Adaptivt lokalsøk for boolske optimeringsproblemer

Lars Magnus Hvattum
Høgskolen i Molde
Lars.M.Hvattum@himolde.no

Arne Løkketangen
Høgskolen i Molde
Arne.Lokketangen@himolde.no

Fred Glover
Leeds School of Business, UCB 419, University of Colorado, Boulder, CO 80309, USA
Fred.Glover@Colorado.edu

Utdrag

Vi presenterer et adaptivt lokalsøk for boolske optimeringsproblemer. Søket balanserer mellom å oppfylle restriksjoner i form av et boolsk uttrykk og å finne gode verdier for en tilhørende målfunksjon, samtidig benyttes enkle mekanismer fra tabusøk. Det rapporteres resultater oppnådd for en portefølje med testproblemer hentet fra publisert litteratur, og disse antyder at metoden er konkurransedyktig både med hensyn til løsningskvalitet og –tid.

1 Innledning

Boolske optimeringsproblemer (BOOP) representerer en stor klasse binære optimeringsmodeller, inkludert vektede utgaver av Set Covering-, Graph Stability-, Set Partitioning- og Maximum Satisfiability-problemer. Disse problemene er såkalt NP-harde, og bruk av heuristiske søkemetoder er godt egnet selv for problemer av moderat størrelse.

Her beskrives en rimelig enkel, iterativ søkeprosedyre for denne problemklassen, hvor det benyttes adaptiv hukommelse og læringsprinsipper hentet fra tabusøk. Styring av søket er basert på en strategisk oscillasjon rundt avgrensninger uttrykt i en boolsk likning, og koordinerer samspillet mellom endringer i målfunksjonen og endringer i tilfredsstillelse av den boolske likningen. Dette modifiseres så av mekanismer fra tabusøk, som sammen med periodisk å starte søket på nytt, sørger for å bre søket utover løsningsrommet.

Tidligere arbeid med heuristikker for BOOP er hovedsaklig utført av Davoine, Hammer og Vizvári (2001). De benytter en grådig heuristikk basert på pseudo-boolske funksjoner, med avkutting av lokalt optimale løsninger som man finner. Tilnærmingen deres har likheter med Lagrange-relaksasjon, og bruker en formulering i DNF (disjunktiv normalform). Vi baserer våre resultater på problemporteføljen deres, og sammenligner oss både med det som er oppnådd av Davoine, Hammer og Vizvári, så vel som av XPRESS/MP (http://www.dash.co.uk/) og CPLEX (http://www.ilog.com/products/cplex/).

Denne introduksjonen etterfølges av problemformuleringer i seksjon 2. Seksjon 3 beskriver vår tilnærming samt tester for parametre knyttet til søket, mens seksjon 4 antyder resultatene som er oppnådd. Konklusjonene oppsummeres i seksjon 5, sammen med muligheter for fremtidig arbeid.

2 Problemformulering

Det boolske optimeringsproblemet (BOOP), først formulert i Davoine, Hammer og Vizvári (2001), er basert på logiske uttrykk i første ordens utsagnslogikk, med en ekstra kostnad (eller profitt) assosiert med variablene som har verdien *sann* (eventuelt *usann*). En formulering kan være (hvor maksimering forutsettes)

$$Max \quad f = \sum_{i=1}^{n} (c_i \mid x_i = sann / usann)$$

slik at

$$\phi(X) = \phi(x_1,...,x_n) = \begin{cases} sann \\ usann \end{cases}$$

Hvor $\phi(X)$ er det logiske uttrykket, og n er antall variabler. Løsningen av dette problemet er de tildelingene av sannhetsverdier til variablene $x_1,...,x_n$ som gir høyest verdi for målfunksjonen f, samtidig som det logiske uttrykket er tilfredsstilt. Det logiske uttrykket kan generelt være vilkårlig, men vi avgrenser oss til formuleringer i konjunktiv normalform, CNF (disjunktiv normalform kan fremkomme gjennom en enkel transformasjon). Uformelt kan et BOOP oppfattes som et Satisfiability problem (SAT) med en ekstra målfunksjon. (For mer om SAT, se for eksempel Cook, 1997 og Du et al., 1997.)

