Classified Dynamic Programming

Robert Giegerich

Bielefeld University

Bled, Feb. 2009

Motivation

Our topic: Programming methodology

- A trade-off in dynamic programming between search space design and evaluation of candidates
- A trade-off between modifying your code and adding to it
- A simple technique with a broad scope
- More fun in the life of a dynamic programmer ...

Overview

- An informal formalism
- Example problems and first solutions
- Classified Dynamic Programming – definition and central theorem
- Alternative solutions using cDP
- Conclusion

Framework of discussion

DP solves combinatorial optimization problems via

- DP recurrences
- scoring schemes
- Bellmans Principle of Optimality (BPO)

Framework of discussion

DP solves combinatorial optimization problems via

- DP recurrences (search space definition)
- (no explicit representation of candidates)
- scoring schemes (evaluation of candidates)
- Bellmans Principle of Optimality (BPO)

Abstract view on DP

$\mathcal{G}$: Generator of candidate space

$\mathcal{E}(h, c)$: Evaluation of candidate (set) c and objective function h

BPO is a property of $\mathcal{E}$ and h (!)

Abstract view on DP

$\mathcal{G}$: Generator of candidate space

$\mathcal{E}(h, c)$: Evaluation of candidate (set) c and objective function h

BPO is a property of $\mathcal{E}$ and h (!)

Example:

$$\begin{array}{lcl} \mathcal{G} & = & \textit{RNAfold}\text{ recurrences} \\ \mathcal{E} & = & \text{Turner energy rules} \\ h & = & \text{free energy minimization} \end{array}$$

Conceptual view on computation

View computation as a succession of 3 phases:

$$\mathcal{G}(\mathcal{E}(h), x) \xrightarrow{\mathcal{G}} \left\{ \begin{array}{ccc} c_1 & & c_2 \\ & c_3 & \\ c_4 & & c_5 \end{array} \right\} \xrightarrow{\mathcal{E}} \left\{ \begin{array}{ccc} v_1 & & v_2 \\ & v_3 & \\ v_4 & & v_5 \end{array} \right\} \xrightarrow{h} \left\{ \begin{array}{cc} v_3 & \\ & v_5 \end{array} \right\}$$

candidates — candidate "scores" — optimal result

Exponential explosion is avoided via phase amalgamation.

Trade-offs between $\mathcal{G}$ and $\mathcal{E}$:

(1) Common practice:

$$c \in \mathcal{G}(x) \text{ illegal } \rightarrow \mathcal{E}(\min)(c) = +\infty$$

-- a short-sighted trick

(2) Honorable consideration:

No illegal candidates in $\mathcal{G}(x)$, but an additional property P of interest – this creates the trade-off

$$\mathcal{G} \rightarrow \mathcal{G}^P \quad \text{versus} \quad \mathcal{E} \rightarrow \mathcal{E} + P$$

Problem 1: pknotsRG $\rightarrow$ pknotsRG-enf

[Reeder & Giegerich 2004]

Pseudoknot folding in pknotsRG

$$\begin{aligned}
\mathcal{G} &= \mathcal{G}_{RNAsubopt} + \\
&\quad \{\texttt{struct} \rightarrow PK(a, U, b, V, \bar{a}, W, \bar{b}), \\
&\quad \texttt{maxhel} \rightarrow SR(a, \texttt{maxhel}, b)\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{E} &= \mathcal{E}_{Turner} + \\
&\quad \{\mathcal{E}(PK(a, U, b, V, \bar{a}, W, \bar{b}) = \\
&\quad \mathcal{E}(U) + \mathcal{E}(V) + \mathcal{E}(W) + \\
&\quad \mathcal{E}(\texttt{maxhel}(a, \bar{a}) + \mathcal{E}(\texttt{maxhel}(b, \bar{b}) \\
&\quad +\text{correction terms}
\end{aligned}$$

User satisfaction ...

Standard folding including PKs

$$\mathcal{G}\,(\mathcal{E}(\min), x) \quad \text{finds} \quad S_{MFE}(x)$$

Most often, $S_{MFE}(x)$ does not hold a pseudoknot

$\Rightarrow$ Is there a knotted structure "nearby"?

pknotsRG $\longrightarrow$ pknotsRG-enf

(1) Duplicate Rules in $\mathcal{G}$:

$$X \rightarrow F(a\ B\ C\ d)$$

becomes

$X \rightarrow F(a\ B\ C\ d)$	X holds no PK
$X' \rightarrow F(a\ B'\ C'\ d)$	X' holds PK somewhere

(2) Make connections

$$\texttt{struct'} \rightarrow PK(a, U, b, V, \bar{a}, W, \bar{b})$$

Roughly, size of program doubles.

