7. Factor graphs and contraction

CS-E4500 Advanced Course on Algorithms
Spring 2020

Petteri Kaski
Department of Computer Science
Aalto University

Problem Set 7 not yet online

[Caveat: Problem Set 7 is not online yet—but will be there either later today or tomorrow]

Lecture schedule

Tue 7 Jan:	1. Polynomials and integers
Tue 14 Jan:	2. The fast Fourier transform and fast multiplication
Tue 21 Jan:	3. Quotient and remainder
Tue 28 Jan:	4. Batch evaluation and interpolation
Tue 4 Feb:	5. Extended Euclidean algorithm
Tue 11 Feb:	*Break — no lecture*
Tue 18 Feb:	*Exam week — no lecture*
Tue 26 Feb:	6. Interpolation from erroneous data
Tue 3 Mar:	7. Factor graphs and contraction
Tue 10 Mar:	8. Verifiable and error-tolerant inference for factor graphs
Tue 17 Mar:	9. Further applications of factor graphs

CS-E4500 Advanced Course in Algorithms (5 ECTS, III–IV, Spring 2020)

2020 KALENTERI 2020

Tammikuu

1	Ke	Uudenvuodenpäivä
2	To	
3	Pe	
4	La	
5	Su	
6	Ma	Loppiainen
7	Ti	L1
8	Ke	
9	To	Q1
10	Pe	
11	La	
12	Su	D1
13	Ma	Vk 03 T1
14	Ti	L2
15	Ke	
16	To	Q2
17	Pe	
18	La	
19	Su	D2
20	Ma	Vk 04 T2
21	Ti	L3
22	Ke	
23	To	Q3
24	Pe	
25	La	
26	Su	D3
27	Ma	Vk 05 T3
28	Ti	L4
29	Ke	
30	To	Q4
31	Pe	

Helmikuu

1	La	
2	Su	D4
3	Ma	Vk 06 T4
4	Ti	L5
5	Ke	
6	To	Q5
7	Pe	
8	La	
9	Su	
10	Ma	Vk 07
11	Ti	Break
12	Ke	week
13	To	
14	Pe	
15	La	
16	Su	
17	Ma	Vk 08
18	Ti	Exam
19	Ke	week
20	To	
21	Pe	
22	La	
23	Su	D5
24	Ma	Vk 09 T5
25	Ti	Laskiainen
26	Ke	
27	To	Q6
28	Pe	
29	La	

Maaliskuu

1	Su	D6
2	Ma	Vk 10 T6
3	Ti	L7
4	Ke	
5	To	Q7
6	Pe	
7	La	
8	Su	D7
9	Ma	Vk 11 T7
10	Ti	L8
11	Ke	
12	To	Q8
13	Pe	
14	La	
15	Su	D8
16	Ma	Vk 12 T8
17	Ti	L9
18	Ke	
19	To	Q9
20	Pe	Kevätpäiväntasaus
21	La	
22	Su	D9
23	Ma	Vk 13 T9
24	Ti	
25	Ke	
26	To	
27	Pe	
28	La	
29	Su	Kesäaika alkaa
30	Ma	Vk 14
31	Ti	

Huhtikuu

1	Ke	
2	To	
3	Pe	
4	La	
5	Su	Palmusunnuntai
6	Ma	Vk 15
7	Ti	
8	Ke	
9	To	
10	Pe	Pitkäperjantai
11	La	
12	Su	Pääsiäispäivä
13	Ma	2. pääsiäispäivä
14	Ti	
15	Ke	
16	To	
17	Pe	
18	La	
19	Su	
20	Ma	Vk 17
21	Ti	
22	Ke	
23	To	
24	Pe	
25	La	
26	Su	
27	Ma	Vk 18
28	Ti	
29	Ke	
30	To	

Toukokuu

1	Pe	Vappu
2	La	
3	Su	
4	Ma	Vk 19
5	Ti	
6	Ke	
7	To	
8	Pe	
9	La	
10	Su	Äitienpäivä
11	Ma	Vk 20
12	Ti	
13	Ke	
14	To	
15	Pe	
16	La	
17	Su	Kaatuneiden muistopäivä
18	Ma	Vk 21
19	Ti	
20	Ke	
21	To	Helatorstai
22	Pe	
23	La	
24	Su	
25	Ma	Vk 22
26	Ti	
27	Ke	
28	To	
29	Pe	
30	La	
31	Su	Helluntaipäivä