For å kunne håndtere dette som et tradisjonelt optimeringsproblem, med tall heller enn sannhetsverdier, lar vi den logiske verdien *sann* bli representert med 1 og verdien *usann* med 0, som gir oss målfunksjonen

$$Max \qquad f = \sum_{i=1}^{n} c_i x_i$$

Det logiske uttrykket $\phi(X)$, i CNF, består av konjunksjoner av klausuler $\phi(X) = k_1 \wedge k_2 \wedge ... \wedge k_m$, hvor hver klausul er en disjunksjon av negerte og ikke-negert variabler, med m lik antall klausuler. Som et enkelt eksempel, la

$$\phi(X) = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_3})$$

Ved å erstatte sann/usann med 1/0, disjunksjoner med + , ved å representere hver konjunksjon med en separat rad og dele hver variabel i sine negerte og ikke-negerte forekomster, får man følgende samling restriksjoner for eksempelet, der variabelparet y_i og $y_{i\#}$ representerer x_i:

$$\begin{aligned} y_1 + y_2 \quad &\geq 1 \\ y_1 \quad + y_{3\#} &\geq 1 \end{aligned}$$

$$y_i + y_{i\#} = 1$$

Vår endelige modell er dermed

$$Max\ f = \sum_{i=1}^{n} c_i x_i \qquad (1.1)$$

gitt

$$Dy \geq 1 \qquad (1.2)$$

$$y_i + y_{i\#} = 1 \qquad (1.3)$$

Hvor D er 0-1 matrisen som følger ved å erstatte y_i og $y_{i\#}$ for x_i. Den siste restriksjonen (1.3) håndteres implisitt av heuristikken vi beskriver.

3 Adaptivt lokalsøk (tabusøk)

Lokalsøket vi har implementert er basert på en elementær utgave med tabu-aktivering av variablene, og en selvtilpassende trekkevalueringsfunksjon. Trekkevalueringen forsøker å holde fokus på å oppfylle restriksjonene (1.2), men samtidig opprettholde gode målfunksjonsverdier (1.1).

3.1 Implementasjon av søket

Det er forsøkt å ha en enkel implementasjon, for å kunne legge til mer sofistikerte mekanismer i senere arbeid. Det benyttes eksempelvis tilfeldige startløsninger samtidig som søket blir startet på nytt fra tilfeldig løsninger undervegs, hvilket presumptivt kan forbedres i tabusøk-paradigmet slik det indikeres i Glover og Laguna (1997). Implementasjonen har følgende komponenter.

1. Den *initielle løsningen* (startpunktet) er basert på tilfeldige tildelinger til variablene. Ettersom denne løsningen ikke nødvendigvis tilfredsstiller restriksjonene til problemet (1.2), må søket kunne navigere gjennom søkerommet uavhengig av om (1.2) er tilfredsstilt eller ikke.
2. Et *trekk* er et ”flip” av en variabel. Et ”flip” betyr å tildele den motsatte verdi til en variabel (dvs. endre $1 \rightarrow 0$ eller $0 \rightarrow 1$).
3. *Nabolaget* er mengden av mulige flytt, med en *nabolagsstørrelse* lik n, antall variabler.
4. *Trekkevalueringen* er basert både på endring i verdi av målfunksjonen (1.1), og endringen i graden restriksjonene (1.2) tilfredstilles.
5. *Langtidslæring* er implementert ved å endre vekten av hver rad i (1.2), hvor hver rad tilsvarer en klausul i det opprinnelige logiske uttrykket.
6. *Valg av trekk* er grådig (dvs. man tar det beste trekket ifølge trekkevalueringen).
7. Enkle *tabu*- og *aspirasjonskriterier* er gjeldende.
8. Søket *startes på nytt* med nye tilfeldige variabeltildelinger etter et antall trekk, for å spre søket tilstrekkelig utover (diversifisere søket).

9. *Stoppkriteriet* er basert på en enkel tidsfrist, eller begrensninger i antall trekk.

3.2 Tabu- og aspirasjonskriterier

Ettersom trekkene består av å flippe variabler, vil endringen i målfunksjonens verdi, Δf, skifte fortegn nesten for hvert trekk. Dette fører til at svært mange lokale optimum besøkes av søket, slik at bruk av tabukriterier er nyttig. Det er flere måter å anvende tabukriterier i et søk, men vårt valg er å ikke flippe tilbake en variabel som nylig er flippet. Vår hovedinteresse er å holde tabu-mekanismene enkle, og samtidig oppnå god styring av søket. Det er viktig å finne gode verdier for antall iterasjoner en variabel skal holdes tabuaktiv (tabu tenure - TT), og at antallet iterasjoner endres dynamisk, ettersom statisk TT kan være for lite fleksibelt. Vårt valg av dynamisk tabuaktivering er å la antall tabuaktive iterasjoner velges tilfeldig fra et gitt intervall. Passende verdier for antall iterasjoner med TT er eksperimentelt funnet i 3.4. For en generell behandling av disse aspektene av tabusøk, se Glover og Laguna (1997).

