen til Bård Henning Tvedt . Kraner er modellert som alternative ressurser og for ...

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 varme ressursen 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 IBM ILOG Solver og IBM ILOG Scheduler bibliotekene. 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 ressursbegrensningene. 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 `IloAssignAlternated` 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

Proessen med eksperimenteringen av implementasjonen blir evaluert ved å undersøke makespan opp mot teoretisk øvre grense (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 10 lokasjoner vil også bli evaluert opp mot løsningene med 25 lokasjoner, men da vil gjennomsnittsverdien 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.

Som nevnt evalueres løsningene på probleminstansene med et eksternt program. 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 og varme.

5.2 Evalueringsstrategi

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

5.2.1 Teoretisk øvre grense

Teoretisk øvre grense for makespan er

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

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.

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

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

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

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

⁽¹⁾ prosentandel løste probleminstanser ⁽²⁾ $\#1$ er løsninger uten varmebegrensning ⁽³⁾ $\#2$ 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).

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

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 begrensede 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 varmebegrensning 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.

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 løsningsstrategien, er makespan 14.2% over teoretisk nedregrense og den finner løsninger på 36.88% av probleminstansene.

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 løsningsstratteien, 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 løsningsstrategien er gjennomsnittsverdien 81.0% over teoretisk nedregrense.

Med den andre løsningsstrategien finner den bedre løsninger, men generelt sett

færre av probleminstansene blir løst. Den beste løsningen her har gjennomsnittet 4.1% over teoretisk nedregrense. Den dårligste løsningen ligger på 31.3% over teoretisk nedregrense. Der har 18.18% og 30.91% av probleminstansene blitt løst. Totalt sett med den andre løsningsstrategien ligger det 23.2% over teoretisk nedregrense og 40% av probleminstansene er løst med denne løsningsstrategien.

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 varmekapasitet 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 løsningsstrategien 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, uavhengig 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 nedregrense. 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%.

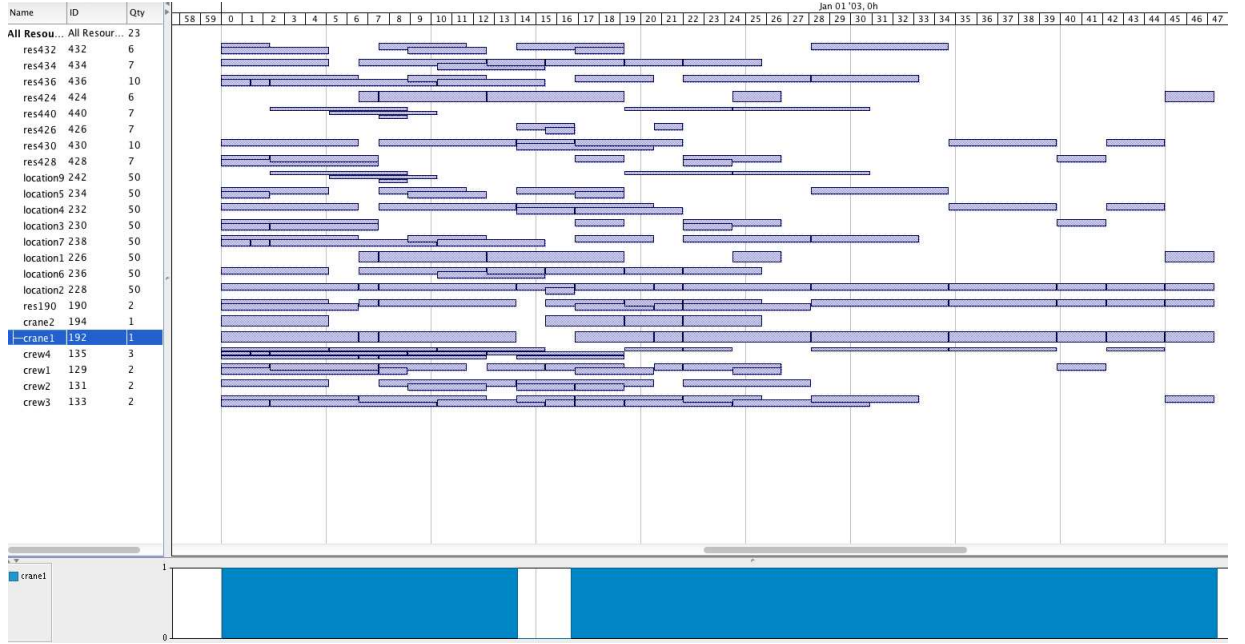
7.3 Resultater

I dette prosjektet er en eksisterende løsning utvidet med en ressurs kalt varme-ressurs. 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 av resultatene baserer seg på gjennomsnittsverdiene i tabell 1. En tilsvarende oppsummering av løsningene eksisterer også for 25 lokasjoner [15].

