Automatisk Planlegging i Oljeindustrien

av

Teis Lindemark

v0.2.1

AVHANDLING

for en grad av

MASTER I INFORMATIKK

Masteroppgave, Institutt for informatikk

DRAFT

Universitetet i Bergen

Juni 2012

Universitetet i Bergen

Sammendrag

TEST

Innhold

Forord

Forkortelser

AI	Kunstig intelligens (engelsk: artificial inteligence)
APO	Automatisk Planlegging i Oljeindustrien
Avgjøringsvariable	På engelsk: decision variable
Avledningsvariable	Derived variable
Concert	IBM ILOG Concert Technology
CP	Begrensningsprogrammering (engelsk: Constraint programming)
JSSP	Job Shop Scheduling Problem
LS1	Løsningsstrategi 1
LS2	Løsningsstategi 2
Monooperatorressurs	På engelsk: Unary resource
MSProject	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 planleggingsproblem i industrien og det blir brukt en fiktiv platform for å kunne gjøre problemet så likt som mulig virkeligheten. Implementjonen i IBM ILOG Scheduler er utvidet med ressurser på varmebegrensning. Med varme er det ikke varme i tradisjonell forstand, men et mål for hvor mye kapasitet en ressurs bruker.

1.1 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.

1.2 Bakgrunn

Bakgrunn på utvides yterligere! 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.[16] Automatisk planlegging er endel av det som kalles kunstig intelligens (AI) .

Problemer i kategorien *beregningsvitenskap kompleksitets teori*[6] og som også er minst like vanskelige som de vanskeligste probleme i NP-kategorien, kalles *ikke-deterministisk polynomtid problemer vansklige* (NP-hard) . Det er forskjellige kategorier av NP-harde problemer, avgjøreksesproblemer, søkeproblemer og optimaliseringsproblemer. [7] Problemet i dette prosjektet er i kategorien optimaliseringsproblem. Fra wikipedia [8] er et optimaliseringsproblem: Et optimaliseringsproblem er et problem med å finne den beste løsningen ut ifra alle gyldige løsninger".

1.2.1 Relatert arbeid

Skrive om tidligere gjort forskning på området. Nuijten og Pape har brukt ILOG Scheduler til å løse "Job Shop Schedulingproblemet og det er et problem som er

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

- (Taillard) Benchmark for basic scheduling problems [17]
  - Vanskelige problemer; beste makespan langt unna nedre grense
  - Går på hvordan oppretter gode og vanskelige benchmarksett for 3 kjente CP problemer.
- (Wallace) On bencgmarking CLP Platforms [18]
  - Tar for seg positive og negative aspekter ved benchmarking CLP problemer.
  - Benchmarking oppgave 2 oppgaver; 1 - dekke et representativt problem og bredt programmerings begrep
  - 2 måter; application benchmarking og unit tests
  - Utfordring opprette et klar målsetting med ytelsen.
  - Teoretisk idelle måten å velge beste CLP systemet for en applikasjon er å implementere en løsning for applikajsonen for hvert system og velge den beste. Praktisk: kan bare benchmarke systemer med predefinerte problemer og håpe at resultatet kan bli overført til applikasjonen som krever det.
  - Sammenlignet CLP problemer SICStus og IF/Prolog med andre CLP systemer (clp(FD), CHR, ILOG, Oz og B-prolog)
  - Tidsmekanismer vanskelig spesielt i høynivåspråk, hvor det ofte måles CPU tid.
  - Sammenligne CLP systemer gir flere tilfeller som ikke har oppstått ved sammenliggning av mer tradisjonelle systemer. Utførelsestiden for 2 CLP språk løser samme problem ofte avhenger mindre på språkimplementasjon enn presis algoritme utførelse av programmet.
  - For å benchmarke forskjellige CLP språk støtter samme funskjonalitet er det nødt å skrive program i forskjellig språk med samme algoritme.
- (Laborie) IBM ILOG CP Optimizer for Detailed Scheduling Illustrated in Three problems [13]

  - ...
- (Biskup) Enslig-maskin planlegging med læringsbetraktninger
  - ...

Jobber med dette punktet nå. Så punktlistene er kun så jeg husker bedre.

1.3 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?

1.4 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 områder 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".[4]

1.4.2 Utfordringer med begrensningsprogrammering

Utfordringer med CP

- Ytelse - begrensings logikk programmerings system ytelse mindre enn tradisjonelle imperative.
- Ytelse - automatisk paralissering
- Debugging og visualisering Begrensningsprogrammeringssystemer har ofte en svakhet når det kommer til feilsøking (engelsk: debugging). Uten tilstrekkelige måter å kunne være sikkert på riktigheten og muligheter å sjekke ytelse i programmer. En måte å løse utfordringene med manglende feilsøkingsmuligheter er å bruke påstander . Ved bruk av påstander kan

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

det opprettes post- og preforhold. 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].

