# En introduktion til heltalsprogrammering*)

Af Jørgen Tind**)

*Resumé*

*Nærværende artikel giver en indføring i emnet heltalsprogrammering, der i de senere år har udviklet sig til et effektivt operationsanalytisk redskab til løsning af forskellige beslutningsproblemer i bl.a. investering og planlægning.*
*Der gives forskellige illustrerende eksempler på anvendelser tilligemed en oversigt over løsningsmetoder. Endelig bringes en bogfortegnelse, suppleret med enkelte kommentarer.*

*) Foredrag, holdt ved DORS-dag om heltalsprogrammering, november 1975.

**) Lektor ved Institut for Operationsanalyse (IFOR), Aarhus Universitet.

# En introduktion til heltalsprogrammering*)

Af Jørgen Tind**)

*Resumé*

*Nærværende artikel giver en indføring i emnet heltalsprogrammering, der i de senere år har udviklet sig til et effektivt operationsanalytisk redskab til løsning af forskellige beslutningsproblemer i bl.a. investering og planlægning.*
*Der gives forskellige illustrerende eksempler på anvendelser tilligemed en oversigt over løsningsmetoder. Endelig bringes en bogfortegnelse, suppleret med enkelte kommentarer.*

*) Foredrag, holdt ved DORS-dag om heltalsprogrammering, november 1975.

**) Lektor ved Institut for Operationsanalyse (IFOR), Aarhus Universitet.

# 1. Indledning og definition

Heltalsprogrammering er et model- og algoritmeværktøj, som er nært beslægtet med lineær programmering, og som navnlig finder anvendelse på optimeringsproblemer af kombinatorisk natur. Som foreløbige eksempler herpå kan nævnes beslutningsproblemer inden for investerings- og produktionsplanlægning, herunder f.eks. vedrørende oprettelse og nedlæggelse af depoter, diverse bemandingsproblemer o.lign.
Lad os imidlertid først se på den modelteoretiske definition af heltalsprogrammering. Betragt derfor følgende programmeringsproblem:

$$
\begin{aligned}
\max \quad & cx \\
& Ax \leq b \\
& x \geq 0 \\
& x \text{ heltallig,}
\end{aligned}
\tag{1.1}
$$

hvor $c \in R^n$, $A \in R^m \times R^n$ og $b \in R^m$ er konstanter, og hvor $x \in R^n$ er en vektor af variable. At x forlanges heltallig betyder, at hver komponent i x skal være et helt tal. Da dette her gælder samtlige komponenter, kaldes (1.1) et *rent heltalsprogrammeringsproblem* (pure integer programming).
Hvis kravet om heltallighed fjernes, ses vi at være tilbage i lineær programmering, der som bekendt byder på gode løsningsmetoder. Dette er desværre normalt ikke tilfældet for problemer af typen (1.1). Derfor bør man altid overveje, om man kan opnå en tilstrækkelig god løsning til (1.1) ved at løse det som et lineært programmeringsproblem med en efterfølgende afrunding af de variable.
Dette vil normalt være tilfældet ved store løsningsværdier i x-vektoren. Problemet stiller sig imidlertid anderledes, hvis $x = (x_1, \ldots, x_n)$ er begrænset til små værdier, som f.eks. ved følgende specialtilfælde af heltalsprogrammering:

$$\max \quad cx$$

$$Ax \leq b \tag{1.2}$$

$$x_j = \begin{cases} 0 \\ 1 \end{cases} \quad , j = 1,\ldots, n.$$

(1.2) kaldes et *0-1 programmeringsproblem,* og det er normalt modeller af den type eller variationer heraf, der byder på de fleste anvendelser af heltalsprogrammering.