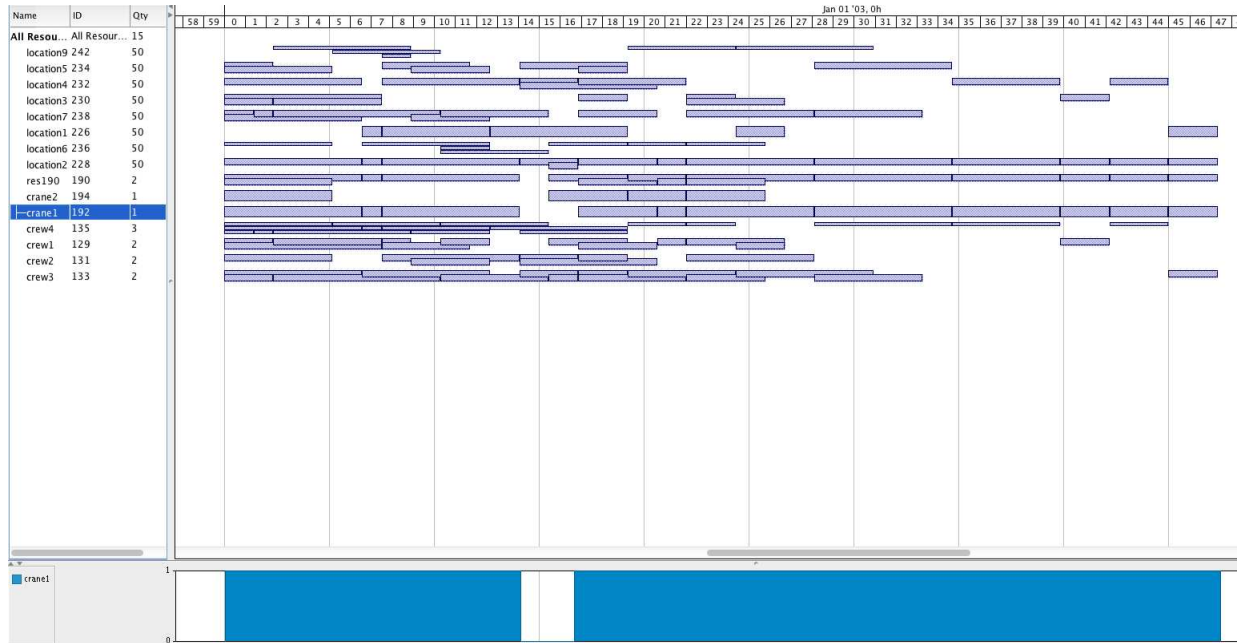
7.3.1 10 lokasjoner

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 1 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 2 er det også kran som er den mest begrensende ressursen. Når ser på figur 3 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.



Figur 1: Ressursoversikt på Act50Loc10Crew5Crane2 med varmebegrensning



Figur 2: Ressursoversikt på Act50Loc10Crew5Crane2 uten varmebegrensning

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



Figur 3: Ressursoversikt på Act50Loc10Crew5Crane3 med varmebegrensning

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

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

| Modell | 2 kraner | | | | 3 kraner | | | | Alle | |
|----------------------|-----------|------------------|-----------|------------------|-----------|------------------|-----------|------------------|----------|------------------|
| $\#Act(\#P)$ | < 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#1 | 1.143 | 100 | 1.076 | 100 | 1.176 | 100 | 1.037 | 91 | 1.114 | 98 |
| Over#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

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 løsningsstrategi 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. Begrensningsbasert planlegging blir ofte brukt for å redusere beregningstiden som trengs. Begrensingene reduserer domenet av variable. I industrien er ofte planleggingsproblemer utført i dynamiske miljøer, som vil øke behovet for reaktive planleggingsalgoritmer. En mulighet for å bruke begrensingsprogrammering 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 dette arbeidet har vært å se på utfordringer innen vedlikeholdsplanlegging. Det går ut på å prøve ut forskjellige løsningsstrategier for å ta hånd om komplekse problemer, i et veldig dynamisk miljø som olje og gass industrien.

Den første løsningsstrategien (LS1) som tildelte kraner først løser alle probleminstansene, uavhengig av størrelsen og kompleksiteten. Løsningene ligger her 74.5% og 81.0% over teo

Den andre løsningsstrategien (LS2) som tildeler starttidspunkt til alle aktivitetene først, løser ikke alle probleminstansene, men har de beste løsningene.

Løsningene med varmebegrensning var med begge løsningsstrategiene noe dårligere enn uten varmebegrensning. Det ser ut som ut ifra disse resultatene det er mer på løsningsstrategien hvor mange probleminstanser som blir løst.

11 Vedlegg

| Benchmark | Nedregrense | Øvregrense | Uten varme Makespan | Med 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 |
| 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 |
| 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 |
| 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 3: Løsninger med løsningsstrategi 1 med tidsgrænse
100 sekunder

| | | | | |
|--|--|--|-----------|------------|
| | | | Med varme | Uten varme |
|--|--|--|-----------|------------|

| Benchmark | Nedregrense | Øvregrense | Makespan | 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 |
| 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 | - |
| 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 |
| 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 4: Løsninger med løsningsstrategi 2 og tidsgrense på 100 sekunder

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. jce:titlejProject Management anf Schedulingi/ce:titlej.
- [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.