Tolv forslag til datalogiprojekter

Keld Helsgaun

Indholdsfortegnelse

1. Planters algoritmiske skønhed

Planters former har optaget matematikere i århundreder. Planters smukke geometriske træk, såsom symmetrien i blade og blomster, har været genstand for intensive studier. Målet har været at udtrykke planters *udseende* på en matematisk form.

I 1968 introducerede en biolog, A. Lindenmayer, en ny matematisk teori for planters *udvikling* [1]. Han indførte de såkaldte Lindenmayer-systemer (L-systemer) til at formulere algoritmer for, hvordan planter udvikler sig.

L-systemer giver en elegant notation til modellering og simulering af planters udvikling. Tidsmæssige forløb udtrykkes ved hjælp af relativt simple grammatiske regler. Et stadium i en plantes udvikling beskrives ved hjælp af en tegnfølge, og de grammatiske regler udtrykker, hvorledes tegnfølgen kan omskrives, dvs. ændres, så den repræsenterer et efterfølgende stadium i plantens udvikling.

Et L-system kan gives en grafisk tolkning gennem såkaldt *skildpaddefortolkning* af de indgående tegnfølger. Hvert tegn i følgen fortolkes som en kommando til en skildpadde, der kan bevæge sig på et stykke papir i forskellige retninger og tegne undervejs. Skildpaddens tilstand er defineret ved triplet (x,y,α), hvor (x,y) er koordinaterne for dens position, og α er en vinkel, der angiver skildpaddens orientering.

Givet en skridtlængde d og en vinkeltilvækst δ, så kunne kommandoerne til skildpadden for eksempel være følgende:

F	Gå et skridt af længde d fremad og tegn samtidig.
f	Gå et skridt af længde d fremad uden at tegne.
+	Drej vinklen δ til venstre.
-	Drej vinklen δ til højre.
[	Gem den aktuelle tilstand på en stak.
]	Lad den aktuelle tilstand være den øverste på stakken og afstak.

Jeg vil her nøjes med at give et enkelt eksempel, der illustrerer slagkraften af L-systemer.

n=4, d=22°
F
F $\Rightarrow$ F[+F]F[-F][F]

I første linje angives, at antallet af omskrivninger skal være 4, mens vinkeltilvæksten skal være 22°. Anden linje udtrykker, at starttegnfølgen skal være F. I den tredje linje defineres en omskrivningsregel, som udtrykker, at F overalt i en tegnfølge kan erstattes med udtrykket på højresiden af pilen.

Ved at fortolke den resulterende tegnfølge ved brug af skildpaddegrafik fås følgende busklignende vækst.

Det nævne eksempel er taget fra Prisinkiewicz og Lindendmayers imponerende bog [2]. Bogen giver en grundig indføring i L-systemer og er rigt illustreret med grafiske udtegninger, heriblandt mange i farver.

Jeg foreslår et datalogisk projekt, der med udgangspunkt i denne bog, har som mål at få udviklet et system til eksperimentering med L-systemer. Systemet skal kunne indlæse et L-system, simulere udviklingen og derefter vise resultatet grafisk ved hjælp af skildpaddefortolkning.

[1] A. Lindenmayer,
Mathematical models of cellular interaction in development. Part I and II,
Journal of Theoretical Biology. 18:280-315, 1968.

[2] P. Prusienkiewicz and A. Lindenmayer,
The Algorithmic Beauty of Plants,
Springer Verlag, 1990.

[3] M. Allen, P. Prusienkiewicz, and T. DeJong,
Using L–Systems for Modeling the Architecture and Physiology of Growing Trees:
The L–PEACH Model,
Proceedings of the 4th International Workshop on Functional–Structural Plant Models,
pp. 220–225, 2004
http://algorithmicbotany.org/papers/lpeach.fsmp2004.pdf

2. Systemsimulering

Her er ikke tale om et specifikt projektforslag, men snarere en temaoverskrift. Indenfor temaet kan f.eks. laves projekter af typen:

- simulering af X (hvor X er et eller andet system)
- undersøgelse af simuleringsparadigmer
- simuleringssprog
- effektiv organisering af hændelseskøer
- animation

Ved simulering forstås en efterligning af et tidsligt forløb.