Kesäkuu

1	Ma	Vk 23
2	Ti	
3	Ke	
4	To	
5	Pe	
6	La	
7	Su	
8	Ma	Vk 24
9	Ti	
10	Ke	
11	To	
12	Pe	
13	La	
14	Su	
15	Ma	Vk 25
16	Ti	
17	Ke	
18	To	
19	Pe	
20	La	Juhannus
21	Su	Kesäpäivänseisaus
22	Ma	Vk 26
23	Ti	
24	Ke	
25	To	
26	Pe	
27	La	
28	Su	
29	Ma	Vk 27
30	Ti	

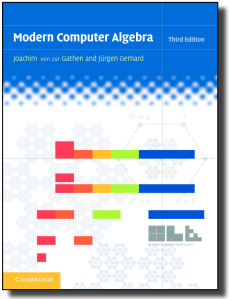
L = Lecture; hall T5, Tue 12–14
Q = Q & A session; hall T5, Thu 12–14
D = Problem set deadline; Sun 20:00
T = Tutorial (model solutions); hall T6, Mon 16–18

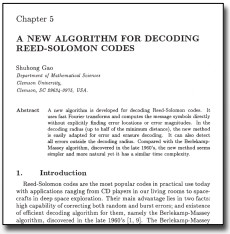
Recap of last week

- Coping with **errors in data** using **error-correcting codes**
- A family of error-correcting codes (**Reed–Solomon codes**) based on evaluation–interpolation duality for univariate polynomials
  - Key observation: low-degree polynomials have few roots (exercise)
  - Fast **encoding** and **decoding** of Reed–Solomon codes via the fast univariate polynomial toolkit and **Gao's** (2003) **decoder**

Have: Near-linear-time toolbox for univariate polynomials

- Multiplication
- Division (quotient and remainder)
- Batch evaluation
- Interpolation
- Extended Euclidean algorithm (gcd)
- Interpolation from partly erroneous data





Chapter 5

A NEW ALGORITHM FOR DECODING REED-SOLOMON CODES

Shuhong Gao
Department of Mathematical Sciences
Clemson University,
Clemson, SC 29634-1907, USA.

Abstract: A new algorithm is developed for decoding Reed-Solomon codes. It uses fast Fourier transforms and computes the message symbols directly without explicitly finding error locations or error magnitudes. In the decoding radius (up to half of the minimum distance), the new method is easily adapted for error and erasure decoding. It can also detect all errors outside the decoding radius. Compared with the Berlekamp-Massey algorithm, discovered in the late 1960's, the new method seems simpler and more natural yet it has a similar time complexity.

1. Introduction

Reed-Solomon codes are the most popular codes in practical use today with applications ranging from CD players in our living rooms to spacecrafts in deep space exploration. Their main advantage lies in two facts: high capability of correcting both random and burst errors; and existence of efficient decoding algorithm for them, namely the Berlekamp-Massey algorithm, discovered in the late 1960's [1, 9]. The Berlekamp-Massey

Motivation for this week

- Factor graphs are a formalism for working with multilinear sum–product expressions
- Such expressions have many applications, e.g.
  - Multilinear algorithm designs (e.g. [1])
  - Probabilistic inference with graphical models (e.g. [12])
  - Verifiable and error-correcting inference (next week; cf. [9])
  - Computational physics applications (e.g. [15])
  - ...

