

Special thanks go to Wiebke Höhn and Max Klimm for providing the original template of this guide book.

Supporters:



Basic information	5
Sightseeing, restaurants, bars, cafés and more	10
Lecture notes	19
István Fekete, István László - Mathematical analysis of satellite images	21
1 Introduction	21
2 Principles of remote sensing	22
3 Pixel-based classification	23
4 Segment-based classification	26
5 Data fusion	30
6 Object-based image analysis	33
Róbert Freud - Hundred thousand dollars for a prime number	39
1 Introduction	39
2 Mersenne primes	39
3 Fermat primes	41
4 General primality tests	42
5 Factorization of integers	43
6 Revolutionary new ideas: public key cryptosystems and the RSA scheme	44
7 Exercises	44
Katalin Gyarmati - Pseudorandom binary sequences and lattices	47
1 Outline of the lecture	47
Zoltán Halasi - Permutation group algorithms	51
1 Introduction	51
2 Some basic algorithms	52
3 Polynomial time algorithm for deciding isomorphism of graphs with bounded valency	59
Tamás Király - Graph algorithms and LP duality	65
1 Introduction	65

2	Linear programming duality – all we need to know	65
3	Dijkstra’s algorithm in light of LP duality	66
4	Minimum cost arborescences	67
5	The Hungarian Method	68
6	Vertex cover – A primal-dual approximation algorithm	70
7	Online matching	71
Zoltán Király - Recent techniques in algorithm design		75
1	Introduction	75
2	Definitions	75
3	Main tool: Kernelization	76
4	Second tool: bounded search tree	77
5	Smaller kernel via crown reduction	78
6	Another parameter for VERTEX-COVER	80
7	Longest path, randomization and color coding	80
8	Feedback vertex set and iterative compression	81
9	Suggested reading	82
Tamás Kis - Scheduling problems and algorithms		83
1	Introduction	83
2	Single machine scheduling problems	84
3	Parallel machine scheduling problems	86
4	Open shop scheduling	88
5	Recommended readings	91
Péter Simon - Modeling propagation processes on networks by using differential equations		93
1	Introduction	93
2	Networks	94
3	Processes	95
4	Mean-field approximations for epidemic propagation	98

5 Analysis of the closed SIS systems 104



Contact

If you have any questions or problems, please contact the organizers or the lecturers.

István Ágoston

Email: agoston@cs.elte.hu
Tel.: +36 1 372 2500 / 8422
Room: 3.708

Kristóf Bérczi

Email: berkri@cs.elte.hu
Tel.: +36 1 372 2500 / 8582
Room: 3.519

István Fekete

Email: fekete.istvan@inf.elte.hu
Tel.: +36 1 372 2500 / 8490
Room: 2.609

Róbert Freud

Email: freud@cs.elte.hu
Tel.: +36 1 209 0555 / 8413
Room: 3.202

Katalin Gyarmati

Email: gykati@cs.elte.hu
Tel.: +36 1 372 2500 / 8418
Room: 3.205

Zoltán Halasi

Email: zhalasi@cs.elte.hu
Tel.: -
Room: 3.711

Alpár Jüttner

Email: alpar@cs.elte.hu
Tel.: +36 1 372 2500 / 8126
Room: 3.501

Tamás Király

Email: tkiraly@cs.elte.hu
Tel.: +36 1 372 2500 / 8053
Room: 3.503

Zoltán Király

Email: kiraly@cs.elte.hu
Tel.: +36 1 372 2500 / 8584
Room: 3.507

Tamás Kis

Email: kis.tamas@sztaki.mta.hu
Tel.: -
Room: 3.505

István László

Email: laszlo.istvan@fomi.hu
Tel.: +36 1 460 4233
Room: -

Péter Simon

Email: simonp@cs.elte.hu
Tel.: +36 1 209 5555 / 8417
Room: 3.701

Welcome to Summer School in Mathematics 2016!

In this short guide we would like to provide you with some basic information about practical issues as well as a rather incomplete list of sights, museums, restaurants, bars and pubs.

If you have any question concerning the summer school or your stay in Budapest, do not hesitate to ask us!

We wish you a pleasant stay in Budapest!

The Organizers

Venue

The summer school takes place in the Southern building ("Déli tömb") at the Lágymányosi Campus of the Eötvös Loránd University. The lectures are given in **room 4-710**, refreshments are provided in **room 4-713**.

Internet access

Wireless internet connection is available in the lecture room. If you need a scanner or a printer, please contact the organizers.

Lunch

There are several places around the university where you can have lunch. A cafeteria with a limited menu (including vegetarian choices) is in the northern building of the university. For a more complete list of restaurants, see the section "Around the campus".

Getting around by taxi

The taxi fares are uniformly calculated as follows: the base fee is 450 HUF with an additional 280 HUF/km or 70 HUF/min. Altogether, getting around by taxi is rather expensive and you might want to consider using public transport instead.

Getting around by bike

You can rent bikes for 2500 - 3500 HUF per day at many places. Here are some possibilities:

- **Budapestbike** - <http://www.budapestbike.hu/>
- **Yellow Zebra Bikes and Segways** - <http://www.yellowzebrabikes.com/>
- **Bikebase** - <http://bikebase.hu/home>
- **Bestbikerental** - <http://www.bestbikerental.hu/>

Since last year a large bicycle rental network (so called "Bubi") with many rental and return locations is run as part of the public transport system. You will find many automated rental places mostly in the downtown areas. If you want to rent a bicycle, you will have to pay a one week access fee of 2000 HUF plus the usage fee which is free for the first 30 minutes. A deposit is also required. Please, note that the fares are designed to encourage short term use. For the fares and further details see their website:

- **MOL Bubi** - <http://molbubi.bkk.hu/dijszabas.php>

Getting around by public transport

The public transportation system in Budapest is a favourite internal travel option for a number of Budapest visitors. The system is efficient, inexpensive and runs throughout all of the major tourist areas of Budapest. The system consists of a combination of the bus, trolley-bus, tram, metro, and train lines and is streamlined so that tickets for all of them can generally be purchased at the same locations.

All regular transportation services stop around midnight (varies by route). However, night buses (blue coloured buses, marked with black in the schedule, numbered 900-999 and tram line 6) replace the metro lines, major tram and bus routes and run through the night until normal service resumes in the morning. Separate schedules for night and day buses are posted at every stop. In the inner areas buses run very frequently (appr. 10-15 min.) Please note: there's front door boarding-only on most lines, except tram 6 and articulated buses.

Budapest currently has four metro lines - M1 (Yellow), M2 (Red), M3 (Blue) and M4 (Green). The Yellow line is the oldest underground transportation line in continental Europe and retains much of its old-fashioned charm. Lines M1, M2 and M3 meet at Deák Ferenc Tér in central Pest. Line M4 opened on 28 March 2014 and connects the Kelenföld railway station located in Buda, and the eastern Keleti station in Pest. Trains run frequently (2-5 minutes on weekdays, 5 minutes on weekends, 10 minutes late night).

Budapest has an extensive system of above-ground trams. The most useful lines for tourists are the famous 4 and 6, which follow the large ring road that encircles the Budapest

city center and crosses Margaret bridge before terminating at Széll Kálmán Tér on the Buda side on the North - and Petőfi bridge before it terminates at Móricz Zsigmond Körtér, also on the Buda side; no 47 and 49, which runs through central Pest and across the river all the way to Kelenföldi railway station; no. 2, which follows the Danube River on the Pest side; and no. 19, which follows the Danube River on the Buda Side.

Bus lines of use to most tourists are the 7 and 107 which connect the busy Keleti railway station and the area around the Kelenföld railway station on the Buda side. Some other notable places that they stop along is Blaha Lujza tér (connection to the red M2 metro line, also trams 4 and 6), Ferenciek tere (connection to metro line M3, also very near Váci utca), and in front of the Rudas bath and the Gellért Hotel both on the Buda side. Bus 27 takes you to the top of Gellért Hill from Móricz Zsigmond körtér while bus 26 connects Nyugati station (Metro line 3) and Árpád bridge (on the same metro line) via Margaret Island.

All public transport in Budapest is run by the company BKK. Connections can be easily checked at <http://www.bkk.hu/en/timetables/> or by using the convenient smart phone apps available for **Android** or **iPhone**. Online information - based on onboard GPS devices - about position of buses and timetable of bus, tram and trolley stops is available at <http://futar.bkk.hu>. The map also contains information about the number of available bicycles at the Bubi stations.

On the metro lines, **tickets need to be bought and validated before boarding** while on buses and trams you have to validate your ticket on the spot. For a complete list of tickets and conditions see <http://www.bkk.hu/en/prices/>.

<i>Single ticket</i>	350 HUF
Valid for one uninterrupted trip without change. On the metro lines the ticket must be validated before the start of the trip; on other vehicles immediately after boarding or after the vehicle has departed. Validity period is 80 minutes after stamping; it is 120 minutes on night services.	

<i>Block of 10 tickets</i>	3000 HUF
You can buy 10 tickets in a block with some discount compared to buying 10 single tickets separately.	

<i>Transfer ticket</i>	530 HUF
Valid for one trip with one change. Trip interruptions - other than changes - and return trips are not permitted. The ticket must be validated at the printed number grids at either end: first when starting a trip at one end and at the other end when changing, with the exception of changes between metro lines.	

<i>Short section metro ticket for up to 3 stops</i>	300 HUF
Valid on the metro for one short trip of up to 3 stops for 30 minutes after validation. Trip interruptions and return trips are not permitted.	

<i>Single ticket for public transport boat</i>	750 HUF
24-hour, 72-hour, 7-day travelcards and Monthly Budapest pass is valid on weekdays.	

<i>Budapest 24-hour travelcard</i>	1650 HUF
Valid for 24 hours from the indicated date and time (month, day, hour, minute) for an unlimited number of trips.	

<i>5/30 BKK 24-hour travelcard</i>	4550 HUF
The 5/30 BKK travelcard consists of 5 slips, each with a validity period of 24 hours. The block can be purchased with any starting day with a validity period of 30 days from the starting day.	

<i>Budapest 72-hour travelcard</i>	4150 HUF
Valid for 72 hours from the indicated date and time (month, day, hour, minute) for unlimited number of trips on the public transport services ordered by BKK on tram, trolleybus, underground, metro, cogwheel railway on the whole length of the lines on all days; on the whole length of boat services but only	

on working days.

Budapest 7-day travelcard 4950 HUF
Valid from 00:00 on the indicated starting day until 02:00 on the following 7th day for an unlimited number of trips. The travelcard is to be used only by one person; it is non-transferable as it is issued specifically for the holder.

Monthly Budapest pass for students 3450 HUF
Valid from 0:00 of the indicated optional starting day to 2:00 of the same day of the following month. Valid for students in higher education together with a Hungarian or EU or ISIC student ID.

Giraffe Hop On Hop Off

Giraffe Hop On Hop Off tours offer 2 bus, 1 boat and 1 walking tour during the day and 1 bus tour in the evening in Budapest for tourists. They pass several sights on their way; the RED and YELLOW lines are audio guided in 20 languages and the BLUE boat line is audio guided in English and German. The ticket is valid on the day of the first departure while the next day is free.

Things to do in Budapest

- Walk along the Danube on the Pest side between Elisabeth Bridge and Chain Bridge. Then cross the Danube and continue towards Margareth Bridge to see the Parliament Building from Batthyány tér.
- Take Metro 1 from Vörösmarty tér to Hősök tere and see the monument there. You may take a walk in the City Park (Városliget) or go to the Zoo or the Museum of Fine Arts.
- After 7:00pm take a short walk along Ráday utca from Kálvin tér. You may want to enter one of the cafés or restaurants.
- Go to the Great Market Hall on Várház körút, close to Liberty Bridge (Szabadság híd). After all, this is one of the few things Margaret Thatcher did when she visited Budapest in the 1980's (she bought garlic :)).
- Go to Gellért Hill to get a glimpse of the city from above.
- Go to a concert in one of the major or smaller concert halls, churches or open air locations. Some of them are free.
- Go to Margaret Island and see the fountain on the southern end or the music tower on the northern end of the island. In between you will find a garden of roses and a small zoo.
- Go at night to the Palace of Arts (across Eötvös University, close to Rákóczi Bridge on the Pest side) and enjoy the view of the National Theatre or of the glass walls of the Palace of Arts.
- See the bridges at night. You get a good view from Castle Hill.
- Go to some of the baths in Budapest (Széchenyi bath in the City park, Rudas bath at the Buda side of Elisabeth Bridge or Gellért bath in Hotel Gellért at the Buda side of Liberty Bridge).

Words of caution

- Don't go to a restaurant or café without checking the price list first. A reasonable dinner should not cost you more than 20 Euros (6000 HUF). (Of course you may be willing to pay more but you should know in advance.)
- Don't leave your valuables unattended, especially not in places frequently visited by tourists. Be aware of pickpockets on crowded buses or trams.
- Budapest is relatively safe even at night, nevertheless if possible, try to avoid being alone on empty streets at night. Some pubs should also be avoided.
- Don't carry too much cash with you: direct payment banking cards and most credit cards are widely accepted. If you withdraw money from a banking machine, be careful and try to do it in a public place.
- If you get on a bus, tram, trolley or metro, usually you have to have a pass or a prepaid ticket which you have to validate upon boarding (or when entering the metro station). Most of the tickets are valid for a single trip only (even if it is only for a short distance). If you get a pass for a week, you have to enter on the ticket the number of a photo id (passport, id card) which you have to carry with you when using the pass. - On some buses you may get a single ticket from the driver but be prepared to have change with you. Even tickets bought from the driver have to be validated.

Shopping centres

Mammut. 1024 Budapest, II. district, Lövház utca 2-6, +36 1 345 8020 www.mammut.hu

Westend City Center. 1062 Budapest, VI. district, Váci út 1-3, +36 1 238 7777 www.westend.hu

Corvin Plaza. 1083 Budapest, VIII. district, Futó utca 37, +36 1 799 2440 www.corvinplaza.hu

Arena Plaza. 1087 Budapest, VIII. district, Kerepesi út 9, +36 1 880 7010 www.arenaplaza.hu

Allee. 1117 Budapest, XI. district, Október huszonharmadika utca 8-10, +36 1 372 7208 www.allee.hu

Market halls

Batthyány téri Vásárcsarnok. 1011 Budapest, I. district, Batthyány tér 5

Rákóczi téri Vásárcsarnok. 1084 Budapest, VIII. district, Rákóczi tér 7-9, Mon 6:00am–4:00 pm, Tue–Fri 6:00am–6:00pm, Sat 6:00am–1:00pm

Várház körúti Vásárcsarnok. 1093 Budapest, IX. district, Várház körút 1-3, Mon 6:00am–5:00 pm, Tue–Fri 6:00am–6:00pm, Sat 6:00am–3:00pm

Fehérvári úti Vásárcsarnok. 1117 Budapest, XI. district, Kőrösi J. utca 7-9, Mon 6:30am–5:00 pm, Tue–Fri 6:30am–6:00pm, Sat 6:30am–3:00pm

Supermarkets

CBA

www.cba.hu/uzletlista,
www.primaboltok.hu/uzletlista

CBA Ferenciek Tere. 1053 Budapest, V. district, Ferenciek tere 2, Mon–Fri 6:00am–10:00pm, Sat 7:00am–10:00pm, Sun 8:00am–6:00pm

CBA Élelmiszer. 1053 Budapest, V. district, Veres Pálné u. 12, Mon–Fri 6:00am–10:00pm, Sat 7:00am–10:00pm, Sun 9:00am–9:00pm

CBA Millenium Príma élelmiszerüzlet. 1061 Budapest, VI. district, Andrásy út 30, Mon–Fri 7:00am–10:00pm, Sat 8:00am–8:00pm

Príma Élelmiszer Károly Krt. 9. 1075 Budapest, VII. district, Károly krt. 9, Mon–Fri 6:30am–9:30pm, Sat 8:00am–8:00pm, Sun 7:30am–2:30pm

Görög Csemege. 1085 Budapest, VIII. district, József krt. 31, Mon–Fri 6:00am–10:00pm, Sat 8:00am–10:00pm

CBA Prima. 1085 Budapest, VIII. district, József krt 84, Mon–

Sat 6:00am–9:30pm

Corvin Átrium CBA. 1082 Budapest, VIII. district, Futó utca 37-45, Mon–Fri 7:00am–10:00pm, Sat 7:00am–8:00pm, Sun 8:00am–8:00pm

Bakáts csemege. 1092 Budapest, IX. district, Bakáts tér 3, Mon–Fri 6:00am–7:30pm, Sat 7:00am–1:00pm

CBA Élelmiszer. 1111 Budapest, XI. district, Bartók Béla út 32, Mon–Sat 6:30am–10:00pm, Sun 7:00am–5:00pm

CBA Élelmiszer Karinth F. u. 30-32. 1111 Budapest, XI. district, Karinthy Frigyes u. 30-32, Mon–Fri 6:30am–9:00pm, Sat 8:00am–8:00pm

CBA Élelmiszer. 1117 Budapest, Kőrösi József u. 9., Mon 6:30am–5:00pm, Tue–Fri 6.30am–6.00pm, Sat 6:30am–3:00pm, Sun 6:30am–2:00pm

Spar/Interspar

www.spar.hu

SPAR. 1011 Budapest, I. district, Batthyány tér 5-6, Mon–Fri 7:00am–9:00pm, Sat 8:00am–8:00pm, Sun 8:00am–5:00pm

SPAR. 1024 Budapest, II. district Lövház utca 2-6, Mon–Sat 6:30am–10:00pm, Sun 8:00am–8:00pm

City SPAR. 1052 Budapest, V. district, Károly körút 22-24, Mon–Sat 7:00am–10:00pm, Sun 8:00am–6:00pm

SPAR. 1066 Budapest, VI. district, Teréz körút 28, Mon–Sat 6:30am–10:00pm, Sun 8:00am–6:00pm

SPAR. 1066 Budapest, VI. district, Nyugati tér 1-2, Mon–Sat 6:30am–10:00pm, Sun 8:00am–6:00pm

SPAR. 1073 Budapest, VII. district, Erzsébet körút 24, Mon–Sat 7:00am–10:00pm, Sun 8:00am–4:00pm

City SPAR. 1076 Budapest, VII. district, Thököly út 8, Mon–Sat 6:30am–10:00pm, Sun 8:00am–6:00pm

SPAR. 1085 Budapest, VIII. district, Blaha Lujza tér 1-2, Mon–Sat 6:30am–10:00pm, Sun 8:00am–6:00pm

City SPAR. 1092 Budapest, IX. district, Ráday utca 32, Mon–Fri 7:00am–10:00pm, Sat 7:00am–8:00pm, Sun 8:00am–4:00pm

City SPAR. 1095 BUDAPEST, IX. district, Mester utca 1, Mon–Sat 6:30am–10:00pm, Sun 8:00am–6:00pm

SPAR. 1095 Budapest, IX. district, Soroksári út 1, Mon–Sat 6:30am–9:00pm

SPAR Market. 1111 Budapest, XI. district, Bartók Béla út 14, Mon–Sat 6:00am–10:00am, Sun 9:00am–10:00pm

SPAR. 1117 Budapest, XI. district, Irinyi József utca 34, Mon–Fri 7:00am–8:30pm, Sat 8:00am–5:00pm, Sun 8:00am–2:00pm

INTERSPAR. 1117 Budapest, XI. district, Október 23-a utca 8-10 (in the basement of the Allee shopping center), Mon–Sat 7:00am–10:00pm, Sun 8:00am–7:00pm

SPAR. 1123 Budapest, XII. district, Alkotás utca 53, Mon–Sat 7:30am–10:00pm

Penny Market

www.penny.hu

Penny Market. 1085 Budapest, VIII. district, József Körút 45, Mon–Sat 6:00am–9:00pm, Sun 7:00am–8:00pm

ALDI

www.aldi.hu

ALDI. 1053 Budapest, V. district, Kossuth Lajos utca 13, Mon–Sat 7:00am–10:00pm, Sun 8:00am–6:00pm

ALDI. 1054 Budapest, V. district, Báthory utca 8, Mon–Sat 7:00am–9:00pm, Sun 8:00am–6:00pm

ALDI. 1081 Budapest, VIII. district, Rákóczi út 65, Mon–Sat 7:00am–10:00pm, Sun 8:00am–6:00pm

ALDI. 1093 Budapest, IX. district, Vámház körút 1-3, Mon–Sat 6:00am–9:00pm, Sun 8:00am–6:00pm

ALDI. 1094 Budapest, IX. district, Tűzoltó utca 10-16, Mon–Sat 7:00am–10:00pm, Sun 8:00am–6:00pm

Lidl

www.lidl.hu

Lidl. 1060 Budapest, VI. district, Bajcsy-Zsilinszky út 61, Mon–Sat 6:30am–9:00pm, Sun 7:00am–5:00pm

Lidl. 1061 Budapest, VI. district, Király utca utca 112, Mon–Sat 6:30am–9:00pm, Sun 7:00am–5:00pm

Lidl. 1082 Budapest, VIII. district, Leonardo da Vinci utca 23, Mon–Sat 7:00am–10:00pm, Sun 7:00am–5:00pm

Lidl. 1114 Budapest, XI. district, Bartók Béla út 47, Mon–Sat 7:00am–10:00pm, Sun 7:00am–5:00pm

Tesco

www.tesco.hu

Tesco Expressz. 1072 Budapest, VII. district, Rákóczi út 20, Mon–Sat 6:00am–10:00pm, Sun 8:00am–4:00pm

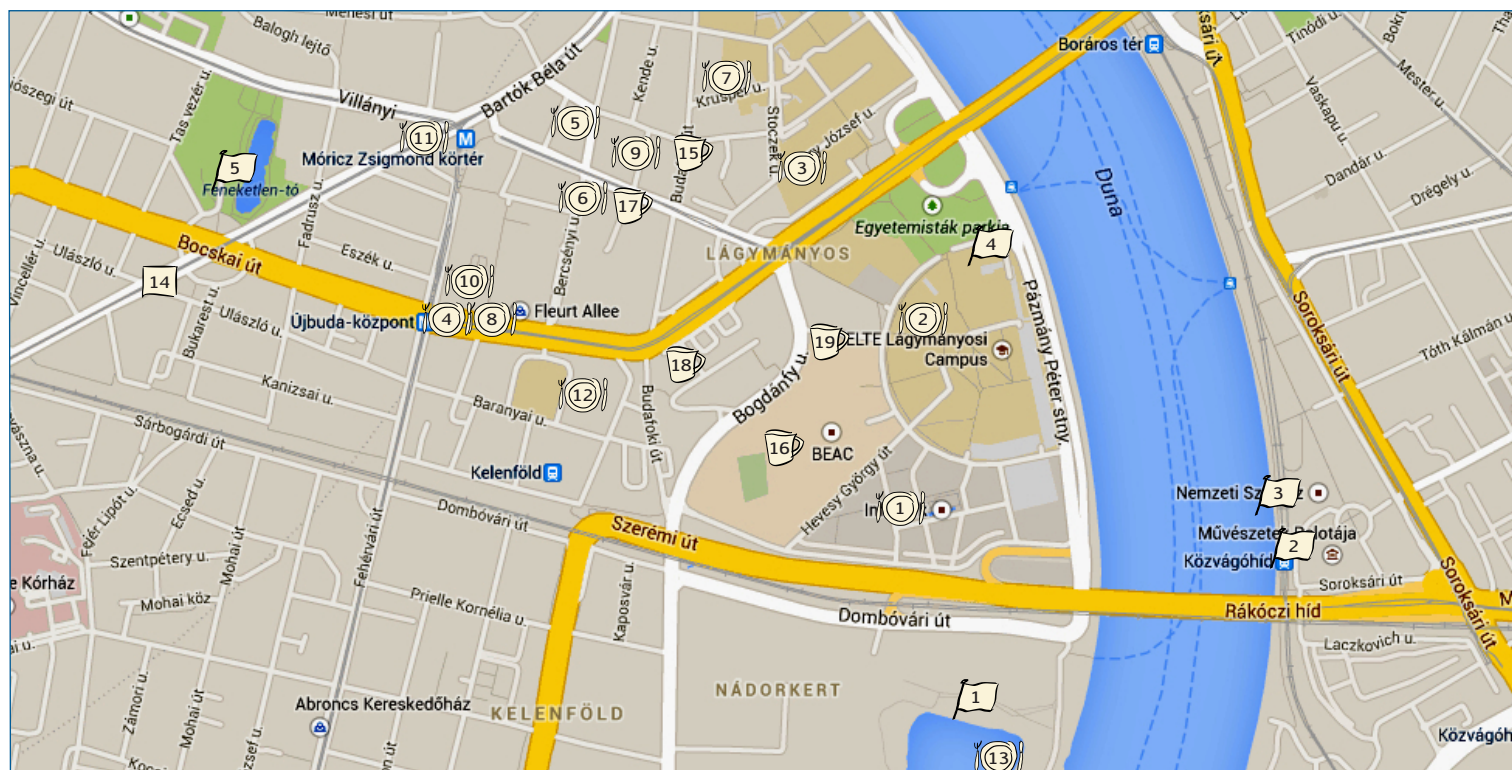
Tesco Arena Plaza Hipermarket. 1087 Budapest, VIII. district, Kerepesi út 9-11, Mon–Sat 6:00am–10:00pm, Sun 7:00am–8:00pm

Tesco Astoria Szupermarket. 1088 Budapest, VIII. district, Rákóczi út 1-3, Mon–Sat 6:00am–10:00pm, Sun 7:00am–6:00pm

Tesco Expressz. 1091 Budapest, IX. district, Kálvin tér 7, Mon–Sat 6:00am–10:00pm, Sun 7:00am–6:00pm

Tesco Soroksári úti Hipermarket. 1097 Budapest, IX. district, Koppány utca 2-4, Mon–Sat 6:00am–10:00pm, Sun 7:00am–8:00pm

Tesco Új Buda Center Hipermarket. 1117 Budapest, XI. district, Hengermalom út 19-21, Mon–Sat 6:00am–10:00pm, Sun 7:00am–8:00pm



Sights

- 1 Kopaszi-gát.** 1117 Budapest, Kopaszi gát 5, Bus 153, 154, 154B, 6:00am–2:00am.