Som eksempler på systemer, hvor simulering med fordel er blevet anvendt, kan nævnes:

- Menneskets kredsløb
- Solsystemets dannelse
- Økosystemer (f.eks. livet i en sø)
- Kødannelse (f.eks. lægekonsultationer og posthuse)
- Trafik (f.eks. S-togstrafik)
- Procesanlæg (f.eks. kraftværker)
- Opsendelse af rumraketter
- Legemers bevægelse (f.eks. himmellegemer eller billardkugler)

Disse eksempler er blot tænkt som inspiration til mulige projektvalg. Det skal imidlertid understreges, at der allerede i projektets start bør foreligge en model af det system, der ønskes simuleret. Desuden bør inddata være tilgængelige. Hvis disse to forudsætninger ikke er opfyldt, er det sædvanligvis umuligt, inden for projektets afgrænsede tidsramme, at nå frem til selve simuleringen af systemet. Al tiden vil nemlig så gå med opstilling af model, dataindsamling og verifikation af modellen. Det kan i denne forbindelse nævnes, at jeg har mulighed for at fremskaffe modeller og inddata til de fleste af ovennævnte eksempler.

Sædvanligvis skelner man skarpt mellem to typer af simulering, nemlig *kontinuert* simulering og *diskret* simulering.

I en kontinuert simulering er den dynamiske model udtrykt ved en sæt af koblede differentialligninger. Simuleringen omfatter numerisk løsning af de indgående differentialligninger.

I en diskret simulering er modellen udtrykt ved hjælp af *hændelser*, d.v.s. øjeblikkelige ændringer af systemets tilstand. Metoder til fremskrivning af modeltiden til næste hændelse er centrale i denne type af simuleringer.

Herudover kan det i visse situationer være hensigtsmæssigt at sammenblande de to simuleringstyper, nemlig i form af *kombineret kontinuert og diskret simulering*. Som et simpelt eksempel kan nævnes simulering af et køleskab. Den kontinuerte ændring af køleskabets temperatur som følge af dets varmeudveksling med omgivelserne kan udtrykkes ved hjælp af en sædvanlig førsteordens differentialligning. Termostaten, som slår til og fra, når temperaturen i køleskabet passerer visse tærskelværdier, giver anledning til hændelser. Et andet eksempel er simulering af blodsukkerkoncentrationen hos sukkersyge. Den kontinuerte variation af blodsukkerkoncentrationen ændres pludseligt som følge af fødeindtagelse og insulininjektioner.

Der findes mange udmærkede indføringer i simuleringsteknik, bl.a. [1]. En udmærket indføring i simuleringsparadigmer findes i [2].

[1] R. E. Shannon,
Systems simulation: the art and science,
Prentice-Hall, 1975.

[2] W. Kreutzer,
System simulation: programming styles and languages.,
Addison-Wesley, 1986.

[3] J. G. Vaucher,
Comparison of simulation event list algorithms.,
Comm. ACM, Vol. 18, 1975 (pp. 223-230).
http://portal.acm.org/citation.cfm?id=360758&coll=portal&dl=ACM

[4] J. H. Kingston,
Analysis of Three Algorithms for the Simulation Event List,
Acta Informatica, Vol. 22, April 1985 (pp. 15-34).

[5] K, Chung, J, Sang, and vernon Rego,
A Performance Comparison of Event Calendar Algorithms: an Empirical Approach,
Software - Practice and Experience, Vol. 23(10), 1993, pp. 1107-1138.
http://citeseer.ist.psu.edu/581982.html

3. Genetiske algoritmer

I 1839 udgav Charles Darwin sit hovedværk, *The Origin of Species*, hvori han fremlagde princippet om evolution gennem naturlig udvælgelse:

- Hvert individ vil videreføre egenskaber til sit afkom.
- Ikke desto mindre producerer naturen forskellige individer.
- De stærkeste individer får mere afkom end de svage, hvorved populationen som helhed får fordelagtige egenskaber.
- Over en lang periode kan variation ophobes og resultere i nye arter med særlige egenskaber.