Key content for Lecture 7

- **Factor graphs**
- Factor graphs **represent** multilinear sum–product expressions
- **Inference problem** for factor graphs
- Examples: multilinear maps, conditional probability and independence
- Inference by **contraction** of factors
- **Order-invariance** for contraction
- **Cost** of inference by contraction
- **Minimizing** the cost inference by contraction

Factor graphs (1/2)

- Let $X_1, X_2, \ldots, X_n$ be **variables**
- Let $f_1, f_2, \ldots, f_m$ be **factors**
- Associated with each variable X_i, there is a finite nonempty set $\mathcal{D}_i$, the **domain** of X_i
- Each factor f_k is **incident** to a subset $S_k \subseteq U = \{1, 2, \ldots, n\}$ of the variables
- For a subset $S \subseteq U$, let us write $\prod_{i \in S} \mathcal{D}_i$, or simply $\mathcal{D}_S$, for the Cartesian product of the sets $\mathcal{D}_i$ with i ranging over S in the natural order of U
- The Cartesian product over the empty set is assumed to be the set of consisting of the empty set
- Let $B \subseteq U$ that indicates the **boundary** or **query** variables

Factor graphs (2/2)

- Let R be a ring (commutative and nontrivial)
- Associate with each factor f_k a map $\mathcal{D}_{S_k} \to R$
- For a point $v \in \mathcal{D}_{S_k}$, let us write $f_k(X_i = v_i : i \in S_k)$, or simply $f_k(v)$, for the value of the map at v
- The variables and their domains, the incidence sets, the boundary, and the factors and their associated maps together constitute a **factor graph** G
- To avoid degenerate cases, we assume each variable is incident to at least one factor, each variable not in the boundary is incident to at least two factors, and $|\mathcal{D}_i| \geq 2$ for all $i \in U$
- For a factor graph G, let us write $\|G\|$ for the **total size** of G, given by $\sum_{k=1}^{m} |\mathcal{D}_{S_k}|$

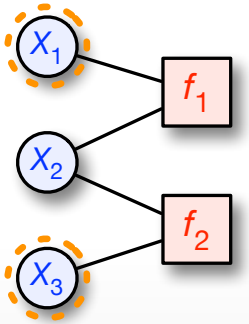
Example: A factor graph (over $\mathbb{Q}$)

$B = \{1, 3\}$

$D_1 = \{a, b\}$

$D_2 = \{c, d\}$

$D_3 = \{e, f\}$



$S_1 = \{1, 2\}$

$S_2 = \{2, 3\}$

f_1	$X_2 = c$	$X_2 = d$
$X_1 = a$	0.90	0.10
$X_1 = b$	0.40	0.60

f_2	$X_3 = e$	$X_3 = f$
$X_2 = c$	0.30	0.70
$X_2 = d$	0.80	0.20

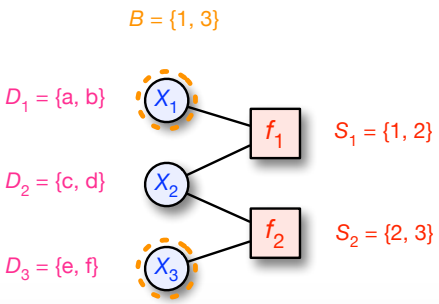
The inference problem for factor graphs

- Let G be a factor graph
- Associated with G is a map $f_G : \mathcal{D}_B \to R$, the map **represented** by G, defined for all $v \in \mathcal{D}_B$ by

$$f_G(v) = \sum_{w \in \mathcal{D}_{U \setminus B}} \prod_{k=1}^{m} f_k(v, w)\,,$$

  where for conciseness we have abbreviated $f_G(v) = f_G(X_i = v_i : i \in B)$ and $f_k(v, w) = f_k(X_i = v_i, X_j = w_j : i \in S_k \cap B, j \in S_k \setminus B)$
- In what follows we will tacitly use such abbreviations
- For a factor graph G given as input, the **inference problem** is to compute a complete table of values for the map f_G
- Inference is a hard computational problem, even if one allows for approximations [4]

Example: The map represented by a factor graph



f_1	$X_2 = c$	$X_2 = d$
$X_1 = a$	0.90	0.10
$X_1 = b$	0.40	0.60

f_2	$X_3 = e$	$X_3 = f$
$X_2 = c$	0.30	0.70
$X_2 = d$	0.80	0.20

f_G	$X_3 = e$	$X_3 = f$
$X_1 = a$	0.35	0.65
$X_1 = b$	0.60	0.40

Examples: Multilinear maps

[whiteboard]

Examples: Conditional probability and independence

[whiteboard]