*Eksempel, investeringsplanlægning*

En investor har hvert år et på forhånd fastsat beløb til rådighed for fortsat investering i n mulige projekter. Lad $b_i$ være beløbet, der er til rådighed i det i'te år. Lad endvidere $a_{ij}$ betegne kapitalbehovet for det j'te projekt i det i'te år, og lad $c_j$ betegne den samlede gevinst fra det j'te projekt. Betragtes en periode på m år, bliver problemet herefter at udvælge projekter, således at den totale gevinst maksimeres uden at budgettet overskrides i noget år. Lad

$$x_j = \begin{cases} 1, \text{ hvis j'te projekt sættes i gang} \\ 0, \text{ ellers,} \end{cases}$$

og vi får derfor følgende 0-1 programmeringsproblem:

$$\max \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad , i = 1,\ldots,m$$

$$x_j = \begin{cases} 0 \\ 1 \end{cases} \quad , j = 1,\ldots, n,$$

som i øvrigt i matrixsprog er identisk med (1.2).
Ovenstående meget simple model kan udvides og generaliseres på forskellige måder. Der henvises til Weingartner [21] og Thygesen [20].

*Eksempel, flybemanding*

Et luftfartsselskab skal have fastlagt en bemandingsplan for f.eks. piloter i deres fly inden for en periode. Selskabet kender på forhånd, hvilke ruter (flyben) der skal bemandes i løbet af perioden. Med en periodelængde på en uge kunne eksempelvis tænkes på flyben af følgende tilsnit:

1. København - New York, mandag
2. Paris - Rom, mandag

–
–
–
–

n. Tokyo - København, søndag

Man opregner nu, normalt ved hjælp af en datamat, samtlige kombinationsmuligheder af flyben, som kan bemandes med en pilot. Dette sker under hensyntagen til bemandingsregler omkring maksimal flyvetid for en pilot, antal sammenhængende fridage o.s.v. Lad n være det samlede antal kombinationer, og lad m være antallet af flyben. Lad endvidere $a_{ij}$ være elementer i en m × n matrix, hvor

$$a_{ij} = \begin{cases} 1, & \text{hvis j'te kombination bemander det i'te flyben} \\ 0, & \text{ellers.} \end{cases}$$

Selskabet ønsker nu at minimere antallet af piloter, som er nødvendige til bemanding af de givne flyben. Dette problem er ækvivalent med følgende 0-1 programmeringsproblem:

$$\min \sum_{j=1}^{n} x_j$$

$$\sum_{j=1}^{n} a_{ij} \; x_j \geq 1 \qquad \text{for } i = 1,\ldots, m \tag{1.3}$$

$$x_j = \begin{cases} 0 \\ 1 \end{cases} \quad \text{for } j = 1,\ldots, n \quad ,$$

hvor $x_j = 1$, hvis j'te kombination bruges.

Uligheden i (1.3) sikrer, at ethvert fly bliver bemandet. Det tillades endda at tage en pilot med »som passager«. Ønskes dette ikke, indføres i stedet for ulighedstegnet et lighedstegn. (1.3) tilhører typen af såkaldte overdækningsproblemer (setcovering). For yderligere materiale vedr. flybemanding henvises til Arabeyre et al. [14] og Ellervik [17]. For overdækningsproblemer generelt henvises til Garfinkel and Nemhauser [3].

Der har hidtil kun været omtalt rene heltalsprogrammeringsproblemer, hvori samtlige variable kræves heltallige. Man opererer imidlertid ofte med modeller af blandet type, hvor kun nogle variable kræves heltallige, og resten er kontinuerte. Man taler da om *blandet heltalsprogrammering* (mixed integer programmering), som formelt kan opstilles på følgende måde:

$$\begin{aligned} \max \quad & c_1x_1 + c_2x_2 \\ & A_1x_1 + A_2x_2 \leq b \\ & x_1 \geq 0 \text{ og heltallig} \\ & x_2 \geq 0 \quad , \end{aligned}$$

hvor $c_1 \in R^p$, $c_2 \in R^q$, $x_1 \in R^p$, $x_2 \in R^q$,

$A_1 \in R^m \times R^p$, $A_2 \in R^m \times R^q$ og $b \in R^m$.