Darwins evolutionsprincip er i dag almindeligt accepteret, men ikke mange ved, at princippet kan bruges til at konstruere effektive optimeringsalgoritmer, de såkaldte *genetiske algoritmer*. Denne type af algoritmer blev introduceret af J. H. Holland i 1986 [1].

En genetisk algoritme simulerer udviklingen i en population. Hvert individs arveanlæg er fastlagt ved dets gener. Individerne kan parre sig, hvorved afkommet arver nogle af forældrenes egenskaber efter fastlagte regler. Et "stærkt" afkom er et individ med egenskaber, som svarer til en løsning tæt på optimum.

Idet sandsynligheden for overlevelse stiger med individets styrke, vil populationen tendere mod stærke individer. Efter et vist antal generationer standses algoritmen med en population, hvor det stærkeste af individerne svarer til en løsning tæt på optimum.

Projektet går ud på at afprøve genetiske algoritmers effektivitet gennem et, eventuelt flere, udvalgte eksempelproblemer. Et muligt eksempel kunne være *den rejsende sælgers problem*, som, kort fortalt, går ud på at finde den korteste rejserute for en sælger, der skal besøge en række byer. Men der er også andre muligheder. En god ide kunne være, i første omgang, at forsøge at løse det problem, som P. H. Winston benytter i sin lærebog om kunstig intelligens [2], nemlig oplæring af en bager til at optimere mængden af sukker og mel i sine kager.

[1] J. H. Holland, K. J. Holyoak, R. E. Niebett and P. R. Thagard,
Induction: Processes of Inference, Learning and Discovery,
MIT Press, 1986.

[2] P. H. Winston,
Artificial Intelligence,
Addison-Wesley, 1992 (3rd ed.).

4. Kombinatorisk optimering

Området *kombinatorisk optimering* omhandler løsning af problemer, hvor der blandt et endeligt, men ofte meget stort, antal muligheder skal bestemme en optimal løsning. Antallet af muligheder kan i visse tilfælde antage astronomiske størrelser, f.eks. 10^{10000}, hvilket nødvendiggør, at særdeles effektive søgemetoder må tages i anvendelse.

Blandt søgemetoderne skelnes der mellem *eksakte* metoder og *approksimative* metoder. Med de eksakte metoder bestemmes det eksakte optimum for et givet problem. De approksimative metoder giver derimod kun en tilnærmelse til optimum, men med et relativt lille tidsforbrug.

En eksempel på en approksimativ metode er *simuleret nedkøling* (engelsk: simulated annealing) [1]. Metoden, der er baseret på en fysisk analogi, langsom nedkøling, har vist sig effektiv i forbindelse med løsning af mange kombinatoriske optimeringsproblemer. Dens styrke ligger i dens evne til at undslippe lokale optima.

Et andet eksempel er *tabusøgning* [2][3]. Denne metode, der er forholdsvis ny, har i flere tilfælde vist sig at være mere effektiv end simuleret udglødning.

Jeg kunne forestille mig et projekt, hvor en af de to metoder (eller dem begge), studeres og afprøves på et simpelt eksempel. Som eksempel kunne f.eks. vælges farvelægningsproblemet for grafer [4].

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi,
Optimization by simulated annealing,
Science, Vol. 220, 1983 (4458).
http://citeseer.ist.psu.edu/kirkpatrick83optimization.html

[2] F. Glover,
Future paths for integer programming and links to artificial intelligence,
Computers Oprns. Res., Vol. 13, 1986 (pp. 533-549).

