Combinatorial Optimization in Computer Vision
Lecture: F. R. Schmidt and C. Domokos
Exercises: T. Möllenhoff and T. Windheuser
Winter Semester 2015/2016

Computer Vision Group
Institut für Informatik
Technische Universität München

Weekly Exercises 1

Room: 02.05.014

Tuesday, 27.10.2015, 14:15-15:45

Submission deadline: Tuesday, 27.10.2015, 11:15 , Room 02.09.023

Distributive Lattices (7 Points)

Exercise 1 (2 Points)**.** Consider the set D_n of positive divisors of some natural number $n \geq 2$, partially ordered by the divisibility relation $a \preceq b \Leftrightarrow a \mid b$.

a) What are meet and join in this case? Show that D_n is a lattice.

b) Draw the Hasse diagram of D_{36}.

Solution.

a) The meet $x \wedge y$ is the greatest common divisor (gcd), which is defined as the biggest positive integer that is both a divisor of x and y. The join $x \vee y$ is the least common multiple (lcm), which is the smallest positive integer that is divisible by both x and y.

First we note that every number $n = p_1^{i_1} p_2^{i_2} \ldots p_k^{i_k}$ has a unique prime factorization. We show that D_n along with gcd and lcm is a lattice by constructing a lattice isomorphism to $P = [i_1] \times \ldots \times [i_k]$, with meet being the component-wise min and join the component-wise max. We use the notation $[n] = \{0, 1, \ldots, n\}$. It is easy to verify that the latter is a lattice, i.e. fulfills the axioms of meet and join.

Since the prime factorization is unique, the function $\varphi : D_n \to P$, with

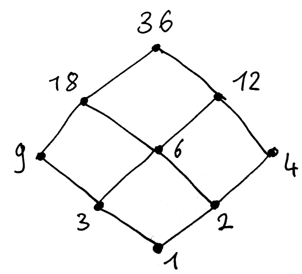
$$\varphi(n) = \begin{pmatrix} i_1 & i_2 & \ldots & i_k \end{pmatrix}^\top,$$

is well-defined and bijective. Clearly it holds (by definition of gcd and lcm) that

$$\begin{aligned} &a \mid b \;\Leftrightarrow\; \varphi(a) \preceq \varphi(b), \\ &\varphi(\gcd(a, b)) = \min(\varphi(a), \varphi(b)), \\ &\varphi(\mathrm{lcm}(a, b)) = \max(\varphi(a), \varphi(b)), \end{aligned}$$

where $\preceq$ denotes the component-wise conjunction of $\leq$ on $\mathbb{N}$.

b)



Exercise 2 (2 Points)**.** Let $(\Omega, \preceq)$ be a distributive lattice, i.e.,

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z). \tag{1}$$

a) Show that if $x \preceq y$, then

$$x \wedge y = x, \qquad x \vee y = y.$$

b) Verify that the second distributivity law,

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z),$$

follows from the first one (1).

Solution.

a) $\underline{x \wedge y = x}$: We want to show that $x \wedge y = x$, i.e.,

$$a \preceq x \;\Leftrightarrow\; a \preceq x \;\text{ and }\; a \preceq y.$$

The direction "$\Leftarrow$" is trivial. The direction "$\Rightarrow$" follows from transitivity of $\preceq$: $(a \preceq x \text{ and } x \preceq y) \Rightarrow a \preceq y$.

$\underline{x \vee y = y}$: Here we want to prove:

$$a \succeq y \;\Leftrightarrow\; a \succeq x \;\text{ and }\; a \succeq y.$$

The proof is analogous to the previous one.