Contraction of two factors (1/2)

- Let G be a factor graph and let $1 \leq a \neq b \leq m$
- The operation of **contracting** the factors f_a and f_b in G is as follows
- For each $i \in U$, let $T_i = \{1 \leq k \leq m : i \in S_k\}$
- We say that i is **internal** to the contraction if $i \notin B$ and $T_i \subseteq \{a, b\}$
- Let us write $I_{ab} \subseteq U$ for the set of all i that are internal to the contraction
- We have $I_{ab} \subseteq S_a \cup S_b$ by our assumption on nondegeneracy

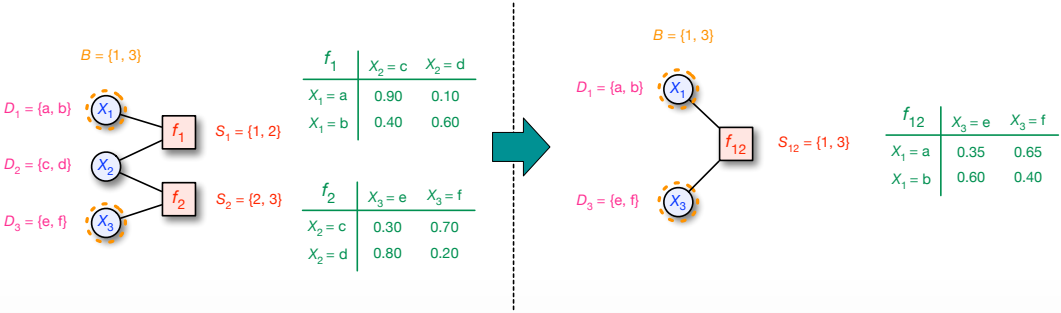
Contraction of two factors (2/2)

- Delete f_a and f_b from G and introduce the factor f_{ab} that is incident to $(S_a \cup S_b) \setminus I_{ab}$ and defined for all $v \in \mathcal{D}_{(S_a \cup S_b) \setminus I_{ab}}$ by

$$f_{ab}(v) = \sum_{w \in \mathcal{D}_{I_{ab}}} f_a(v, w) f_b(v, w)$$

- Finally, delete the variables X_i with $i \in I_{ab}$ from the factor graph to obtain G_{ab}, the factor graph obtained from G by **contracting** f_a and f_b
- The **cost** of the contraction is $|\mathcal{D}_{S_a \cup S_b}|$
- The factor graphs G and G_{ab} represent the same map; that is, we have $f_G = f_{G_{ab}}$ (exercise)
- That is, contraction is **order-invariant** in terms of the result—if one contracts pairs of factors repeatedly until only a single factor remains, the single remaining factor is always associated with the map f_G

Example: Contracting factors f_1 and f_2



Cost of inference

- The **cost** of contracting the factors f_a and f_b is $|\mathcal{D}_{S_a \cup S_b}|$; this is a proxy for the number of arithmetic operations in R needed to compute f_{ab} from f_a and f_b
- The **cost** of a sequence of contractions is the sum of the costs of each contraction
- While contraction is order-invariant *in terms of the result* (when only a single factor remains, the associated map is the map f_G), it is **not** order-invariant *in terms of the cost*
- The cost of a sequence of contractions so that a single factor remains is the **cost of inference** for this sequence
- A sequence of contractions resulting in a single remaining factor can be represented by a rooted binary tree with the factors $f_1, f_2, \ldots, f_m$ at the leaves—each internal node is the result of contracting its two child nodes

Example: Cost of inference

[whiteboard]

Minimizing the cost of inference

- Given a factor graph G as input, we would like to minimize the cost of a sequence of contractions that produces f_G
- This is a hard optimization problem
- In the exercises we will develop a dynamic programming algorithm that runs in time exponential in the number of factors m (exercise)
- Efficiently solvable special cases exist, such as when G has the topology of a path

Recap of Lecture 7

- **Factor graphs**
- Factor graphs **represent** multilinear sum–product expressions
- Examples: multilinear operators, conditional probability
- **Inference problem** for factor graphs
- Inference by **contraction** of factors
- **Order-invariance** for contraction
- **Cost** of inference by contraction
- **Minimizing** the cost of inference

