Automatisk planlegging i oljeindustrien

av

Teis Lindemark

v0.1

AVHANDLING

for en grad av

MASTER I INFORMATIKK

Masteroppgave, Institutt for informatikk

Universitetet i Bergen

Juni 2012

Universitetet i Bergen

Sammendrag

TEST

Innhold

Forord

Dette er absolutt bare et utkast og vil bli fullført tilslutt. 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.

Teis Lindemark, 6. Mai 2012

Nomenclature

AI	Kunstig intelligens (engelsk: artificial inteligence)
Avgjøringsvariable	På engelsk: decision variable
Avledningsvariable	Derived variable
Concert	IBM ILOG Concert Technology
CP	Begrensningsprogrammering (engelsk: Constraint programming)
Monooperatorressurs	På engelsk: Unary resource
Scheduler	IBM ILOG Scheduler
Solver	IBM ILOG Solver

1 Introduksjon

I dette prosjektet vil utvikleren bli gitt et eksisterende ILOG Scheduler program og benchmarksett. Det eksisterende ILOG Scheduler programmet skal utvides med flere ressurser og benchmarksettene skal kjøres for å kunne sammenligne løsningene med de nye resurssene mot de løsningene uten de nye resurssene.

Denne skriver jeg tilslutt tenker jeg

1.1 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 ved å tildele ressurser til aktiviteter og tildele en tid til aktiviteter så det ikke er noen konflikt med begrensingene.[5] Automatisk planlegging er endel av det som kalles kunstig intelligens (AI) .

I prosjektet vil også utvikleren evaluere løsningene og sammenligne løsningene med og uten tilleggsbegrensinger.

Det overordnede målet med denne avhandlingen er å utvide ILOG Scheduler løsningen til Bård Henning Tvedt med flere resursser for å sjekke om det å legge til flere ressurser vil gi bedre og flere løsninger enn de opprinnelige løsningene til Bård Henning Tvedt .

1.1.1 Relatert arbeid

Skrive om tidligere gjort forskning på området.

1.2 Målet med prosjektet

Målet med prosjektet er todelt, og består i å vurdere den modifiserte problemstillingen mot den opprinnelige i forhold til:

- minimere *makespan*
- antall begrensninger
- implementasjon 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.

- 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?

1.3 Beskrivelse av kommende kapitler

I kapittelet om metode, vil fremgangsmåten for prosjektet bli lagt frem og hvordan løsningene har blitt evaluert. I kapittelet om eksperimenteringen vil det legges frem løsninger ved forskjellige strategier og med og uten ressursene. Her vil også løsningene bli evaluert og tilslutt vil det bli sammenfattet en konklusjon over det arbeidet som er gjort.

1.4 Motivasjon

Forskningen er motivert av praktisk erfaring at planleggingsproblemer er veldig aktuelt i bedrifter og i samfunnet idag. Det er ikke bare i oljeindustrien som jeg fokuserer oppgaven på hvor planleggingsløsninger er aktuelt, men generelt bemmaningsproblematikken som alle personalavdelinger sitter med i det daglige. I hverdagen er det også mange planleggingsproblemer fra buss- og togtabeller til personlige gjøremål med å få tid til alt man skal ha tid til.

Planlegging i oljeindustrien er viktig for å på en mest mulig effektiv måte benytte seg av de ressursene som er tilgjengelig til enhver tid, samtidig som visse begrensinger blir fastsatt med tanke på sikkerheten. Det er mye penger involvert i olje- og gassindustrien og det å utføre aktiviteter på en ineffektiv måte kan koste selskapene veldig mye penger. 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. Det å imøtekomme disse kompliserte og utfordrende målene, kan det i enkelte situasjoner oppstå konflikter.