1.4.3 Begrensingsprogrammeringsverktøy idag

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

Planlegging er ofte tett knyttet opp mot prosjektstyring og prosjektstyringsverktøy finnes det veldig mange av etterhvert [5]. 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, men det er 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[3]. I MSProject er det mulig å legge til begrensninger, men ser ganske begrenset ut til aktiviteters starttidspunkt[2].

Det finnes få verktøy for å gjøre begrensingprogrammering som er kan brukes rett ut av boksen idag, enten det er logisk problemløsning eller planleggingsløsning. Noen verktøy er bedre på logisk problemløsning og andre på planleggingsproblemer. Det finnes også mange forskjellige problemmer innen begge kategorier som er komplekse og det er derfor nødvenndig å finne verktøyet som passer best mulig til problemet som skal løses. Innen industrien er det ofte ressurser som stiller krav utover standardiserte problemer.

1.4.4 Forbedringerspotensiale i verktøyene

1.4.5 Relevans for forskingen

1.5 Problembeskrivelse

Problemstillingen tar utgangspunkt i den opprinnelige problemstillingen til Bård Henning Tvedt . Beskrivelsen følger i første omgang [15], som jeg oversetter 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 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 (RCPSP)"[9], 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 $Crew_j \in Crews = \{Crew_1, \ldots, Crew_n\}$.

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, 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:

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

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

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

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

Kraner er 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:

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

Den gjensidge uttelukkelsen opprettet av den monopolatoriskeressursbegrensningen blir da:

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

for alle aktiviteter som krever kran.

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

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

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

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

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

Inntil nå har jeg oversatt [15]. Nå kommer en ny type begrensning kalt *varmebegrensning*. Med varme, 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 ikke kan ha ubegrenset med mannskap til å jobbe der samtidig. Forskjellige mannskaper kan også ha forskjellig årsaker ut ifra hva slags arbeid de utfører hvor mye av denne kapasiteten de bruker. Mannskaper som driver arbeid som sveising kan for eksempel ha en høyere varmeverdi enn et mannskap med elektrikkere. 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. Varmebegrensingen skal også etterhvert kunne erstatte sikkerhetsbegrensningene på foreksempel kranbruk, ved at lokasjoner med kran får en varmekapasitet og aktiviter som bruker kran bruker opp denne varmekapasiten på lokasjonen. Den er uttrykt ved:

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

1.5.5 Målfunksjon

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

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

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, $\#Acts \in \{50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 5000\}$

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 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. 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 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.[12]

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.[11]

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.[11]

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.[10]

2.5 Forskningsmetoder

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

2.5.1 Implementeringsprosessen

Problemet med å finne ut om en RCPSP løsning i tillegg til sikkerhetsbegrensningene med makespan mindre enn en gitt frist finnes er NP-hard. Dette betyr at den 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 biblotekene. I begge løsningsstrategiene brukes standard søkemål i ILOG Solver.

Den første løsningsstrategien (LS1) bruker IloAssignAlternatives, som blir brukt til å tildele kraner til aktivitetene. Den neste søkemålet er IloRankForward... og tilslutt brukes søkemålet IloSetTimesForward ...

Den andre løsningsstrategien (LS2)

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 (se 2.6.1) og teoretisk nedregrense (se 2.6.2) 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

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

2.6.1 Teoretisk øvregrense

Teoretisk øvregrense for makespan er

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

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

2.6.2 Teoretisk nedregrense

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

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

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

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

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 65 benchmarksett som implementasjonene er evaluert på, hvor mange det ble funnet en løsning varierte med eller uten tilleggsressursene og tidsgrensen som var satt.

Modell	**2 kraner**				**3 kraner**				**Alle**	
$\sharp Act(\sharp P)$	$< 100(25)$		$> 100(55)$		$< 100(25)$		$> 100(55)$		(160)	
Modell	w_{rq}	$\%^{(1)}$	w_{rq}	$\%^{(1)}$	w_{rq}	$\%^{(1)}$	w_{rq}	$\%^{(1)}$	w_{rq}	$\%^{(1)}$
$LS1\sharp1^{(2)}$	1.164	80	1.115	44	1.105	64	1.027	20	1.113	44
$LS1\sharp2^{(3)}$	1.228	32	-	-	-	-	-	-	1.183	5
$LS2\sharp1^{(2)}$	1.691	100	1.855	100	1.545	100	1.693	100	1.725	100
$LS2\sharp2^{(3)}$	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 $^{(2)}$ $\sharp 1$ er løsninger med varmeressurs $^{(3)}$ $\sharp 3$ er løsninger uten varmeressurs

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

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

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

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

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

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

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

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

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

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

[9] 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.

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

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

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

[13] Philippe Laborie. Ibm ilog cp optimizer for detailed scheduling illustrated on three problems. In *Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, CPAIOR '09, pages 148–162, Berlin, Heidelberg, 2009. Springer-Verlag.

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

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

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

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

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