Kopaszi-gát is a beautifully landscaped narrow peninsula in south Buda, next to Rákóczi Bridge. Nested in between the Danube on one side and a protected bay, it has a lovely beach feel. Kopaszi-gát is also a favourite picnic spot and the park offers lots of outdoor activities from biking to ball games. The sign in the park says it all: “Füre lépni szabad!”, which means “Stepping on the grass is permitted!”

- 2 Palace of Arts.** 1095 Budapest, Komor Marcell utca 1, Suburban railway 7, Tram 1, 2, 24, Mon-Sun 10:00am–10:00pm (varies).

The Palace of Arts in Budapest, also known as MÜPA for short (Művészetek Palotája), is located within the Millennium Quarter of the city, between Petőfi and Lágymányosi bridges. It is one of the most buzzing cultural and musical centres in Budapest, and as such one of the liveliest Budapest attractions.

- 3 National Theatre.** 1095 Budapest, Bajor Gizi park 1, Suburban railway 7.

The building lies on the bank of the Danube, and is a five-minute walk from the Csepel HÉV (Suburban railway 7). The area of the theatre can be functionally separated into three parts. The central part is the nearly round building of the auditorium and stage, surrounded by corridors and public areas. The second is the U-shaped industrial section around the main stage. The third section is the park that surrounds the area, containing numerous memorials commemorating the Hungarian drama and film industry.

- 4 A38 Ship.** 1117 Budapest, a little South from Petőfi bridge, Buda side, Trams 4 and 6 (Petőfi híd, budai hídfő), Buses 153, 154, 154B, Mon–Sat 11:00am–11:00pm.

The world’s most famous repurposed Ukrainian cargo ship, A38 is a concert hall, cultural center and restaurant floating on the Danube near the abutment of Petőfi Bridge on the Buda side with a beautiful panorama. Since its opening it has become one of Budapest’s most important venues, and according to artists’ feedback, one of Europe’s coolest clubs.

- 5 Feneketlen-tó.** 1114 Budapest, Bus 7, 107, Tram 19, 41, 47, 48, 49.

Feneketlen-tó, which means bottomless lake, is surrounded by a beautiful park filled with paths, statues and children’s playgrounds. The lake is not as deep as its name suggests. In the 19th century there was a brickyard in its place and the large hole dug by the workers filled with water when they accidentally hit a spring. Ever since, locals cherish the park and they come to feed the ducks, relax on the benches or take a stroll around the lake. The lake’s water quality in the 1980s began to deteriorate, until a water circulation device was built. The lake today is a popular urban place for fishing.

Restaurants & Eateries

- 1 Infopark.** Next to the university campus, Mon–Fri 8:00am–6:00pm.

Infopark is the first innovation and technology park of Central and Eastern Europe. There are several cafeterias and smaller sandwich bars hidden in the buildings, most of them are really crowded between 12:00am–2:00pm.

- 2 University Cafeteria.** University campus, Northern building, Mon–Fri 8:00am–4:00pm.

The university has a cafeteria on the ground floor of the Northern building. You can also buy sandwiches, bakeries, etc here.

- 3 Goldmann restaurant.** 1111 Budapest, Goldmann György tér 1, Mon–Fri 11:00am–3:00pm.


Goldmann is a cafeteria of the Technical University, popular among students for its reasonable offers. Soups are usually quite good.

- 4 Fehérvári úti vásárcsarnok.** 1117 Budapest, Kőrösi József utca 7–9, Mon 6:30am–5:00pm, Tue–Fri 6:30am–6:00pm, Sat 6:30am–2:00pm.


A farmers market with lots of cheap and fairly good native can- teens (e.g. Marika Étkezde) on the upper floors. You can also find cheese, cakes, fruits, vegetables etc.

- 5 Anyu.** 1111 Budapest, Bercsényi utca 8, Mon–Fri 11:00am–4:00pm.


Tiny bistro selling home-made soup, sandwiches and cakes.

 **Turkish restaurant.** 1111 Budapest, Karinty Frigyes út 26, Mon–Sun 10:00am–0:00am.


This tiny Turkish restaurant offers gyros, baklava and salads at a reasonable price.

 **Stoczek.** 1111 Budapest, Stoczek utca 1-3, Mon–Fri 11:00am–3:30pm.


Stoczek is a cafeteria of the Technical University. It offers decent portions for good price. There are two floors, a café can be found downstairs.

 **Allee.** 1117 Budapest, Október huszonharmadika utca 8-10, Mon–Sat 9:00am–10:00pm, Sun 9:00am–8:00pm.


A nearby mall with several restaurants on its 2nd floor.

 **Íz-lelő étkezde.** 1111 Budapest, Lágymányosi utca 17, Mon–Fri 10:00am–3:00pm.


Decent lunch for low price, and student friendly atmosphere. Only open from Monday to Friday!

 **Cserpes Milk Bar.** 1117 Budapest, Október Huszonharmadika utca 8-10, Mon–Fri 7:30am–10:00pm, Sat 9:00am–8:00pm, Sun 9:00am–6:00pm.


A milk bar just next to the shopping center Allee. Great place for having a breakfast or a quick lunch.

 **Wikinger Bistro.** 1114 Budapest, Móricz Zsigmond körtér 4, Mon–Sun 11:00am–10:00pm.

If you are up for hamburgers, Wikinger Bistro offers a huge selection of different burgers.


 **Hai Nam Bistro.** 1117 Budapest, Október huszonharmadika utca 27, Mon–Sat 10:00am–9:00pm, Sun 10:00am–3:00pm.

If you like Vietnamese cuisine and Pho, this may be the best place in the city. It is a small place, so be careful, it is totally full around 1:00pm.

 **Vakvarjú.** 1117 Budapest, Kopaszi gát 2, Mon–Sun 11:30am–10:00pm.


Vakvarjú can be found on the Kopaszi gát. It is a nice open-air restaurant where you can have lunch and relax next to the Danube for a reasonable price.

Others

 **Gondola.** 1115 Budapest, Bartók Béla út 69-71, Mon–Sun 10:00am–8:00pm.

This is a nice little ice cream shop right next to the Feneketlen-tó (Bottomless Lake).


Pubs

 **Bölcső.** 1111 Budapest, Lágymányosi utca 19, Mon–Fri 11:30am–11:00pm, Sat–Sun 12:00am–11:00pm.


Bölcső has a nice selection of Hungarian and Czech craft beers and one of the best all-organic homemade burger of the city. Other than burgers, the menu contains homemade beer snacks such as pickled cheese, hermelin (a typical Czech bar snack), and breadsticks. Bölcso also boasts a weekly menu that makes a perfect lunch or dinner.

 **Szertár.** 1117 Budapest, Bogdánfy utca 10, Mon–Fri 8:30am–4:30am.


Szertár is a small pub close to the university campus. It is located at the BEAC Sports Center and offers sandwiches and hamburgers as well. A perfect place to relax after a long day at the university where you can also play kicker.

 **Pinyó.** 1111 Budapest, Karinty Frigyes út 26, Mon 10:00am–0:00am, Tue–Sat 10:00am–1:00am, Sun 4:00pm–0:00am.

Squeezed to a basement, Pinyó looks like being after a tornado: old armchairs, kicker table, tennis racket on the wall, ugly chairs and tables. It does not promise a lot, but from the bright side, it is completely foolproof. Popular meeting place among students.

 **Lusta Macska.** 1117 Budapest, Irinyi József utca 38, Mon–Wed 4:00pm–0:45am, Thu–Sat 4:00pm–2:00am.

Lusta Macska is a cheap pub for students close to the Schönherz dormitory of the Technical University. It is a tiny place with very simple furniture.

 **Kocka.** 1111 Budapest, Wargha László út 1, Mon–Fri 6:30am–7:50pm.

Nearby the campus, the Kocka Pub is a rather cheap place mainly for students.



Sights

1

Great Market Hall. 1093 Budapest, Vármház körút 1-3, Metro 4, Tram 47, 48, 49, Mon 6:00am–5:00 pm, Tue–Fri 6:00am–6:00pm, Sat 6:00am–3:00pm.

Central Market Hall is the largest and oldest indoor market in Budapest. Though the building is a sight in itself with its huge interior and its colourful Zsolnay tiling, it is also a perfect place for shopping. Most of the stalls sell fruits and vegetables but you can also find bakery products, meat, dairy products and souvenir shops. In the basement there is a supermarket.

2

Károlyi Garden. 1053 Budapest, Károlyi Mihály utca 16, Metro 2, Bus 5, 7, 8, 107, 133, 233, Tram 47, 48, 49.

Károlyi Garden is maybe the most beautiful park in the center of Budapest. It was once the garden of the castle next to it (Károlyi Castle, now houses the Petőfi Literature Museum). In 1932 it was opened as a public garden. In the nearby Ferenczy utca you can see a fragment of Budapest's old town wall (if you walk in the direction of Múzeum körút).

3

Gellért Hill and the Citadel. 1118 Budapest, Metro 4, Bus 5, 7, 8, 107, 133, 233.

The Gellért Hill is a 235 m high hill overlooking the Danube. It received its name after St. Gellért who came to Hungary as a missionary bishop upon the invitation of King St. Stephen I. around 1000 a.d. If you approach the hill from Gellért square, you can visit the Gellért Hill Cave, which is a little chapel. The fortress of the Citadel was built by the Habsburgs in 1851 to demonstrate their control over the Hungarians. Though it was equipped with 60 cannons, it was used as threat rather than a working fortification. From the panorama terraces one can have a stunning view of the city, especially at night. By a short walk one can reach the Liberation Monument.

4

Sziklatemplom (Cave Church). 1111 Budapest, Szent Gellért tér, Metro 4, Buses 7, 107, 133, 233, Trams 19, 41, 47, 48, 49, 56, 56A, Mon–Sat 9:30am–7:30pm, 500 HUF.

The Cave Church, located inside Gellért Hill, isn't your typical church with high ceilings and gilded interior. It has a unique setting inside a natural cave system formed by thermal springs.

5

Rudas Gyógyfürdő és Uszoda. 1013 Budapest, Döbrentei tér 9 (a little South from Erzsébet bridge, Buda side), Buses 5, 7, 8, 107, 109, 110, 112, 233, 239, Trams 19, 41, 56, 56A, Mon–Sun 6:00am–10:00pm; Night bath: Fri–Sat 10:00pm–4:00am, 1350–4200 HUF.

Centered around the famous Turkish bath built in the 16th century, Rudas Spa offers you several thermal baths and swimming pools with water temperatures varying from 16 to 42 Celsius degrees.

6

Gellért Gyógyfürdő és Uszoda. 1118 Budapest, Kelenhegyi út 4 (at Gellért tér), Metro 4, Buses 7, 107, 133, 233, Trams 19, 41, 47, 48, 49, 56, 56A, Mon–Sun 6:00am–8:00pm, 5100–5500 HUF.

Gellért Thermal Bath and Swimming Pool is a nice spa in the center of the city.

7

National Museum. 1088 Budapest, Múzeum körút 14-16, Metro 3, 4, Bus 9, 15, 115, Tram 47, 48, 49, Tue–Sun 10:00am–6:00pm, 800 HUF.

The Hungarian National Museum (Hungarian: Magyar Nemzeti Múzeum) was founded in 1802 and is the national museum for the history, art and archaeology of Hungary, including areas not within Hungary's modern borders such as Transylvania; it is not to be confused with the collection of international art of the Hungarian National Gallery. The museum is in Budapest VIII in a purpose-built Neoclassical building from 1837-47 by the architect Mihály Pollack.

Restaurants & Eateries

1


Pagony. 1114 Budapest, Kemenes utca 10, Mon–Sun 11:00am–10:00pm.

If you are looking for a cool spot in the blazing summer heat of Budapest, look no further. This joint was created by its resourceful proprietor by converting an unused toddler's pool section of the Gellért bath into a trendy pub. While there is no water (yet) in the pools, you can definitely find a table with comfy chairs which are actually in a wading pool.


2

Hummus Bar. 1225 Budapest, Bartók Béla út 6, Mon–Fri 10:00am–10:00pm, Sat–Sun 12:00am–10:00pm.


The famous homemade Hummus can be enjoyed in variety of different dishes. The menu offers everything from a wide variety of quality salads, soups, desserts, meats and vegetarian dishes. The food is prepared with great care using only high quality products, and focusing on the simplicity of preparation - thus allowing affordable pricing.

 **Főzelékfaló Ételtár.** 1114 Budapest, Bartók Béla út 43–47, Mon–Fri 10:00am–9:30pm, Sat 12:00–8:00pm.

Főzelékfaló Ételtár boasts a selection centered on főzelék, a Hungarian vegetable dish that is the transition between a soup and a stew, but you can get fried meats, several side dishes, and desserts as well.

 **Főzelékfaló Ételtár.** 1053 Budapest, Petőfi Sándor utca 1 (Ferenciek tere), Mon–Fri 10:00am–8:00pm, Sat 12:00–8:00pm.

Főzelékfaló Ételtár boasts a selection centered on főzelék, a Hungarian vegetable dish that is the transition between a soup and a stew, but you can get fried meats, several side dishes, and desserts as well.


 **Púder.** 1091 Budapest, Ráday utca 8, Mon–Sun 12:00am–1:00am.

Restaurant and bar with a progressive, eclectic interior that was created by Hungarian wizards of visual arts. Its back room gives home to a studio theatre. Many indoor and outdoor cafés, bars, restaurants and galleries are located in the same street, the bustling neighborhood of Ráday Street is often referred to as “Budapest Soho”.


Cafés

 **CD-fű.** 1053 Budapest, Fejér György utca 1, Mon–Thu 6:00pm–12:00pm, Fri–Sat 4:00pm–12:00pm.

As the third teahouse of Budapest, CD-fű is located in a slightly labyrinth-like basement. With its five rooms it is a bit larger than usual, and also gives place for several cultural events.


 **Hadik kávéház.** 1118 Budapest, Bartók Béla út 36, Mon–Sat 9:00am–11:00pm.

A lovely place to relax and soak up the atmosphere of pre-war years in Budapest. Hadik is a pleasant, old-fashioned café serving excellent food.


 **Sirius Teaház.** 1088 Budapest, Bródy Sándor utca 13, Mon–Sun 12:00am–10:00pm.

Sirius teahouse has the perfect atmosphere to have a cup of tea with your friends, but it is better to pay attention to the street numbers, this teahouse is very hard to find, there is no banner above the entrance. Customers can choose from 80 different types of tea.

Pubs

 **Mélypont Pub.** 1053 Budapest, Magyar utca 23, Mon–Tue 6:00pm–1:00am, Wed–Sat 6:00pm–2:00am.


Basement pub in the old city center. Homey atmosphere with old furniture.

 **Trapéz.** 1093 Budapest, Imre utca 2, Mon–Tue 10:00am–0:00am, Wed–Fri 10:00am–2:00am, Sat 12:00am–2:00am.


Nice ruin pub in an old house behind the Great Market Hall which also has an open-air area. You can watch sports events and play kicker on the upper floor.

 **Élesztő.** 1094 Budapest, Tűzoltó utca 22, Mon–Sun 3:00pm–3:00am.

Élesztő is the Gettysburg battlefield of the Hungarian craft beer revolution; it's a like a mixture of a pilgrimage site for beer lovers, and a ruin-pub with 17 beer taps, a home brew bar, a theater, a hostel, a craft pálinka bar, a restaurant and a café.


 **Mr. & Mrs. Columbo.** 1013 Budapest, Szarvas tér 1, Mon–Sat 4:00pm–11:00pm.

A nice pub with excellent food and czech beers. Their hermelin is really good.

 **Aréna Corner.** 1114, Budapest, Bartók Béla út 76, Sun–Fri 12:00am–11:00pm, Sat 12:00am–12:00pm.

A nice place to watch World Cup matches while drinking Czech beer.

Others

 **Mikszáth square.** 1088 Budapest, Mikszáth Kálmán tér.

Mikszáth tér and the surrounding streets are home to many cafés, pubs and restaurants usually with nice outdoor terraces. Many places there provide big screens to watch World Cup matches.



Sights

1

St. Stephen's Basilica. 1051 Budapest, Szent István tér 1, Metro 1, 2, 3, Bus 9, 16, 105, Guided tours Mon–Fri 10:00am–3:00pm, 1200 HUF.

This Roman Catholic Basilica is the most important church building in Hungary, one of the most significant tourist attractions and the third highest building in Hungary. Equal with the Hungarian Parliament Building, it is one of the two tallest buildings in Budapest at 96 metres (315 ft) - this equation symbolises that worldly and spiritual thinking have the same importance. According to current regulations there cannot be taller building in Budapest than 96 metres (315 ft). Visitors may access the dome by elevators or by climbing 364 stairs for a 360° view overlooking Budapest.

2

Opera. 1061 Budapest, Andrásy út 22, Metro 1, Bus 105, Trolleybus 70, 78, Tours start at 2:00pm, 3:00pm and 4:00pm, 1990 HUF.

The Opera House was opened in 1884 among great splendour in the presence of King Franz Joseph. The building was planned and constructed by Miklós Ybl, who won the tender among other famous contemporary architects.

3

Kossuth Lajos Square. 1055 Budapest, Metro 2, Bus 15, 115, Tram 2.

The history of Kossuth Lajos Square goes back to the first half of the 19th century. Besides the Parliament, other attractions in the square refer to the Museum of Ethnography (which borders the square on the side facing the Parliament) and to several monuments and statues. The square is easily accessible, since the namesake metro station is located on the south side of the square.

4

Fishermen's Bastion. 1014 Budapest, Hess Andras Square, Bus 16, 16A, 116, all day, tower: daily 9:00am–11:00pm; free, tower: 350 HUF.

On the top of the old fortress walls, the Fishermen's Bastion was only constructed between 1895–1902. It is named after the fishermen's guild because according to customs in the middle ages this guild was in charge of defending this part of the castle wall. As a matter of fact it has never had a defending function. The architect was Frigyes Schulek, who planned the building in neo-gothic style.

5

Parliament. 1055 Budapest, Kossuth Lajos tér 1-3, Metro 2, Bus 15, 115, Tram 2, Mon–Sun 8:00am–6:00pm, 3500 HUF, EU citizens and students 1750 HUF, EU students 875 HUF.

The commanding building of Budapest Parliament stretches between Chain Bridge and Margaret Bridge on the Pest bank of the Danube. It draws your attention from almost every river-side point. The Gellért Hill and the Castle Hill on the opposite bank offer the best panorama of this huge edifice. The Hungarian Parliament building is splendid from the inside too. You can visit it on organised tours. Same-day tickets can be purchased in limited numbers at our ticket office in the Museum of Ethnography. Advance tickets are available online at www.jegymester.hu/parlament.

6

Buda Castle and the National Gallery. 1014 Budapest, Szent György tér 2, Bus 16, Funicular, Tue–Sun 10:00am–6:00pm, 900 HUF.

Buda Castle is the old royal castle of Hungary, which was damaged and rebuilt several times, last time after World War II. Now it houses the Széchényi Library and the National Gallery, which exhibits Hungarian paintings from the middle ages up to now. The entrance to the castle court is free (except if there is some festival event inside). One of the highlights of the court is the Matthias fountain which shows a group of hunters, and the monument of Prince Eugene Savoy. From the terrace of the monument you have a very nice view of the city.

7

Matthias Church. 1014 Budapest, Szentháromság tér 2, Bus 16, 16A, 116, Mon–Fri 9:00am–5:00pm, Sat 9:00am–12:15pm, Sun 1:00pm–5:00pm, 1000 HUF.

Matthias Church (Mátyás-templom) is a Roman Catholic church located in front of the Fisherman's Bastion at the heart of Buda's Castle District. According to church tradition, it was originally built in Romanesque style in 1015. The current building was constructed in the florid late Gothic style in the second half of the 14th century and was extensively restored in the late 19th century. It was the second largest church of medieval Buda and the seventh largest church of medieval Hungarian Kingdom. Currently it also regularly houses various concerts.

8

Heroes Square. 1146 Budapest, Hősök tere, Metro 1, Bus 20E, 30, 105, Trolleybus 72, 75, 79.

The monumental square at the end of Andrásy Avenue sums up the history of Hungary. The millennium memorial commemorates the 1000th anniversary of the arrival of the Hungarians in the Carpathian Basin.

9 **Városliget.** 1146 Budapest, Városliget, Metro 1, Bus 20E, 30, 105, Tram 1, Trolleybus 70, 72, 74, 75, 79.

Városliget (City Park) is a public park close to the centre of Budapest. It is the largest park in the city, the first trees and walkways were established here in 1751. Its main entrance is at Heroes Square, one of Hungary's World Heritage sites.

10 **Vajdahunyad vára.** 1146 Budapest, Városliget, Metro 1, Bus 20E, 30, 105, Trolleybus 70, 72, 75, 79, Courtyard always open, Castle Tue–Sun 10:00am–5:00pm, Courtyard free, Castle 1100 HUF.

