

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

Sammendrag

TEST

Innhold

1	Introduksjon	7
1.1	Beskrivelse av kommende kapitler	7
1.2	Bakgrunn	7
1.2.1	Relatert arbeid	8
1.3	Målet med prosjektet	8
1.4	Motivasjon	9
1.4.1	Kort om begrensingsprogrammering	10
1.4.2	Utfordringer med begrensingsprogrammering	10
1.4.3	Begrensingsprogrammeringsverktøy idag	10
1.4.4	Forbedringspotensiale i verktøyene	11
1.4.5	Relevans for forskingen	11
1.5	Problembeskrivelse	11
1.5.1	Notasjoner og terminologi	12
1.5.2	Ressurser	13
1.5.3	Aktiviteter	13
1.5.4	Begrensinger	14
1.5.5	Målfunksjon	15
1.5.6	Probleminstanser	15
2	Metode	16
2.1	Verktøy brukt i prosjektet	16
2.1.1	Kort om IBM ILOG Concert Technology	16
2.1.2	Kort om IBM ILOG Solver	17
2.1.3	Kort om IBM ILOG Scheduler	17
2.2	Forskningsmetoder	18
2.2.1	Implementeringsprosessen	18
2.2.2	Evaluering av prosessen	18
2.3	Evalueringsstrategi	19
2.3.1	Teoretisk øvre grense	19
2.3.2	Teoretisk nedre grense	19
3	Eksperimenter	20
3.1	Uten varmeressurs	20
3.2	Med varmeressurs	21
4	Evaluering	21
4.1	Probleminstanser	21
4.2	Løsningsstrategier	22
4.3	Resultater	22
4.3.1	Med 10 lokasjoner	22
4.3.2	10 lokasjoner mot 25 lokasjoner	22

5	Fremtidig arbeid	22
6	Konklusjon	22
7	Vedlegg	23

Forord

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
NP-hard	Ikke-deterministisk polynomtid hard (engelsk: non-deterministic polynomial-time hard)
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 et planleggingsverktøy for å løse et problem med planlegging av bruk av ressurser i oljeindustrien. Det blir brukt en fiktiv oljeplattform for å kunne gjøre problemet så likt som mulig virkeligheten. Implementasjonen i IBM ILOG 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 kapasitet en ressurs bruker. Løsningene blir evaluert med et eksternt program.

1.1 Beskrivelse av kommende kapitler

I kapitlet om metode, vil fremgangsmåten for prosjektet bli lagt frem og hvordan løsningene har blitt evaluert. Under kapitlet om eksperimenteringen vil det legges frem løsninger ved forskjellige strategier og med og uten ressursene. 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 ILOG Scheduler som er endel av IBM sitt ILOG CP. ILOG 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) .

Problemer i kategorien *beregningsvitenskaps kompleksitets teori*[7] og som også er minst like vanskelige å løse som de vanskeligste probleme i NP-kategorien, kalles *ikke-deterministisk polynomtid problemer vanskelige* (NP-hard) . Det er forskjellige kategorier av NP-harde problemer, avgjøreskesproblemer, 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”.

1.2.1 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]

Taillard 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 [17]. Wallace 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 [18].

1.3 Målet med prosjektet

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

- minimere *makespan*
- øke antall begrensninger
- 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 ILOG

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 ILOG Scheduler som er årsaken til at ILOG Scheduler implementerte løsningsstrategien sliter med å finne løsninger i mange av probleminstansene?

1.4 Motivasjon

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 på 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 begrensninger blir fastsatt med tanke på blandt annet sikkerheten. Det er mye penger involvert i olje- og gassindustrien og det å utføre aktiviteter 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 aktivitetene 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.