Spesifikke planleggingssituasjoner 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 plattformer på land kan ha mange aktiviteter utført på et relativt stort område. Selvom 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 begrensningsprogrammering

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

1.4.2 Utfordringer med begrensningsprogrammering

Utfordringer med CP

1.4.3 Begrensingsprogrammeringsverktøy idag

Det finnes idag flere forskjellige verktøy for begrensningsprogrammering, både i form av egne programmeringsspråk som er skreddersydd for begrensningsprogrammering og biblioteker til godt kjente programmeringsspråk som Java[2] og C++[3]. 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[4]. Sistnevnte er et programmeringspråk for begrensningsprogrammering med lokalt søk og er en kommersiell løsning. Eksempler på begrensningsprogrammeringsbiblotek så er det IBM ILOG CP.

Det er ingen løsninger som passer uansett hva slags del innenfor begrensningsprogrammering du skal gjøre. Logisk programløsning eller planleggingsløsning må tas med når det skal bestemmes hvilke verktøy som brukes.

[1] Imperativ programmeringsspråk har sekvenser med som blir utført.
[2] Java
[3] C++
[4] Comet

1.5 Problembeskrivelse

Problemstillingen tar utgangspunkt i den opprinnelige problemstillingen til Bård Henning Tvedt . 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 bli planlagt. 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", som kjennetegnes ved:

- Et sett av ressurser med en gitt kapasitet
- Et sett av ikke-forstyrrede aktiviteter som er gitt en prosseseringstid
- Et netverk 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 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 planleggingspløsninger 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 et veldefinert problembeskrivelse. En løsning til et problem $S(P_i)$ er en planlegging hvor aktiviteter er tilegnet en starttid og begrensningene er holdt.

1.5.1 Notasjoner og termonologi

En probleminstans P inneholder aktiviteter som skal gjennomføres, ressurser som er påkrevd for å gjennomføre aktivitene 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* $Loc_l \in Locs = \{Loc_1, \ldots, Loc_n\}$ er stedet hvor aktiviter blir utført. Selvom lokasjoner blir vist som ressurser, så er det ikke noen begrensinger på hvor mange aktiviter 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 $Crew_j \in Crews = \{Crew_1, \ldots, Crew_n\}$. Utfylling her

En *kran* $Crane_k \in Cranes = \{Crane_1, \ldots, Crane_n\}$ er en potensiell ressurs for aktiviteter. Noen aktiviteter trenger kran og alle probleminstanser har et mindre antall av aktiviteter som krever kranbruk. Kraner er 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 loksjonen hvor aktiviteten som krever kranbruk er utført og kranens egen lokasjon. Sikkerhetssonen som blir satt vil derfor variere ut ifra hvilken kran som er tilegnet til aktiviteten.

1.5.3 Aktiviteter

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

En aktivitet Act_i krever et mannskap $c_{crew}(Act_i) \in Crews$ for å utføre den 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 nødvenndig for alle aktivitetene.

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

Sjekk om dette stemmer også etter varme!

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:

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

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

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

hvor $c_capCrew_j$ er kapasiteten av j's mannskap.

Kraner er unik induvidielle og er derfor modellert som et sett av monopolatorressursbegrensninger. 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:

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

Den gjensidge uttelukkelsen opprettet av den monopolatoriskeressursbegrensningen blir da:

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

for alle aktiviteter som krever kran.

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

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

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

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

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

Den siste begrensningen er varmebegrensningen. De er uttrykt som kummulativressursbegrensning og er påført lokasjon for å være sikkert på at total varme bruk ikke oversitiger varmekapasiteten tilgjengelig på hver lokasjon. Den er uttrykt ved:

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

$$\wedge c_{loc}(Act_i) = Loc_l]\} \leq c_{heatcap}(Loc_l)$$

Skriv denne ferdig

1.5.5 Mål

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

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

som sier 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, $\#Acts \in \{50, 60, 100, 200, 300, 400, 500, 600, 800, 900, 1000, 5000\}$ og kraner, $\#Cranes = [2, 3]$. Det ble generert totalt 5 probleminstanser for hver av de 24 problem størrelsene, som summert opp blir 120 instanser.