*Eksempel, Plant Location (Warehouse Location).*

En virksomhed overvejer at etablere en række depoter til forsyning af n markeder med en vare. Efterspørgslen efter varen kendes på forhånd for hvert marked. Lad $b_j$ betegne efterspørgslen i det j'te marked i en valgt periode. Der er m lokaliteter for mulig placering af depoter. For hver lokalitet kendes på forhånd en øvre grænse for leverance, bestemt af kapaciteten af et evt. depot. Lad $a_i$ betegne kapaciteten af det i'te depot. I forbindelse med oprettelse af depoter påføres endvidere en fast omkostning, som for det i'te depot, er $k_i$ kroner. Det aktuelle tal kan være amortisationsomkostninger i den valgte periode ($k_i \geq 0$). Herudover findes omkostninger i forbindelse med transport af varer, som udgør $c_{ij}$ kroner pr. enhed for transport fra i'te depot til j'te marked.

Lad $x_{ij}$ betegne den transporterede varemængde fra det i'te depot til det j'te marked. Problemet for virksomheden består nu i at finde, hvilke depoter der skal etableres, og bestemme varestrømmene $x_{ij}$, således at de samlede udgifter til transport og etablering minimeres under hensyntagen til, at al efterspørgsel tilfredsstilles.

Vi indfører en 0-1 vektor $y = (y_1, \ldots, y_m)$, hvor

$$y_i = \begin{cases} 1, & \text{hvis i'te depot etableres} \\ 0, & \text{ellers.} \end{cases}$$

Problemet kan herefter formuleres som følgende blandede heltalsprogrammeringsproblem:

$$\begin{aligned} \min_{y,x} \quad & \sum_{i=1}^{m} k_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ & \sum_{i=1}^{m} x_{ij} = b_j \quad , j = 1, \ldots, n \\ & \sum_{j=1}^{n} x_{ij} \leq a_i y_i \quad , i = 1, \ldots, m \\ & x_{ij} \geq 0 \quad \text{for alle i,j} \\ & y_i = \begin{cases} 0 \\ 1 \end{cases} , i = 1, \ldots, m. \end{aligned} \tag{1.4}$$

Det skal bemærkes, at for fastholdte værdier af de heltalsvariable y, reduceres (1.4) til *det klassiske transportproblem*. Dette problem har en særlig struktur, idet koefficienterne på venstre side af bibetingelserne udgør en såkaldt unimodulær matrix. Heraf følger, at hvis konstanterne $a_i$ og $b_j$ er heltallige, findes en optimal løsning til (1.4), hvori $x_{ij}$ også er heltallig. En sådan (basis-) løsning opnås med alle gængse algoritmer til løsning af (1.4).

Plant location problemet kan modificeres og generaliseres på forskellige leder, hvilket har haft betydning for anvendelserne. Emnet har endvidere været underkastet en omfattende behandling i litteraturen. Der skal her henvises til en oversigt i Salkin [11] og til 0. Bilde [15].

## 2. To klassiske eksempler

*Rygsæksproblemet* (knapsack problem, cargo loading).

En bjergbestiger skal pakke sin rygsæk med varer før bjergbestigning. Han kan vælge mellem n forskellige varer, $c_j$ og $a_j$ betegner henholdsvis værdien og vægten af den j'te vare. Bjergbestigeren ønsker nu at pakke sin rygsæk, således at den samlede værdi maksimeres uden at overskride en vægtbegrænsning på b. Han skal derfor løse følgende heltalsprogrammeringsproblem:

$$\max \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_j x_j \leq b$$

$$x_j = \begin{cases} 0 \\ 1 \end{cases}, \quad j = 1, \ldots, n,$$

hvor $x_j$ antager værdien 1, hvis j'te vare medbringes i rygsækken.