Vajdahunyad Castle is one of the romantic castles in Budapest, Hungary, located in the City Park by the boating lake / skating rink. The castle, despite all appearances, was built in 1896, and is in fact a fantasy pastiche showcasing the architectural evolution through centuries and styles in Hungary. The castle is the home of several festivals, concerts and the exhibitions of the Hungarian Agricultural Museum.

11 **Museum of Fine Arts (Szépművészeti Múzeum).** 1146 Budapest, Dózsa György út 41, Metro 1, Trolleybus 72, 75, 79, Temporarily closed till 2018 due to renovation.

The Museum of Fine Arts is a museum in Heroes' Square, Budapest, Hungary. The museum's collection is made up of international art (other than Hungarian), including all periods of European art, and comprises more than 100,000 pieces. The Museum's collection is made up of six departments: Egyptian, Antique, Old sculpture gallery, Old painter gallery, Modern collection, Graphics collection.

12 **Zoo Budapest (Fővárosi Állat és Növénykert).** 1146 Budapest, Állatkerti körút 6-12, +36 1 273 4900, Metro 1, Trolleybus 72, 75, 79, Mon–Thu 9:00am–6:00pm, Fri–Sun 9:00am–7:00pm, 1900 HUF.

The Budapest Zoo and Botanical Garden is one of the oldest in the world with its almost 150 years of history. Some of its old animal houses were designed by famous Hungarian architects. Nowadays it houses more than 1000 different species. Currently the greatest attraction is Asha, the child elephant.

13 **Great Synagogue.** 1072 Budapest, Akácfa utca 47., Metro 2, Bus 5, 7, 8, 9, 107, 133, 178, 233, Tram 47, 48, 49, Trolleybus 74, Sun–Thu 10:00am–6:00pm, Fri 10:00am–4:30pm, 2950 HUF.

The Great Synagogue in Dohány Street is the largest Synagogue in Europe and the second largest in the world. It can accommodate close to 3,000 worshipers. It was built between 1854 and 1859 in Neo-Moorish style. During World War II, the Great Synagogue was used as a stable and as a radio communication center by the Germans. Today, it's the main center for the Jewish community.

14 **Millenáris.** 1024 Budapest, Kis Rókus utca 16-20, Tram 4, 6, 17 (Széna tér), Bus 6, 11, 111, Mon–Sun 6:00am–11:00pm.

Located next to the Mammut mall, at the site of the one-time Ganz Electric Works, Millenáris is a nice park and venue for exhibitions, concerts, performances. You can also see a huge hyperbolic quadric and its two reguli.

15 **Batthyány tér.** 1011 Budapest, Metro M2, Tram 19, 41, Bus 11, 39, 111, Suburban railway 5.

Batthyány square has a great view on the beautiful Hungarian Parliament, one of Europe's oldest legislative buildings, a notable landmark of Hungary.

16 **Erzsébet tér.** 1051 Budapest, Metro 1, 2, 3, Buses 15, 16, 105, Trams 47, 48, 49, Metro 1, 2, 3.

Erzsébet Square is the largest green area in Budapest's inner city. The square was named after Elisabeth, 'Sisi', wife of Habsburg Emperor Franz Joseph, in 1858. The square's main attraction is the Danubius Fountain, located in the middle of the square, symbolizing Hungary's rivers. One of the world's largest mobile Ferris wheels can be also found on the square. The giant wheel offers fantastic views over Budapest day and night. Standing 65 meters tall, the wheel with its 42 cars is Europe's largest mobile Ferris wheel.

17 **Hungarian Academy of Sciences.** 1051 Budapest, Széchenyi István tér 9, Tram 2, Bus 15,16, 105, 115.

The Hungarian Academy of Sciences is the most important and prestigious learned society of Hungary. Its seat is at the bank of the Danube in Budapest.

18 **Playground for adults.** 1124 Budapest, Vérmező.

A playground for adults? Yes, this indeed exists and can be found in a nice park on the Buda side, close to the castle.

Restaurants & Eateries

1 **Onyx restaurant.** 1051 Budapest, Vörösmarty tér 7-8, Tue–Fri 12:00am–2:30pm, 6:30pm–11pm; Sat 6:30pm–11:00pm.

Exclusive atmosphere, excellent and expensive food – Onyx is a highly elegant restaurant with one Michelin Star. Do not forget to reserve a table.

2 **Pizza King.** 1072 Budapest, Akácfa utca 9, Mon–Fri 10:00am–0:00am, Sat–Sun 10:00am–3:00am.

During lunchtime on weekdays offers nice menus for 900 HUF, and you can buy cheap pizza there at any time of the day. Also runs pizza takeaways at many locations in the city.

Pubs

3 **Snaps Galéria.** 1077 Budapest, Király utca 95, Mon–Fri 2:00pm–0:00am, Sat 6:00pm–0:00am.

Snaps is a tiny two-floor pub located in the sixth district. From the outside it is nothing special, but entering it has a calm atmosphere. The beers selection - Belgian and Czech beers - is quite extraordinary compared to other same level pubs.

4 **Noiret Pool and Darts Hall, Cocktail Bar and Pub.** 1066 Budapest, Dessewffy utca 8-10, Mon–Sun 10:00am–4:00am.

A good place to have a drink and play pool, darts, snooker, or watch soccer.

5 **Szimpla Kert.** 1075 Budapest, Kazinczy utca 14, Mon–Thu 12:00am–4:00am, Fri 10:00am–4:00am, Sat 12:00am–4:00am, Sun 9:00am–5:00am.

Szimpla Kert (Simple Garden) is the pioneer of Hungarian ruin pubs. It is really a cult place giving new trends. Undoubtedly the best known ruin pub among the locals and the tourists, as well.

Others

6 **Mammut Shopping and Entertainment Centre.** 1024 Budapest, Lövház utca 2-6, Mon–Sat 6:30am–10:00pm, Sun 9:00am–10:00pm.

A twin mall in the heart of Buda.

7 **WestEnd City Center.** 1062 Budapest, Váci út 1-3,
Mon–Sun 8:00am–11:00pm.

A big mall with stores, restaurants etc. and a roof garden.

8 **Corvintető.** 1085 Budapest, Blaha Lujza tér 1-2,
Mon–Sun 6:00pm–6:00am.

Situated on the rooftop of once-glorious Corvin Department Store, Corvintető offers world-class DJs and concerts every day of the week. Recommended by The New York Times. Do not mess it up with Corvin Negyed, another stop of trams 4 and 6.



Margaret Island (Margitsziget) is the green heart of Budapest. It lies in the middle of the Danube between Margaret Bridge and Árpád Bridge. Apart from a couple of hotels and sport facilities, there are no buildings on the Island, it is a huge green park with promenades and benches, great for a date or a picnic. Everyone can find their own cup of tea here: there is the Hajós Alfréd National Sports Swimming Pool, the Palatinus and the running track for the sporty, the petting zoo, the music fountain and the Water Tower for families, and we recommend the Japanese Garden or a ride on a 4-wheel bike car for couples. If you're hungry for culture, check out the open-air stages and the medieval ruins of the Island.

Sights

- 1** **Entrance of Margit-sziget.** Budapest, Margit híd, Trams 4 and 6 (Margit híd, Margit-sziget).

Here you can enter the beautiful Margit-sziget (Margaret Island) on foot. However, you may also take bus 26 to get to the Island.

- 2** **Kiscelli Múzeum.** 1037 Budapest, Kiscelli utca 108, Tram 17, 19, 41, Bus 165, Tue–Sun 10:00am–6:00pm, 800 HUF.

Kiscelli Múzeum is located in a beautiful baroque monastery in Old-Buda. It offers exhibitions on the history of Budapest between the 18–21. centuries.

- 3** **Görzenál.** 1036 Árpád fejedelem útja 125, Bus 29, Suburban railway 5, Mon–Fri 2:00pm–8:00pm, Sat–Sun 9:30am–8:00pm, 900 HUF.

Görzenál currently is the biggest outdoor roller skating rink in Europe. The skating surface of the Gorzenal Roller Skate and Recreational Park is 14,000 square meters. This rink, which is located in picturesque surroundings along the Danube and Margaret Island, has a skating track as well as park structures for aggressive roller sports and BMX.

- 4** **Pál-völgyi Cave.** 1025 Budapest, Szépvölgyi út 162, Bus 65, Tue–Sun 10:00am–4:00pm, 1100 HUF.

An 500-metre long route in a cave with narrow, canyon-like corridors, large level differences, astonishing stone formations, drip stones, glittering calcium-crystals and prints of primeval shells. Even with the 120 steps and the ladder that have to be mounted, the whole tour can easily be fulfilled in normal clothes and comfortable shoes.

- 5** **Gül baba's türbe.** 1023 Budapest, Mecset utca 14 (entrance: Türbe tér 1), Tram 4, 6, Mon–Sun 10:00am–6:00pm, free.

The tomb of Gül Baba, “the father of roses”, who was a Turkish poet and companion of Sultan Suleiman the Magnificent. He died shortly after the Turkish occupation of Buda in 1541 and his tomb is said to be the northernmost pilgrimage site of the muslims in the world. It is located on a hilltop, surrounded by a beautiful garden which offers a nice view of the city.

Bars

- 1** **Holdudvar Courtyard.** 1138 Budapest, Margitsziget, Mon–Wed 11:00am–2:00am, Thu 11:00am–4:00am, Fri–Sat 11:00–5:00, Sun 11:00–2:00am.

A great entertainment spot in Budapest where everybody finds something to do: an open-air cinema, café, bar. The gallery exhibits works of contemporary fine art. Holdudvar hosts fashion shows and various cultural events.

Lecture notes

	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 – 10:30	Róbert Freud: Hundred thousand dollars for a prime number	Tamás Kis: Scheduling problems and algorithms	Péter Simon: Modeling propagation processes on networks by using differential equations	Tamás Kis: Scheduling problems and algorithms	Péter Simon: Modeling propagation processes on networks by using differential equations
10:30 – 10:45	Coffee break	Coffee break	Coffee break	Coffee break	Coffee break
10:45 – 12:15	Zoltán Király: Recent techniques in algorithm design	Katalin Gyarmati: Pseudorandom binary sequences and lattices	Tamás Király: Graph algorithms and LP duality	Zoltán Király: Recent techniques in algorithm design	Tamás Király: Graph algorithms and LP duality
12:15 – 13:30	Lunch break	Lunch break	Excursion	Lunch break	Lunch break
13:30 – 15:00	Zoltán Halasi: Permutation group algorithms	István Fekete, István László: Mathematical analysis of satellite images		Zoltán Halasi: Permutation group algorithms	Alpár Jüttner: LEMON: Library for Efficient Modeling and Optimization in Networks
15:00 – 15:15	Get together party	Coffee break		Coffee break	
15:15 – 16:45		Alpár Jüttner: LEMON: Library for Efficient Modeling and Optimization in Networks		István Fekete, István László: Mathematical analysis of satellite images	

1 Introduction

Remote sensing (RS) has an increasing role in survey, observation and control activities related mainly to agriculture. In the past years, several monitoring tasks were supported by remote sensing in Hungary. Crop mapping and yield forecasting, flood, waterlog and drought monitoring, applications related to area based agricultural subsidies and ragweed (*ambrosia*) control and exemption can be mentioned, among others. Remote sensing provides a sophisticated and cost effective technology to monitoring activities. Nowadays, very high resolution and hyperspectral images show the most spectacular technical development. On the other hand, the research and methodology that support the applications are also essential parts of the successful solution.

Institute of Geodesy, Cartography and Remote Sensing (FÖMI) has an experience in the usage of *satellite images* in different agricultural applications. The projects require a versatile expertise also in informatics and mathematics beside geoscience, agriculture and others. Despite the diversity of project domains the main goal is usually to identify the real-world content (land cover, land use, specific objects) that is represented by remotely sensed images in digital form. Thus the *classification* can be regarded as a central issue of *image analysis*.

After introducing basic concepts of RS, this paper gives an overview on the relevant topics in classifying satellite images. Most of the classification methods, even so far, use a basically traditional per-point approach. The *pixel-based classification* method ignores the spectral identity or similarity of the neighboring pixels in homogeneous areas.

A possible improvement of the traditional pixel-based classification method is the *segment-based classification*, i.e. to group certain pixels into segments and to apply the further classification steps to them. The choice of the appropriate method of *segmentation*, *clustering* and *classification* can be a crucial point when analyzing an image.

The application of segmentation is not only an option, but a necessity in the processing of very high resolution images, as their pixels usually cannot be interpreted individually. Several attributes, derived from geometrical properties (e.g. texture), are assigned to segments. This leads to the advanced approach called *object-based image analysis (OBIA)*.

In monitoring tasks, it is usual to use several remote sensing images of different kinds: multispectral and panchromatic optical satellite images, aerial photos and radar images. *Data fusion* aims to integrate the advantages of the images. In current applications, most often used data fusion methods improve the quality of the input by merging the beneficial attributes of different images. This approach has been generalized at FÖMI's projects.

There are further relevant topics which remain out of scope (e.g. preprocessing, filtering images, big data phenomena, distributed computing, professional image processing software systems).

Some words about mathematics which is heavily used in image analysis and especially in image classification. Most chapters of *applied mathematics* are addressed, but instead of stating theorems and constructing proofs, math is often present "out of sight" and has been used "unnoticed" when constructing and fine tuning algorithms. The present overview—not for that reason alone—has somewhat *descriptive style*, which fairly highlights the whole processes of remote sensing applications. It is recommended to readers who are interested in mathematics used in detail to turn to the original publications referred in this overview.

2 Principles of remote sensing

As technology develops, *remote sensing (RS)* is continuously gaining importance: the quality of RS images is improving while unit costs are decreasing. Thus a need for handling large, sometimes enormous data sets is arising, which is mainly supported by high-speed computers. This paper concentrates on *agricultural applications* of remotely sensed images. In this section a brief summary of the related parts of remote sensing and image processing is given. For a detailed introduction to remote sensing and its applications, see e.g. the excellent books of Richards [1] or McCloy [2].

Remotely sensed images convey information about a certain area of the Earth's surface. Satellites (e.g. Landsat, NOAA, SPOT, IRS, Sentinel series) surveying our environment detect the reflected radiation in various electromagnetic wavelength intervals. These data are stored in the form of *digital image*. Figure 1 shows the area covered by a pass of the satellite NOAA-14 as seen on the screen of an acquisition program. A definite part of the whole electromagnetic

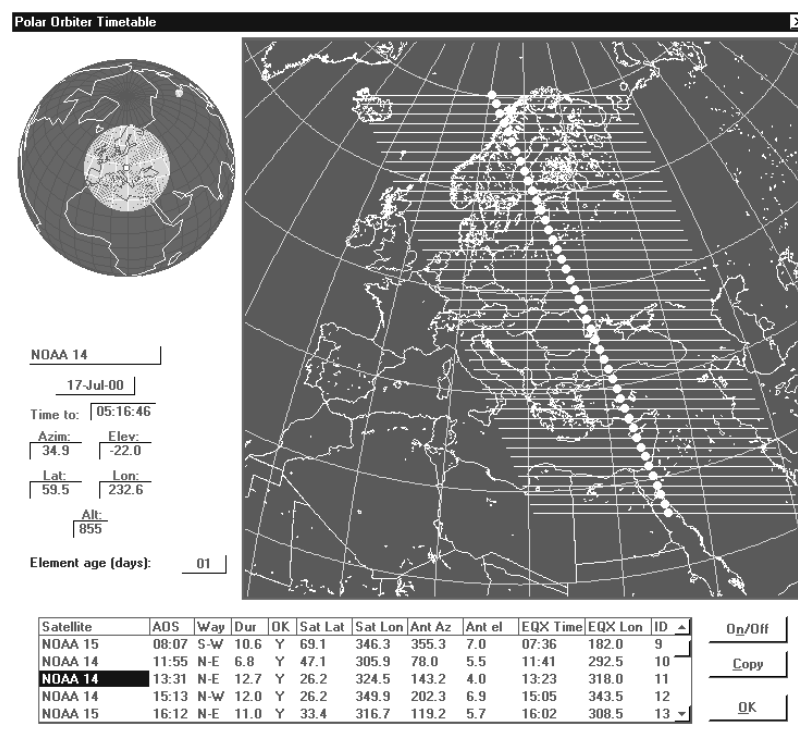


Figure 1: A pass of the satellite NOAA-14

spectrum, called optical band, is involved in RS. The radiation of the surface is measured by sensors, each capturing a given subinterval of the spectrum. A multispectral image can be regarded as a matrix, whose elements correspond to a given spot of the surface. These elements themselves, called *pixels*, are r -dimensional vectors, containing the intensity values recorded by the r sensors.

RS images are categorized by four principal parameters. *Pixel size* is the smallest distinguishable area on the surface. It varies between 1m by 1m and 1km by 1km. Most images in our practice (taken by Landsat TM and ETM+) have a pixel size of 30m by 30m. The second type of parameters are number of the *spectral bands* and the respective wavebands. The spectral bands are selected to requirements of applications. *Radiometric resolution* refers to the number of distinguishable intensity levels, yielding pixel values of 0..255. *Period of acquisition* (e.g. 16 days) is the time between the passes of a given satellite over the same area.

Before starting the analysis of images, some *preprocessing steps* are necessary. Geometric and radiometric correction transform raw images into a uniform (spatial and spectral) system, which conforms to the way the Earth is usually pictured by man. The map projection used holds the area, but does not hold the distances. Nevertheless, the length

distortion is not significant compared to the pixel size and the extent of the sample area. Effects disturbing detection and data transfer can be decreased by atmospheric correction and noise filtering.

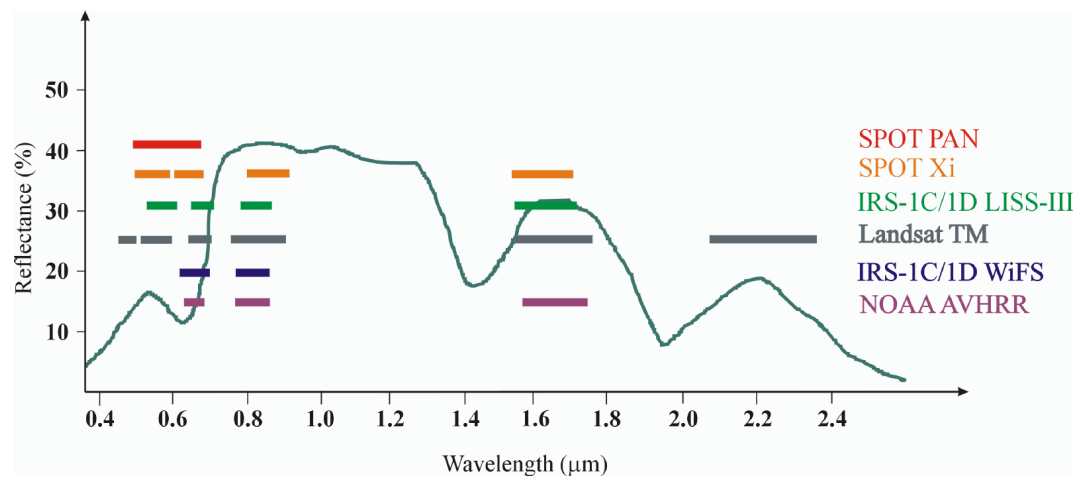


Figure 2: A typical reflectance function of vegetation and spectral bands of satellite sensors

The reflectance (i.e. the percentage of reflected radiation) of land covers heavily depends on the wavelength. The so-called *reflectance function* for a given land cover gives the reflectance along the optical band. Every land cover category has its own characteristic reflectance function, which depends on the phenological phase of the vegetation. This fact can be used to make distinction between land covers, which is a fundamental requirement of RS. Figure 2 shows the reflectance function of a typical cultivated crop. A representative sample of this function is taken by the sensors. Horizontal lines in the figure show a few examples of the spectral bands covered by some satellite sensors.

Unfortunately, there are time periods when spectral response of different crops nearly coincide. Therefore, a series of images from different dates have to be used in order to separate covers. Besides, they greatly help progress monitoring.

Straight application of determining land covers is *vegetation mapping*, which serves as a basis for area estimation. Similar technical and theoretical tools are used at flood and waterlog monitoring, or drought monitoring in another period of the year. The sophisticated task of early yield forecast can be solved with extensive use of RS image time series.

The importance of RS is obvious: less field work is required, it gives immediate results and the accuracy and reliability is higher than those of human information resources.

3 Pixel-based classification

The task of thematic classification is to produce a digital thematic map of a certain area. The pixels of a thematic image usually refer to categories of land cover (i.e. to classes). This is an important difference from satellite images, where pixel values contain the vector of measurement value: radiation intensities in spectral bands. The set of categories can be rather varying among the different applications: they can not only represent crop species, but they can show the severity of drought, and they can make distinction between the areas affected by waterlog.

The input of classification methods consists of one or several satellite images. They usually use a thematic map, called reference data, describing the parts of examined area that are known in advance. Reference data are divided into two parts: training data (training areas) are used to determine the distributions describing land cover categories,

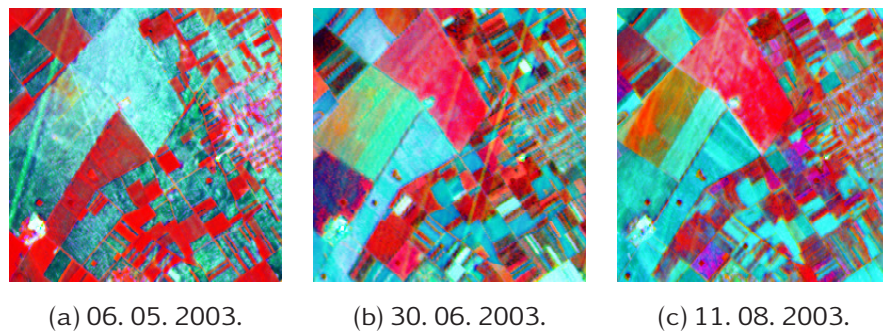


Figure 3: A satellite image series of the sample area

while test data are the basis of accuracy assessment. The complex procedure of classification is usually not completely automatic; it is often improved by the interaction of a human expert.

A *vegetation mapping* (crop mapping) application is taken as an example, where the task is to determine the crop species on agricultural areas with $25\text{ m} \times 25\text{ m}$ spatial resolution (see Csornai et al., [3]). This great early practical result was completed by an accuracy assessment (see Nádor et al. [4]).

The vegetation period and the progress of development of different crops can be rather varying throughout the year, but their reflectance often coincides in a given period. This is why images from multiple dates are necessary to create an adequate classification. Figure 3 shows a satellite image series of three dates, taken of the same area. The simplest way to involve multi-temporal data into the classification is when several images of the target area, having different acquisition dates, are “stacked together” to compose a multi-layer image.

It is a known observation in remote sensing that a large amount of pixels representing the same land cover category (e.g. crop species) show nearly normal distribution, or can be approximated with several normal distributions. The following assumption is made as the principle of traditional classification task: each thematic category (e.g. wheat) can be described with the composition of appropriate *spectral subclasses*, each subclass having multivariate normal distribution in the multi-dimensional intensity space.

The pixel-based classification method, which was somewhat intuitive, has been described more precisely by László et al. (see [6]) and later has been theoretically significantly improved (see [7]). As a consequence, deep analogy can be observed between the pixel-based and segment-based classification. Together with accuracy assesment, the classification procedure consists of four major parts.