Application:

$\mathcal{G}_{pknotsRG-enf}\,(\mathcal{E}_{pknotsRG}(\min), x)$ finds best knotted structure.

Note: $S_{MFE}(x)$ still interesting as reference.

Problem 2: RNAbor

[Freyhult, Moulton, Clote 2007]

Evaluate structure space relative to a target structure t

for $d = 0, 1, \ldots, k$ find

$$\mathtt{argmin}_c \left\{ \mathcal{E}_{Turner}(c) | \mathtt{bpdist}(c, t) = d \right\}$$

$\mathcal{G} = \mathcal{G}_{Mfold}^{[0..k]}$ where

$$X \rightarrow F(a \ B \ C \ d)$$

becomes

$$X^i \rightarrow F(a \ B^l \ C^r \ d) \text{ such that } i = l + r + \delta_{ab}$$

$\mathcal{E} = \mathcal{E}_{Turner}$

Example of Recurrence

$$\mathrm{MFE}_{ij}^{\delta} = \min \Bigg\{ \mathrm{MFE}_{ij-1}^{\delta-b_0},$$

$$\min_{\substack{s_k s_j \in \mathbb{B}, \\ i \leq k < j}} \min_{w+w'=\delta-d_1} \mathrm{MFE}_{i,k-1}^{w} + \mathrm{MFEB}_{kj}^{w'} + E_d \Bigg\}$$

Developed in two steps,

- implemented first for base pair maximization $\mathcal{E}_{BPmax}$
- then for energy minimization $\mathcal{E}_{Turner}$

Problem 3: KNOT $\rightarrow$ NEST (K2N)

[Smit, Rother, Heringa, Knight 2008]

Given pseudo-knotted structure $K(x)$,
find "best" un-knotted structure $U(x)$ compatible with $K(x)$

Different techniques

- 5 heuristics implemented
- 1 optimization approach using
  $\mathcal{G} = \mathcal{G}_{Nussinov}$ (allows for all optimal solutions)
  $\mathcal{E} = \mathcal{E}_{BPmax}$

Why not use

$$\mathcal{G}^{K2N}_{RNAsubopt}\,(\mathcal{E}_{Turner}, x)$$

or

$$\mathcal{G}_{RNAsubopt}\left(\mathcal{E}^{K2N}_{Turner}, x\right)$$

What is the cost of modifying $\mathcal{G}_{RNAsubopt}$ or $\mathcal{E}_{Turner}$?

Problem 4: RNAshapes

[Giegerich, Voss, Rehmsmeier 2004/2006]

Classify candidates by their abstract shapes, e.g.
`((((...))..((...)..))) -> [[][]]`

Compute accumulated Boltzmann-Probabilities over all shapes

$$\begin{aligned} \mathcal{G} &= \mathcal{G}_{Voss} \\ \mathcal{E} &= \mathcal{E}_{shape} + \mathcal{E}_{Boltzmann} \end{aligned}$$

Summary

Features of Interest:

pknotsRG-enf	PK or not?	$\mathcal{O}(1)$
RNAbor	$\text{bpdist}(c, t)$	$\mathcal{O}(n)$
K2N	$\text{BP}(c) \subseteq \text{BP}(k)$	$\mathcal{O}(1)$
RNAshapes	$\mathcal{E}_{shape}(c)$	$\mathcal{O}(\alpha^n)$

Summary

Features of Interest:

pknotsRG-enf	PK or not?	$\mathcal{O}(1)$
RNAbor	$\text{bpdist}(c, t)$	$\mathcal{O}(n)$
K2N	$\text{BP}(c) \subseteq \text{BP}(k)$	$\mathcal{O}(1)$
RNAshapes	$\mathcal{E}_{shape}(c)$	$\mathcal{O}(\alpha^n)$

Note: The feature can be calculated by candidate evaluation.

Classified Dynamic Programming (cDP)

Given $\mathcal{G}, \mathcal{E}(h)$.

Let $\mathcal{A}(\_, c)$ be an evaluator that computes the classification attribute from c.

Build classified evaluator $\mathcal{E}_{\mathcal{A}}(h')$:

$$\begin{aligned} \text{where } \mathcal{E}_{\mathcal{A}}(c) = & \ (\mathcal{A}(c), \mathcal{E}(c)) \\ h'\{(a_i, e_i)\} = & \ \{(a, e) | a \in \{a_i\}, e \in h\{(a, e_i)\}\} \end{aligned}$$

$\mathcal{E}_{\mathcal{A}}(h')$ computes $\mathcal{E}(h)$ class-wise by $\mathcal{A}$

Shorthand notation:$\mathcal{E}_{\mathcal{A}}(h') := \mathcal{A}(id) * \mathcal{E}(h)$

Central Theorem

If $\mathcal{E}(h)$ satisfies BPO, then so does $\mathcal{E}_{\mathcal{A}}(h')$.