Det kan vises, at ethvert begrænset heltalsprogrammeringsproblem (1.1) kan omformuleres til et rygsæksproblem. Da rygsæksproblemer er noget simplere at løse end (1.1) foregår der for tiden en betydelig forskningsaktivitet omkring forskellige omformuleringer af (1.1) til et rygsæksproblem. Det skal dog påpeges, at disse omformuleringer ofte medfører, at koefficienterne i rygsæksproblemet får store numeriske værdier, hvilket forårsager nogle komplikationer ved selve omformuleringen og ved den efterfølgende løsning af problemet. Rygsæksproblemet spiller også en central rolle i de såkaldte gruppeteoretiske metoder for løsning af heltalsprogrammeringsproblemer.

*Den handelsrejsende* (Travelling Salesman).

En handelsrejsende skal besøge n byer. Det antages, at der findes direkte veje mellem samtlige par af byer. Lad $d_{ij}$ betegne afstanden mel-

lem by i og by j. Han skal nu søge at vælge en rute, som minimerer den samlede rejseafstand, således at han vender tilbage til sit udgangspunkt, som er en af byerne, efter at have besøgt hver af de øvrige byer netop en gang.

Indfør for hvert par af byer en 0-1 variabel $x_{ij}$, hvor

$$x_{ij} = \begin{cases} 1, \text{ hvis han rejser direkte fra by i til by j} \\ 0, \text{ ellers.} \end{cases}$$

Den handelsrejsende skal herefter løse følgende 0-1 programmeringsproblem:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

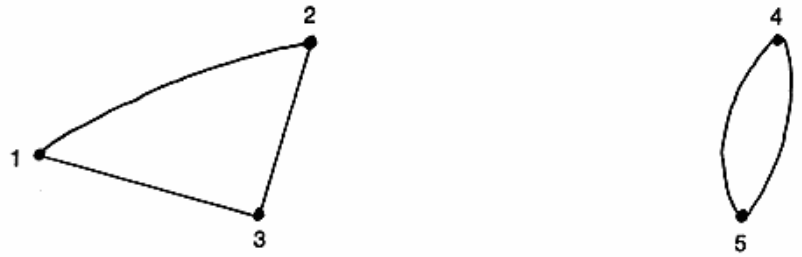
(2.1)
$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1,\ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1,\ldots,n$$

»ingen subcykler«

$$x_{ij} = \begin{cases} 0 \\ 1 \end{cases} \text{ for alle i,j.}$$

De to første bibetingelser i (2.1) sikrer, at der foretages netop en indrejse og en udrejse for hver by. Dette er imidlertid ikke tilstrækkeligt til at sikre at løsningen bliver en sammenhængende rute, idet de to bibetingelser også opfyldes af flere uafhængige lukkede ruter, subcykler, som illustreret på følgende figur for 5 byer:




Kravet om »ingen subcykler« kan udtrykkes ved hjælp af lineære uligheder og ligninger, men det undlades her. Der er ikke her gjort nogen antagelse om at $d_{ij} = d_{ji}$, og man kan udmærket operere med det asymmetriske tilfælde, hvor $d_{ij}$ kan være forskellig fra $d_{ji}$.
Travelling salesman problemet er stadig en udfordring for konstruktører af løsningsmetoder for problemet. Der udvikles hele tiden nye og forbedrede algoritmer. For øjeblikket er det muligt at løse travelling salesman problemet med en ca. 100 byer, d.v.s. med næsten 10000 0-1 variable, eller (100-1)! mulige ruter. Blandt de senere arbejder kan nævnes K. Helbig Hansen og J. Krarup [19].
Travelling salesman modellen dækker over en idealiseret problemstilling, men den har haft betydelig indflydelse på forskellige løsningsmetoder til udformning af ruteplanlægning o.lign.
Der findes flere såkaldt klassiske heltalsprogrammeringsproblemer, hvoraf de fleste beskæftiger sig med problemer omkring kombinatoriske forhold i netværk. Se f.eks. Garfinkel and Nemhauser [3] eller J. Clausen [16].

## 3. Logiske bindinger

Heltalsprogrammering bliver ofte benyttet ved formuleringer af restriktioner, som svarer til sætninger af typen »enten... eller...«, »hvis... så...« o.lign.