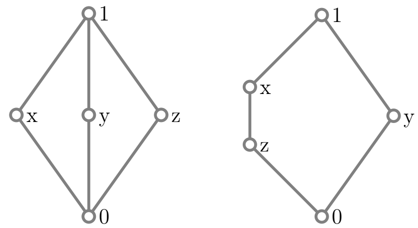
b) For showing distributivity, we first note that using a) it holds that

$$(x \vee y) \wedge x = x, \tag{2}$$

$$(x \wedge z) \vee x = x, \tag{3}$$

since $x \preceq (x \vee y)$ and $(x \wedge z) \preceq x$. This follows from the definition of join respectively meet with specific choice of $a = x \vee y$ and $a = x \wedge y$ and using reflexivity of the partial order. Now:

$$\begin{aligned} (x \vee y) \wedge (x \vee z) &= [(x \vee y) \wedge x] \vee [(x \vee y) \wedge z] && \text{Using (1).} \\ &= x \vee (x \wedge z) \vee (y \wedge z) && \text{Using (2).} \\ &= x \vee (y \wedge z). && \text{Using (3).} \end{aligned}$$



Exercise 3 (1 Point)**.** Show that the diamond and pentagon lattices sketched below on the set $\Omega = \{0, 1, x, y, z\}$ are both non-distributive.

Solution.

- Diamond lattice:

$$x \wedge (y \vee z) = x \wedge 1 = x \neq 0 = 0 \vee 0 = (x \wedge y) \vee (x \wedge z).$$

- Pentagon lattice:

$$x \wedge (y \vee z) = x \wedge 1 = x \neq z = 0 \vee z = (x \wedge y) \vee (x \wedge z).$$

Exercise 4 (2 Points)**.** Let $(\Omega, \preceq)$ be a finite totally ordered set. Give a well-defined bijective mapping $\varphi : \Omega \to \mathcal{I}_\Omega \setminus \{\varnothing\}$ such that:

$$x \preceq y \Leftrightarrow \varphi(x) \subset \varphi(y),$$

i.e., prove that there exists the following homomorphism:

$$(\mathcal{I}_\Omega \setminus \{\varnothing\}, \subset) \approx (\Omega, \preceq).$$

Solution. We first show that the map $\varphi : \Omega \to \mathcal{I}_\Omega \setminus \{\varnothing\}$ defined as

$$\varphi(x) = x_\preceq, \quad x_\preceq := \{y \mid y \preceq x\},$$

is bijective.

Surjectivity: Take a non-empty lower ideal $A \in \mathcal{I}_\Omega$. Since Ω is finite, A has a maximal element

$$z = \max_{x \in A} x.$$

We want to prove that $A = z_\preceq$.
$\underline{A \subset z_\preceq}$:
Take $x \in A$. Since $z \in A$ is the maximum, and due to the totality property of the total order, we have $x \preceq z$ and thus $x \in z_\preceq$.
$\underline{z_\preceq \subset A}$:
Take $x \in z_\preceq$. Then $x \preceq z$, and since

$$z \in A \Rightarrow [x \in A \text{ for all } x \preceq z],$$

we have $x \in A$.

Injectivity: Take two different $x, y \in \Omega$. Without loss of generality assume $x \prec y$, $x \neq y$. Then $y \in \varphi(y) = y_{\preceq}$ by reflexivity. Further, $y \notin \varphi(x) = x_{\preceq}$, since $y \succ x$, so $\varphi(x) \neq \varphi(y)$.

"$\Leftarrow$": Since $x \in \varphi(x) \subset \varphi(y)$, we have $x \in \varphi(y)$ and thus $x \preceq y$.

"$\Rightarrow$": We want to prove that $\varphi(x) \subset \varphi(y)$. For $a \in \varphi(x)$ it holds $a \preceq x \preceq y$ and thus $a \preceq y$ hence $a \in \varphi(y)$.

Submodular Functions (8 Points)

Exercise 5 (4 Points)**.** Let Ω be a finite set and let 2^Ω denote the power set of Ω. Prove the following statements:

a) The function $E : 2^\Omega \to \mathbb{R}$ defined by

$$E(A) = \sum_{i \in A} f(i),$$

is modular for any choice of $f : \Omega \to \mathbb{R}$.

b) Any modular function $E : 2^\Omega \to \mathbb{R}$ can be written as

$$E(A) = E(\varnothing) + \sum_{i \in A} \big[E(\{i\}) - E(\varnothing)\big].$$

c) Let $\Omega \subset \mathbb{Z}^2$. The function $E : 2^\Omega \to \mathbb{R}$ given as

$$E(A) = \sum_{i \in A} \sum_{\substack{j \notin A, \\ |i-j|=1}} 1,$$

is a submodular function.