1.4.1 Kort om begrensingsprogrammering

Begrensingsprogrammering 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 begrensingsprogrammering som Constraint Satisfaction problems og planleggingsproblemer. Det mest kjente planleggingsproblemet er Job Shop Scheduling.[5]

1.4.2 Utfordringer med begrensingsprogrammering

Systemer for begrensings-logikk programmeringssystemer som sammenlignes med begrensingsløsningssystemer, 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]

Begrensingsprogrammeringssystemer har ofte en svakhet når det kommer til feilsøking (engelsk: debugging). Uten tilstrekkelige måter å kunne være sikkert på riktigheten og muligheter å sjekke ytelse i programmer. En måte å løse utfordringene med manglende feilsøkingsmuligheter er å bruke påstander . Ved bruk av påstander kan det opprettes post- og preforhold. Påstander kan bli sjekket ved kompilering eller når programmet kjører. Det er også mulig å generere påstander av kompilatoren, som utvikleren kan sjekke om det eksisterer høynivåfeil[1].

1.4.3 Begrensingsprogrammeringsverktøy idag

Det finnes idag flere forskjellige verktøy for begrensingsprogrammering, både i form av egne programmeringsspråk som er skreddersydd for begrensingsprogrammering 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 begrensingsprogrammering er Prolog og Comet. Sistnevnte er et programmeringsspråk

¹Imperativ programmeringsspråk har sekvenser med som blir utført.

for begrensingsprogrammering med lokalt søk og er en kommersiell løsning. Eksempler på begrensingsprogrammeringsbiblotek så er det IBM ILOG CP.

Planlegging er ofte tett knyttet opp mot prosjektstyring og prosjektstyringsverktøy finnes det veldig mange av etterhvert [6]. Både programmer du har lokalt på maskinen og også webbaserte tjenester. Fellesnevneren for veldig mange av disse tjeneste, enten de er lokalt eller webbaserte er at de skal hjelpe til med alt fra prosjektplanlegging, dokumentdeling, oversikt over oppgaver, oversikt over frister, møteplanlegging og mye mer. Denne typen programvare er ofte gjort ganske enkle å bruke. I samme kategori er det også noen programmer som gjør det mulig å definere aktiviteter (eller 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 Software er et eksempel på et slikt system, hvor det er mulig å utføre planlegging og produksjons gjennomføring. Dette systemet har funksjonalitet å opprette aktiviteter, planlegge aktiviteter, materialkravplanlegging osv [2].

1.4.4 Forbedringspotensiale i verktøyene

1.4.5 Relevans for forskningen

Denne forskningen er ment for å undersøke om resultatene blir bedre når et eksisterende problem får lagt til en ekstra ressurs, 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 ressursen.

1.5 Problembeskrivelse

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 plattformen. 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.

1.5.1 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.

1.5.2 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 monooperatorressurser 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.

1.5.3 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 redusere 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.

1.5.4 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:

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

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

Kraner er unikt induviduelle 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 varmekapasiteten på lokasjonen. Den er uttrykt ved:

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

1.5.5 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.

1.5.6 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 aktivitetenes 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. Mannskapers varme er tilfeldig generert i området: $c_{heat}(Crew_j) \in [5, 15]$, mens lokasjoners varmekapasitet er tilfeldig generert i området: $c_{heatcap}(Loc_l) \in [5, 20]$.

2 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.

2.1 Verktøy brukt i prosjektet

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

2.1.1 Kort om IBM ILOG Concert Technology

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

2.1.2 Kort om IBM ILOG 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++.

Det å gjøre det enkelt mulig å omgjøre applikasjoner fra plattform til plattform, Solver og Concert utelukkes karaktertrekk som skiller seg fra forskjellige systemer. Av den grunn, anbefales det å bytte ut de enkle typene i C++ med ILOG sine egne:

- `IloInt` som er signed long integers
- `IloAny` som er pekere
- `IloNum` som er double presisjon floating-point verdier
- `IloBool` som er boolean verdier: `IloTrue` og `IloFalse`

Solver bruker begrensingsprogrammering for å finne løsninger til optimaliseringsproblemer. Det å finne løsninger med Solver er basert på tre steg: beskrive, modell 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 kombinasjoners av verdier.[12]

2.1.3 Kort om IBM ILOG Scheduler

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.[11]

2.2 Forskningsmetoder

Forskningsmetoden som er brukt i prosjektet er å eksperimentere med implementasjonen av ressursene og løsningsstrategien.