Aspirasjonskriteriet virker ved å tillate trekk som ellers er tabubelagt, dersom trekket gir en ny beste løsning. I seksjon 3.4 illustrerer vi fordelen med denne enkle aspirasjonen.

3.3 Adaptiv trekkevaluering

Evalueringsfunksjonen for hvert trekk, F_{Mj}, har to komponenter. Den ene komponenten er endring i verdien av målfunksjonen. Vektene av variablene, c_i, blir initielt normalisert til å ligge i intervallet $\langle 0,1 \rangle$. Dette betyr at endringen i verdi av målfunksjonen per trekk, Δf_i, ligger i området $\langle -1,1 \rangle$.

Den andre komponenten er endring i antall ikke-tilfredsstilte klausuler (eller rader i restriksjonen (1.2)) gitt at variabelen flippes. Dette tallet, ΔT_i, er i utgangspunktet et lite, positivt eller negativt heltall, og kan finnes fra endringen i en funksjon av sammensatte restriksjoner. (Se f.eks. Løkketangen og Glover, 1996.) Vi utvider, som beskrevet i 3.4, dette til å inkludere individuell vekting av hver klausul, slik at ΔT_i nå blir et flyttall.

Disse to komponentene kombineres for å gi et balansert utgangspunkt for å opprettholde fokus på å innfri restriksjonene, samt å finne gode verdier for målfunksjonen. Den relative vekten for komponentene endres dynamisk for å holde søket i interessante områder.

Dette gir følgende trekkevalueringsfunksjon:

$$F_{Mj} = \Delta T_i + w * \Delta f_i$$

Verdien av w, den adaptive komponenten, blir initielt satt lik 1. Den oppdateres deretter for hvert trekk som følger:

- Dersom den gjeldende løsningen innfrir restriksjonene: $w = w + \Delta w_{ink}$

- Dersom den gjeldende løsning ikke innfrir restriksjonene: $w = w - \Delta w_{dek}$

Ulike verdier brukes for henholdsvis inkrement og dekrement av w. Passende verdier for vektendringene, Δw_{ink} og Δw_{dek}, presenteres i 3.5. Effekten av tilpassningen er å oppnå en strategisk oscillasjon rundt grensene hvor restriksjonene oppfylles. En annen tilnærming finnes i Glover og Kochenberger (1996), hvor oscillasjonen er koblet med bruk av erindring om kritiske hendelser for å tvinge søket til nye områder.

3.4 Langtidslæring

Utgangspunktet for trekkevalueringsfunksjonen var $F_{Mj} = \Delta T_i + w * \Delta f_i$, hvor $\Delta T_i \in Z$ skal representere endring i antall tilfredsstilte rader i restriksjonene (1.2). Vi er interessert i å unngå at den samme mengden av rader veksler mellom ikke å være oppfylt. Dynamisk vekting av radene innføres derfor på følgende vis; initielt settes vekten av alle radene lik 1, men denne økes med en liten additiv konstant hver gang den tilhørende klausulen får en uønsket sannhetsverdi. For å forhindre at noen av radvektene skal bli prohibitivt store, divideres samtlige med en fast konstant hver gang en av dem passerer en øvre grense.