1. The first step in determining classes is *clustering*. This is an unsupervised procedure, in which no preliminary information (reference data) is used from the target area. Clusters are compact groups of the intensity space that characterise land cover categories. During this procedure, in each iteration the cluster map shows the assignment of cluster identifiers to pixels. In the procedure, the pixels of clusters with few elements and ones being far from every cluster are not classified. At the end of procedure the mean vectors and covariance matrices (together: the signatures) of the clusters are calculated. Clusters determine the initial values of the spectral subclasses of land covers.
2. In the second step, in the *training phase* the aim is to assign a land cover category to each spectral subclass. In this step, the spectral subclasses that build up the distribution of pixels belonging to crops, as introduced above, are formed, starting from the clusters evolved in the first step. The input of this step is the original image, the map of training reference data and the cluster map. Its output is the signature set of spectral subclasses together with the assignment between them and land cover categories.

Firstly, the pixels of image are assigned again to the clusters. This yields a new “cluster map”. The assignment is done with the maximum-likelihood method using normal distribution functions. Every x pixel is classified to the cluster ω_k for which the following probability is maximum:

$$p(x|\omega_k) = (2\pi)^{-N/2} |\Sigma_k|^{-1/2} e^{-(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)/2} \quad (1)$$

In practice, instead of the distribution function itself the natural logarithm of the above expression is calculated and is used to choose the appropriate cluster for pixels, in which the quadratic form $((x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k))$ is an additive member. Following the assumption that x is a random variable with normal distribution, the quadratic form has a χ^2 distribution with N degrees of freedom. It represents the confidence of x fitting a normal distribution with parameters μ_k and Σ_k . Comparing its value to the critical value belonging to the desired significance level, one can accept or reject the matching. In the implementation described here, this test of hypothesis will be used several times.

In this step, the test is the following: if, for a given pixel, even the maximum probability is small (i.e., according to the χ^2 test of hypothesis, the pixel does not fit into the cluster with a probability of 99%), then it is not classified.

Next, the correspondence between clusters and reference areas is examined. For each cluster, the intersection of its points and the reference categories is determined. The next procedure assigns a label to every cluster:

if there is not enough reference area compared to the cluster size

then

if the cluster is large

then the cluster is *to be reclassified*

else the cluster is *to be abandoned*

else

do search for the crop that intersects the most with the cluster

if this crop does not intersect sufficiently

then the cluster is *to be reclassified*

else

do search for all the relevant crops

if there is only one relevant crop

then the cluster is *classified*

else the cluster is *to be cut*

After this decision, we remember the only relevant crop for the *classified* clusters, and all the relevant crops for the clusters *to be cut*.

3. In the *classification phase* the pixels belonging to the labelled clusters are classified into land cover categories.

Firstly, the pixels of clusters *to be reclassified* are classified with maximum-likelihood decision into one of the clusters that are *classified* or *to be cut*. The points not fitting into any cluster are abandoned.

Next, the signatures of the reference pixels belonging to the relevant crops within clusters *to be cut* are determined. The parameters of distributions assumed to be normal are estimated on the basis of the intersection of the cluster and the reference areas of relevant crops. Then the points belonging to the cluster are classified into one of these “subclusters” with maximum-likelihood decision, but at this point, the pixels not fitting into the subcluster are not abandoned.



Figure 4: The result of pixel-based processing

Finally, the pixels of *classified* clusters are assigned to the class of the only relevant crop.

In the resulting image, the majority of pixels are assigned with a class. The pixels abandoned in the initial clustering, the ones of the clusters *to be abandoned* and the ones abandoned during the reclassification fall into the unknown category, that is, they have not been classified.

4. Finally, an *accuracy assessment* is carried out with the usage of the test reference areas. In the case of high error rate, some of the previous steps are executed again with a modified parameter setting.

It has to be noted that this pixel-based classification method completely ignores the identity or similarity of the neighbouring pixels in homogeneous areas (i.e., falling into the same agricultural parcel), as it uses only the intensity of pixels. Figure 4 illustrates the result of the clustering (a) and the classification (b) for a known area.

4 Segment-based classification

In the *thematic classification of remote sensing images* it is always a key question to find the link between objects of “real world” and those of images (see [1]). To achieve proper detection of features in mapping, image objects that fit land cover objects need to be delineated and they must be classified into predefined classes. Since the birth of remote sensing, the rich spectral content of satellite images has been showing its strength in the differentiation of land cover classes. But it is not always enough to utilize only spectral information. Traditional pixel-based classification methods completely disregard spatial relations.

To overcome this drawback, *segmentation* was introduced with the aim of extracting neighborhood information and preserving natural homogeneity. Segment is a contiguous set of spectrally similar pixels. Segments form a complete disjoint coverage of an image. Segmentation methods are usually iterative, and segments in different iteration steps can be organized into a hierarchical system.

Within the wide range of segmentation methods, authors primarily deal with region-based algorithms. During the last years, gradually a fully segment-based classification framework has been designed and implemented. The interest of the authors and their community was basically influenced by Schoenmakers’ monography (see [8]).

4.1 Segmentation methods

First, the Sequential linking method ([9]) has been improved (see [5]). Afterwards, a special variant of pixel-based classification has been used for the segments obtained in previous step ([6]). Finally, a real segment-based clustering and classification procedure has been implemented ([7]). This is a modular framework, that is, all the components – segmentation, clustering, classification – can be changed. The following segmentation methods were implemented and analyzed.

- *Merge-based, “bottom-up” methods* start from pixels: in the beginning, every pixel of the image forms an individual segment. During the iterations of algorithm, segments are gradually merged.
 - *Sequential linking* ([9]) deals with the statistical homogeneity of segments. In a preparatory step, small groups of pixels (for example, 2 by 2 squares) are joined into cells. Iteration consists of row-wise traversing through cells, attempting to join the current cell to a segment. Because of sequential traversing, time requirement is a linear function of image size. It is, however, a serious drawback of the original algorithm that the resulting segments are strongly influenced by traverse order. Some improvement has been made in the above-mentioned framework (in detail see [7]). For curiosity, this method will be below explained in detail.
 - *Best merge* algorithm ([10] and [8]) overcomes the above mentioned strict traverse order: it chooses any two neighboring segments over the image if their contraction is optimal with respect to certain criteria. Best merge is a greedy algorithm. In the variant implemented by the authors and their colleagues, image is represented by a graph. The efficiency of implementation is assured by advanced disjoint-set data structures. This method will be illustrated below.
 - The idea of *graph-based merge* ([11]) comes from graph theory. Segments are characterized by their heterogeneity, which is formally defined using graph notation. In each iterative step, adjacent segments are merged. Edges are taken in the descending order of their weight, and it is decided whether the two segments belonging to the two end nodes can be contracted. The decision is based on the internal variability of segments. An efficient implementation is based on union-find data structure.
- *Cut-based, “top-down” methods* behave the opposite way. In the beginning, the whole image is considered as one large segment. In each iteration step, an appropriately chosen segment is cut into two smaller ones. All the cut-based methods the authors dealt with are developed and implemented on the basis of graph representation.
 - In the *Minimum mean cut* algorithm ([12]), the edge weights are proportional to the spectral difference between segments that are represented by the respective subgraph.
 - *Minimum ratio cut* algorithm ([13]) is an improved and generalized variant of minimum mean cut. Two weight functions are used, and criteria are defined with their ratio.
 - *Normalized cut* algorithm ([14]) is also derived from minimum mean cut, using the same weight function.

In general, images and segmentation algorithms can be represented by undirected graphs. Vertices correspond to pixels. Edges connect vertices of neighboring pixels. Weights of edges are assigned according to the relation between pixels they are connecting. Within the graph, segments can be described by subgraphs. Graph representation of cut-based algorithms subdivides edge set into disjoint subsets. This way only those edges remain in the graph

that connect pixels within segments. Our implementation of graph-based algorithms presented above is strongly supported by efficient data structures and algorithms, which resulted in the asymptotic decrease of running time. In a part of cases, this improvement has been theoretically justified as well ([15], [16]).

The result of segmentation is a thematic image, called segment map, containing the numbers of segment the pixel belongs to. As a result of earlier research and development, a fully segment-based classification method has been developed. The steps of pixel-based classification have been adapted to segments. Segmentation is followed by *clustering*, and then the final *classification* is carried out.

With the introduction of fully segment-based classification, the thematic accuracy grew to 91-95%, about 2-3% more than that of pixel-based method. Procedures are sensitive to parameters, but they less depend on the segmentation algorithm chosen. Run-time has significantly decreased, as the number of objects to be dealt with is less by magnitudes.

According to the above remarks, in the following two segmentation methods will be presented in detail.

4.2 Sequential linking: an example

The input of the algorithm for segmentation is the digital image, the parameters are the size of a cell (usually $2 * 2$) and some statistical thresholds. The output is a thematic map, called the *segment map*, which is a matrix of non-negative integer labels. A matrix element assigns a segment number to a cell, where the value of zero designates inhomogeneous cell. The result highly depends on the mentioned statistical thresholds. Currently they have to be set up manually, by the examination of the frequency function of the image intensity and density function of the used similarity statistics.

First, the image is divided into cells, as the elementary units of segmentation. Let $X = (x_1, x_2, \dots, x_n)$ represent pixels of a homogeneous cell. The cell is considered homogeneous if the estimation of its normalized deviation (c_j in the j th band) falls below a certain predefined limit, i.e.

$$\forall j(1 \leq j \leq b) : c_j = \frac{s_j}{\bar{x}_j} \leq C_H,$$

where $\bar{x}_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$, the mean in the j th band, and $s_j^2 = \frac{1}{n-1} \sum_{i=1}^n x_{ij}^2 - \frac{n}{n-1} \bar{x}_j^2$, the empirical variance. This test selects the cells with pixels intensity disperses close to the mean. On the other hand, it discards the cells with high variability in proportion to the cell mean, among them, the cells with reflectance level differing from zero non-significantly.

In the following, statistically similar, neighbouring cells will be merged into segments. Inhomogeneous cells are ignored in the segmentation steps. The annexation criterion is a hypothesis test. It takes the cells in a prescribed order, detailed in 3.2.

Outline of the annexation test. Let $Y = (y_1, y_2, \dots, y_m)$ be the pixels of a segment's current extent and X a neighbouring cell.

X can be connected to Y if the inequalities $L_1 \geq C_1$ and $L_2 \geq C_2$ hold, where C_1 and C_2 are predefined thresholds. L_1 is an ANOVA-like statistics, which measures the equality of means. L_2 measures the equality of covariances of X and Y . The comparison with the resulting L_1 tests for the hypothesis of equal mean vectors (first-order statistics), while L_2 is used for testing of equal covariances (second-order statistics). This way we assure the approximate equality of both the mean and the covariance.

L_1 and L_2 can be calculated independently. Let \bar{x} and \bar{y} mean the average of X and Y , respectively, and M denote

the new segment centre (in the spectral space), i. e. $M = (n\bar{x} + m\bar{y})/(n + m)$. A_X, A_Y, B_X and B_Y are covariance-like quantities:

$$A_X = \sum_{i=1}^n (x_i - \bar{x})^2 \quad A_Y = \sum_{i=1}^m (y_i - \bar{y})^2 \quad B_X = \sum_{i=1}^n (x_i - M)^2 \quad B_Y = \sum_{i=1}^m (y_i - M)^2$$

Using this notation, and $N = n + m - 2$, $A = A_X + A_Y$, $B = B_X + B_Y$:

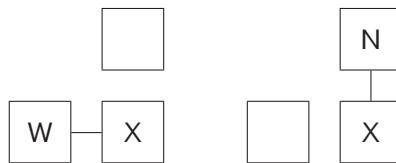
$$L_1 = (A/B)^{N/2}$$

$$L_2 = \left(\frac{(A_X/n)^{n-1} * (A_Y/m)^{m-1}}{(A/N)^N} \right)^{1/2}$$

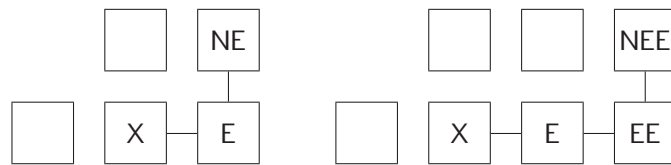
Once X is annexed to Y , their points will be treated together in the following.

Merging starts from the upper left corner of the image, proceeds row-wise, and only homogeneous cells are examined. For each new cell, attempt is made to merge it to an existing segment, represented by the western (W), northern (N), first or second north-eastern (NE or NEE) neighbouring cell. Remote sensing images of the examined area have a special texture. Majority of the agricultural parcels appear as relatively simple polygons. The elaborated method eliminate the faults caused by the order of annexation.

First the procedure tries to merge the cell to its western or northern adjacent cell. These cells have already been assigned to a segment. If the cell can be connected to both segments (and they are not the same), the one with the closer centre is chosen.



If both cases fail, the algorithm investigates one or two cells to the east to find a way to another northern segment, as described by Fekete and Farkasfalvy [5]. The annexation criterion is examined for the east or second east neighbouring cells and their northern adjacent cells. If the criterion meets, an attempt is made to merge the cell X to this segment.



If all previous attempts fail, X will start a new segment.

4.3 Best merge: illustration

An illustration of Best-merge method can be seen in Figure 5. The upper row shows a time series chosen from several sample areas used for development and testing. The rest of figure shows the result of three steps of classification process. The colors seen in the thematic images showing different phases are independent of each other.

The illustration of segment map (5.d) is a six-colored image, where neighboring segments are distinguished by different colors. During clustering, segments are merged into clusters; segments belonging to the same cluster may

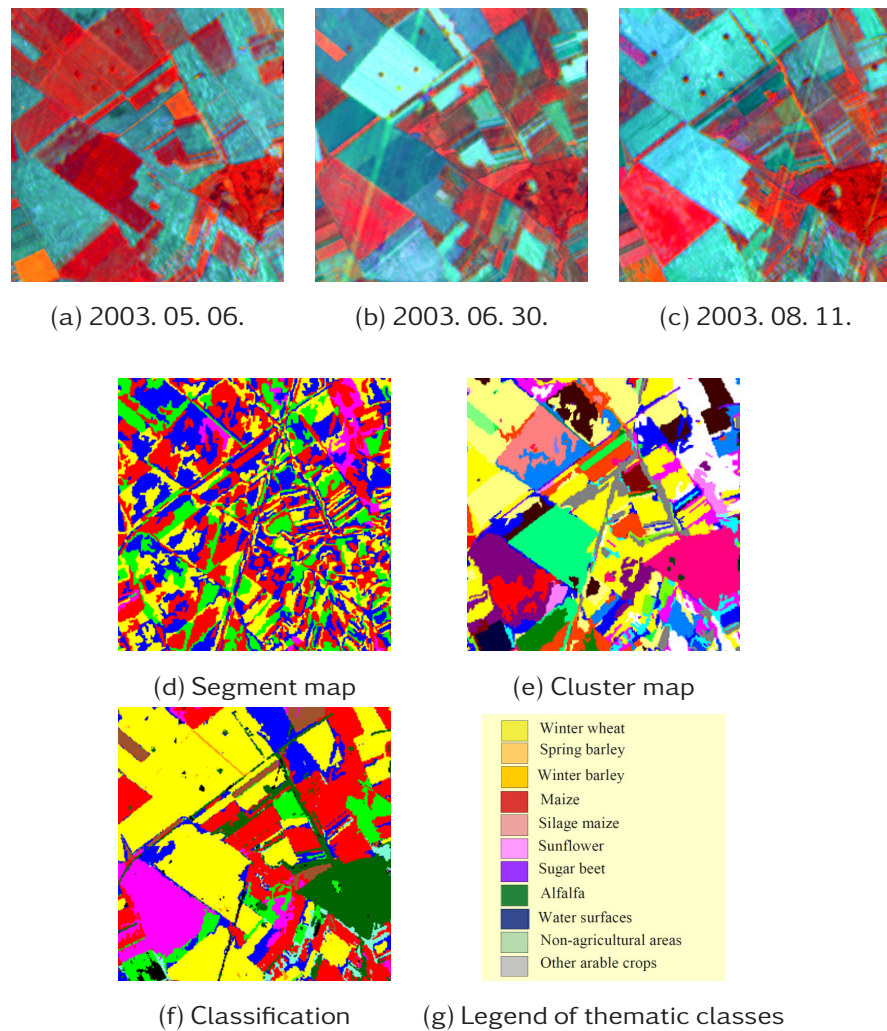


Figure 5: The illustration of Best-merge method

appear at different places of the image, depending on their statistical properties. Clusters in 5.e have randomly chosen unique colors. In the image showing classification result (5.f) thematic classes appear according to a pre-defined legend (yellow = winter wheat, red = corn etc.; see Figure 5.g).

5 Data fusion

This section will describe the general meaning of the image fusion. Then, an alternative data fusion approach will be presented, in which the time dimension is highlighted, and which plays an emphasized role in some agricultural monitoring programmes.

In the present paper, within the data fusion the authors will deal with image fusion, involving remotely sensed images, mostly satellite images. *Image fusion* refers to procedures that integrate the advantages in the spatial, spectral and temporal characteristics of several image data sources. In remote sensing, both “data fusion” and “image fusion” are used interchangeably for the same operations. Although there are general methods to use together several types of images, but the implementation – with special emphasis on the preservation of the advantageous properties of each image – highly depends on the actual inputs.

The most often used image fusion procedures are the *pixel-based fusion methods*, which deal only with the spatial and spectral properties. Their aim is to improve the spatial characteristics of a multispectral image via “fusing” or “merging” it with a higher spatial resolution (usually panchromatic) image; see Fig. 6.

Wald [17] gives a comprehensive survey on pixel-based fusion methods. The improvement can mean the increasing of the spatial resolution, or the emphasizing of the boundaries of the objects, the direction of the linear elements, or the texture. While improving the spatial properties of the multispectral image, its spectral characteristics usually have to be preserved. Some examples of the pixel-based fusion methods are the IHS (Intensity-Hue-Saturation), the PCA (Principal Component Analysis) and the WS (Wavelet Substitution) transformation. With the exception of the IHS transformation, the spectral characteristics of the multispectral image are kept.

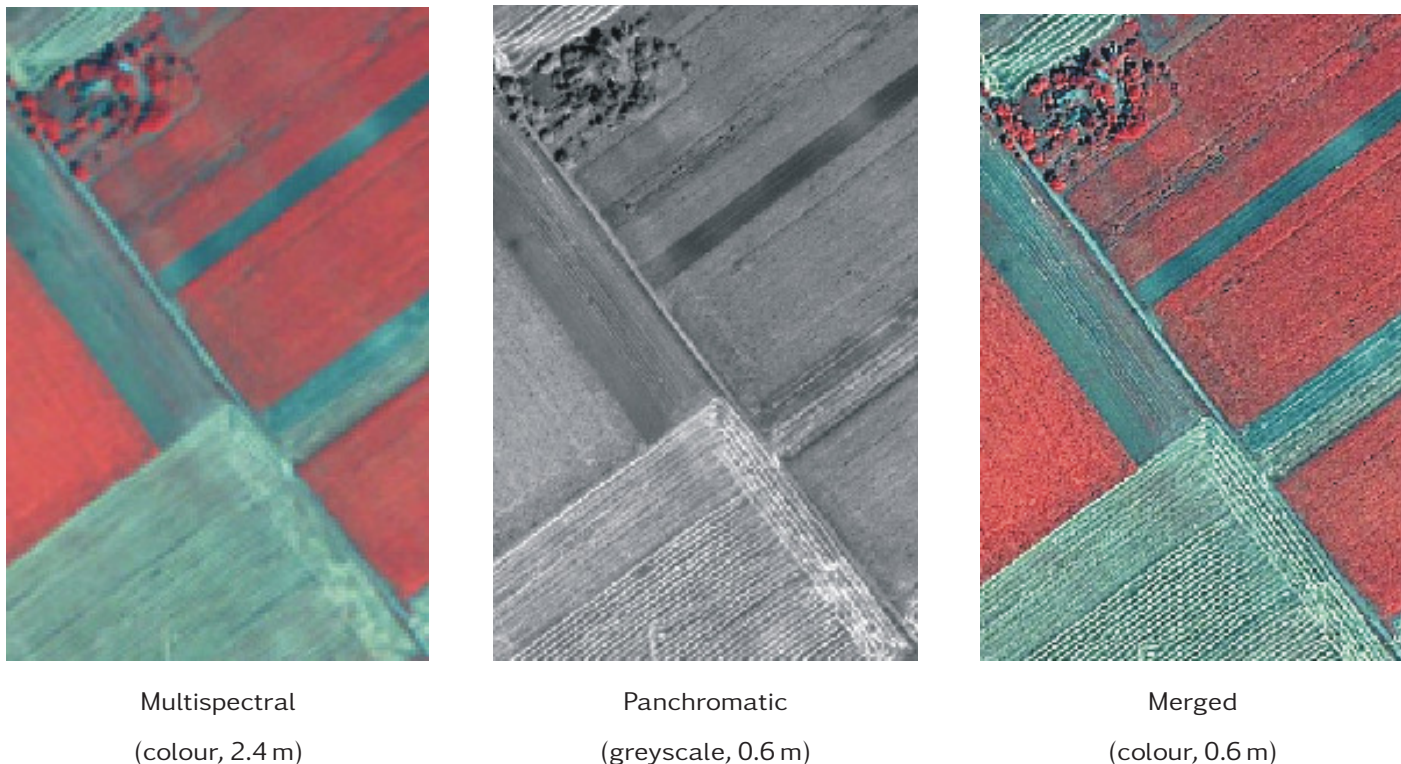


Figure 6: An example of pixel-based fusion methods

5.1 A generalized approach

The following part will give another approach to image fusion according to [18], where László et al. present three applications of data fusion (with several references). The most important feature is that the time dimension is much more involved than in the case of pixel-based fusion methods. Several satellite images – usually of different kind – are simultaneously used in an application, but it is not necessary to derive intermediate “fused” images. Instead, the usage of several images – and the difference between their characteristics – is “built in” into the model.

To give a description analogous to that of the pixel-based fusion, it can be said that the temporal enhancement of some images is done via the substitution of the weaker spatial or spectral characteristics with better temporal resolution. The usual situation is to have one or a few images with higher spatial resolution and/or better spectral characteristics and a series of images with weaker spatial or spectral properties. The latter carries the additional temporal information.

The proper incorporation of temporal information has a fundamental role in the agricultural applications, or more specifically, in the crop development monitoring.

For the differentiation among crops it is only rarely enough to have one image. Instead, the temporal progress has to be examined in the different wavebands, within the year. For this purpose, several images, taken in proper time periods, are needed. For example, in a dominantly agricultural area, usually one or two images are enough to make

a distinction between winter and summer crops. But if one wants to differentiate among the crop species, several spring and early summer images have to be used for the winter crops, and late summer images are needed as well for summer crops.

The crop development assessment is also carried out via the examination of the temporal progress within the year, but more frequent acquisitions are necessary for this purpose, and instead of the measured values in the different wavebands, mainly a vegetation index is used. In some applications, the investigation is not bounded to one year; instead, it is vital to compare the status of the vegetation between different years. In this case, a data set is needed that is coherent over several years.

Beside crop monitoring and production forecast there are several projects that are related to agricultural subsidies. The disaster (flood, waterlog and drought) monitoring, despite its autonomous competency, is strongly connected to mentioned programmes.

5.2 Disaster monitoring: an application

An important group of monitoring applications is *disaster monitoring*. In Hungary, several natural disasters happened in the past years due to extreme weather. Therefore, flood, waterlog and drought monitoring are very important applications. These phenomena can also be monitored by remote sensing, utilizing the advantages of different optical and radar images.