Learning objectives (1/2)

- Terminology and objectives of modern algorithmics, including elements of algebraic, online, and randomised algorithms
- Ways of coping with uncertainty in computation, including error-correction and proofs of correctness (next week)
- The art of solving a large problem by reduction to one or more smaller instances of the same or a related problem
- (Linear) independence, dependence, and their abstractions as enablers of efficient algorithms

Learning objectives (2/2)

- Making use of duality
  - Often a problem has a corresponding **dual** problem that is obtainable from the original (the **primal**) problem by means of an easy transformation
  - The primal and dual control each other, enabling an algorithm designer to use the interplay between the two representations
- Relaxation and tradeoffs between objectives and resources as design tools
  - Instead of computing the exact optimum solution at considerable cost, often a less costly but principled approximation suffices
  - Instead of the complete dual, often only a randomly chosen partial dual or other relaxation suffices to arrive at a solution with high probability

References I

[1] P. Austrin, P. Kaski, and K. Kubjas, Tensor network complexity of multilinear maps, in *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA* (A. Blum, Ed.). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019, pp. 7:1–7:21. [doi:10.4230/LIPIcs.ITCS.2019.7].

[2] R. Brent and P. Zimmermann, *Modern Computer Arithmetic*, Cambrigde University Press, 2011. [WWW].

[3] D. A. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms*, fourth ed., Springer, Cham, 2015. [doi:10.1007/978-3-319-16721-3].

References II

[4] P. Dagum and M. Luby, Approximating probabilistic inference in Bayesian belief networks is NP-hard, *Artif. Intell.* 60 (1993), 141–153. [doi:10.1016/0004-3702(93)90036-B].

[5] M. Fürer, Faster integer multiplication, *SIAM J. Comput.* 39 (2009), 979–1005. [doi:10.1137/070711761].

[6] S. Gao, A new algorithm for decoding Reed–Solomon codes, in *Communications, Information, and Network Security* (V. K. Bhargava, H. V. Poor, V. Tarokh, and S. Yoon, Eds.), Springer, 2003, pp. 55–68.

[7] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, third ed., Cambridge University Press, Cambridge, 2013. [doi:10.1017/CBO9781139856065].

References III

[8] D. Harvey, J. van der Hoeven, and G. Lecerf, Even faster integer multiplication, *J. Complexity* 36 (2016), 1–30. [doi:10.1016/j.jco.2016.03.001].

[9] N. Karimi, P. Kaski, and M. Koivisto, Error-correcting and verifiable parallel inference in graphical models, in *Proceedings of AAAI'20*, to appear.

[10] P. Kaski, Engineering a delegatable and error-tolerant algorithm for counting small subgraphs, in *Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments, ALENEX 2018, New Orleans, LA, USA, January 7-8, 2018.* (R. Pagh and S. Venkatasubramanian, Eds.). SIAM, 2018, pp. 184–198. [doi:10.1137/1.9781611975055.16].

[11] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, 1998.

References IV

[12] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, 2009.

[13] S. Lang, *Algebra*, third ed., Springer-Verlag, New York, 2002. [doi:10.1007/978-1-4613-0041-0].

[14] N. Möller, On Schönhage's algorithm and subquadratic integer GCD computation, *Math. Comp.* 77 (2008), 589–607. [doi:10.1090/S0025-5718-07-02017-0].

[15] R. Orús, A practical introduction to tensor networks: Matrix product states and projected entangled pair states, *Annals of Physics* 349 (2014), 117–158. [doi:10.1016/j.aop.2014.06.013].

[16] A. Schönhage, Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2, *Acta Informat.* 7 (1976/77), 395–398. [doi:10.1007/BF00289470].

References V

[17] A. Schönhage and V. Strassen, Schnelle Multiplikation grosser Zahlen, *Computing (Arch. Elektron. Rechnen)* 7 (1971), 281–292.

[18] A. Shamir, How to share a secret, *Comm. ACM* 22 (1979), 612–613. [doi:10.1145/359168.359176].

[19] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, 1992. [doi:10.1137/1.9781611970999].