[3] F. Glover,
Tabu Search - Part I,
ORSA J. Comput., Vol. 1, 1989 (pp. 190-206).
http://leeds-faculty.colorado.edu/glover/TS%20-%20Part%20I-ORSA.pdf

[4] A. Hertz and D. de Werra,
Using Tabu Search Techniques for Graph Coloring,
Computing, Vol. 39, 1987 (pp. 344-351).
http://www.springerlink.com/content/y766j232466l1674/

5. Myrekolonier til kombinatorisk optimering

Jo, du læste faktisk rigtigt. Det er muligt at løse visse problemer ved at efterligne dyrs evner til at løse problemer. I dette tilfælde efterlignes myrers instinktive evne til at finde vej i et ukendt terræn. Metoden er inspireret af forskning i myrers kollektive adfærd. Etologerne har forsøgt at forstå, hvordan de næsten blinde myrer kan finde den korteste vej fra myretuen hen til et sted med føde. En hypotese er, at myrerne 'kommunikerer' ved hjælp af deres ekskrementer. En myre, der bevæger sig, lægger undervejs sine ekskrementer, og når en anden myre så senere kommer til et sted på denne vej, vælger den med høj sandsynlighed at følge den forrige myres spor. Derved forstærkes 'lysten' hos andre myrer til at følge sporet.

Der er med andre ord tale om kollektiv indlæring af en hensigtsmæssig adfærd. Datalogisk udtrykt, er der tale om "distribuerede beregninger foretaget af samarbejdende, men ikke centralt kontrollerede, processorer". Problemløsningsmetoden er naturligvis særlig relevant i forbindelse med løsning af problemer på paralleldatamater, men også på traditionelle datamater har metoden vist sin berettigelse.

I dette projekt implementeres og afprøves en algoritme, inspireret af myrekoloniers adfærd, til løsning af et klassisk kombinatorisk optimeringsproblem, nemlig "Den rejsende sælgers problem". Problemet går i korthed ud på at bestemme den korteste rejserute for en person, der skal besøge en række byer.

Som udgangspunkt for projektet benyttes nedenstående artikler, specielt den første af referencerne.

[1] M. Dorigo and L. M. Gambardella,
ANT-Q. A Cooperative Learning Approach to Combinatorial Optimization,
Technical Report 95-01. IRIDIA, Université Libre de Bruxelles, 1995.

[2] A.. Colorni, M. Dorigoand V. Maniezzo,
Distributed Optimization by Ant Colonies,
Proc. ECAL91, Paris 1991, pp. 134-142.
http://www.cs.ualberta.ca/~bulitko/F02/papers/IC.06-ECAL92.pdf

[3] A.. Colorni, M. Dorigo and V. Maniezzo,
An Investigation of some properties of an Ant algorithm,
Proc. PPSN,, Bussels, 1992 (pp. 509-520).
ftp://iridia.ulb.ac.be/pub/mdorigo/conferences/IC.08-PPSN92.ps.gz

[4] M. Dorigo, V. Maniezzo and A.. Colorni,
The Ant System: Optimization by a colony of cooperating agents,
IEEE Trans. Sys., Man and Cyb.-part B. Vol. 26, No. 1, 1996 (pp. 1-13).
ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.10-SMC96.ps.gz

6. Parallelle algoritmer

Overskriften skal blot ses som et tema. To mulige projekter kunne være at undersøge (1) parallelle algoritmer til sortering og (2) parallelle algoritmer til løsning af grafproblemer.

[1] H. T. Kung,
The Structure of Parallel Algorithms,
Advances in Computers (ed. M. C Yovits), Vol. 19, 1980 (pp. 65-112).

[2] D. J. Kuck.,
A Survey of parallel machine organization and programming.
Computing Surveys, Vol. 9, No. 1, 1977 (pp. 29-59).
http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=356686

[3] S. Lakshmivarahan, S. K. Dhall and L. L. Miller,
Parallel Sorting Algorithms,
Advances in Computers (ed. M. C. Yovits), Vol. 23, 1984 (pp. 295-394).

[4] K. M. Chandy and J. Mishra,
Distributed Computations on Graphs: Shortest Path Algorithms,
Comm. ACM, Vol. 25, No. 11, 1982 (pp. 833-837).