d) Show that $E_h : 2^\Omega \to \mathbb{R}$ given by

$$E_h(A) := h(|A|),$$

is submodular if $h : \mathbb{R} \to \mathbb{R}$ is a concave function.
Hint: If $h : \mathbb{R} \to \mathbb{R}$ is concave, we have $h\frac{x+y}{2}) \geq \frac{h(x)+h(y)}{2}$.

Solution.

a) Let $S = A \cap B, P = A \setminus B, Q = B \setminus A$. Then $P \cup S = A$, $Q \cup S = B$ and $A \cup B = P \cup Q \cup S$. Applying the definition yields:

$$\begin{aligned} E(A \cup B) + E(A \cap B) &= \sum_{i \in A \cup B} f(i) + \sum_{i \in A \cap B} f(i) \\ &= \sum_{i \in P} f(i) + \sum_{i \in Q} f(i) + 2 \sum_{i \in S} f(i) \\ &= \sum_{i \in A} f(i) + \sum_{i \in B} f(i) = E(A) + E(B). \end{aligned}$$

b) We prove the statement by induction.
Base case:
$A = \varnothing$. Then $E(A) = E(\varnothing)$. $\checkmark$
Induction step:
We assume the statement holds for set A with n elements.

$$\begin{aligned} E(A \cup \{j\}) &= E(A) + E(j) - E(\varnothing) \\ &= E(\varnothing) + \Big[\sum_{i \in A} E(i) - E(\varnothing)\Big] + E(j) - E(\varnothing) \qquad \text{(Using induction hypothesis).} \\ &= E(\varnothing) + \Big[\sum_{i \in A \cup \{j\}} E(i) - E(\varnothing)\Big]. \end{aligned}$$

c) We can rewrite then length term as:

$$E(A) = \sum_{\substack{i \in \Omega, j \in \Omega, \\ |i-j|=1}} E_2(i \in A, j \in A),$$

for a pseudo-Boolean function $E_2 : \mathbb{B} \times \mathbb{B} \to \mathbb{R}$. It is defined as:

$$E_2(0,0) = E_2(1,1) = E_2(0,1) = 0, \quad E_2(1,0) = 1.$$

Since E_2 is submodular ($E_2(0,0) + E_2(1,1) \leq E_2(0,1) + E_2(1,0)$) and the sum over submodular functions is a submodular function again, E is also submodular.

d)

$$\begin{aligned} F(A \cup \{i,j\}) + F(A) = h(|A|+2) + h(|A|) &\overset{\text{concavity}}{\leq} 2h\left(\frac{2|A|+2}{2}\right) \\ &= F(A \cup \{i\}) + F(A \cup \{j\}). \end{aligned}$$

Exercise 6 (4 Points)**.** Let $\Omega = \{p, q\}$. A real-valued function on 2^Ω can be represented by $E : \{0,1\} \times \{0,1\} \to \mathbb{R}$. Let $E_1, E_2 : \{0,1\} \times \{0,1\} \to \mathbb{R}$ be two such functions, defined by the following table:

Here $a, b, c, d \in \mathbb{R}$ are constants.

x_1	x_2	$E_1(x_1, x_2)$	$E_2(x_1, x_2)$
0	0	a	0
0	1	b	-1
1	0	c	-1
1	1	d	0

a) Write down the Lovász extensions $E_1^L, E_2^L : [0,1] \times [0,1] \to \mathbb{R}$ of E_1, E_2.

b) Write down the convex closure $E_2^- : [0,1] \times [0,1] \to \mathbb{R}$ of E_2.

c) Write down the multilinear extensions $\overline{E}_1, \overline{E}_2 : [0,1] \times [0,1] \to \mathbb{R}$ of E_1, E_2.

d) Is E_2 submodular? Under which circumstances is E_1 submodular?