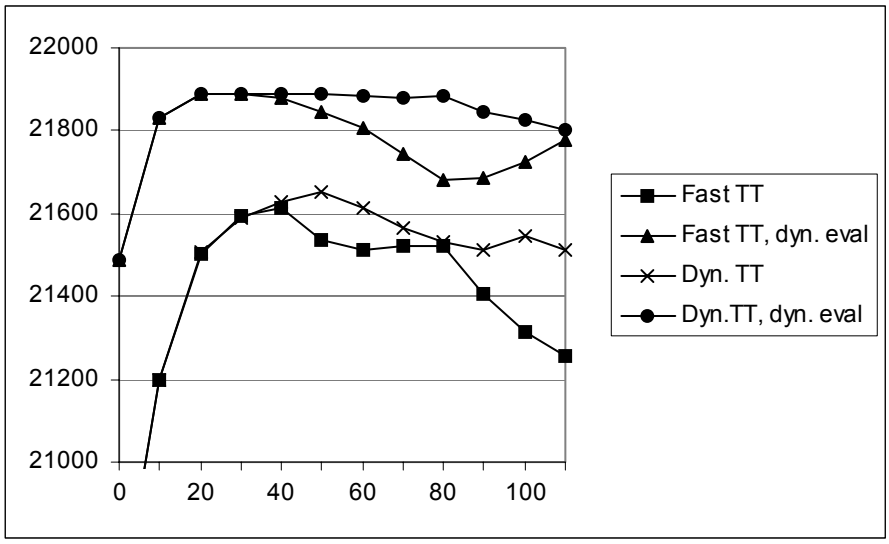
Resultatet av dynamisk radvekting vil være at man i det lange løp vil oppfordre søket til å bryte restriksjoner som ellers ikke ville bli brutt, og dermed bre søket ut over en større del av søkerommet.

3.5 Innledende testing av søkeparametre

Selv om søket vi har implementert er nokså enkelt, er det mange valg som må gjøres med hensyn til verdier for søkeparametre. Ettersom problemporteføljen som søket skal prøves mot består av et stort antall problemer (5485 i alt, se seksjon 4), har vi valgt ut en liten mengde testkasus for å stille inn parametrene i søket. Tre problemer ble valgt (noe tilfeldig), ett i hver av kategoriene *lite* (fra klasse 4 - rn50m200t10s0c0num0 med 50 variabler og 200 klausuler), *medium* (fra klasse 34 - rn200m400t10s0c50num0 med 200 variabler og 400 klausuler) og *stort* (fra klasse 38 - rn500m1000t25s0c50num0 med 500 variabler og 1000 klausuler).

Det bør bemerkes at effekten av, og verdiene for, de ulike parametrene ikke er uavhengig. Man kunne derfor ønske å gjøre et fullt søk blant mulige verdier for parametrene, men dette virker ganske tidkrevende. Derfor har vi valgt en grådig tilnærming, hvor vi velger gode verdier for en søkeparameter om gangen. De øvrige parametrene er enten satt med rimelige verdier, eller med de beste verdiene funnet tidligere i testingen. Rekkefølgen av testingen er derfor viktig, men vi har ikke tatt høyde for dette.

Ikke alle resultatene i denne seksjonen er rapportert i sin helhet, men heller oppsummert ved å angi relativ ytelse.