[5] F. Y. Chin, J. Lam and I. Chen,
Efficient Parallel Algorithms for Some Graph Problems,
Comm. ACM, Vol. 25, No. 9, 1982 (pp. 659-665).
http://portal.acm.org/citation.cfm?id=358650&coll=portal&dl=ACM

[6] J. Mishra and K. M. Chandy,
A Distributed Graph Algorithm: Knot Detection,
ACM. Trans. Prog. Lang. and Syst., October 1982.
http://www.cs.utexas.edu/users/misra/scannedPdf.dir/KnotDetection.pd

[7] S. G. Akl,
Parallel Sorting Algorithms,
Academic Press, 1985.

7. NESL - et sprog til parallelprogrammering

NESL er et sprog til parallelprogrammering, dvs. udvikling af programmer, som tillader samtidig udførelse af operationer på data. Sproget er forholdsvist simpelt og er velegnet både til undervisning i parallelprogrammering og til udvikling af parallelle algoritmer.

En grundlæggende egenskab ved NESL er, at sproget tillader samtidige operationer på sekvenser - endimensionale tabeller. For at give et indtryk af nogle af sprogets muligheder er nedenfor vist en implementering i NESL af Quicksort, en velkendt algoritme til sortering af tabeller.

```
function Quicksort(S) =
if (#S <= 1) then S
else
  let a = S[rand(#S)];
    S1 = {e in S | e < a};
    S2 = {e in S | e == a};
    S3 = {e in S | e > a};
    R = {Quicksort(v): v in [S1,S3]};
  in R[0] ++ S2 ++ R[1] $
```

Operatoren `#` returnerer længden af en sekvens. Funktionen `rand(n)` returnerer et tilfældigt heltal mellem `0` og `n`. Udtrykket `S[rand(#S)]` returnerer således et tilfældigt element i `S`. Notationen `{e in S | e < a}` betyder: "find i parallel alle elementer `e` i `S`, for hvilke `e < a`". Notationen `{Quicksort(v): v in [S1,S3]}` betyder "udfør `Quicksort(v)` i parallel for alle `v` i `S1` og `S3`". Operatoren `++` sætter to sekvenser i forlængelse af hinanden.

Oversætteren til NESL oversætter til et mellemsprog, der kaldes VCODE. Den resulterende mellemkode kan så fortolkes, eller oversættes til en paralleldatamat, f.eks. til Connection Machines CM-2/CM-5 eller Cray C90.

Der er mange muligheder for projekter, der omhandler NESL. Blandt disse kan nævnes:

(1) Udvikling af en NESL-oversætter.
(2) Udvikling af en VCODE-fortolker.
(3) Evaluering af NESL.

[1] G. E. Blelloch,
Programming Parallel Algorithms,
C. ACM, Vol. 39, No. 3, 1996 (pp. 85-97).
http://citeseer.ist.psu.edu/36476.html
http://portal.acm.org/citation.cfm?id=227246&coll=portal&dl=ACM

8. Intervalaritmetik

Det er velkendt, at data kan være forbundet med usikkerhed. Tænk for eksempel på data, som aflæses på måleinstrumenter. Alligevel foretages der ofte beregninger på sådanne data, som om der var tale om eksakte data. Ofte undlades egentlige følsomhedsanalyser, hvor den beregningsmæssige betydning af dataenes usikkerhed analyseres. Man stoler på, at de beregnede resultater kan bruges, på trods af inddataene er fejlbehæftede.

Dette projekt går ud på at stille et værktøj til rådighed, som kan benyttes i forbindelse med følsomhedsanalyser, nemlig en Java-klasse, der kan regne med intervaller. Programmellet skal kunne foretage aritmetiske beregninger på tal, hvis fejlen kan angives i form af intervaller.

Hvis værdien for x vides at ligge i intervallet [xmin,xmax], så skal alle beregninger, hvor x indgår benytte dette interval. Antag f.eks. at x tilhører intervallet [xmin,xmax] og y tilhører intervallet [ymin,ymax], så kan det konkluderes, at deres sum x+y tilhører intervallet [xmin+xmax,ymin+ymax]. Tilsvarende intervalregneregler kan angives for andre aritmetiske operationer.