2.2.1 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 ILOG 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 rangert, 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.

Den andre løsningsstrategien (LS2) tillegner starttid til alle aktivitetene først, ved å bruke `IloSetTimesForward`. Når alle aktivitetene har fått tildelt en starttid brukes `IloAssignAlternatives` 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.

2.2.2 Evaluering av prosessen

Proessen med eksperimenteringen av implementasjonen blir evaluert ved å undersøke makespan opp mot teoretisk øvre grense (se 2.3.1) og teoretisk nedre-

grense (se 2.3.2) i både løsningene uten tilleggsressurser og med tilleggsressurser. Løsningene fra prosessen med og uten tilleggsressursene vil også bli evaluert opp mot hverandre. Dette innebærer bruk av kvantitative metoder.

Forskningsmetoden vil bli evaluert ved å bruke genererte benchmarksett, som er generert av et eksternt program. Benchmarksettene som genereres kan bestemmes hvor mange av de forskjellige ressursene som skal være med i benchmarksettet. I dette prosjektet er det et sprang på 50 - 5000 aktiviteter som implementasjon blir evaluert på.

2.3 Evalueringsstrategi

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

2.3.1 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.

2.3.2 Teoretisk nedre grense

En teoretisk nedre grense 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.

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

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

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

3 Eksperimenter

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.

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}	$\%^{(1)}$
$LS1\#1^{(2)}$	1.164	80	1.115	44	1.105	64	1.027	20	1.113	44
$LS1\#2^{(3)}$	1.228	32	-	-	-	-	-	-	1.183	5
$LS2\#1^{(2)}$	1.691	100	1.855	100	1.545	100	1.693	100	1.725	100
$LS2\#2^{(3)}$	1.651	40	1.860	1.8	-	-	-	-	1.670	6.8

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

⁽¹⁾ prosentandel løste probleminstanser ⁽²⁾ $\#1$ er løsninger uten varmeressurs ⁽³⁾ $\#2$ er løsninger med varmeressurs