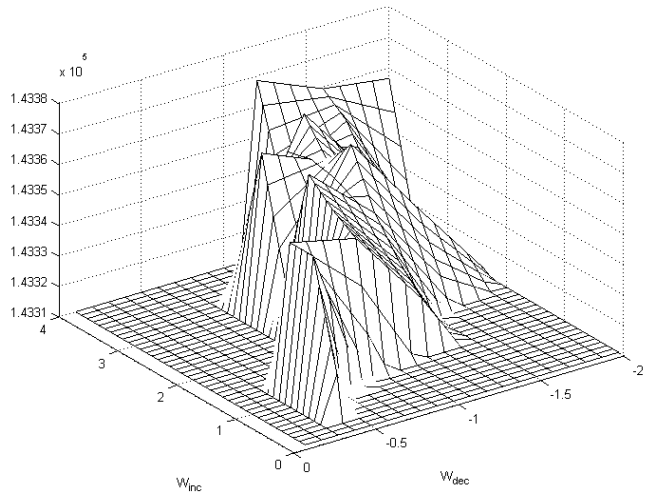


Figur 1

Verdier for tabu-aktivering

For å finne gode verdier for tabu-aktivering (antall iterasjoner en variabel skal være tabu-aktiv etter å ha blitt flippet – tabu tenure - TT), og effekten av å benytte dynamisk TT, kjørte vi en mengde tester for hver av de tre testproblemene. Hver test ble kjørt 20 ganger med ulike (tilfeldig generert) initielle løsninger. Aspirasjonskriteriet var inkludert, men ellers ingen andre mekanismer. Figur 1 viser gjennomsnittlige resultater for kjøringer med fast TT for det utvalgte *medium* store testproblemet, med TT i området fra 0 til 110. Den optimale løsning er 21891. Som man kan se er den beste verdien rundt 40. I tillegg viser figuren gjennomsnittlig resultat med dynamisk tabu-aktivering, med dynamisk oppdatering av den adaptive komponenten i trekkevalueringen, w, samt med begge mekanismene. I tilfellet vist i figur 1 har antall iterasjoner med dynamisk tabuaktivering nedre grense lik 10 og øvre grense like tallet på x-aksen. Hver gang en variabel flippes blir den altså tabuaktiv i et tilfeldig valgt antall iterasjoner fra dette intervallet. I testene med dynamisk trekkevaluering ble det benyttet verdier $\Delta w_{ink} = 0.10$ og $\Delta w_{dek} = 0.05$.

Åpenbart er effekten av disse mekanismene ikke uavhengige, ettersom langt kortere TT er nødvendig når man benytter den selvtilpassende vekten, w, i trekkevalueringen. Figuren indikerer at søket blir nokså ufølsomt overfor spennet av tabuaktivering når både dynamisk TT og adaptiv trekkevaluering brukes. Grafer tilsvarende figur 1 vil naturligvis være forskjellig for hvert problem. Tester for de øvrige utvalgte testproblemene viser tilsvarende resultater, og en dynamisk tabuaktivering i intervallet [10-15] ble brukt for videre testing.



Figur 2

Verdier for oppdatering av adaptiv trekkevaluering

Trekkevalueringen er som nevnt basert på funksjonen $F_{Mj} = \Delta T_i + w * \Delta f_i$, hvor den relative vekten av endringer i målfunksjonen i forhold til endring i antall tilfredsstilte rader i restriksjonen (1.2) kontrolleres av w. Denne parameteren endrer verdi dynamisk som forklart i seksjon 3.3. Av viktighet er å finne gode verdier for oppdatering (inkrementering og dekrementering) av w, altså verdier for Δw_{ink} og Δw_{dek}. Det synes som om det er forholdet mellom verdiene, $\Delta w_{ink} / \Delta w_{dek}$, og ikke størrelsen på dem, som er viktigst. Dette er illustrert i figur 2, hvor gjennomsnittlig verdi for målfunksjonen er vist for ulike kombinasjoner av Δw_{ink} og Δw_{dek} for det *store* testproblemet. Tilsvarende resultater oppnås for de andre testproblemene. Det beste forholdet ser ut til å være et inkrement rundt 2.5 ganger større enn dekrementet, og for videre testing ble det valgt å bruke $\Delta w_{ink} = 0.90$ og $\Delta w_{dek} = 0.35$.

Effekt av aspirasjonskriteriet

Bruken av aspirasjonskriterier er antatt å være viktig i tabusøk, da tabukriteriet ellers vil begrense søket for mye. Denne antagelsen er sjelden dokumentert i litteraturen. Effekten av vårt valg av aspirasjonskriterium (å akseptere en ny beste løsning) er vist i tabell 1 for de 13 første klassene av testproblemer (se seksjon 4 for mer om testproblemene). Søket ble tildelt 5 sekunder per problem, hvor søket startes på nytt undervegs – som beskrevet nedenfor. Klassene 1-13 er blant de minste testproblemene, hvor optimum finnes nokså enkelt i de fleste tilfellene. Selv om resultatene uten bruk av aspirasjon er bedre enn resultatene rapportert av Davoine, Hammer og Vizvári (2001), er resultatene enda bedre, både med tanke på kvalitet og tid brukt til å finne beste løsning, når aspirasjonskriteriet benyttes.

Tabell 1. Effekt av aspirasjon

	Uten asp. i % av med asp.	Uten asp. Tid til beste	Med asp. Tid til beste
Klasse 1	99.985	0.05	0.01
Klasse 2	100.000	0.01	0.00
Klasse 3	100.000	0.02	0.00
Klasse 4	100.000	0.05	0.00
Klasse 5	99.992	0.02	0.01
Klasse 6	100.000	0.02	0.00
Klasse 7	100.000	0.04	0.00
Klasse 8	99.985	0.03	0.03
Klasse 9	100.000	0.02	0.00
Klasse 10	100.000	0.03	0.00
Klasse 11	99.955	0.11	0.08
Klasse 12	99.980	0.06	0.01
Klasse 13	99.998	0.03	0.00