For at tilbyde en bekvem brugergrænseflade anbefales et objektorienteret design. Et tals interval er et objekt med tilhørende aritmetiske operationer. Således kan addition af x og y (hvor x og y er talintervaller) f.eks. udtrykkes ved x.Add(y).

Alternativt kunne projektet rette sig mod C^{++} i stedet for Java.

9. Problemløsning med båndlagte variable

I nedennævnte artikel af Sussmann og Steele [1] beskrives et sprog, CONSTRAINTS, til løsning af problemer, der involverer variable, som er båndlagt ved ligninger og uligheder. Som problemdomæne benyttes elektriske kredsløb, idet variablerne bl.a. er strømstyrke, spænding og modstand, og bindingerne er udtryk for elektriske love, f.eks. Ohm´s lov. Projektet går ud på at implementere et tilsvarende sprog, eller dele heraf, i form af en Java-pakke.

[1] G. J. Sussmann and G. L. Steele,
CONSTRAINTS - A Language for Expressing Almost Hierarchical Descriptions,
Artificial Intelligence, Vol. 14, No. 1, 1980 (pp. 1-39).

10. Et sprog til håndtering af grafstrukturer

Inspirationen til dette projekt stammer fra artiklen [1], hvori beskrives et sprog, GRAPHIX, til håndtering af grafstrukturer. Projektet går ud på at programmere dette sprogs faciliteter samt foretage en vurdering heraf på baggrund af en række anvendelseseksempler.

[1] G. Sutcliffe,
GRAPHIX - a Graph Theory Sub-Language,
Int. J. Computer Math., Vol. 17, 1985 (pp. 275-296).

11. Løsning af ligningssystemer

Løsning af ligningssystemer er et hyppigt forekommende problem i forbindelse med naturvidenskabelige problemstillinger.

Ofte er der tale om *lineære* ligningssystemer (systemer på formen Ax = b, hvor A er en kvadratisk matrix, og x og b er vektorer). I sådanne tilfælde kan en løsning bestemmes ved brug af et tilgængeligt numerisk programbibliotek. Det kræver sædvanligvis en smule programmeringskendskab, men er forholdsvist simpelt.

Hvis ligningssystemerne derimod *ikke* er lineære (hvis der f.eks. indgår produkter af de ubekendte), kan det være vanskeligt at finde programmel, som kan benyttes til løsning. I sådanne tilfælde kan det derfor blive nødvendigt selv at udvikle en algoritme til formålet.

I artiklen [1] er beskrevet en simpel algoritme til løsning af såvel lineære ligningssystemer som visse typer af ikke-lineære ligningssystemer. Dens styrke i forhold til andre tilsvarende algoritmer hævdes i artiklen at være dens simpelhed og hastighed.

Målet med projektet er at implementere og vurdere den angivne algoritme. Er de i artiklen fremførte påstande korrekte? Kan algoritmen forbedres?

[1] E. Derman and C. J. Van Wyk,
A Simple Equation Solver and its Application to Financial Modelling,
Software - Practice and Experience, Vol. 14, No. 12, 1984 (pp. 1169-1184).

12. Fraktionelle kaskader

Fraktionelle kaskader en navnet på en datastruktur, der med fordel kan anvendes i forbindelse med databaser, der indeholder geometrisk information (f.eks. geografiske databaser). I projektet undersøges og afprøves denne datastruktur nærmere. Projektarbejdet kan muligvis resultere i udformning af kernen til et databasesystem, der baserer sig på datastrukturen.

[1] B. Chazell and J. L. Guibas,
Fractional Cascading: I. A Data Structuring Technique,
Algorithmica, Vol. 1, No. 2, 1986 (pp. 133-162).
http://www.springerlink.com/content/w00156688761426v/

[2] B. Chazell and J. L. Guibas,
Fractional Cascading: II. Applications,
Algorithmica, Vol. 1, No. 2, 1986 (pp. 163-191).
http://www.springerlink.com/content/j5028h162331u315/

[3] K. Mehlhorn and S. Näher,
Dynamic Fractional Cascading,
Algorithmica, Vol. 5, No. 2, 1990 (pp. 215-241).
http://www.springerlink.com/content/l1691644h32352r0/