Consequence

1. If you can describe the feature of interest by an adequate $\mathcal{A}$, then cDP always does the job.
2. If you implement $\mathcal{A}(id) * \mathcal{E}(h)$ generically, a simple classifier $\mathcal{A}$ is all you must write.
3. You add to your program, rather than changing it.

Classified solutions for Problem 1 – 4

pknotsRG-enf	$\mathcal{A}(c)$	$=$	true, if c holds a pseudoknot false, otherwise
RNAbor	$\mathcal{A}(c)$	$=$	bpdist (c, t)
K2N	$\mathcal{A}(c)$	$=$	true, if $\text{BP}(c) \subseteq \text{BP}(k)$ false, otherwise
RNAshapes	$\mathcal{A}(c)$	$=$	$\mathcal{E}_{shape}(c)$

Think it through, class-wise ...

Problem 5: Gen structure prediction

[Brejova, Brown, Vinar 2007]

$\mathcal{G}$ = Hidden Markov Model $\mathcal{M}$

$\mathcal{E}(max)(c) = Prob(\mathcal{G}^{\mathcal{M}} \Rightarrow c)$

(maximum likelihood path through $\mathcal{M}$ for x)

Path labelling: $\{c\} \rightarrow$ gene structure for x

Gene structures and path labellings

Simple HMM for gene structure: Path cycling through 3 states, with loops in each state, for example:

$$\textit{intergenic} \longrightarrow \textit{exon} \longrightarrow \textit{intron} \longrightarrow \textit{exon} \longrightarrow \textit{intergenic}$$

A gene structure is a labelling of bases with states:

```
aaattagttaaccacgtccccagttagaggatatcccccc
-----eeeeeeeeeiiiiiiieeeeeeeeeee-------
```

Viterbi algorithm returns the most likely path = most likely labelling = most likely gene structure

Refined gene structure models

Let us split up intron state in 3 states; early (1), middle (2), near-end (3) intron to better model intron length distribution and significance of splice sites.

```
aaattagttaaccacgtccccagttagaggatatccccccc
-----eeeeeeeeeiiiiiiieeeeeeeeeee-------
```

now splits up into

```
aaattagttaaccacgtccccagttagaggatatccccccc
-----eeeeeeeee1222223eeeeeeeeeee-------
-----eeeeeeeee1122223eeeeeeeeeee-------
-----eeeeeeeee1123333eeeeeeeeeee-------
```

with different individual probabilities

When Viterbi goes wrong ...

The most likely path is not the most likely gene structure!
Different paths that correspond to the same original labelling should be summed over.
This problem (named "HMM path labelling problem") has been shown to be NP-complete.
(See Brejova, Brown, Vinar in Journal of Computer and System Sciences Volume 73, Issue 7, November 2007, Pages 1060-1077)

When Viterbi goes wrong ...

The most likely path is not the most likely gene structure!
Different paths that correspond to the same original labelling should be summed over.
This problem (named "HMM path labelling problem") has been shown to be NP-complete.
(See Brejova, Brown, Vinar in Journal of Computer and System Sciences Volume 73, Issue 7, November 2007, Pages 1060-1077)

Note: Viterbi returns garbage, while the FORWARD algorithm correctly returns the joint probability of *all* gene structures for the given sequence.

- Several optimal paths map to the same gene structure
- Viterbi algorithm goes wrong
- this is also known as "semantic ambiguity"

Ambiguity compensation:
Evaluator $\mathcal{E}(max)(c)$ = maximum (path) probability
Evaluator $\mathcal{L}(c)$ = "path labelling" of c

$\mathcal{G}_{HMM}(\mathcal{L} * \mathcal{E}(+), x)$ computes most likely gene structure.

Conclusion

Write your DP code in perfect separation

$$\mathcal{G} + \mathcal{E}$$

Provide generic implementation of

$$\mathcal{E}_1 * \mathcal{E}_2$$

Have more fun in your lifetime ...
...because extensions never change code, they only add to it.

…and even improve your sleep:

Let $\mathcal{N}(c) =$ a canonical representation for c
Let $\mathcal{C}(c) = 1$
Then $\mathcal{G}(\mathcal{N} * \mathcal{C}(+), x)$ allows for ambiguity checking.

Versatile family modelling

Tools such as

- *HMMer*
- *Infernal*

generate stochastic family models, implemented via DP.
Why not make provision for a user specified, add-on scoring scheme?

Apologies

My apologies go to

Robert Giegerich, Jens Reeder
Eva Freyhult, Vince Moulton, Peter Clote
Sandra Smit, Jap Heringa, Rob Knight