Hint: It is useful to a make a distinction between the cases $x_1 < x_2$ and $x_1 \geq x_2$.

Solution. a)

$$\begin{aligned} E_1^L(x_1, x_2) &= \begin{cases} x_1 d + (x_2 - x_1) b + (1 - x_2) a & \text{if } x_1 < x_2 \\ x_2 d + (x_1 - x_2) c + (1 - x_1) a & \text{otherwise} \end{cases} \\ E_2^L(x_1, x_2) &= \begin{cases} -(x_2 - x_1) & \text{if } x_1 < x_2 \\ -(x_1 - x_2) & \text{otherwise} \end{cases} = -|x_1 - x_2| \end{aligned} \tag{4}$$

b) $E_2^-(x_1, x_2) = -1 + |1 - x_1 - x_2|$.

c)

$$\begin{aligned} \overline{E_1}(x_1, x_2) &= a\overline{x_1 x_2} + b\overline{x_1} x_2 + c x_1 \overline{x_2} + d x_1 x_2 \\ \overline{E_2}(x_1, x_2) &= -\overline{x_1} x_2 - x_1 \overline{x_2} \end{aligned} \tag{5}$$

d) By definition E_1 is submodular iff

$$E_1(0,0) + E_1(1,1) \leq E_1(0,1) + E_1(1,0). \tag{6}$$

I.e. E_1 is submodular iff $a + d \leq b + c$. If follows that E_2 is not submodular (but supermodular).

Programming (4 Points)

Exercise 7 (0 Points)**.** This exercise is only necessary, if you want to program on your own computer or laptop. Download, compile and install *OpenGM* (Version ≥ 2) on your machine. The code is available from

`http://hci.iwr.uni-heidelberg.de/opengm2`

and the manual can be downloaded from

`http://hci.iwr.uni-heidelberg.de/opengm2/download/opengm-2.0.2-beta-manual.pdf`

Make sure you compile with the external libraries `MaxFlow`, `QPBO` and `TRWS`.

Make sure you have installed *Doxygen* and *cmake* on your machine. On Ubuntu just type `sudo apt-get install doxygen cmake`.

Building is a little bit tricky. You need to call `cmake` first, then `make externalLibs` and then `cmake` again. On Linux you can do this by just typing these lines on your terminal:

```
wget http://hci.iwr.uni-heidelberg.de/opengm2/download/opengm-2.3.5.
    zip
unzip opengm-2.3.5.zip
mkdir opengm-master/build
cd opengm-master/build
cmake ..
make externalLibs
cmake -DCMAKE_INSTALL_PREFIX=~/usr -DWITH_HDF5:BOOL="1" -
    DWITH_MAXFLOW:BOOL="1" -DWITH_QPBO:BOOL="1" -DWITH_TRWS:BOOL="1"
       ..
make -j 4
make doc
mkdir ~/usr
make install
```

The include files for Open GM should now be in the folder `~/usr/include/opengm`, the documentation should be located at `~/usr/doc/opengm/html` and the compiled libraries in `build/src/external`.

Exercise 8 (4 Points)**.** Install the *ImageMagick* C++ library called *Magick++*. The library is already installed on the machines in the lab. On Ubuntu you can do this by typing `sudo apt-get install libmagick++-dev`.

You can find information on Magick++ on the website

`http://www.imagemagick.org/Magick++/`

Most information you need and some easy examples are contained in the documentation of the `Magick::Image` class:

`http://www.imagemagick.org/Magick++/Image.html`

Write a program that does the following:

1. Read an image from a file.
2. Depending on the command-line arguments, perform one or more of the following operations:

   Convert the image from color to grey scale.

   Flip the x-axis of the image.

   Swap color channels.

   Display the image.
3. Save the image to a file.

The program should recognize the filenames and type of operation from the command-line arguments. For example, if the executable is called `exercise9` the call

```
./exercise9 -flipx -swaprg -display input.png output.jpg
```

would load an PNG image from the file `input.png`, flip the x-axis of the image, swap the red and green color channel, display the image and then write the result into the file `output.jpg`.