*Eksempel, foderblanding*

I en foderblanding kan indgå en række forskellige råvarer. Hvis den i'te råvare indgår, skal det dog ske med mindst $a_i$ enheder af hensyn til prisrabatter ved indkøb og transport. Lad $x_i$ betegne mængden af den i'te råvare. D.v.s. at netop en af to følgende relationer skal holde:

$$\begin{aligned} x_i &= 0 \\ x_i &\geq a_i. \end{aligned} \tag{3.1}$$

Indfør en 0-1 variabel $y_i$, hvor

$$y_i = \begin{cases} 1, \text{ hvis i'te råvare indgår} \\ 0, \text{ ellers.} \end{cases}$$

Lad $b_i$ betegne en øvre grænse for forbrug af den i'te råvare (eller et stort tal). (3.1) kan da omformuleres til følgende to lineære relationer, som skal holde samtidigt:

$$\begin{aligned} & a_i y_i \leqq x_i \\ \text{og} \quad & x_i \leqq y_i b_i. \end{aligned}$$

Der kan også være et krav om, at der højst må indgå m ud af ialt n forskellige råvarer i blandingen på grund af et begrænset antal blandesiloer eller lign. Med den indførte notation kan dette krav formuleres på følgende måde ved hjælp af lineære relationer:

$$\begin{aligned} y_1 + \ldots + y_n &\leq m \\ x_i &\leq y_i b_i. \end{aligned}$$

Det ses, at disse relationer sikrer at højst m variable x kan være positive.

Som bekendt benyttes lineær programmering ofte som et hjælpemiddel til at opnå en billig blanding under hensyn til betingelser som f.eks. minimalt indhold af forskellige næringsstoffer. Med en eventuel

tilføjelse af krav af ovenstående type får vi herefter udvidet vores problemstilling til et blandet heltalsprogrammeringsproblem.

Det skal endelig bemærkes, at heltalsprogrammering er et ofte anvendt formuleringsværktøj for adskillige *sekvensproblemer*. I den forbindelse skal det nævnes, at her hyppigt indgår restriktioner svarende til diverse logiske bindinger. Sæt f.eks.:

$$x_{ij} = \begin{cases} 1, & \text{hvis job i udføres før job j på en maskine} \\ 0, & \text{ellers} \end{cases}$$

Det må da gælde, at

$$x_{ij} + x_{ji} = 1.$$

## 4. Algoritmer

Løsningsmetoder til heltalsprogrammering falder i det væsentlige i følgende kategorier:

a) Branch and Bound
b) Implicit Enumeration
c) Cutting Plane
d) Benders partitioning metode
e) Gruppeteoretiske metoder
f) Dynamisk Programmering

Der vil ikke her blive givet en beskrivelse af de enkelte metoders virkemåde. Vi vil indskrænke os til en summarisk redegørelse for deres udbredelse og effektivitet.

*Ad a) Branch and Bound*

Dette algoritmeprincip er så langt det mest anvendte af dem allesammen. Efter dette princip er således alle standardprogrampakkerne fra datamatfirmaerne udformet, og herudover er der udviklet en lang række branch and bound algoritmer for mere specielle problemer. Blandt standardprogrampakkerne kan omtales CDC's Ophelie Mixed og IBM's MPSX-MIP. Antallet af heltalsvariable er en kritisk størrelse i relation til forbrugt regnetid, og disse programpakker er normalt i stand til at løse heltalsprogrammeringsproblemer af generel natur op til ca. 50 heltalsvariable. Det skal dog understreges, at beregningstiden er ret afhængig af det aktuelle problem, ligesom brugeren normalt har indflydelse på regnetiden ved fastlæggelse af forskellige parametre til styring af beregningsgangen.