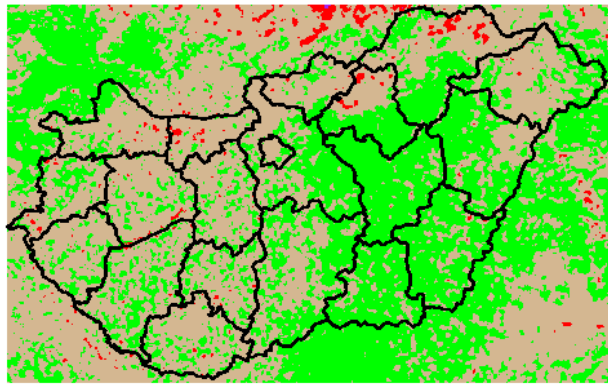
Flood monitoring and prevention is important in Hungary as 95% of its surface water flows in from abroad. In the flood monitoring, the task is to follow the extent of the inundation and to forecast its progress. In the operational applications carried out by FÖMI, this task was mainly accomplished by visual evaluation. The time dimension has dual role in this application. Firstly, frequent acquisitions are needed to continuously monitor the flood situation. Secondly, the images have to be available in short time after the acquisition.

In the previous practice of FÖMI, all the possible and appropriate satellite images were used, including thermal and radar images. The importance of temporal dimension supercedes that of spectral properties (open water surfaces are well recognizable in most kinds of images) and of spatial resolution (the needed area accuracy is satisfied by any pixel size in the usual range).

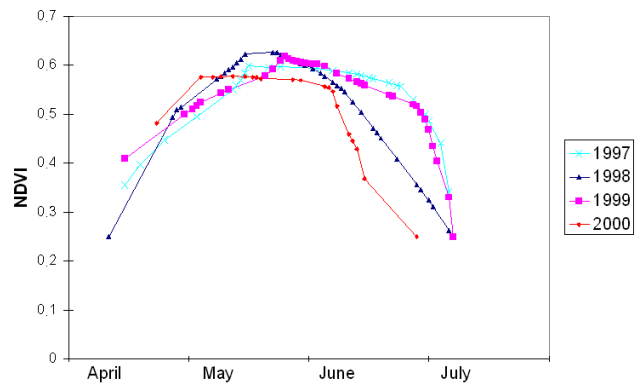
Drought monitoring was carried out in several years when drought seriously affected the growth of vegetation. This is often a retrospective monitoring, not a forecast. The remote sensing drought monitoring uses numeric, quantitative methods. The calculation is based on a vegetation index. Its physical background and implementation is similar to that of crop development monitoring, but in a “larger scale”. It gives the general status of the vegetation, and it is not specific to the individual crop species. Occasionally, the elementary mapping unit of the monitoring is larger than the pixel size of the satellite images used. Temporal dimension is also used in a “larger scale”, which means measurement values are examined over longer time periods (usually over weeks or decades). Although an appropriately dense time series is required for monitoring, the pixel values belonging to different days are not treated individually; instead, they are summarized over a certain time period.

The basis of the drought monitoring is the comparison of status of the vegetation at different dates – either between different periods of the same year, or between different years, as seen in Fig. 7. In general, remote sensing evaluates the status of the vegetation via quantities derived from physical measurements. In the specific case of development assessment and drought monitoring, this evaluation is rather sensitive to the proper knowledge of the correspondence between physical properties and values stored in the images. This is particularly true when different kinds of

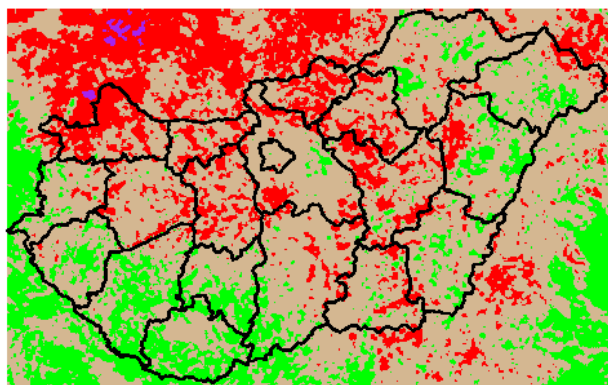
images are used. The derivation of compatible data sets from different sensors is called inter-calibration, and can be viewed as a case of image fusion. With proper usage of several sensors, more dates can be utilised in the examined periods, and the scope of the monitoring can be extended to several years – another issue of the temporal dimension.



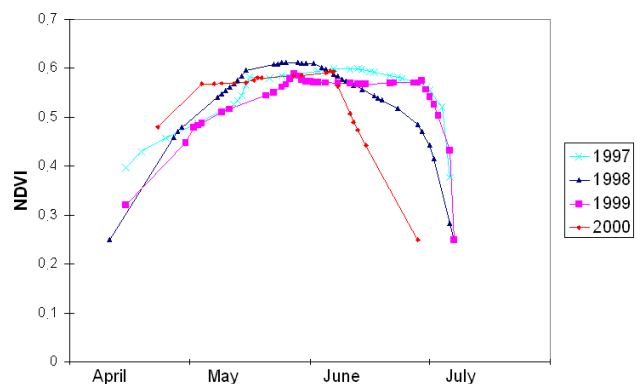
1991



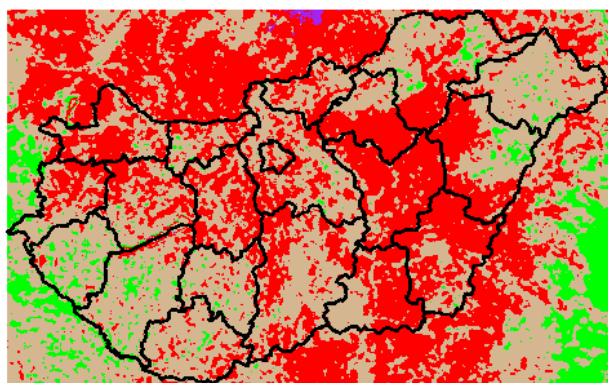
Jász-Nagykun-Szolnok



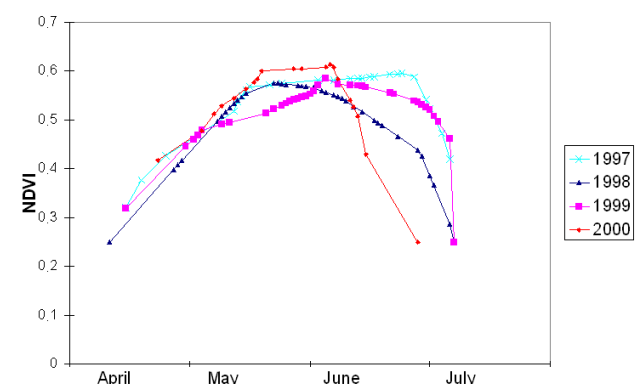
1993



Hajdú-Bihar



2000



Tolna

Figure 7: Drought monitoring: the drought map of Hungary in three different years (left) and the time series of a vegetation index in three counties (right). On the maps, lighter shades indicate favorable crop development conditions, while darker shades indicate drought. On the charts, higher values of the vegetation index (NDVI) refer to more favorable crop development. The serious drought in year 2000 can be obviously seen in the figure.

6 Object-based image analysis

Very high resolution satellite images and ortho-photos gain increasing importance in current operational applications. Individual pixels of these images usually cannot be interpreted in themselves. Therefore, the application of segmentation is not only an option, but a necessity in the solution. Recently a new paradigm, called *Object-based Image*

Analysis (OBIA), has emerged (see e.g. [19] and [20]). Its aim is to partition remote sensing images into meaningful objects, and assessing their spatial, spectral and temporal characteristics. Its basic step is segmentation. It includes the attribution of segments, that is, the assignment attributes that usually express geographic information, e.g. textural, shape and neighborhood properties. A higher level objective of OBIA is the replication of human interpretation. This way work processes may be automatized, and with the reduction of subjectivity, results become more repeatable. OBIA suits the trend of object-oriented approach in software technology.

There are several recent projects which can be regarded as OBIA applications. These are:

- the task of delimiting tree groups and scattered trees in pastures (which will be presented below in detail),
- the surveying of red mud spill, an industrial disaster,
- the recognition of ragweed and
- the recognition of built infrastructure in rural areas (which be mentioned next for an interesting effect).

These applications are tightly connected to projects of FÖMI. In the operational processing chain, either pixel-based classification or visual interpretation is used to recognize and delineate land cover categories. In our research, the target is to provide a useful solution using object-based image analysis. These results have been partially presented in [21].

6.1 Identification of ineligible land

One of the largest tasks of FÖMI is the operation and *updating of the Land Parcel Identification System (LPIS)*. In Hungary, this is the exclusive reference system of area-based EU subsidies since 2004, in which all the agricultural parcels can be identified. As cultivation structure continuously changes, LPIS must be regularly updated to reflect current status. The basis of updating is the ortho-photo coverage, which is renewed in every year for about one fourth or one third of the country.

The basic areal unit is called physical block, which is bounded by borders stable in time. Beside block boundaries, LPIS contains many geoinformatic coverages, called *thematic layers* (e.g., Less Favored Areas, Nitrate Sensitive Areas, layers related to Agri-Environmental Measures). In LPIS, an important property of areas is whether they are *subsidized (eligible for agricultural payments)*. The delineation of eligible and ineligible areas within physical blocks is also stored as a thematic layer.

The goal of application is *to automatically delineate (ineligible) scattered trees and bushes* appearing on (otherwise eligible) pastures. This task cannot be correctly solved with pixel-based classification, as the land cover units (trees, bushes, spots of pasture) are significantly larger than pixels. Furthermore, the differentiation cannot be made based solely on the intensity values of pixels; neighborhood information must be taken into account as well. The half meter spatial resolution of ortho-photos provides great geometrical accuracy. However, local spectral properties make classification harder, as it is difficult to distinguish between objects belonging to tree groups and similar segments actually belonging to other types of land cover. In some cases the question cannot be decided even by visual interpretation. Continuous effort is made to automatize the task of delineation. Segment-based classification was proven to be appropriate in the majority of cases examined.

To solve the delineation task, the eCognition software suite is applied. It contains several built-in proprietary segmentation procedures, which are not exactly the same that were presented in the last section. But the theoretical

knowledge of segmentation and the experience gained with their implementation greatly helps in the proper use of this software suite.

Input consists of raster images. For the delineation of scattered trees, color infrared (CIR) ortho-photos with sub-meter resolution are applied. Besides, very high resolution (VHR) satellite images can also be used. The main output is a thematic vector (shapefile) with the classification results, which can be used in other GIS programs.

The software eCognition deals with a hierarchical system of images and image objects. Segmentation and classification steps are executed to derive objects from the level of pixels. Depending on the complexity of task, it may consist of several steps. The eCognition suite provides plenty of procedures built from elementary algorithms. Their choice and parameterization requires the theoretical knowledge of segmentation and photogrammetry. Publicly known and proprietary segmentation algorithms have been implemented under the names Chessboard, Quadtree, Contrast Split, Spectral Difference and Multi-resolution etc. Classification takes segments as input. A plentiful system of sophisticated criteria is used for classification. Beyond usual statistical measures, geometrical and textural features, membership functions in class hierarchy and statistical distributions are available. Commands and functions are organized into so-called Rule Sets.

When processing large images, eCognition is able to apply the “divide and conquer” approach: the image is split into smaller, equal-sized tiles, which may be processed in parallel; finally the results of tiles are “stitched” together.

The following segmentation algorithms were applied during the delimitation of pastures with trees: Quadtree-based, Multiresolution, Spectral difference and Contrast split segmentation. Evaluations carried out on several test areas have proven that the segment map obtained in these steps is appropriate for the further examinations. It is difficult to define the properties of target categories, that is, vegetation with trees in our case. Spectral properties are significantly influenced by tree species, vegetation density and illumination. However, we can address these difficulties with the possibilities of OBIA.

First of all, we assume that the geometric and texture properties of vegetation to be detected (trees and bushes) are different from the other kinds of vegetation (primarily arable land, grassland and other natural land cover types). This assumption is justified by preliminary visual examination. The texture of foliage of trees is less homogeneous and less structured than that of other vegetation. Besides, one can formulate other geometrical criteria, primarily in connection with the size of objects.

Texture is measured in two ways in this application. Inhomogeneity is simply characterized by the spectral deviation of segment. Besides, several GLCM-based measures (homogeneity, entropy) are examined. GLCM (Grey Level Co-occurrence Matrix) is a square matrix. Its height and width is equal to the number of grey levels in the spectral band to be characterized (radiometric resolution). In the simplest case, a P_{ij} element of the GLCM matrix P is the number of neighboring pixel pairs in the image having value of i and j , respectively. Instead of direct neighbors, pixels situated at a given distance can also be used. After counting pixel pairs, GLCM matrix is made symmetric by adding its transposed matrix to it, and finally it is normalized by the number of elements. This way the elements of GLCM constitute an empirical probability distribution.

Texture is characterized by the statistics derived from GLCM. *Entropy* measures the disorder, or in other words, unpredictability or randomness of texture. Considering the elements of GLCM as probabilities, entropy is calculated in the same way as in information theory. *Homogeneity* measures the closeness of elements in the GLCM to the GLCM diagonal. Its value is high if the non-zero elements of the matrix are situated in the proximity of diagonal.

It is not the aim to make distinction between tree species. The definition of target categories is based on spectral

properties extracted from samples of different areas with trees, independently of species. Classification is based on maximum-likelihood decision. Results can be improved using geometric properties. A part of them is defined by the parameters of application (minimum area to be delimited, the handling of surrounded areas). The other parts are related to image properties (surrounded shadows in foliage, local inhomogeneities within contiguous areas).

Some steps of the classification process on a sample area is shown in Figure 8. In the picture showing final classification results (8. d.), yellow color indicates segments classified as trees based on spectral and textural information, while in the case of segments marked with cyan, only geometrical properties were taken into account.

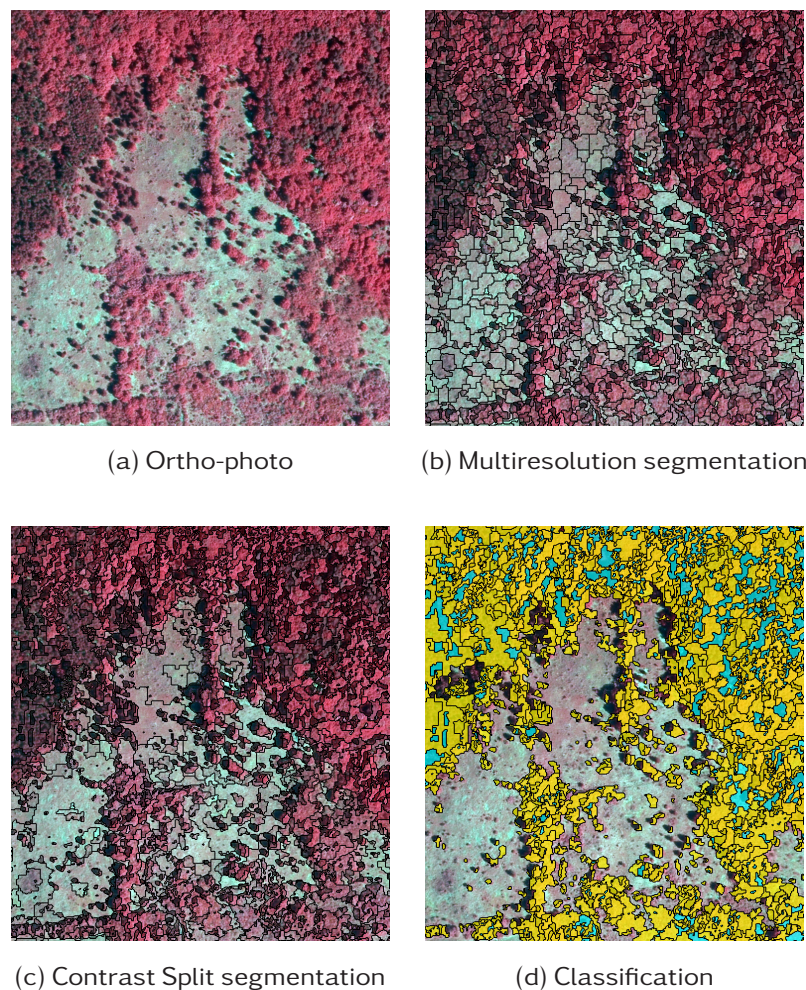


Figure 8: The steps of delineation of tree groups

6.2 The recognition of built infrastructure in rural areas

Beside agricultural and disaster monitoring tasks, *the surveying of rural or built environment* is also an important application of remote sensing. Different kinds of monitoring tasks – for example, the examination of expansion and *re-structuring of cities*, ground tasks and aboveground mining – can be accomplished by using remote sensing data of appropriate resolution.

In this subsection the object-based analysis of ortho-photos in rural monitoring will be introduced. In order to achieve accurate evaluation results at the spatial resolution of ortho-photos (0.5 m pixels), auxiliary information is needed to complement spectral information; for example, LIDAR measurements must be used.

Using OBIA methods, it is vital to carry out appropriate segmentation. In rural environment, spectral information is not always enough for the proper classification of some kinds of objects. For example, roads, concrete surfaces and buildings with grey roof are difficult to distinguish. In the case of vegetation, classification is supported by several

indices like NDVI, but there is no such tool in the classification of infrastructure. On the other hand, geometry provides useful help in OBIA. It is a basic requirement that segment map shall preserve geometric properties, spatial relations of “real world” objects.

Similarly to the processing of pasture with trees (presented in 6.1), segment map is derived in four steps. Quadtree-based and multiresolution segmentation is followed by two runs of spectral difference segmentation, using different parameters. As the result, initial objects are formed.

In the classification step, the principle is the tracking of geometric properties of surface objects. Compound geometric measures are primarily used in the recognition of road and railroad networks. While radiometric properties of roads and roofs are rather similar, the geometric measure *density* effectively makes a distinction between them. The density measure of the segment emphasized with red in Figure 9 (a) is significantly greater than that of other, spectrally similar segments; this makes it unambiguous that this segment belongs to a road.

Density describes the distribution of the pixels of a segment in space. It is calculated by the number of pixels forming the image object divided by its approximated radius. According to this calculation, the most “dense” shape is a square, while elongated segments have lower density ([22]).

The usage of this measure is made difficult by shadows, vehicles and even road markings, as they occasionally separate segments. Another difficulty is that buildings in a row may behave geometrically like roads. This is illustrated in Figure 9 (b), where the roof of the apartment building (with low density) has been contracted into one segment with a part of the neighboring road. These difficulties may be overcome by the usage of accurate height data (for example, LIDAR or stereo evaluation).



Figure 9: Segmentation of rural environment

- [1] Richards, J. A., *Remote Sensing Digital Image Analysis: an Introduction (5th ed.)*, Springer, 2013.
- [2] McCloy, K. R., *Resource Management Information Systems: Remote Sensing, GIS and Modelling (2nd ed.)* Taylor&Francis, 2006.
- [3] Csornai, G., Dalia, O., Farkasfalvy, J., Nádor, G., Crop inventory studies using Landsat data on large area in Hungary. In: *Applications of Remote Sensing in Agriculture*, 159–165, Butterworths, 1990.
- [4] Nádor, G., Csornai, G., Kocsis, A., Methods of accuracy assessment of thematic vegetation mapping by remote sensing. In: *7th Seminar on Earth and Meteorological Observations by Satellites*, 198–205, Budapest, 1997 (in Hungarian)
- [5] László, I., Nádor, G., Fekete, I., Csornai, G. and Kocsis, A., A Segment-based Classification Method for Satellite Images, in: Kovács, E. and Winkler, Z. (Eds.) *Proceedings of the 5th International Conference of Applied Informatics (ICAI)* (Eger, 2001), 151–163.
- [6] László, I., Pröhle, T., Fekete, I. and Csornai, G., A Method for Classifying Satellite Images Using Segments, *Annales Univ. Sci. Budapest, Sectio Computatorica*, **23** (2004), 163–178.
- [7] László, I., Dezső, B., Fekete, I. and Pröhle, T., A Fully Segment-based Method for the Classification of Satellite Images, *Annales Univ. Sci. Budapest, Sectio Computatorica*, **30** (2009), 157–174.
- [8] Schoenmakers, R., *Integrated methodology for segmentation of large optical satellite images in land applications of remote sensing*, Joint Research Centre, 1995.
- [9] Kettig, R. L. and Landgrebe, D. A., Classification of Multispectral Image Data by Extraction and Classification of Homogeneous Object, *IEEE Transactions on Geoscience Electronics*, **14**(1) (1976), 19–26.
- [10] Tilton, J. C., Image segmentation by Iterative Parallel Region Growing and Splitting, in: *Proc. of the International Geoscience and Remote Sensing Symp. (IGARSS89)*, (1989), 2235–2238.
- [11] Felzenszwalb, P. F. and Huttenlocher, D. P., Efficient Graph-Based Image Segmentation, *Int. J. Comput. Vis.*, **59**, no. 2 (2004), 167–181.
- [12] Wang, S. and Siskind, J. M., Image Segmentation with Minimum Mean Cut in: *8th IEEE International Conference on Computer Vision*, 1 (2001) 517.
- [13] Wang, S. and Siskind, J. M., Image Segmentation with Ratio Cut *IEEE Trans. Pattern Anal. Mach. Intell.* **25**, no. 6, 675–690.
- [14] Shi, J. and Malik, J., Normalized Cuts and Image Segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.*, **22**, no. 8 (2000), 888–905.
- [15] Dezső, B., Giachetta, R., László, I. and Fekete, I., Experimental study on graph-based image segmentation methods in the classification of satellite images, *EARSel eProceedings (ISSN 1729-3782)*, **11**(1) (2012), 12–24.
- [16] Dezső, B., Optimization methods in remote sensing and geoinformatics, Ph.D. Dissertation (Supervisor: Dr. István Fekete), Eötvös Loránd University, Faculty of Informatics, Doctoral School of Informatics, Budapest, 2012.
- [17] Wald, L., *Data Fusion: Definitions and Architectures - Fusion of images of different spatial resolutions*. L'Ecole des Mines de Paris, 2002.
- [18] László, I., Csornai, G., Fekete, I., An Alternative Approach to Data Fusion in Remote Sensing, In: *Proceedings of the 7th International Conference on Applied Informatics, Eger, 2007*. (Vol.2. pp. 391-401)
- [19] Hay, G. J. and Castilla, G., Object-based Image Analysis: Strengths, Weaknesses, Opportunities and Threats, *1st International Conference on Object-based Image Analysis* (Salzburg, 2006)
- [20] Blaschke, T., Lang, S. and Hay, G. (Eds.), *Object-Based Image Analysis, Spatial Concepts for Knowledge-Driven Remote Sensing Applications, Series: Lecture Notes in Geoinformation and Cartography*, Springer, 2008, ISBN 978-3-540-77057-2
- [21] László, I., Ócsai, K., Gera, D., Giachetta, R. and Fekete, I., Object-based Image Analysis of Pasture with Trees and Red Mud Spill, in: *Proceedings of the 31th EARSel Symposium* (Prague, 2011)
- [22] *Definiens eCognition Developer 8, Reference Book*, Definiens AG, München (2009)

The problem of distinguishing prime numbers from composites, and of resolving composite numbers into their prime factors is one of the most important and useful in all of arithmetic. The dignity of science seems to demand that every aid to the solution of such an elegant and celebrated problem be zealously cultivated.