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

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

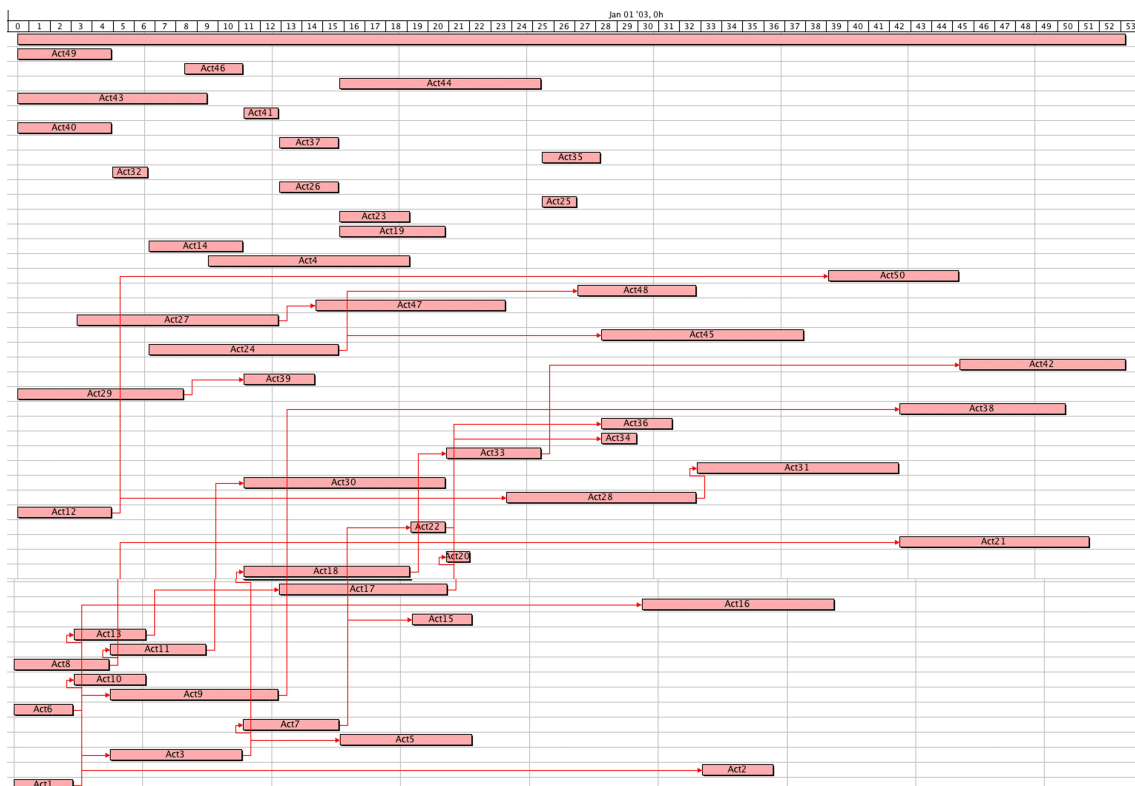
P_{sol} er det settet med probleminstanser som er løst ved hver enkelt løsningsstrategi. Verdien av P_{sol} varierer fra løsningsstrategi til løsningsstrategi og disse gjennomsnittene er derfor ikke helt sammenlignbare, men de vil gi en indikasjon på kvaliteten på løsningen. w_{rq} skiller ikke på strategier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

3.1 Uten varmeressurs

Resultatene når benchmarksettene er kjørt gjennom løsningen uten varmebegrensning er markert med $\#1$, viser det er løsning i tilfellene med både 2 og 3 kraner. Der hvor den første løsningsstrategi (LS1) er benyttet varierer antallet løste benchmarksett fra 44% og opp til 80%, mens for den andre løsningsstrategien (LS2) er det løsninger på alle benchmarksettene.

3.2 Med varmeressurs

Resultatene som er kjørt med varmebegrensing, finner ikke i nærheten av like mange løsninger på banchmarksettene. Med den første løsningsstrategien (LS1) finner den ikke løsninger for mer enn banchmarksettene med under 100 aktiviteter og 2 kraner. Der finner den heller ikke løsning på mer enn 32% av benchmarksettene.



Figur 1: Gant skjema over en av løsningene med 50 aktiviteter og 2 kraner

4 Evaluering

Evaluering intro

4.1 Probleminstanser

Evaluere probleminstansene

4.2 Løsningsstrategier

Evaluering av løsningsstrategiene

4.3 Resultater

Resultater intro I dette prosjektet er en eksisterende løsning utvidet med en ressurs kalt varmeressurs. For å evaluere resultatene fra den utvidete løsningen, kommer det en evaluering av den eksisterende løsningen med 25 lokasjoner sett opp mot den utvidede løsningen med 10 lokasjoner. Evalueringen av resultatene baserer seg på gjennomsnittsverdiene i tabell 1. En tilsvarende oppsummering av løsningene eksisterer også for 25 lokasjoner [15].