*Ad b) Implicit Enumeration*

Implicit enumeration er nært beslægtet med Branch and Bound, og undertiden skelnes der ikke synderligt herimellem. Man plejer dog at anvende betegnelsen implicit enumeration i forbindelse med algoritmer for 0-1 programmeringsproblemer uden kontinuerte variable. Algoritmer for sådanne problemer opbygget efter implicit enumerations princippet er forholdsvis simple at programmere for en datamat, og de har derfor fået en vid udbredelse.

*Ad c) Cutting Plane*

Cutting Plane princippet er historisk set det første princip, der blev anvendt ved udformning af algoritmer for heltalsprogrammering. Der forskes og skrives stadig meget om cutting plane, og der er opnået flere interessante teoretiske resultater. Desværre er dette ikke tilfældet for anvendelserne, idet algoritmer, der bygger på cutting plane princippet, ofte kræver en lang beregningstid. Der er dog en enestående undtagelse, og den gælder for setcovering problemer, hvor flybemandingsproblemer med op til 4000 0-1 variable er løst med en cutting plane metode (Martin, se Geoffrion og Marsten [5].

*Ad d) Benders partitioning metode*

Denne metode går tilbage til 1962 og har ført en relativ upåagtet tilværelse indtil for nylig, hvor den har fundet anvendelse flere steder. Metoden er specielt beregnet på blandede heltalsprogrammeringsproblemer.

*Ad e) Gruppeteoretiske metoder*

Der udfoldes for tiden store anstrengelser med udvikling af algoritmer af nævnte type. Men man har hidtil opnået relativt få eksperimentelle erfaringer.

*Ad f) Dynamisk programmering*

Dynamisk programmering er et vigtigt rekursivt beregningsprincip, som har fundet anvendelse på en udstrakt klasse af optimeringsproblemer. Heri er også indbefattet heltalsprogrammeringsproblemer, idet dog dynamisk programmering især egner sig for visse typer af heltalsprogrammeringsproblemer. Som et eksempel herpå kan nævnes rygsæksproblemet, som er omtalt i afsnit 2.

## 5. Ikke lineær heltalsprogrammering

Vi har indtil nu udelukkende beskæftiget os med problemer, hvor både objektfunktion og bibetingelser er lineære funktioner i x. Der findes situationer, også i praksis, hvor det er formålstjenligt at operere med en videre klasse af funktioner, f.eks. polynomier i x.
I et problem fra investeringsplanlægning (se afsnit 1) kan der f.eks. være en vis økonomisk afhængighed mellem projekter, således at

igangsættelse af både det i'te og det j'te projekt vil medføre nogle stordriftsfordele i form af besparelser. Lad f.eks. d betegne besparelsen i et bestemt år. Denne besparelse kan udtrykkes algebraisk ved at indføre følgende ikke lineære led i kapitalforbruget det pågældende år:

$$- dx_i x_j.$$

Et klassisk eksempel indenfor ikke lineær 0-1 programmering er det såkaldte *kvadratiske assignment problem:*
Lad der være givet m placeringsmuligheder (steder) for anbringelse af n afdelinger i en virksomhed ($m \geq n$). Lad $c_{ikjh}$ angive kommunikationsomkostningerne mellem den i'te afdeling, hvis denne anbringes på sted k, og den j'te afdeling, hvis denne anbringes på sted h. Typisk kan $c_{ikjh}$ være produktet af transportbehovet $a_{ij}$ mellem afdelingerne og afstanden $d_{kh}$, d.v.s. $c_{ikjh} = a_{ij} d_{kh}$.
Problemet består nu i at anbringe afdelingerne, således at de samlede kommunikationsomkostninger minimeres. Lad derfor

$$x_{ik} = \begin{cases} 1, \text{ hvis afdeling i anbringes på sted k} \\ 0, \text{ ellers.} \end{cases}$$