Karl Friedrich Gauss

Suppose that the cleaning lady gives p and q by mistake to the garbage collector, but the product pq is saved. How to recover p and q ? It must be felt as a defeat for mathematics that the most promising approaches are searching the garbage dump and applying memo-hypnotic techniques.

Hendrik W. Lenstra Jr.

1 Introduction

In October 2009 the 100000 dollar prize of EFF (Electronic Frontier Foundation) was handed over to GIMPS (Great Mersenne Prime Internet Search) for being the first in presenting explicitly a prime consisting of at least 10 million decimal digits. This prime was found in August 2008, it is $2^{43112609} - 1$, and has 12978189 digits.

The next goal is a 100 million plus digit prime which will be rewarded by 150000 dollars. The present prime record is $2^{74207281} - 1$, discovered in January 2016 by GIMPS, it has 22338618 decimal digits, while the above mentioned prize winner was degraded to a bronze medalist by now. Each of the ten largest known primes is of the form $2^k - 1$, where k itself is a prime number.

Why is it so hard to exhibit big primes, and what are they good for? Why are all record-holders of this special form, and what do we know about this type of primes, called Mersenne primes? Why is GIMPS so effective in finding them?

We shall organize the lecture as follows. In Section 2 we summarize the basic facts about Mersenne primes, their connection to perfect numbers, and present an effective algorithm for their search, used also by GIMPS. We shall have a glimpse on their brothers, the Fermat primes in Section 3. In Section 4 we deal with general algorithms for deciding whether a big number is prime or composite (primality testing), and in Section 5 we have a few words on how to find the factors of a big composite number (prime factorization). We shall see that we have quick algorithms for primality testing but by our present knowledge we do not have efficient algorithms for factoring large composite numbers. This discrepancy serves as a base for the most widely used public key cryptosystem, the RSA-scheme, which will be discussed in Section 6, followed by some exercises in Section 7.

2 Mersenne primes

These are primes of the form $M_p = 2^p - 1$. Here p must be a prime, since if m is composite, then also $2^m - 1$ is composite: using the identity

$$a^n - b^n = (a - b)(a^{n-1} + a^{n-2}b + a^{n-3}b^2 + \dots + ab^{n-2} + b^{n-1})$$

for m composite, i.e. $m = rs$, $r > 1$, $s > 1$ we have

$$2^m - 1 = 2^{rs} - 1 = (2^r)^s - 1^s = (2^r - 1)((2^r)^{s-1} + (2^r)^{s-2} + \dots + 1)$$

yielding non trivial divisors since $r > 1$ and $s > 1$.

The Mersenne primes are closely connected to the perfect numbers. A positive integer was called perfect by the ancient Greeks, if it equals the sum of its (positive) divisors, excluding itself. E.g. $6 = 1 + 2 + 3$ or $28 = 1 + 2 + 4 + 7 + 14$ are perfect. Around 300 BC Euclid gave a recipe to find perfect numbers (see Elements, Theorem IX.36): “If starting from the unity we form a geometric sequence of quotient 2 till we get a sum being a prime, then the sum multiplied by the last term yields a perfect number.” Indeed, if $1 + 2 + 4 + \dots + 2^{k-1} = 2^k - 1$ is a prime q , then the positive divisors of $n = 2^{k-1}(2^k - 1) = 2^{k-1}q$ less than n are 2^j for $0 \leq j \leq k-1$ and $q2^j$ for $0 \leq j \leq k-2$, and summing these up we have

$$2^k - 1 + q(2^{k-1} - 1) = 2^k - 1 + (2^k - 1)(2^{k-1} - 1) = (2^k - 1)2^{k-1} = n.$$

It took 2000 years till Euler proved a partial converse: If an **even** number n is perfect then it must be of the form $n = 2^{p-1}(2^p - 1)$ where $2^p - 1$ is a (Mersenne) prime. For the proof consider an even perfect number n in the form $n = 2^{k-1}v$ with v odd and $k > 1$, and apply the simple formula $\sigma(ab) = \sigma(a)\sigma(b)$ for any relatively prime integers a and b , where $\sigma(t)$ denotes the sum of all positive divisors of t , including t itself. By this notation n is perfect iff $\sigma(n) = 2n$. Hence

$$2^k v = 2n = \sigma(n) = \sigma(2^{k-1}v) = \sigma(2^{k-1})\sigma(v) = (2^k - 1)\sigma(v).$$

Subtracting $(2^k - 1)v$ we can rearrange it as $v = (2^k - 1)(\sigma(v) - v)$ which means that $\sigma(v) - v$ is a divisor of v and it is not equal to v (since $k \geq 2$), hence the sum of all divisors $\sigma(v) \geq v + (\sigma(v) - v) = \sigma(v)$. Therefore equality must hold, hence v has just one divisor less than v , i.e. $1 = \sigma(v) - v$, $v = 2^k - 1$ and v is a prime, which proves the claim.

Euler’s theorem is very probably a complete converse of Euclid’s theorem, since it is widely conjectured that there do not exist odd perfect numbers. But even for even perfect numbers this necessary and sufficient criterion does not seem to be satisfactory, since we do not know for which (prime) exponents p will $q = 2^p - 1$ be a Mersenne prime. It is unsolved whether there exist infinitely many Mersenne primes (i.e. infinitely many even perfect numbers), this is one of oldest unsolved problems in mathematics, though the problem itself can be well understood even by a small child. Erdős told several times that “this is probably the most difficult, though not the most important problem mankind is facing.”

Marin Mersenne (1588–1648) was a friar and had a widespread correspondence with the leading scientists of his era. He too was interested in this type of primes in order to find perfect numbers. He knew that it is hard to determine whether a big integer is prime or composite. He wrote in 1644: “To tell if a given number of 15 or 20 digits is prime or not, all time would not suffice for the test, whatever use is made of what is already known.”

On the other hand, he still gave a list of much larger prime numbers in the same(!) book: $2^p - 1$ is prime for $p = 2, 3, 5, 7, 13, 17, 19, 31, 67, 127, 257$ and for no other p less than 257. For centuries nobody knew whether the list was correct or not. The first error was discovered in 1876(!) by É. Lucas: $2^{67} - 1$ is composite. The other errors are: $2^{61} - 1$ is prime (Pervoushin 1883), $2^{89} - 1$ and $2^{107} - 1$ are primes (Powers, 1911,1914), and finally $2^{257} - 1$ is composite (Kraitchik 1922).

It is not clear what principles led Mersenne to create his list, especially concerning the primes he included. For some of the missing ones he might have had some idea why not to insert them. E.g. he could have been aware that the prime 43 should not belong to the list, for the following reasons. Assume that there exists a prime r dividing $2^{43} - 1$, how could we find it. Using the elementary properties of congruences this means that $2^{43} \equiv 1 \pmod{r}$, hence the order $o_r(2)$ of 2 modulo r must divide 43. As the only divisor of 43 greater than 1 is 43 itself, we have that $o_r(2) = 43$. Using Fermat’s Little Theorem this implies $43 \mid r - 1$, thus r is of the form $r = 43u + 1$, and since u must be even, $r = 86w + 1$. The first such primes are 173 and 431, and checking the latter we find that it divides $2^{43} - 1$, which means that $2^{43} - 1$

is not a Mersenne prime. (Much longer calculations were quickly and happily done at that time by hand in lack of computers.)

Another interesting fact is that Lucas proved the compositeness of $2^{67} - 1$ without exhibiting any factors of it. The factorization $193707721 \cdot 761838257287$ was found only in 1903(!) by F. N. Cole who spent three years of Sundays to cope with the problem (remember, he had to work by hand without computers, since these were invented half a century later). These numbers are tested by computers using the slightly improved version of Lucas' method, the so called Lucas–Lehmer-test. Put $a_1 = 4$ and $a_{n+1} = a_n^2 - 2$ and let $p > 2$ be a prime. Then $M_p = 2^p - 1$ is prime if and only if $M_p \nmid a_{p-1}$. This test is very efficient, since it is a necessary and sufficient condition, and can be implemented very well on computers. Of course, the numbers to be tested are so big nowadays, that it is a major problem how to handle them even on a supercomputer at all. GIMPS made the innovation by making possible to perform the elements of the testing procedure parallelly on thousands of home PC-s in their idle time, and thus now more than 200000 private persons collaborate with their computers in searching new Mersenne primes. (You can join them via the home page of GIMPS.)

The list of the presently known 49 Mersenne primes is: $2^p - 1$ where $p = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209, 44497, 86243, 110503, 132049, 216091, 756839, 859433, 1257787, 1398269, 2976221, 3021377, 6972593, 13466917, 20996011, 24036583, 25964951, 30402457, 32582657, 37156667, 42643801, 43112609, 57885161, 74207281$. As mentioned in the introduction, the last number, $2^{74207281} - 1$ is the biggest known prime, it has 22338618 decimal digits!

3 Fermat primes

These are primes of the form $F_n = 2^{2^n} + 1$. The special form of the exponent is explained by the fact that $2^m + 1$ cannot be prime if m is not a power of 2 (see Exercise 6).

Pierre Fermat (1601–1665), lawyer and “amateur” mathematician thought that F_n is prime for every $n \geq 0$. For $n \leq 4$ this is true (these primes are 3, 5, 17, 257 and 65537), but Euler disproved it for $n = 5$ by showing that $641 \mid F_5 = 2^{32} + 1$.

Today we know that F_n is composite for $5 \leq n \leq 32$ and also for some larger values of n . The record is $F_{3329780}$ (with more than $10^{1000000}$ decimal digits!) having a factor $193 \cdot 2^{3329782} + 1$. No other Fermat primes were found than the F_n with $n \leq 4$. We have no information about F_{33} . No factors of F_{20} or F_{24} are known (though they are known to be composite).

The Fermat numbers are tested by Pepin's test (see Exercise 7): Let $n \geq 1$. Then F_n is prime if and only if $3^{(F_n-1)/2} \equiv -1 \pmod{F_n}$.

To find factors of Fermat numbers, the following theorem is helpful: Any (prime) divisor r of F_n must be of the form $r = 2^{n+1}t + 1$ (see Exercise 8), moreover for $n \geq 2$ of the form $r = 2^{n+2}s + 1$.

The Fermat primes are related to the construction of the regular polygons using compass and ruler. Gauss' theorem states that a regular N -gon can be constructed if and only if $(3 \leq) N = 2^\alpha p_1 \dots p_r$, where $\alpha \geq 0, r \geq 0$, and p_i are distinct Fermat primes. These values of N are characterized by the property that $\varphi(N)$ is a power of 2 where the Euler function $\varphi(N)$ denotes the number of positive integers not greater than N and coprime to N . The first few values are $N = 3, 4, 5, 6, 8, 10, 12, 15, 16, 17, 20, \dots$

4 General primality tests

The simplest quick test is based on Fermat's Little Theorem: if p is a prime and c is not a multiple of p then $c^{p-1} \equiv 1 \pmod{p}$. Hence for a big odd number N we can try to determine its compositeness by computing the remainder of $2^{N-1} \pmod{N}$. If this remainder is not 1 then N must be composite, and this brings to light the compositeness of "most" composite numbers (see below). The test is quick if we compute the powers via repeated squarings and reduce the result mod N in every step. E.g. to compute c^{78} we compute $c^2, c^4, c^8, c^{16}, c^{32}$ and $c^{64} \pmod{N}$, and then $c^{78} = c^{64}c^8c^4c^2$. This algorithm requires at most $2\log_2(N-1)$ steps to compute 2^{N-1} which is proportional to the number of data as binary digits in the input of N .

What to do if we get $2^{N-1} \equiv 1 \pmod{N}$? We cannot be certain that N is a prime, e.g. $2^{340} \equiv 1 \pmod{341 = 11 \cdot 31}$ (see Exercise 9), hence 341 is a pseudoprime for the base 2, a composite number which imitates Fermat's Little Theorem for $c = 2$. We can try to evaluate $3^{N-1} \pmod{N}$, and in fact $3^{340} \not\equiv 1 \pmod{341}$, hence the test revealed the compositeness of 341 (i.e. 341 is not a pseudoprime for the base 3). There exist, however, infinitely many universal pseudoprimes, i.e. composite numbers N for which $c^{N-1} \equiv 1 \pmod{N}$ for every $(c, N) = 1$, e.g. $N = 1729$ (see Exercise 10) (and generally it is hopeless to find a c not coprime to N).

Still, we can use Fermat's Little Theorem as a probabilistic primality test, in various senses.

(I) It can be proven that the number of pseudoprimes for the base 2 (and thus also the number of universal pseudoprimes) up to some big K is much smaller than the number of primes till this limit. E.g. for $K = 10^9$ there are 5597 pseudoprimes for the base 2, only 646 universal pseudoprimes, whereas the number of primes is 50847534 in this interval. This means that obtaining $2^{N-1} \equiv 1 \pmod{N}$ we can say that " N is probably prime".

(II) We can choose, say, 1000 random values for $c \not\equiv 0 \pmod{N}$, and test whether or not $c^{N-1} \equiv 1 \pmod{N}$. If we obtain the answer no for at least one value of c , then we can be certain that N is composite. And if the congruence holds for all 1000 values of c then N is a universal pseudoprime or it can be composite with at most the very tiny probability $1/2^{1000}$, since for composite numbers not being universal pseudoprimes the probability of satisfying the congruence for one value of c is at most $1/2$ (see Exercise 11).

(III) We can improve the Fermat test to defeat even the pseudoprimes. The Miller–Rabin–Lenstra test checks for (say) 1000 values of random $c \not\equiv 0$ the following. Write $N-1 = 2^j t$, where t is odd. Then compute $c^t, c^{2t}, \dots, c^{2^{j-1}t} = c^{(N-1)/2} \pmod{N}$. If N is prime, then either $c^t \equiv 1$, or $c^{2^h t} \equiv -1 \pmod{N}$ for some $0 \leq h \leq j-1$ (see Exercise 12). The main strength of the test is that no composite numbers can escape, since in case of a composite N the probability of satisfying the conditions for one c is at most $1/2$ (in fact, at most $1/4$), i.e. there are no universal pseudoprimes for this stronger test.

(IV) The Miller–Rabin–Lenstra test (and the Solovay–Strassen test based on the elementary theory of the Legendre and Jacobi symbols) are still probabilistic tests guaranteeing that N is prime with a probability arbitrarily close to 1. Using famous (and hopeless) conjectures from number theory they can be transformed into a deterministic test yielding full, 100% certainty, when choosing the values of c not randomly but checking the condition for the first M positive integers c where M is a function of N not too large compared to N .

(V) In 2002, however, three mathematicians from India, Agrawal, Kayal, and Saxena presented a quick deterministic "bomb-proof" primality test based on somewhat different ideas, and not relying on any unproven hypothesis.

These are important theoretical developments though a simple Fermat test with $c = 2$ is nearly completely perfect for practical purposes.

5 Factorization of integers

To get an impression why this may take an enormously long time, assume that $N = pq$ is the product of two primes each close to \sqrt{N} . Then the natural way is to check whether $3 \mid N, 5 \mid N$ etc. A simple way is to check all odd numbers up to \sqrt{N} which in our case means nearly $\sqrt{N}/2$ divisibility checks. (It would be enough to check the divisibility by primes, but first we have no list of primes up to \sqrt{N} for N large, and secondly this would not help significantly since the Prime Number Theorem states that there are about $\sqrt{N}/\ln \sqrt{N}$ primes up to \sqrt{N} .)

Hence if N is, say, a 100 digit number, then billions of years would not be sufficient to factor this single composite N .

Of course, there are some quicker methods, but by our present knowledge we do not have a really quick algorithm for factorization.

The following simple combinatorial problem may help a little to improve some of the methods.

Given $k + 1$ positive integers, none of them divisible by a prime greater than the k -th prime. Prove that the product of some of our numbers (maybe just one in itself, maybe the product of all) is a square.

Proof 1: Concerning this problem the only relevant thing is whether the exponent of a prime in the standard factorization is odd or even. For k primes this means 2^k options, and there are $2^{k+1} - 1$ non empty products formed from our $k + 1$ numbers. By the pigeon hole principle we must have two products where the exponent of every prime has the same parity, say $P_1 = a_1 a_3$ and $P_2 = a_1 a_4 a_5$ are such products. Then the product $P_1 P_2 = a_1^2 a_3 a_4 a_5$ is a square. Finally making a cancellation by the squares of our numbers appearing both in P_1 and P_2 we obtain that the product of the remaining numbers ($a_3 a_4 a_5$ in our example) is a square.

Proof 2: We assign to a number a a vector of length k , where the i -th entry is 0 or 1 according to whether the exponent of the i -th prime in the standard factorization of a is even or odd. We can consider these vectors mod 2, thus we have $k + 1$ vectors in this vector space of dimension k . Hence these are linearly dependent, a non trivial linear combination of them equals the zero vector. Then multiplying those numbers for which the coefficient of the corresponding vector is 1 we obtain a square.

Both solutions are elegant. The second one, however, has the extra advantage that we can exhibit such a product quickly since we have to solve a homogeneous system of linear equations which can be done efficiently by Gaussian elimination.

Now consider first for the sake of simplicity a large odd composite N which is the product of just two big primes: $N = pq$. If we can find some C and D for which $N \mid C^2 - D^2 = (C - D)(C + D)$ and N divides neither $C - D$, nor $C + D$, then the gcd of N and $C - D$ is a non trivial factor of N . This gcd can be quickly computed by the Euclidean algorithm.

How to find suitable values of C and D ? We pick random values of c , compute $d \equiv c^2 \pmod{N}$, and if this remainder d is not divisible by primes greater than the k -th prime, then we keep it, otherwise we discard it. If we collect $k + 1$ such pairs c and d then we know from our combinatorial problem that the product of some of the d -s is a square D^2 , and the product of the relevant c^2 -s is a square C^2 too, obviously. Hence $C^2 \equiv D^2 \pmod{N}$. We cannot be certain that $C \not\equiv \pm D \pmod{N}$, but it has probability at least $1/2$ since we have the following four roughly equally probable cases: (i) both p and q divide $C - D$; (ii) both p and q divide $C + D$; (iii) p divides only $C - D$, and q divides only $C + D$; (iv) p divides only $C + D$, and q divides only $C - D$, and the last two are good for us. If N contains more prime factors then the probability is even larger.

Why is this algorithm not really efficient? If you choose a large k then you have to produce many pairs c and d , and if

you choose a small k then nearly all pairs must be discarded, so it takes a long time to collect some of them. By deep number theoretic methods we can find an optimal k depending on N , but even so we do not arrive at a good factoring algorithm.

6 Revolutionary new ideas: public key cryptosystems and the RSA scheme

In 1975 Diffie and Hellman introduced a new cryptosystem, assuming that we make the encoding function E public while keeping the decoding function D secret. For first sight this seems to be a nonsense, since E and D are the inverses of each other, hence knowing E we know also D , at least in theory. E.g. if we want to find out $D(3)$, then we compute $E(1), E(2), \dots$ and watch when the 3 will appear. This can be, however, a very long procedure: if the domain of E is large, say, of size 10^{50} , then we have no chance to recover $D(3)$ before the end of the history of mankind. The case is similar to the situation when we intend to use an English-Hungarian dictionary also as a Hungarian-English one: wanting to find the English word for “ablak” we start to go through the alphabetic order of an English-Hungarian dictionary till we arrive on the right hand side to this Hungarian word. This happens to occur at the English word “window” which we could thus find only after checking many thousand other words.

Assuming the existence of such pairs of functions E and D , Alice and Bob can communicate the following way. Each of them has a publicly known function E_A and E_B , resp., and also the inverse of these, D_A and D_B , resp., which is known only by Alice and Bob, resp.

If Alice wants to send a message M to Bob, then she sends $L = E_B(D_A(M))$. Then Bob can get M via $E_A(D_B(L)) = M$. Nobody else can read the message since nobody else knows D_B . Nobody can send a false message in the name of Alice as nobody else knows D_A needed for the encoding. Moreover, neither Alice, nor Bob can deny the proper value of the message, by the same reasons. Also, no preliminary code exchange is necessary, everybody can use his own one pair E and D even in multilateral corresponding, which is important in business, security of credit cards etc.

This is a nice dream, but to realize it we have to construct such a scheme. A few months later Rivest, Shamir, and Adleman invented the RSA scheme which is the most widely used cryptosystem, and its security relies on the fact that at our present knowledge one can find large prime numbers quickly but nobody else can factor their product. Hence the domain of E and D should be the set $\{1, 2, \dots, N\}$ where N is the product of two large primes $N = pq$. We make N public, but keep p and q secret. Also we make public that $E(M) = M^e \pmod{N}$, where we choose e to be relatively prime to $(p-1)(q-1)$. Then we search D in the form $D(L) = L^d \pmod{N}$ with a suitable secret d . We need $M = DE(M) = ED(M) = M^{de} \pmod{N}$. Applying Fermat’s Little theorem for p and q we see that $M \equiv M^{1+h(p-1)(q-1)} \pmod{N}$ for every M , so choose d as the solution of the Diophantine equation $de = 1 + h(p-1)(q-1)$. This equation is solvable since e and $(p-1)(q-1)$ are coprime, and a solution can be found quickly using the Euclidean algorithm. To do this we need the value of $(p-1)(q-1) = N - (p+q) + 1$ which is available only for the person who knows p and q .

Of course, we cannot be absolutely certain that our function D is perfectly secret for others; maybe a different type of representation of D can be obtained from E , or one can find a quick algorithm to factor N . This, however, seems to be very improbable.

7 Exercises

1. For which positive integers is the sum of all positive divisors (including the number itself) (a) odd; (b) a power

of two?

2. What is the sum of the reciprocals of all positive divisors (including the number itself) of a(n even or eventually odd) perfect number?
3. What can be the last digit of an even perfect number?
4. Prove that an odd perfect number (if it exists at all) must be the product of a square and a prime of form $4h+1$.
5. We call a positive integer *superperfect* if $\sigma(\sigma(n)) = 2n$. Prove that an even number n is superperfect iff $n = 2^{p-1}$, where $2^p - 1$ is a Mersenne prime.
6. (a) Verify $(a+b)(a^{n-1} - a^{n-2}b + a^{n-3}b^2 - \dots - ab^{n-2} + b^{n-1})$ if n is odd.
(b) Prove that if m is not a power of 2 then $2^m + 1$ is composite.
7. Prove Pepin's test: Let $n \geq 1$. Then $F_n = 2^{2^n} + 1$ is prime iff $3^{(F_n-1)/2} \equiv -1 \pmod{F_n}$. (One direction requires the elements from the theory of Legendre symbols.)
8. Show that any (prime) divisor r of the Fermat number $F_n = 2^{2^n} + 1$ must be of the form $r = 2^{n+1}t + 1$.
9. Show that the composite number $341 = 11 \cdot 31$ is a pseudoprime for the base 2, but not for the base 3, i.e. $2^{340} \equiv 1 \pmod{341}$, but $3^{340} \not\equiv 1 \pmod{341}$.
10. Verify that 1729 is composite and a universal pseudoprime, i.e. $c^{1728} \equiv 1 \pmod{1729}$ for every $(c, 1729) = 1$.
11. Prove that if N is composite but not a universal pseudoprime then $c^{N-1} \equiv 1 \pmod{N}$ is satisfied with probability at most $1/2$ for a random c .
12. Show that the Miller–Rabin–Lenstra test is satisfied for any prime N : Write $N - 1 = 2^j t$, where t is odd, and compute $c^t, c^{2t}, \dots, c^{2^{j-1}t} = c^{(N-1)/2} \pmod{N}$ for any $c \not\equiv 0 \pmod{N}$. If N is prime, then either $c^t \equiv 1$, or $c^{2^h t} \equiv -1 \pmod{N}$ for some $0 \leq h \leq j - 1$.