4.3.1 Med 10 lokasjoner

Evaluering av resultatene med 10 loc mot hverandre

4.3.2 10 lokasjoner mot 25 lokasjoner

Evaluere løsninger med 10 loc mot Bård Henning Tvedt 25 lokasjoner

5 Fremtidig arbeid

Videre arbeid

6 Konklusjon

Konklusjon

7 Vedlegg

Benchmark	Nedregrense	Øvregrense	Uten varme Makespan	Med varme Makespan
Act50Loc10Crew5Crane2_1	28	177	34	34
Act50Loc10Crew5Crane2_4	21	172	22	22
Act50Loc10Crew5Crane2_5	22	177	27	28
Act60Loc10Crew5Crane2_1	29	211	32	32
Act60Loc10Crew5Crane2_2	31	215	39	41
Act60Loc10Crew5Crane2_3	31	209	38	36
Act60Loc10Crew5Crane2_4	27	203	27	27
Act60Loc10Crew5Crane2_5	27	205	46	46
Act70Loc10Crew5Crane2_1	39	249	39	-
Act70Loc10Crew5Crane2_2	38	238	54	-
Act70Loc10Crew5Crane2_3	34	267	35	-
Act70Loc10Crew5Crane2_4	39	270	40	-
Act70Loc10Crew5Crane2_5	26	231	27	-
Act80Loc10Crew5Crane2_2	35	272	38	-
Act80Loc10Crew5Crane2_4	40	278	47	-
Act80Loc10Crew5Crane2_5	47	258	52	-
Act90Loc10Crew5Crane2_1	50	298	53	-
Act90Loc10Crew5Crane2_2	42	340	53	-
Act90Loc10Crew5Crane2_3	43	314	48	-
Act90Loc10Crew5Crane2_4	43	303	51	-
Act100Loc10Crew5Crane2_1	50	356	50	-
Act100Loc10Crew5Crane2_2	66	364	67	-
Act100Loc10Crew5Crane2_3	55	352	56	-
Act200Loc10Crew5Crane2_1	92	711	103	-
Act200Loc10Crew5Crane2_2	124	742	124	-
Act200Loc10Crew5Crane2_3	66	704	89	-
Act200Loc10Crew5Crane2_4	71	734	109	-
Act200Loc10Crew5Crane2_5	88	698	93	-
Act300Loc10Crew5Crane2_1	144	1049	144	-
Act300Loc10Crew5Crane2_4	137	1071	148	-
Act400Loc10Crew5Crane2_1	177	1400	192	-
Act400Loc10Crew5Crane2_2	137	1428	174	-
Act400Loc10Crew5Crane2_3	198	1369	198	-
Act400Loc10Crew5Crane2_4	168	1362	186	-
Act400Loc10Crew5Crane2_5	136	1361	190	-
Act500Loc10Crew5Crane2_2	234	1723	257	-
Act500Loc10Crew5Crane2_4	212	1798	243	-
Act600Loc10Crew5Crane2_2	282	2201	297	-

Act600Loc10Crew5Crane2_3	269	2102	304	-
Act700Loc10Crew5Crane2_1	291	2407	328	-
Act1000Loc10Crew5Crane2_1	416	3529	447	-
Act1000Loc10Crew5Crane2_2	416	3528	451	-
Act50Loc10Crew5Crane3_1	18	196	34	-
Act50Loc10Crew5Crane3_2	33	186	34	-
Act50Loc10Crew5Crane3_3	20	157	22	-
Act50Loc10Crew5Crane3_4	29	177	29	-
Act50Loc10Crew5Crane3_5	27	196	27	-
Act60Loc10Crew5Crane3_1	34	190	35	-
Act70Loc10Crew5Crane3_1	40	259	44	-
Act70Loc10Crew5Crane3_2	40	259	41	-
Act70Loc10Crew5Crane3_4	38	254	38	-
Act80Loc10Crew5Crane3_1	29	265	30	-
Act80Loc10Crew5Crane3_2	43	265	43	-
Act80Loc10Crew5Crane3_3	47	272	48	-
Act80Loc10Crew5Crane3_4	40	287	41	-
Act80Loc10Crew5Crane3_5	29	288	38	-
Act90Loc10Crew5Crane3_3	39	300	40	-
Act90Loc10Crew5Crane3_4	42	348	46	-
Act100Loc10Crew5Crane3_1	53	347	53	-
Act100Loc10Crew5Crane3_3	37	350	43	-
Act100Loc10Crew5Crane3_4	44	364	45	-
Act100Loc10Crew5Crane3_5	35	362	38	-
Act200Loc10Crew5Crane3_1	101	688	102	-
Act200Loc10Crew5Crane3_3	97	728	97	-
Act200Loc10Crew5Crane3_4	92	713	93	-
Act300Loc10Crew5Crane3_2	149	1061	149	-
Act400Loc10Crew5Crane3_2	187	1360	188	-
Act400Loc10Crew5Crane3_3	188	1378	188	-
Act400Loc10Crew5Crane3_5	210	1450	211	-