Vi får da følgende kvadratiske 0-1 programmeringsproblem:

$$\min \tfrac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{m} \sum_{j=1}^{n} \sum_{h=1}^{m} c_{ikjh}\, x_{ik}\, x_{jh}$$

$$\sum_{i=1}^{n} x_{ik} \leq 1 \quad \text{for } k = 1,\ldots,m$$

$$\sum_{k=1}^{m} x_{ik} = 1 \quad \text{for } i = 1,\ldots,n$$

$$x_{ik} = \begin{cases} 0 \\ 1 \end{cases} \text{ for alle i,k.}$$

Problemet antages symmetrisk, d.v.s. $c_{ikjh} = c_{jhik}$. Herved tælles identiske led, hvor $(i,k) \neq (j,h)$, med to gange i objektfunktionen. Dette

forklarer tilstedeværelsen af faktoren ½, idet vi for simpeltheds skyld her yderligere antager, at $c_{ikik} = 0$ for alle (i,k). Bibetingelserne sikrer, at hver afdeling bliver anbragt et sted, og at hvert sted får højst en afdeling.

Der findes algoritmer til løsning af ikke-lineære heltalsprogrammeringsproblemer. De minder meget om dem, som er nævnt i afsnit 4, og er ofte af type a) eller b). Der kan henvises til Garfinkel and Nemhauser [3] eller Hammer and Rudeanu [7]. Vedrørende en specialoversigt over løsningsmetoder for kvadratisk assignment henvises til Francis and White [18].

Man bør dog her holde sig for øje, at hvis der kun findes få led, som ikke er lineære i x, vil man med fordel kunne omformulere problemet til et lineært problem, illustreret ved følgende eksempel:

Lad $x_1$ og $x_2$ være 0-1 variable.

Et sted i et heltalsprogrammeringsproblem indgår de i et produktled:

$$x_1 \cdot x_2$$

Dette led kan erstattes med en ny 0-1 variabel $x_3$ ved tilføjelse af denne variabel til problemet plus følgende to lineære bibetingelser:

$$x_1 + x_2 - x_3 \leq 1$$

$$-x_1 - x_2 + 2x_3 \leq 0.$$

De vil sikre, at $x_3 = 1$, hvis og kun hvis $x_1 \cdot x_2 = 1$.

*Nogle bøger og oversigtsartikler om heltalsprogrammering:*

[1] Abadie, J., editor, *»Integer and Nonlinear Programming«*, North Holland, 1970. Proceedings fra en NATO sommerskole i Bandol, Frankrig.

[2] Balinski, M. L., *»Integer Programming, Methods, Uses, Computation«*, *Management Science*, vol 12, pp. 253-313. En oversigtsartikel.

[3] Garfinkel, R., and G. L. Nemhauser: *»Integer Programming«*, Wiley 1972. En omfattende bog på over 400 sider. Ret matematisk orienteret.

forklarer tilstedeværelsen af faktoren ½, idet vi for simpeltheds skyld her yderligere antager, at $c_{ikik} = 0$ for alle (i,k). Bibetingelserne sikrer, at hver afdeling bliver anbragt et sted, og at hvert sted får højst en afdeling.

Der findes algoritmer til løsning af ikke-lineære heltalsprogrammeringsproblemer. De minder meget om dem, som er nævnt i afsnit 4, og er ofte af type a) eller b). Der kan henvises til Garfinkel and Nemhauser [3] eller Hammer and Rudeanu [7]. Vedrørende en specialoversigt over løsningsmetoder for kvadratisk assignment henvises til Francis and White [18].

Man bør dog her holde sig for øje, at hvis der kun findes få led, som ikke er lineære i x, vil man med fordel kunne omformulere problemet til et lineært problem, illustreret ved følgende eksempel:

Lad $x_1$ og $x_2$ være 0-1 variable.

Et sted i et heltalsprogrammeringsproblem indgår de i et produktled:

$$x_1 \cdot x_2$$

Dette led kan erstattes med en ny 0-1 variabel $x_3$ ved tilføjelse af denne variabel til problemet plus følgende to lineære bibetingelser:

$$x_1 + x_2 - x_3 \leq 1$$

$$-x_1 - x_2 + 2x_3 \leq 0.$$

De vil sikre, at $x_3 = 1$, hvis og kun hvis $x_1 \cdot x_2 = 1$.