1 Outline of the lecture

In the last hundred years some important applications such as Monte Carlo methods, wireless communications or famous encrypting algorithms (e.g. Vernam cipher) inspired intensive study of pseudorandomness of different objects. Initially, random and pseudorandom objects were generated by physical methods, but these methods have several disadvantages: they are slow, expensive, it is difficult to store the data and their pseudorandom properties cannot be proved mathematically. In order to avoid these difficulties, pseudorandom objects are generated nowadays from a small secret key by mathematical algorithms, with the intent that they appear random to a computationally bounded adversary.

Different approaches and definitions of pseudorandomness exist. Menezes, Oorschot and Vanstone [8] wrote an excellent monograph about these approaches. The most frequently used interpretation of pseudorandomness is based on complexity theory; Goldwasser [3] wrote a survey paper about this approach. In this approach usually sequences of length tending to infinity are tested while in the applications only *finite* sequences are used. Unfortunately, most of the results are based on certain unproved hypotheses (such as the difficulty of factorization of integers or the difficulty of the discrete logarithm problem). Finite pseudorandom $[0, 1)$ sequences have been studied by Niederreiter (see for example [9], [10], [11], [12]). Niederreiter [13] also studied random number generation and quasi-Monte Carlo methods and their connections.

In the second half of the 1990s, Christian Mauduit and András Sárközy [7] introduced a new constructive quantitative approach, in which the pseudorandomness of *finite binary* sequences is well characterized. Since then it is a fast developing area, several authors work in this field and several constructions, results and generalizations are presented in numerous papers. In [5] I gave a survey of the most important results.

In [7] Mauduit and Sárközy introduced the following pseudorandom measures:

Definition 1.1 (Mauduit, Sárközy). For a binary sequence $E_N = (e_1, \dots, e_N) \in \{-1, +1\}^N$ of length N , write

$$U(E_N, t, a, b) = \sum_{j=0}^t e_{a+jb}.$$

Then the well-distribution measure of E_N is defined as

$$W(E_N) = \max_{a,b,t} |U(E_N, t, a, b)| = \max_{a,b,t} \left| \sum_{j=0}^t e_{a+jb} \right|,$$

where the maximum is taken over all a, b, t such that $a, b, t \in \mathbb{N}$ and $1 \leq a \leq a + tb \leq N$.

The well-distribution measure studies how close are the frequencies of the $+1$'s and -1 's in arithmetic progressions (for a binary sequence with strong pseudorandom properties these two quantities are expected to be very close.) But often it is also necessary to study the connections between certain elements of the sequence. For example, if the subsequence $(+1, +1)$ occurs much more frequently than the subsequence $(-1, -1)$, then it may cause problems in the applications, and we cannot say that our sequence has strong pseudorandom properties. In order to study connections of this type Mauduit and Sárközy [7] introduced the correlation and normality measures:

Definition 1.2 (Mauduit, Sárközy). For a binary sequence $E_N = (e_1, \dots, e_N) \in \{-1, +1\}^N$ of length N , and for $D = (d_1, \dots, d_\ell)$ with non-negative integers $0 \leq d_1 < \dots < d_\ell$, write

$$V(E_N, M, D) = \sum_{n=1}^M e_{n+d_1} \dots e_{n+d_\ell}.$$

Then the correlation measure of order ℓ of E_N is defined as

$$C_\ell(E_N) = \max_{M,D} |V(E_N, M, D)| = \max_{M,D} \left| \sum_{n=1}^M e_{n+d_1} \cdots e_{n+d_\ell} \right|,$$

where the maximum is taken over all $D = (d_1, \dots, d_\ell)$ and M such that $0 \leq d_1 < \dots < d_\ell < M + d_\ell \leq N$.

Definition 1.3 (Mauduit, Sárközy). For a binary sequence $E_N = (e_1, \dots, e_N) \in \{-1, +1\}^N$ of length N , and for $X = (x_1, \dots, x_\ell) \in \{-1, +1\}^\ell$, write

$$T(E_N, M, X) = |\{n : 0 \leq n < M, (e_{n+1}, e_{n+2}, \dots, e_{n+\ell}) = X\}|.$$

Then the normality measure of order ℓ of E_N is defined as

$$N_\ell(E_N) = \max_{M,X} |T(E_N, M, X) - M/2^\ell|,$$

where the maximum is taken over all $X = (x_1, \dots, x_\ell) \in \{-1, +1\}^\ell$, and M such that $0 < M \leq N - \ell + 1$.

We remark that *infinite* analogues of the functions U, V and T had been studied before (see, for example, [2], [6] and [14]), but the quantitative analysis of pseudorandom properties of *finite* sequences has started by the work of Mauduit and Sárközy [7].

The combined (well-distribution-correlation) pseudorandom measure [7] is a common generalization of the well-distribution and the correlation measures. This measure has an important role in the multidimensional extension of the theory of pseudorandomness.

Definition 1.4 (Mauduit, Sárközy). For a binary sequence $E_N = (e_1, \dots, e_N) \in \{-1, +1\}^N$ of length N , and for $a, b, t \in \mathbb{N}$, $D = (d_1, \dots, d_\ell)$ with non-negative integers $0 \leq d_1 < \dots < d_\ell$, write

$$Z(E_N, a, b, t, D) = \sum_{j=0}^t e_{a+jb+d_1} \cdots e_{a+jb+d_\ell}.$$

Then the combined (well-distribution-correlation) measure of order ℓ of E_N is defined as

$$Q_\ell(E_N) = \max_{a,b,t,D} |Z(E_N, a, b, t, D)| = \max_{a,b,t,D} \left| \sum_{j=0}^t e_{a+jb+d_1} \cdots e_{a+jb+d_\ell} \right|,$$

where the maximum is taken over all a, b, t and $D = (d_1, \dots, d_\ell)$ such that all the subscripts $a+jb+d_i$ belong to $\{1, 2, \dots, N\}$.

When Mauduit and Sárközy introduced quantitative pseudorandom measures, their starting point was to balance the requirements possibly optimally. They decided to introduce functions which are real-valued and positive and the pseudorandom properties of the sequence are characterized by the sizes of the values of these functions. It was also an important requirement that one should be able to present constructions for which these measures can be estimated well. It turned out that the measures W and C_ℓ do not only satisfy these criteria, but later Rivat and Sárközy [15] showed that if the values of W and C_ℓ are “small”, then the outcome of many (previously used a posteriori) statistical tests is guaranteed to be (nearly) positive.

Although by W, C_ℓ, N_ℓ and Q_ℓ many pseudorandom properties of the sequence can be characterized, but obviously not all. For example, in [4] I introduced the symmetry measure in order to study symmetry properties of finite binary sequences (later the symmetry measure was generalized by Sziklai [16]). In [17] Winterhof gave an excellent survey on different pseudorandom measures and certain constructions. However it was also important to determine a not too large set of certain basic pseudorandom measures, which can guarantee the adequate security in the applications.

The measures introduced by Mauduit and Sárközy seem to satisfy these criteria. In the case of binary sequences the most studied measures are W and C_ℓ , and many papers use only these measures, while in multidimensional extensions, the most important measure is Q_ℓ .

In [1] Cassaigne, Ferenczi, Mauduit, Rivat and Sárközy formulated the following principle: “The sequence E_N is considered a “good” pseudorandom sequence if these measures $W(E_N)$ and $C_\ell(E_N)$ (at least for “small” ℓ) are “small”.”

In my talk I will show different applications and generalizations of this theory.

- [1] J. Cassaigne, S. Ferenczi, C. Mauduit, J. Rivat and A. Sárközy, *On finite pseudorandom binary sequences III: The Liouville function, I*, Acta Arith. 87 (1999), 367-384.
- [2] J. W. S. Cassels, *On a paper of Niven and Zuckerman*, Pacific J. Math. 2 (1952), 555-557.
- [3] S. Goldwasser, *Mathematical Foundations of Modern Cryptography: Computational Complexity Perspective*, ICM 2002, vol. 1, 245-272.
- [4] K. Gyarmati, *On a pseudorandom property of binary sequences*, Ramanujan J. 8 (2004), 289-302.
- [5] K. Gyarmati, *Measures of pseudorandomness*, Finite fields and applications: character sums and polynomials, P. Charpin, A. Pott, A. Winterhof (eds.), Radon Series in Computational and Applied Mathematics, de Gruyter 2013, 43-64.
- [6] D. E. Knuth, *The Art of Computer Programming*, Vol. 2, 2nd ed., Addison-Wesley, Reading, Mass., 1981.
- [7] C. Mauduit and A. Sárközy, *On finite pseudorandom binary sequences I: Measures of pseudorandomness, the Legendre symbol*, Acta Arith. 82 (1997), 365-377.
- [8] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1997.
- [9] H. Niederreiter, *On the distribution of pseudo-random numbers generated by the linear congruential method.*, Math. Comp. 26 (1972), 793-795.
- [10] H. Niederreiter, *On the distribution of pseudo-random numbers generated by the linear congruential method. II*, Math. Comp. 28 (1974), 1117-1132.
- [11] H. Niederreiter, *On the distribution of pseudo-random numbers generated by the linear congruential method. III*, Math. Comp. 30 (1976), 571-597.
- [12] H. Niederreiter, *Quasi-Monte Carlo methods and pseudorandom numbers*, Bull. Amer. Math. Soc. 84 (1978), 957-1041.
- [13] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NSF Regional Conference Series in Applied Math., Vol. 63, Soc. Industr. Applied Math., Philadelphia, 1992.
- [14] I. Niven and H. S. Zuckerman, *On the definition of normal numbers*, Pacific J. Math. 1 (1951), 103-109.
- [15] J. Rivat and A. Sárközy, *On pseudorandom sequences and their application*, Lecture Notes in Comput. Sci. 4123, General theory of information transfer and combinatorics, Springer, Berlin / Heidelberg, 2006, 343-361.
- [16] B. Sziklai, *On the symmetry of finite pseudorandom binary sequences*, Uniform Distribution Theory 6 (2011), 143-156.
- [17] A. Winterhof, *Measures of pseudorandomness*, in: S. Boztas, (ed.), CRC Handbook of Sequences, Codes and Applications: Chapman and Hall/CRC Press, to appear.

1 Introduction

The goal of Computational Group Theory (CGT) is to find effective algorithms for analysing (generally big) groups by computer. The input of such an algorithm usually consists of a group given in some way, possibly with some other information. The output can be of different types. For example, it can be an other group (e.g. find a Sylow- p subgroup of the group), a number (calculate the order of the group), or True/False (is the group solvable?).

According to how our initial group is given, different methods must be applied for *permutation groups*, *matrix groups*, *polycyclic representations*, or *groups defined by general finite presentations*. In this short note we only concentrate to permutation group algorithms, which is probably the best developed part of CGT.

1.1 Representing permutation groups

In the following Ω denotes a finite set and $\text{Sym}(\Omega)$ denotes the full symmetric group on Ω consisting of all the permutations of Ω with multiplication defined by usual function composition. The image of $\omega \in \Omega$ under the action of $g \in \text{Sym}(\Omega)$ will be denoted by ω^g . A permutation group G on Ω is just any subgroup (i.e. multiplicatively closed subset) $G \leq \text{Sym}(\Omega)$. The degree of the permutation group G is $|\Omega|$. Usually, we assume that $\Omega = \{1, 2, \dots, n\}$. In that case, any element of G can be represented by an ordered list of length n (also called an array in most programming languages) containing every positive integers up to n in some order, which requires roughly $n \log_2 n$ bits memory to store. If, for example, we would like to take calculation with permutation groups of degree $n = 10^5$ (current computer algebra systems can handle this), then this means that we can store about 5000 group elements in a memory of size 1GB, which is very few compared to $|\text{Sym}(\Omega)| = (10^5)!$ So we cannot even hope that we take a full list containing all the elements of the permutation group.

A solution of this problem is that we define permutation groups $G \leq S_n$ by giving a small set of generators for G , i.e. a subset $X \subseteq G$ such that $G = \langle X \rangle = \{x_1 \cdots x_k \mid k \in \mathbb{N}, x_1, \dots, x_k \in X\}$. (Note that we do not need to use inverses of the generators because of the finiteness of G .) This is a very efficient representation of permutation groups, because most interesting groups in practice possesses a small generating set. (As an example, a consequence of the Classification of Finite Simple Groups is that every finite simple groups can be generated by two elements.)

A disadvantage of this representation is that if some permutation group $G \leq S_n$ is given in that way, then it is not at all trivial to answer even very basic questions such as: What is the order of G , or whether G contains a particular permutation $g \in S_n$ (membership test)? In this short note we provide some algorithms, which answer these questions (and also some other problems) effectively.

1.2 Finding random elements in a group

An algorithm is called *deterministic*, if for any input it always provides a correct answer. Moreover, if you repeatedly run it with the same input string, then it runs on the same way, so its output will be the same each time. An algorithm is called *randomised*, if the running of the program not only depends on the input with which it was called, but also on some random choices, so it can happen that running it for the same input, it gives different outputs. In practice, it frequently occurs that a randomised algorithm run faster, and therefore, it can be used more effectively, than any known deterministic algorithm for the same problem. For example, although there is a known polynomial-time

deterministic algorithm for deciding whether a given integer is prime, it is too slow to use it for large numbers, and a randomised algorithm is used for this purpose.

Randomised algorithms has a great importance in CGT. Such algorithm usually work by choosing uniformly distributed random elements from the group. In the following we describe how one can choose a random element from a group given by some generators.

First of all, we need to assume that we have a *perfect number generator*, which is able to choose a random element from a finite list with uniform distribution. (In practice, such a perfect number generator does not exist, but there are satisfactory algorithms for this.) Now, let us assume that a group $G = \langle X \rangle$ is given by a set of generators $X = \{x_1, \dots, x_k\}$, and we would like to pick a random element $g \in G$.

There are situations, when this can be easily solved (with help of a perfect number generator, of course.) For example, if G is small enough, we can calculate every element of G , and choose an element from this list. We shall see later that this problem can also be easily solved if G is a permutation group such that a base and a strong generating set is known for G . In the following we describe an algorithm which can be used in a more general situation.

The product replacement algorithm

Now, let us assume that a group $G = \langle X \rangle$ is given by a set of generators $X = \{x_1, \dots, x_k\}$, and we would like to pick a (nearly) random element $g \in G$, i.e. the probability of the choice of any particular element of G should be approximately $1/|G|$. There are various algorithms for this purpose. Here we describe the *product replacement algorithm*, which is very fast and usually works well in practice. (There are situations, when this algorithm is unsatisfactory, e.g. when G is a large power of a finite simple group.)

First, a “random replacement step” works the following way. We choose randomly two indices $1 \leq s, t \leq k$ with $s \neq t$, a random sign $\varepsilon \in \{\pm 1\}$, and a random “side” (left or right). Then we replace the generator x_s with the product $x_s x_t^\varepsilon$ or $x_t^\varepsilon x_s$ (depending on which side was chosen). Clearly, after such a step we get a new set of generators for G .

Now, the full algorithm is described the following way: Experience shows that the algorithm works satisfactory well only if $|X| \geq 10$; if this would fail, one can repeat the elements of X until its size reaches this bound. First we initialise the method by applying the replacement step many times (usually 50 steps are used in practice) to get a set of generators of G which is “random” enough. After that, if we would like to get a random element, we apply the replacement step (so the set of generators is changed again) and the algorithm gives back the newly constructed generator x_s . A modification of this algorithm due to Leedham-Green also uses an accumulator x_0 . If one needs a list of random elements, then in each step, we multiply the previous value of x_0 with the newly constructed generator x_s . Then we redefine the value of x_0 to be this product, and this new value will be the next random element provided by the algorithm.

2 Some basic algorithms

In this section we present some basic algorithms regarding permutation groups which play a fundamental role in any more advanced procedures. First we fix our notation. In the following let $\Omega = \{1, 2, \dots, n\}$ and $G \leq \text{Sym}(\Omega)$ be a permutation group given by a set of generators $X = \{x_1, \dots, x_r\}$, so $G = \langle X \rangle = \{a_1 \cdots a_s \mid s \in \mathbb{N}, a_i \in X \forall i\}$. (Note that we do not need to take inverses of the generators because G is finite.)

2.1 Orbit and stabiliser

For elements $\alpha \in \Omega$ and $g \in G$ we denote by α^g the image of α under the permutation g . In the following, we will assume that α^g can be computed fast. Then the *orbit* of α is defined as $\alpha^G := \{\alpha^g \mid g \in G\} \subseteq \Omega$. We say G is *transitive* on Ω if $\alpha^G = \Omega$ for one (and hence for all) element $\alpha \in \Omega$. A related concept is the *stabiliser* of α , which is $G_\alpha := \{g \in G \mid \alpha^g = \alpha\}$. It is always a subgroup of G . The *orbit-stabiliser theorem* says that $|G| = |\alpha^G| \cdot |G_\alpha|$ for any $\alpha \in \Omega$. This can be used effectively to calculate the order of the group G . In fact, the following more general statement holds.

Theorem 2.1. *There is a bijective correspondence between the elements of the orbit of α and the right cosets of the stabiliser G_α , given as*

$$\alpha^g \longleftrightarrow G_\alpha g := \{hg \mid h \in G\},$$

so an element of G maps α to α^g (where $g \in G$ is arbitrary) if and only if it is contained in the right coset $G_\alpha g$.

Now, we would like to find a method for the following problems.

1. Calculate the orbit α^G .
2. For every $\beta \in \alpha^G$, find a “transporter” element $u_\beta \in G$ such that $\alpha^{u_\beta} = \beta$.
3. Determine the stabiliser G_α (by giving a set of generators of G_α).

The orbit algorithm

The solution of (1) is easy: We just apply the generators x_1, \dots, x_r to the already found elements of the orbit of α as long as we do not find any new. More precisely, the algorithm works in the following way. The input is $X = \{x_1, \dots, x_r\}$, the set of generators of G , and an element $\alpha \in \Omega$. We maintain an array Δ containing the already found elements of α^G . The initial step is $\Delta := [\alpha]$. Then we go through the elements of Δ . For any $\beta \in \Delta$ we calculate all the elements β^x , and check whether $\beta^x \in \Delta$. If not, we append β^x to Δ (that is, we write it at the end of the array Δ). The algorithm finishes, if there remains no element of Δ for which the generators should be applied. Then the algorithm returns with Δ , which contains exactly the elements of α^G .

Finding a transporter

To manage problem (2), we extend the orbit algorithm such that once we found a new $\beta \in \alpha^G$, then we also calculate $u_\beta \in G$. More precisely, we maintain an other array T , which has the same length as Δ at every step of the algorithm. If $\Delta[i] = \beta$ for some i (here, and everywhere, $A[i]$ will denote the i -th element of the array A), then $T[i]$ will contain an $u_\beta \in G$ such that $\alpha^{u_\beta} = \beta$. Now, running the orbit algorithm, if we reach $\beta \in \Delta[i]$ at the i -th step and $\beta^{x_t} \notin \Delta$ for some generator x_t , then we do not only append $\gamma = \beta^{x_t}$ to Δ but also append $u_\gamma := T[i] \cdot x_t$ to T (so $\alpha^{u_\gamma} = \alpha^{T[i]x_t} = \alpha^{u_\beta x_t} = \beta^{x_t} = \gamma$ holds). At the end we return with both arrays Δ , T .

Now, in contrast to the orbit algorithm, we have a serious problem, when we try to apply this method for permutation groups of degree, say, $n = 10^5$. If our group is transitive, then we should store 10^5 group elements; as we saw before, there is not enough space for this.

Fortunately, we usually do not need a full list of coset representatives of G_α . So the solution for the memory problem is that we store some other information, which requires much less space, and which is enough to calculate a transporter

u_β for a particular $\beta \in \alpha^G$ only when requested. This is usually done by Schreier vectors/trees, which we present in the following.

Schreier vectors

Given an $\alpha \in \Omega$, we modify the orbit algorithm to calculate the so-called *Schreier vector* for α , which is an array S of length n and is defined as the following. At the initial step, we define $S[\alpha] = -1$ and $S[\beta] = 0$ for every $\beta \neq \alpha$. We run the orbit algorithm with the modification that if $\beta \in \Delta$ is an already found element of α^G and $\beta^{x_t} \notin \Delta$ for some generator x_t , then we do not only append β^{x_t} to Δ but also define $S[\beta] := t$.

At the end of the algorithm, we get $\gamma \in \alpha^G$ if and only if $S[\gamma] \neq 0$. Now, using the Schreier vector we can calculate a transporter u_β for a $\beta \in \Omega$ on demand as follows.

First we check whether $S[\beta] = 0$. If so, $\beta \notin \alpha^G$, so u_β does not exist and the algorithm terminates. Otherwise, we go back on the elements of α^G at the reverse direction as how we reached β from α . This means that start with $\gamma := \beta$ and $u = 1$, then we choose $k = S[\gamma]$, and we apply x_k^{-1} to gamma to “get closer” to α . At the same time we multiply u by x_k from the left. Then we redefine $\gamma := \gamma^{x_k^{-1}}$ and repeat until we reach $S[\gamma] = -1$, i.e. $\gamma = \alpha$. (Note that since we need to use the inverses of the generators repeatedly it can spare time if we precalculate and store them instead of calculating them each time we need.) At this stage, $u_\beta := u$ will satisfy $\alpha^{u_\beta} = \beta$, so we found a transporter for β .

Calculating the stabiliser

A further modification of the orbit algorithm allows us to calculate (a generating set of) the stabiliser G_α of α . The algorithm is based on the following theorem.

Theorem 2.2 (Schreier’s lemma). *Let $G = \langle X \rangle$, $H \leq G$, and R be a right transversal for H in G . For any $g \in G$, let \underline{g} denote the unique element $r \in R$ such that $Hg = Hr$ (so $g\underline{g}^{-1} \in H$). Then the set*

$$Y := \{rx(\underline{rx})^{-1} \mid r \in R, x \in X\}$$

is a generating set of H .

Now, we can extend the orbit–transporter algorithm to also calculate a generator set Y of G_α such that every time a $\gamma := \beta^{x_t}$ is NOT appended to Δ (since γ is already found before) we append the element $u_\beta x_s (u_\gamma)^{-1}$ to Y . The previous theorem guarantees that in that way we finally get a generator set Y of G_α .

Of course, this algorithm in this form is unsatisfactory when the degree is large. One problem is that it uses the transporters u_β . We defined Schreier vectors in order to avoid to calculate, (or, which is an even more serious problem, to store) of all the transporters. This problem can be managed by using the Schreier vectors and storing only a list containing the indices of the generators from X , which we should multiply to get the Schreier generators, without actually calculating their product.

An even more serious problem is the size of the resulting generating set Y . For a transitive permutation group, the total number of Schreier generators is $n|X|$, which is too large. Usually, there are ways to decrease the size of Y . For example, if we have a membership test, which can be used at each step to check whether the newly constructed Schreier generator is already in the subgroup generated by the previously calculated elements of Y . (unfortunately, such a method requires many tests which can result in a long running time). Or one can take (a smaller set of) random

elements of the subgroup $G_\alpha = \langle Y \rangle$ and somehow check whether this random subset itself generates G_α . (Often the probability that a small random subset generates G_α is very high.) You can read more on this in the previously mentioned references.