Tabell 2: Løsninger med løsningsstrategi 1 og tidsgrense på 100 sekunder

Benchmark	Nedregrense	Øvregrense	Uten varme Makespan	Med varme Makespan
Act50Loc10Crew5Crane2_1	28	177	41	41
Act50Loc10Crew5Crane2_2	25	180	28	31
Act50Loc10Crew5Crane2_3	38	184	38	38
Act50Loc10Crew5Crane2_4	21	172	29	30

Act50Loc10Crew5Crane2_5	22	177	40	40
Act60Loc10Crew5Crane2_1	29	211	51	53
Act60Loc10Crew5Crane2_2	31	215	64	69
Act60Loc10Crew5Crane2_3	31	209	54	56
Act60Loc10Crew5Crane2_4	27	203	27	27
Act60Loc10Crew5Crane2_5	27	205	72	73
Act70Loc10Crew5Crane2_1	39	249	39	-
Act70Loc10Crew5Crane2_2	38	238	89	-
Act70Loc10Crew5Crane2_3	34	267	40	-
Act70Loc10Crew5Crane2_4	39	270	67	-
Act70Loc10Crew5Crane2_5	26	231	30	-
Act80Loc10Crew5Crane2_1	34	283	72	-
Act80Loc10Crew5Crane2_2	35	272	62	-
Act80Loc10Crew5Crane2_3	31	276	76	-
Act80Loc10Crew5Crane2_4	40	278	64	-
Act80Loc10Crew5Crane2_5	47	258	87	-
Act90Loc10Crew5Crane2_1	50	298	77	-
Act90Loc10Crew5Crane2_2	42	340	88	-
Act90Loc10Crew5Crane2_3	43	314	64	-
Act90Loc10Crew5Crane2_4	43	303	80	-
Act90Loc10Crew5Crane2_5	40	335	84	-
Act100Loc10Crew5Crane2_1	50	356	70	-
Act100Loc10Crew5Crane2_2	66	364	67	-
Act100Loc10Crew5Crane2_3	55	352	81	-
Act100Loc10Crew5Crane2_4	50	312	90	-
Act100Loc10Crew5Crane2_5	55	337	80	-
Act200Loc10Crew5Crane2_1	92	711	174	-
Act200Loc10Crew5Crane2_2	124	742	175	-
Act200Loc10Crew5Crane2_3	66	704	148	-
Act200Loc10Crew5Crane2_4	71	734	191	-
Act200Loc10Crew5Crane2_5	88	698	137	-
Act300Loc10Crew5Crane2_1	144	1049	202	-
Act300Loc10Crew5Crane2_2	158	1087	271	-
Act300Loc10Crew5Crane2_3	138	1058	207	-
Act300Loc10Crew5Crane2_4	137	1071	229	-
Act300Loc10Crew5Crane2_5	162	1044	280	-
Act400Loc10Crew5Crane2_1	177	1400	323	-
Act400Loc10Crew5Crane2_2	137	1428	327	-
Act400Loc10Crew5Crane2_3	198	1369	305	-
Act400Loc10Crew5Crane2_4	168	1362	309	-
Act400Loc10Crew5Crane2_5	136	1361	337	-
Act500Loc10Crew5Crane2_1	219	1750	422	-
Act500Loc10Crew5Crane2_2	234	1723	456	-