Probleminstansene ble tilfeldig generert, ved å tilegne 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 120 instansene er gyldige.

Alle probleme har 10 lokasjoner, som er redusert fra 25 i de opprinnelige probleminstansene til Bård Henning Tvedt , for at det skal kunne være flere aktiviteter på lokasjonen enn om det var 25 lokasjoner, og 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. Skriv denne ferdig. Sjekk antall kraner i problemene

2 Metode

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

2.1 Verktøy brukt i prosjektet

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

2.2 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 begrensningsprogrammering. Det er ikke noe eget programmeringsspråk, som da 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 er instanser av klassen IloConstraint.[4]

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.[3]

2.3 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, resurs tildeling, 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 enkelest mulig å omgjøre applikasjoner fra plattformer til plattformer, 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 begrensningsprogrammering 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 Concert klassene 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 besluttingsvariablene har et domene med mulige verdier. Begrensningene setter grensene for kombinasjonene av verdier for de besluttingsvariablene.

Det siste steget er å bruke Solver for å løse problemet. Det inneholder å finne verdier for alle besluttingsvariablene samt ikke bryte noen av de definerte begrensningene og dermed enten maksimere eller minimere målet, hvis det er et mål inkludert i modellen. Solver ser etter løsninger i et søkeområdet. Søkeområdet er alle mulige kombinasjoners av verdier.[3]

2.4 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 begrensningsprogrammering. Scheduler har spesifisert funksjonalitet på å løse problemer innen planlegging og ressurs tildeling.[2]

2.5 Forskningsmetoder

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

2.5.1 Implementeringsprosessen

Første delen av implementeringsprosessen er å utvide modellen for å inkludere de nye ressursene. For å gjøre det eksperimenterte jeg litt med Scheduler for prøve å finne den beste måten å implementere det på.

Den andre delen av implementeringsprosessen er å eksperimentere med forskjellige løsningstrategier gitt av i Solver. Her er det mange forskjellige kombinasjoner og rekkefølgen de forskjellige strategiene (i ILOG; Goals) kan ha en innvirkning på løsningen tilslutt.

Tilslutt så kommer det an på tidsgrensen som blir satt på hvor lenge ILOG skal søke etter bedre løsninger.

Denne delen skal utvides ytterligere.

2.5.2 Evaluering av prosessen

Prosessen med eksperimenteringen av implementasjonen blir evaluert ved å undersøke makespan opp mot teoretisk øvregrense og teoretisk nedregrense i både løsningene uten tilleggsressurser og med tilleggsressurser. Løsningene fra prosessen med og uten tilleggsresursene 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.

2.6 Evalueringsstrategi

3 Eksperimenter

Eksperimenteringen er utført på en Mackbook Air med 1.8 GHz Intel Core i7 prosessor og 4 GB 1333 MHz DDR3 minne. Det er totalt sett 71 benchmarksett som implementasjonene er evaluert på, hvor mange det ble funnet en løsning varierte med eller uten tilleggsressursene og tidsgrensen som var satt.

3.1 Uten tilleggsressurser

3.2 Med tilleggsressurser

4 Fremtidig arbeid

5 Konklusjon

6 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
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	-
Act500Loc25Crew5Crane2_2	231	1759	232	-
Act500Loc25Crew5Crane2_5	223	1708	223	-

Tabell 1: Løsninger med løsningsstrategien XXX 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
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	-
Act500Loc25Crew5Crane2_1	212	1714	389	389
Act500Loc25Crew5Crane2_2	231	1759	368	368
Act500Loc25Crew5Crane2_3	156	1707	355	355
Act500Loc25Crew5Crane2_4	220	1751	353	353
Act500Loc25Crew5Crane2_5	223	1708	360	360
Act5000Loc25Crew20Crane5_1	521	17370	3243	-

Tabell 2: Løsninger med løsningsstrategien XXX med tidsgrense 100 sekunder

Referanser

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

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

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

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

[5] 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.