Parametre for langtidslæring

For dynamisk vekting av radene var det 3 parametre som måtte bestemmes; verdi for inkrement, ΔDR_{ink}, verdi for øvre grense, $\Delta DR_{\lim}$, samt divisoren når vektene normaliseres, ΔDR_{div}. Testene ble gjennomført på samme vis som tabu-aktivering, men uten entydige resultater. Dog antydet testingen at dynamiske radvekter jevnt over var bedre enn statiske vekter, og verdier for parametrene ble derfor valgt fra intervaller som syntes lovende – hvilket gav henholdsvis $\Delta DR_{ink} = 0.003$, $\Delta DR_{\lim} = 4$ og $\Delta DR_{div} = 2$. For mer om parameterinnstillingen, og andre forsøk på langtidslæring i dette rammeverket, se Hvattum (2002).

Om å starte søket på nytt undervegs

Uten bruk av spesielle mekanismer for å spre søket er det sannsynlig at søket blir mindre effektivt etter en stund, om det holder seg i den samme delen av søkerommet. Det er derfor vanligvis på sin plass med en metode som sørger for å flytte søket til nye steder. Vårt valg har falt på ganske enkelt å starte søket på nytt, fra et tilfeldig sted, etter et antall iterasjoner uten at en ny beste løsning er funnet. Tabusøk fremmer normalt bruk av mer strategiske metoder for diversifikasjon, slik som de ovenfornevnte dynamiske radvektene, men vi har funnet at en restart-strategi uansett er lønnsomt i dette adaptive lokalsøk-rammeverket.

Vi kjørte en rekke tester for å avgjøre hvor lenge man burde vente før søket ble startet på nytt. Løsningstidene for disse testene indikerte at antall iterasjoner mellom restartene hadde en sammenheng med problemstørrelsen uttrykt i både antall variabler, antall rader i restriksjonen (1.2) samt antall elementer ulik 0 per rad. Valget falt på å la søket bli startet på nytt etter 5* R_I iterasjoner uten forbedring, hvor R_I er produktet av antall variabler og gjennomsnittlig antall elementer ulik 0 per rad.

4 Resultater

For å prøve ut metodene våre har vi benyttet den samme problemporteføljen som Davoine, Hammer og Vizvári (2001). Disse kan hentes med anonym ftp fra *rutcor.rutgers.edu* og katalogen */pub/BOP*. Vi rapporterer våre resultat i samme rammeverk som disse, for å gjøre sammenligninger enklere.

Det er tre grupper av testproblemer, alle tilfeldig generert, hvor klasse 1-49 er tilfeldige problemer, klasse 50-54 er Graph Stability-problemer og klasse 55-63 er Set Covering-problemer – med til sammen 5485 testproblemer. For nærmere forklaring av problemporteføljen, se Davoine, Hammer og Vizvári (2001).

Vi sammenligner våre resultater med resultatene fra Davoine, Hammer og Vizvári (2001), så vel som med XPRESS/MP v.12 (http://www.dash.co.uk/) og CPLEX v.6.5 (http://www.ilog.com/products/cplex/). Vår kode er implementert i Visual C++ 6.0 og kjører på en standard 1 GHz Pentium 3 PC med MS Windows 2000. Våre kjøringer av XPRESS/MP er fra en 400 MHz Sun UltraSparc. Davoine, Hammer og Vizvári (2001) kjørte sine eksperimenter på en 50 MHz Sun Sparcsation 5, og brukte CPLEX 6.0 for sammenligning.

Vi kjørte følgende serier av tester på alle testproblemene (oppgitt er maksimal tid for hver probleminstans):

- Adaptivt lokalsøk, 5 sekunder
- Adaptivt lokalsøk, 60 sekunder
- XPRESS/MP v. 12, 4 timer

For søket brukte vi dynamisk tabuaktivering (10-15 iterasjoner), R_I som forklart ovenfor, $\Delta w_{ink} = 0.90$ og $\Delta w_{dek} = 0.35$. Dynamiske radvekter ble initielt satt lik 1, og de øvrige parametrene $\Delta DR_{ink} = 0.003$, $\Delta DR_{\lim} = 4$ og $\Delta DR_{div} = 2$.