Act500Loc10Crew5Crane2_3	234	1746	398	-
Act500Loc10Crew5Crane2_4	212	1798	416	-
Act500Loc10Crew5Crane2_5	254	1754	385	-
Act600Loc10Crew5Crane2_1	269	2097	555	-
Act600Loc10Crew5Crane2_2	282	2201	532	-
Act600Loc10Crew5Crane2_3	269	2102	526	-
Act600Loc10Crew5Crane2_4	272	2116	543	-
Act600Loc10Crew5Crane2_5	242	2065	455	-
Act700Loc10Crew5Crane2_1	291	2407	561	-
Act700Loc10Crew5Crane2_2	259	2492	551	-
Act700Loc10Crew5Crane2_3	277	2447	508	-
Act700Loc10Crew5Crane2_4	296	2428	679	-
Act700Loc10Crew5Crane2_5	262	2408	529	-
Act800Loc10Crew5Crane2_1	358	2811	636	-
Act800Loc10Crew5Crane2_2	347	2834	666	-
Act800Loc10Crew5Crane2_3	251	2856	628	-
Act800Loc10Crew5Crane2_4	323	2785	623	-
Act800Loc10Crew5Crane2_5	320	2779	647	-
Act900Loc10Crew5Crane2_1	456	3232	789	848
Act900Loc10Crew5Crane2_2	416	3144	775	-
Act900Loc10Crew5Crane2_3	464	3188	733	-
Act900Loc10Crew5Crane2_4	404	3143	721	-
Act900Loc10Crew5Crane2_5	443	3179	745	-
Act1000Loc10Crew5Crane2_1	416	3529	800	-
Act1000Loc10Crew5Crane2_2	416	3528	774	-
Act1000Loc10Crew5Crane2_3	335	3533	901	-
Act1000Loc10Crew5Crane2_4	435	3547	773	-
Act1000Loc10Crew5Crane2_5	481	3462	928	-
Act5000Loc10Crew5Crane2_1	2242	17522	4005	-
Act5000Loc10Crew5Crane2_2	2226	17532	4225	-
Act5000Loc10Crew5Crane2_3	2296	17219	4074	-
Act5000Loc10Crew5Crane2_4	2308	17513	4071	-
Act5000Loc10Crew5Crane2_5	2262	17455	4031	-
Act60Loc10Crew5Crane3_1	34	190	37	-
Act60Loc10Crew5Crane3_2	32	224	45	-
Act60Loc10Crew5Crane3_2	32	224	45	-
Act60Loc10Crew5Crane3_3	31	225	53	-
Act60Loc10Crew5Crane3_4	43	216	43	-
Act60Loc10Crew5Crane3_5	22	191	48	-
Act50Loc10Crew5Crane3_1	18	196	52	-
Act50Loc10Crew5Crane3_2	33	186	38	-
Act50Loc10Crew5Crane3_3	20	157	35	-