*Nogle bøger og oversigtsartikler om heltalsprogrammering:*

[1] Abadie, J., editor, *»Integer and Nonlinear Programming«*, North Holland, 1970. Proceedings fra en NATO sommerskole i Bandol, Frankrig.

[2] Balinski, M. L., *»Integer Programming, Methods, Uses, Computation«*, *Management Science*, vol 12, pp. 253-313. En oversigtsartikel.

[3] Garfinkel, R., and G. L. Nemhauser: *»Integer Programming«*, Wiley 1972. En omfattende bog på over 400 sider. Ret matematisk orienteret.

[4] Geoffrion, A., *»A guided tour of recent practical advances in integer linear programming«, SIGMAP Newsletter,* No. 17, pp. 22-32. Association for Computing Machinery, November 1974. En fortsættelse af [5].

[5] Geoffrion, A., and R. E. Marsten, *»Integer Programming: A Framework and State-of-the-Art Survey«, Management Science,* vol 18, pp. 465-491, 1972. En oversigt og en generel rammebeskrivelse af heltalsprogrammering, suppleret med eksperimentelle resultater om beregningstider.

[6] Greenberg, H., *»Integer Programming«,* Academic Press, 1971. En relativ kort bog.

[7] Hammer, P. L., and S. Rudeanu: *»Boolean Methods in Operations Research«,* Springer, 1968. En bog om relationerne mellem boolsk algebra og 0-1 programming. Herunder algoritmer for ikke lineære 0-1 programmeringsproblemer.

[8] Hu, T. C., *»Integer Programming and Network Flows«,* Addison-Wesley, 1969. Halvdelen (ca. 200 sider) af bogen er viet heltalsprogrammering, især om cutting plane og gruppeteoretiske metoder.

[9] Plane, D. R., and C. Mc. Millan: *»Discrete Optimization, Integer Programming and Network Analysis for Management Decisions«;* Prentice Hall, 1971. Elementær. Ca. 100 sider om heltalsprogrammering.

[10] Roy, B. (editor), *»Combinatorial Programming«,* Reidel, 1975. Konferenceproceedings.

[11] Salkin, H. M., *»Integer Programming«,* Addison-Wesley, 1975. En omfattende bog på ca. 500 sider.

[12] Taha, H. A., *»Integer Programming Theory, Applications and Computations«,* Academic Press, 1975. Ca. 400 sider.

[13] Zionts, S., *»Linear and Integer Programming«,* Prentice Hall, 1974. Ca. 200 sider om heltalsprogrammering. Introducerende uden særlig megen brug af matematik.

*Yderligere referencer til teksten:*

[14] Arabevre, J. P., J. Fearnly, F. C. Steiger and W. Teather, *»The Airline Crew Scheduling Problem: A Survey«, Transportation Science,* vol 3, pp. 140-163, 1969.

[15] Bilde, O., *»Nonlinear and Discrete Programming«,* IMSOR, Danmarks tekniske Højskole, 1970.

[16] Clausen, J., *»Beregningskompleksitet I + II«,* Speciale, Datalogisk Institut, Københavns Universitet, 1975.

[17] Ellervik, P., *»Operationelle kombinatoriske bemandingsproblemer«,* Handelshøjskolen i København, 1974.

[18] Francis and White, *»Facility Layout and Location, An Analytical Approach«,* Prentice Hall, 1974.

[19] Hansen, K. Helbig and J. Krarup, *»Improvements of the Held and Karp Algorithm for the Symmetric Travelling Salesman Problem«, Mathematical Programming,* vol. 7, pp. 87-96, 1974.

[20] Thygesen, I., *»Investeringsplanlægning«,* Polyteknisk Forlag, 1971.

[21] Weingartner, H. M., *»Mathematical Programming and the Analysis of Capital Budgeting Problems«,* Kershaw Publishing Company, 1974.