Tabell II. Resultatoversikt

	Davoine et al.	**ALS – 5 sek.**	**ALS – 60 sek.**	**XPRESS**
Klasse 1 – 22	99.161	100.002	100.002	100.002
Klasse 23 – 49	100.440	101.212	101.214	101.127
Klasse 50 – 54	102.806	107.628	107.866	85.357
Klasse 55 – 63	101.238	102.462	102.464	102.237
Klasse 1 – 63	100.295	101.477	101.497	99.641

Utfallet av kjøringene vises i sammendrag i tabell 2. Vår metode er under overskriften ALS (adaptivt lokalsøk).Tallene er oppgitt i % av resultatene av kjøringer med CPLEX rapportert av Davoine, Hammer og Vizvári (2001).

Radene i tabellen representerer henholdsvis små og støre tilfeldige problemer, Graph Stability-problemer, Set Covering-problemer og til slutt alle problemene under ett.
For mer detaljerte resultater, se Hvattum (2002).

Den beste løsningen finnes svært raskt av ALS for de minste problemene, mens en stor andel av tiden brukes for å finne en slik løsning for større problemer. Å tildele mer kjøretid til de store problemene kan derfor være fordelaktig. Kanskje noe overraskende, vår enkle implementasjon tatt i betraktning, er kjøringene begrenset til 5 sekunder også svært konkurransedyktige – selv for de største testproblemene.

Søket vårt er bedre enn Davoine, Hammer og Vizvári (2001), både med hensyn til løsningskvalitet og –tid (se Hvattum (2002)). For større testproblemer er vi også klart bedre (og raskere) enn XPRESS/MP og CPLEX. (XPRESS bruker vanligvis en betydelig andel av sine 4 tildelte timer på å finne sine beste resulater).

5 Konklusjoner og fremtidig arbeid

Boolske optimeringsproblemer representerer en stor klasse binære optimeringsproblemer, og følgelig er det viktig å kunne løse rimelig store probleminstanser raskt og effektivt. Vi har beskrevet et adaptivt lokalsøk (basert på tabusøk) for å løse slike problemer, utformet for å oppnå en strategisk oscillasjon rundt grensene til restriksjonene i problemene. Søket forsøker å koordinere viktigheten av å oppfylle restriksjonene og verdien av målfunksjonen. Metoden overgår klart tidligere spesialiserte metoder utviklet for disse problemene, både med tanke på løsningskvalitet og –tid, og også kommersielle løsere som XPRESS/MP og CPLEX.

Vår løsning har til gode å inkludere noen av de mer avanserte komponentene fra tabusøk. Det forventes at læring basert på frekvensbaserte mekanismer vil gi ytterligere forbedringer i ytelsen. Konstruktive løsere basert på de samme prinsippene er også en mulig veg å følge.

Referanser

S. A. Cook. (1971). “The complexity of theorem-proving procedures”. Proceedings of the Third ACM Symposium on Theory of Computing, sidene 151-158.

Davoine, Thomas, Peter L. Hammer and Béla Vizvári. (2001). “A Heuristic for Boolean optimization problems”. Kommende i *Journal of Heuristics*.

Glover, Fred and Gary Kochenberger.(1996). “Critical Event tabu Search for Multidimensional Knapsack Problems”, I I.H. Osman and J.P. Kelly, redaktører, *Meta Heuristics: Theory and Applications*, Kluwer Academic Publishers, sidene 407 – 427.

Glover, Fred and Manuel Laguna. (1997). **Tabu Search**. Kluwer Academic Publishers.

Hvattum, Lars Magnus. (2002). ”Heuristikker for boolske optimeringsproblemer”, Hovedoppgave ved Høgskolen i Molde.

Hvattum, Lars Magnus, Arne Løkketangen og Fred Glover. (2002). ”Adaptive Memory Search for Boolean Optimization Problems” Til vurdering for “Discret Applied Mathematics”

Løkketangen, Arne and Fred Glover. (1997). “Surrogate Constraint Analysis - New Heuristics and Learning Schemes for Satisfiability Problems”. I: *Satisfiability Problem: Theory and Applications.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol 35.

Du, Dingzhu, Jun Gu and Panos Pardalos (Eds.). (1997). *Satisfiability Problem: Theory and Applications.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol 35.

Battiti, Roberto.(1996). “Reactive search: Toward self-tuning heuristics**”.** I V. J. Rayward-Smith, redaktør, **Modern Heuristic Search Methods**, kapittel 4, sidene 61-83. John Wiley and Sons Ltd, 1996.