Act50Loc10Crew5Crane3_4	29	177	29	-
Act50Loc10Crew5Crane3_5	27	196	31	-
Act70Loc10Crew5Crane3_1	40	259	53	-
Act70Loc10Crew5Crane3_2	40	259	59	-
Act70Loc10Crew5Crane3_3	27	238	73	-
Act70Loc10Crew5Crane3_4	38	254	44	-
Act70Loc10Crew5Crane3_5	41	247	59	-
Act80Loc10Crew5Crane3_1	29	265	44	-
Act80Loc10Crew5Crane3_2	43	265	43	-
Act80Loc10Crew5Crane3_3	47	272	48	-
Act80Loc10Crew5Crane3_4	40	287	42	-
Act80Loc10Crew5Crane3_5	29	288	58	-
Act90Loc10Crew5Crane3_1	52	319	79	-
Act90Loc10Crew5Crane3_2	52	328	84	-
Act90Loc10Crew5Crane3_3	39	300	57	-
Act90Loc10Crew5Crane3_4	42	348	65	-
Act90Loc10Crew5Crane3_5	38	320	94	-
Act100Loc10Crew5Crane3_1	53	347	58	-
Act100Loc10Crew5Crane3_2	58	358	85	-
Act100Loc10Crew5Crane3_3	37	350	66	-
Act100Loc10Crew5Crane3_4	44	364	83	-
Act100Loc10Crew5Crane3_5	35	362	59	-
Act200Loc10Crew5Crane3_1	101	688	138	-
Act200Loc10Crew5Crane3_2	96	688	188	-
Act200Loc10Crew5Crane3_3	97	728	144	-
Act200Loc10Crew5Crane3_4	92	713	156	-
Act200Loc10Crew5Crane3_5	96	667	158	-
Act300Loc10Crew5Crane3_1	123	1028	215	-
Act300Loc10Crew5Crane3_2	149	1061	209	-
Act300Loc10Crew5Crane3_3	137	1032	264	-
Act300Loc10Crew5Crane3_4	137	1056	265	-
Act300Loc10Crew5Crane3_5	172	1047	199	-
Act400Loc10Crew5Crane3_1	197	1443	227	-
Act400Loc10Crew5Crane3_2	187	1360	260	-
Act400Loc10Crew5Crane3_3	188	1378	272	-
Act400Loc10Crew5Crane3_4	205	1402	333	-
Act400Loc10Crew5Crane3_5	210	1450	337	-
Act500Loc10Crew5Crane3_1	254	1829	394	-
Act500Loc10Crew5Crane3_2	233	1738	406	-
Act500Loc10Crew5Crane3_3	251	1774	390	-
Act500Loc10Crew5Crane3_4	244	1763	317	-
Act500Loc10Crew5Crane3_5	239	1747	402	-
Act600Loc10Crew5Crane3_1	308	2148	444	-

Act600Loc10Crew5Crane3_2	255	2046	481	-
Act600Loc10Crew5Crane3_3	291	2140	423	-
Act600Loc10Crew5Crane3_4	286	2132	514	-
Act600Loc10Crew5Crane3_5	289	2120	439	-
Act700Loc10Crew5Crane3_1	340	2498	606	-
Act700Loc10Crew5Crane3_2	315	2418	564	-
Act700Loc10Crew5Crane3_3	339	2566	537	-
Act700Loc10Crew5Crane3_4	306	2508	614	-
Act700Loc10Crew5Crane3_5	337	2449	577	-
Act800Loc10Crew5Crane3_1	350	2817	666	-
Act800Loc10Crew5Crane3_2	408	2807	690	-
Act800Loc10Crew5Crane3_3	385	2808	581	-
Act800Loc10Crew5Crane3_4	383	2897	613	-
Act800Loc10Crew5Crane3_5	356	2731	541	-
Act900Loc10Crew5Crane3_1	407	3152	639	-
Act900Loc10Crew5Crane3_2	271	3208	761	-
Act900Loc10Crew5Crane3_3	285	3153	680	-
Act900Loc10Crew5Crane3_4	420	3181	773	-
Act900Loc10Crew5Crane3_5	426	3163	687	-
Act1000Loc10Crew5Crane3_1	323	3545	756	-
Act1000Loc10Crew5Crane3_2	457	3446	762	-
Act1000Loc10Crew5Crane3_3	431	3544	761	-
Act1000Loc10Crew5Crane3_4	458	3616	896	-
Act1000Loc10Crew5Crane3_5	481	3554	733	-
Act5000Loc10Crew5Crane3_1	2192	17577	4044	-
Act5000Loc10Crew5Crane3_2	2201	17508	3672	-
Act5000Loc10Crew5Crane3_3	2206	17191	3672	-
Act5000Loc10Crew5Crane3_4	1496	17468	3452	-
Act5000Loc10Crew5Crane3_5	2230	17527	3751	-

Tabell 3: Løsninger med løsningsstrategi 2 med tidsgrense
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. `Project Management and 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.