2.2 Finding block systems

By using the orbit algorithm repeatedly, one can find a partition $\Omega = \Omega_1 \cup \dots \cup \Omega_k$ of Ω to the orbits of $G = X$. Then one can restrict the action of G to Ω_i . This restriction defines a homomorphism $\varphi_i : G \rightarrow \text{Sym}(\Omega_i)$ for every i . If G_i is the image of this homomorphism, then G_i is generated by $X_i := \varphi_i(X) = \{\varphi_i(x) \mid x \in X\}$. Notice that if the elements of X are given by their cycle decomposition, then we can easily calculate $\varphi_i(x)$ for every $x \in X$ by deleting all cycles from x which are disjoint from Ω_i . In that way we can answer questions for G by reducing them to the *transitive* permutation groups $G_i = \langle X_i \rangle \leq \text{Sym}(\Omega_i)$. (Remark: Here G is a so-called *subdirect product* of the G_i -s.)

Therefore, in the following we assume that $G = \langle X \rangle \leq \text{Sym}(\Omega)$ is transitive. Our next goal is to further analyse the structure of G to find a non-trivial *block system* for G . In that way many questions about permutation groups can be reduced to *primitive* permutation groups. First, we define these concepts for completeness and we collect their main properties.

Blocks, congruence and primitivity.

Definition 2.3. Let $G \leq \text{Sym}(\Omega)$ be a transitive permutation group.

- We say a partition $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$ of Ω is a *block system* for G , if G preserves \mathcal{B} , that is, for every $B_i \in \mathcal{B}$ and $g \in G$ we also have $B_i^g \in \mathcal{B}$.
- The elements of \mathcal{B} are called *blocks*.
- An equivalence relation \sim on Ω is called a *G-congruence* if $\alpha \sim \beta$ implies $\alpha^g \sim \beta^g$ for every $g \in G$.
- A block system $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$ is called *trivial* if either $k = 1$ (so $\mathcal{B} = \{\Omega\}$) or if $k = n$ (so $\mathcal{B} = \{\{\omega\} \mid \omega \in \Omega\}$).
- We say the action of G on Ω is *primitive*, if no nontrivial block system for G exists; otherwise we say G is *imprimitive*.

Theorem 2.4. Let $G \leq \text{Sym}(\Omega)$ be a transitive permutation group.

1. If $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$ is a system of blocks, then for every i and $g \in G$ we have either $B_i^g = B_i$ or $B_i^g \cap B_i = \emptyset$. Furthermore, by transitivity, for every i, j there is a $g \in G$ such that $B_i^g = B_j$.
2. Conversely, if $B_1 \subset \Omega$ has the property that for every $g \in G$ either $B_1^g = B_1$ or $B_1^g \cap B_1 = \emptyset$, then B_1 is a block and its translates $\{B_1^g \mid g \in G\}$ form a block system for G .
3. If $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$ is a system of blocks, then $|B_i| = |B_j|$ for any two elements B_i, B_j . In particular $k \cdot |B_1| = |\Omega|$.
4. As a corollary, if $|\Omega|$ is prime, then G is primitive.
5. If \sim is a G -congruence on Ω , then its congruence classes form a system of blocks or Ω . Conversely, if $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$ is a system of blocks for G , then the relation $\alpha \sim \beta \iff \alpha$ and β are in the same block of \mathcal{B} is G congruence.

6. Fix an element $\alpha \in \Omega$. Then there is a bijective correspondence between the blocks for G containing α and the subgroups of G containing G_α . This is given the following way. If $\alpha \in B \subseteq \Omega$ is a block, then for the setwise stabiliser $G_B = \{g \in G \mid B^g = B\}$ we have $G_\alpha \leq G_B \leq G$. Conversely, if $G_\alpha \leq T \leq G$, then the set α^T is a block for G .
7. As corollary, G is primitive if and only if $G_\alpha \leq G$ is a maximal subgroup for once (and hence for all) $\alpha \in \Omega$.

Now, if one finds a non-trivial block system $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$ for the transitive permutation group G , then one can define two related transitive actions. First, there is a “top” action of G on the set \mathcal{B} , which defines a homomorphism $G \mapsto \text{Sym}(\mathcal{B}) \simeq \text{Sym}(k)$ with image $K \leq \text{Sym}(\mathcal{B})$. On the other hand, there is a “bottom” action. Namely, the setwise stabiliser G_{B_1} acts on B_1 in a transitive way, with image $H \leq \text{Sym} B_1$. The full (imprimitive) permutation group is somehow built up from H and K . (For a more precise description, one needs to define the concept of the wreath product $H \wr K$. See any textbook about permutation groups, e.g. [4] or [3].)

Finding minimal blocks with Atkinson’ algorithm

A system of blocks $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$ for G is called minimal if $|B_1| > 1$ but there is no block $B'_1 \subset B_1$ for G such that $1 < |B'_1| < |B_1|$. Equivalently, this means that $G_\alpha < G_{B_1}$ is a maximal subgroup for any $\alpha \in B_1$, and the restricted action of G_{B_1} on B_1 is primitive. The purpose of this section to give an algorithm which finds a minimal block system for G . We present here an algorithm due to Atkinson. Although there are known algorithms with better running time, this algorithm has the advantage that it is relatively simple, and it works satisfactorily in practice even for groups with large degree.

Let $G \leq \text{Sym}(\Omega)$ be a transitive group generated by $X = \{x_1, \dots, x_r\}$. The input consists of X together with a subset $\Gamma \subseteq \Omega$. As an output, it provides the finest block system $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$ for G such that $\Gamma \subseteq B_i$ for some i . (Here “finest” means that the size of the B_i -s are as small as possible.) Applying this algorithm with $\Gamma = \{1, \alpha\}$ for every $\alpha \in \Omega$, $\alpha \neq 1$, one gets all the minimal non-trivial block systems for G . In that way one can also decide, whether G is primitive.

The main idea of the algorithm is very simple. Starting from the relation $\alpha \sim \beta$ for every $\alpha, \beta \in \Gamma$, we repeatedly search for two elements $\alpha \sim \beta$, then we apply the generators $x_t \in X$ to α and β and make $\alpha^{x_t} \sim \beta^{x_t}$, until we get a G -congruence. Of course, we need to do this in a clever way to decrease the number of steps needed to get a faster algorithm. Now, we describe the details.

During the algorithm, we maintain a partition of Ω . We call the elements of the partition cells. We store this partition with help of an array r of length n . In every step, $r[\alpha]$ is a representative element of the cell containing α for every $\alpha \in \Omega$ such that $r[\alpha] = r[\beta] \iff \alpha$ and β are in the same cell. (Note that we can easily the cell representatives from r ; they are exactly the elements $\omega \in \Omega$ satisfying $r[\omega] = \omega$.) The algorithm frequently calls a subroutine called “Union-find” with arguments, which are two cell representatives $\alpha, \beta \in \Omega$. Then it goes through the elements of Ω and for every $\omega \in \Omega$ with $r[\omega] = \beta$ the value of r is redefined to be $r[\omega] := \alpha$. The result is that the cells represented by α and β are merged.

Now, the full algorithm works the following way. At the initial step, we choose a $\gamma_0 \in \Gamma$ and define $r[\gamma] = \gamma_0$ for every $\gamma \in \Gamma$ and $r[\gamma] = \gamma$ for $\gamma \notin \Gamma$. The algorithm also uses an other array q , which stores elements of Ω , which “changed their block” during the algorithm. Initially, q only contains the elements of Γ . Then we go through the elements of q . At the i -th step, we let $\alpha := q[i]$, and $\beta := r[\alpha]$. Then for every $x \in X$ we compare the representatives of the cells of α^x and β^x . If $r[\alpha^x] \neq r[\beta^x]$ then we call “Union-find” with arguments $(r[\alpha^x], r[\beta^x])$ (in that order, so $r[\beta^x]$ will be no more a cell representative), and we append $r[\beta^x]$ to the list q .) The algorithm finishes, if we reached the end of q . Note that

every element of Ω is appended to the list at most once, so the algorithm finishes after at most n steps. One can prove that at the end r defines the finest G -congruence containing Γ .

2.3 The O’Nan–Scott theorem

By using the above procedure repeatedly, one finds that transitive permutation groups are built up from primitive ones. In that way, many (but not every) questions about permutation groups can be reduced to primitive groups, so it would be helpful to know something about finite primitive permutation groups. It turns out that the structure of such permutation groups is quite restricted. (In contrast, by Cayley’s Theorem, every group is isomorphic to a transitive permutation group by acting on itself by multiplication from the right.)

The O’Nan–Scott theorem gives a classification of the finite primitive permutation groups in terms of their socle, i.e. the subgroup generated by all its minimal normal subgroups. These groups can be divided into five classes, which are usually named as: *groups of affine type*, *diagonal groups*, *almost simple groups*, *wreath products with product action* and *twisted wreath products*. They are heavily connected to simple groups, so the Classification Theorem of Finite Simple Groups makes the O’Nan–Scott theorem useful. But this goes far beyond our scope. The interested reader can find more information for example, in ([3], [4], [5], [9])

2.4 Bases and strong generating sets

Definition 2.5. Let Ω be a set and $G \leq \text{Sym}(\Omega)$ a permutation group acting on Ω .

1. A sequence of elements $B = (\beta_1, \beta_2, \dots, \beta_k) \in \Omega$ is called a *base* for G if the identity is the only element of G which fixes each element of B .
2. The *stabiliser chain* defined by the base $B = (\beta_1, \beta_2, \dots, \beta_k)$ is a chain of subgroups

$$G = G^{(0)} \geq G^{(1)} \geq G^{(2)} \geq \dots \geq G^{(m)} = 1,$$

where $G^{(i)} = G_{\beta_i}^{(i-1)} = G_{(\beta_1, \dots, \beta_i)}$ is the subgroup of elements of G fixing each of β_1, \dots, β_i .

3. A set of generators S is called a *strong generating set* for G relative to B if $S \cap G^{(i)}$ is a generating set for $G^{(i)}$ for every $0 \leq i \leq k$.
4. If $B = (\beta_1, \beta_2, \dots, \beta_k) \in \Omega$ is a base for G , then the i -th *fundamental orbit* Δ_i is the orbit of β_i under the action of G^{i-1} , i.e. $\Delta_i := \beta_i^{G^{(i-1)}}$.

Note that if a generating set S and a base $B = (\beta_1, \beta_2, \dots, \beta_k)$ is known for G then one can easily find $S \cap G^{(i)}$ for every i by simply checking which elements of S fix each of β_1, \dots, β_i . Illustrating the importance of these concepts we first show some basic but important applications. So, from now on, let us assume that a base $B = (\beta_1, \beta_2, \dots, \beta_k)$ and a strong generating set S relative to B (BSGS for short) is known for $G \leq \text{Sym}(\Omega)$.

Calculating the order of G

If a BSGS (B, S) is known for G , then we can calculate the order of G as follows. First, for any $1 \leq i \leq k$ we can find a generating set for $G^{(i)}$ by taking $S \cap G^{(i)}$. Then we can use the orbit algorithm described before to find the fundamental

orbit Δ_i . In particular, we can calculate $|\Delta_i|$ for $1 \leq i \leq k$. By the orbit-stabiliser theorem, $|\Delta_i| = |G^{(i-1)} : G^{(i)}|$ for every i . Using the stabiliser chain $G = G \geq G^{(1)} \geq \dots \geq G^{(k)} = 1$, we get

$$|G| = \prod_{i=1}^k |G^{(i-1)} : G^{(i)}| = |\Delta_1| \cdot |\Delta_2| \cdot \dots \cdot |\Delta_k|.$$

Testing membership

Let $g \in \text{Sym}(\Omega)$ be an arbitrary permutation. We would like to decide whether $g \in G$ holds. Assuming that a base $B = (\beta_1, \dots, \beta_k)$ and a strong generating set S relative to B is known for G , we can decide whether $g \in G$ as follows.

We take the first base element $\beta_1 \in B$. We calculate both the fundamental orbit Δ_1 and β_1^g and check whether $\beta_1^g \in \Delta_1$. If not, then $g \notin G$ and the algorithm stops. Otherwise, by using the algorithm described in Section 2.1, we can find a transporter $r_1 \in G$ such that $\beta_1^g = \beta_1^{r_1}$. Then $g \in G$ if and only if $gr_1^{-1} \in G^{(1)}$. Now, we proceed the same algorithm with $g_1 := gr_1^{-1}$, and the base $(\beta_2, \dots, \beta_k)$ and strong generating set $S \cap G^{(1)}$ for $G^{(1)}$. In that way we get a chain of elements $g_i = g_1^{-1} \dots r_i^{-1}$ for $1 \leq i \leq k$. If $\beta_{i+1}^{g_i} \notin \Delta_{i+1}$ for some i , then $g \notin G$. Furthermore, if this holds for every $1 \leq i \leq k$, then $g_k = g_1^{-1} \dots r_k^{-1}$ fixes each element of B , so by the definition of the base $g_k \in G \iff g_k = 1$. So we check whether $g_k = 1$. If not, then $g \notin G$. Finally, if $g_k = 1$, then $g \in G$ follows.

A side effect of this algorithm is that if $g \in G$, then we also get a decomposition $g \in r_k r_{k-1} \dots r_1$ where each r_i is a coset representative of $G^{(i)}$ in $G^{(i-1)}$. This fact can be used to solve some logic puzzles related to groups such as Rubik's cube if we can find a BSGS and coset representatives of $G_i^{(i)}$ in $G^{(i-1)}$ for each $1 \leq i \leq k$ such that all representative can be written (in a relatively short) product of the original generators (in the case of the Rubik's cube these generators are the "quarter turns"). How one can do this effectively (for many groups) is described in [8]. This method was implemented to GAP by Hulpke. With its help, GAP is able to solve Rubik's cube from a random position by using about 100 turns.

Choosing a random element of G

Assuming that we have a perfect random number generator, the concept of BSGS can also be used to find a (perfectly) random element of G . This can be done such that we compute all the fundamental orbits Δ_i along with a complete set of right coset representatives T_i of $G^{(i)}$ in $G^{(i-1)}$. (Depending on the degree, the T_i -s can either be calculated explicitly, or can be coded via Schreier vectors.) Then we choose a random element $\delta_i \in \Delta_i$ for every i , which can be done with help of the perfect random number generator. We determine the unique element $r_i \in T_i$ for every i satisfying $\beta_i^{r_i} = \delta_i$. Then $g := r_1 r_2 \dots r_k$ will be a random element of G .

2.5 The Schreier-Sims algorithm

In this section we describe a basic version of an algorithm due to Sims [10] which finds a BSGS for a permutation group $G \leq \text{Sym}(\Omega)$ given by a set of generators S . This algorithm is usually named Schreier-Sims algorithm, since it substantially uses Schreier's Lemma (See Theorem 2.2). We note that there are many different known versions of this algorithm, and the efficiency of the various versions depends on how big the degree of the permutation group is.

To describe it we need some further notation. Let us assume that a sequence of base points $B = (\beta_1, \dots, \beta_k)$ and a generating set S for G is given. (We do not assume that B is base for G . At the initial state it can happen that $B = \emptyset$.) We will extend both B and S to get a BSGS for G . As before, we define $G^{(i)} := G_{(\beta_1, \dots, \beta_i)}$. Furthermore, let $S^{(i)} := S \cap G^{(i)}$

and $H^{(i)} := \langle S^{(i)} \rangle$ for every i . We define $H^{(0)} = G^{(0)} = G$. The following statement can easily be proved. We will use it continuously to check whether we can stop the running of the algorithm.

Lemma 2.6. *With the above notation, (B, S) is a BSGS for G if and only if $H^{(k)} = 1$ and $H_{\beta_i}^{(i-1)} = H^{(i)}$ for every $1 \leq i \leq k$.*

As an initialisation step, we go through the elements of S and for every $x \in S$ we check whether x fixes each element of B . If yes, we add a new point to B , which is moved by x . As a result we get a sequence of points $B = (\beta_1, \dots, \beta_k)$, and a chain of subgroups $H^{(1)} \geq \dots \geq H^{(k)}$ such that $H^{(i)} = \langle S^{(i)} \rangle$ for $S^{(i)} = S \cap G_{\beta_1, \dots, \beta_i}$. By construction, $S^{(k)} = \emptyset$, so $H^{(k)} = 1$.

Now, we check whether the assumptions of the above Lemma holds for every i . (If yes, then we found a BSGS for G .) We check this backwards, i.e. for $i = k-1, k-2, \dots$. Note that $H_{\beta_i}^{(i-1)} \geq H^{(i)}$ by definition, so we only need to check whether $H_{\beta_i}^{(i-1)} \leq H^{(i)}$. This can be done by calculating all the Schreier generators for $H_{\beta_i}^{(i-1)}$ in $H^{(i-1)}$ and checking whether they are in $H^{(i)}$. At this point of the algorithm we need membership testing; we saw at the last section that such a testing is available *if* we have a BSGS for $H^{(i)}$. But we already checked the equality $H_{\beta_j}^{(j-1)} = H^{(j)}$ for $j > i$ (this is why we done this backwards!), so, by the above Lemma, $(\beta_{i+1}, \dots, \beta_k)$ is a base for $H^{(i)}$ with SGS $S^{(i)}$. In that way we can find the largest i such that $H_{\beta_i}^{(i-1)} > H^{(i)}$. Moreover, we can find a Schreier generator $g \in H_{\beta_i}^{(i-1)}$ outside of $H^{(i)}$. Starting from g , we can find a new generator or a new base point as follows.

When we used the membership algorithm for g we actually found right coset representatives r_{i+1}, \dots, r_l (maybe an empty list) up to level $l \leq k$ such that $h := gr_1^{-1} \dots r_l^{-1}$ fixes β_1, \dots, β_l and either $l < k$ and β_{l+1}^h is not an element of the fundamental orbit Δ_{l+1} , or $l = k$ and $h \neq 1$ fixes every element of B . In both case, we add h to the generator sets $S^{(i)}, \dots, S^{(l)}$, so we redefine the subgroups $H^{(i)} \geq \dots \geq H^{(l)}$. In the second case, i.e. when $l = k$, we see that B is not a base for G (since every point of B is fixed by $1 \neq h \in G$). In that case we add a new element β_{k+1} satisfying $\beta_{k+1}^h \neq \beta_{k+1}$, and we define $S^{(k+1)} = \emptyset$. Then we start this cycle again, i.e. we again check whether the assumption of the above Lemma holds. It is easy to see that this algorithm must terminate after finitely many steps. At the final state, we get B is a base for G and $S := \bigcup_{i=1}^{|B|} S^{(i)}$ is an SGS for G .

3 Polynomial time algorithm for deciding isomorphism of graphs with bounded valency

A celebrated result of L. Babai [2] from last November states that isomorphism of graphs can be determined in quasipolynomial time. (Here “quasipolynomial” means that if two graphs is given each having n vertices, then there is a deterministic algorithm of running time $n^{(\log n)^c}$, which decides whether the two graphs are isomorphic.)

As an application of what we saw so far, we describe below the algorithm of Luks from 1982, which was also a starting point of Babai’s algorithm. It solves the Graph Isomorphism Problem in polynomial time for graphs with bounded valency.

In the following for a graph X let $V(X)$ denote the vertices and $E(X)$ denote the edges of X , respectively. If $a, b \in V(X)$ are adjacent vertices, then (a, b) denotes the edge connecting a and b . The valency of the graph is the maximum of the degrees (e.g. the number of outgoing edges) of the vertices of X . In the next section, we show one can reduce the Graph Isomorphism Problem for graphs with bounded valency in polynomial time to the *Colour Automorphism Problem*. In the subsequent section, we show how the *Colour Automorphism Problem* can be solved in polynomial time for 2-groups, which results a solution for the original problem to *trivalent graphs*.

3.1 Reduction to the Colour Automorphism Problem

Let X_1 and X_2 be two graphs. We would like to decide whether they are isomorphic. A first observation is that we can assume that both of them are connected. (see Exercise 2.) A further remark is that testing isomorphism between connected graphs is reducible in polynomial time to the problem of finding generators of the automorphism group of a graph. To see this, let us assume that X_1 and X_2 are connected graphs. Form the disjoint union $X_1 \cup X_2$ (whose connected components are X_1 and X_2 .) Now, X_1 and X_2 are isomorphic if and only if there is an element of $\text{Aut}(X_1 \cup X_2)$ which reverses the components X_1 and X_2 . But if such an element exists, then any generator set of $\text{Aut}(X_1 \cup X_2)$ must also contain one with this property, which can be easily checked once we found a generator set of $\text{Aut}(X_1 \cup X_2)$. The *Colour Automorphism Problem* is the following.

Definition 3.1. As an input you get a coloured set A and a subgroup $G \leq \text{Sym}(A)$ given by a set of generators. Find a set of generators for the subgroup of G defined as the stabiliser of the colouring of A .

Note that a general polynomial solution for the Colour Automorphism Problem would also provide a polynomial algorithm for the Graph Isomorphism Problem for every graph (see Exercise 3). So we restrict our attention to a smaller class of finite groups. For any natural number $k \geq 2$ let Γ_k denote the class of groups G with the property that all of the composition factors of G are subgroups of S_k . It is easy to see that if $N \triangleleft G$, then $G \in \Gamma_k$ if and only if both N and G/N are. Furthermore, if $G \in \Gamma_k$, then $H \leq \Gamma_k$ holds for every $H \leq G$. Now, we show the following.

Theorem 3.2 (Luks). *The graph isomorphism problem for graphs with valency $\leq t$ is polynomially reducible for the Colour Automorphism Problem for groups $G \in \Gamma_{t-1}$.*

Proof. For a graph X and for an edge $e \in E(X)$ let $\text{Aut}_e(X)$ denote the set of elements of $\text{Aut}(X)$ fixing e . (But elements of $\text{Aut}_e(X)$ may transpose the two endpoints of e .) Let X_1 and X_2 be two graphs with $|V(X_1)| = |V(X_2)| = n$ and joint valency $\leq t$ and fix an edge $e_1 \in X_1$. For any edge $e_2 \in X_2$, we would like to decide in polynomial time (i.e. in $O(n^c)$ time for some c), whether there is a graph isomorphism $X_1 \rightarrow X_2$, which maps e_1 to e_2 . If we have such an algorithm, then running it for every $e_2 \in E(X_2)$, this means running time at most $\binom{n}{2} \cdot O(n^c) = O(n^{c+2})$, which is still polynomial.

Now, for any fixed $e_1 \in X_1$, $e_2 \in X_2$, we form a new graph X as follows. X is almost the union of X_1 and X_2 , but we put new vertices v_1 on e_1 and v_2 on e_2 and we connect v_1 and v_2 with a new edge e . Clearly, X also has valency $\leq t$. If we find a set of generators for $\text{Aut}_e(X)$, then we can check whether there is a generator which transposes v_1 and v_2 . The existence of such a generator is clearly equivalent with the existence of an isomorphism $X_1 \rightarrow X_2$ mapping e_1 to e_2 .

Now, we define a chain $X_1 \subset X_2 \subset \dots \subset X_n$ of (not induced) subgraphs of X such that a vertex or edge of X is contained in X_r for one r if and only if it is a member of a path of length r containing e . (For example, X_1 contain only the vertices v_1, v_2 and the edge e , or $X_n = X$. Furthermore, it is important to notice that if $a, b \in V(X_r) \setminus V(X_{r-1})$, are adjacent vertices, then $(a, b) \in E(X_{r+1}) \setminus E(X_r)$ holds. Clearly, every element of $\text{Aut}_e(X)$ stabilises each subgraph X_r . Moreover, for every $r \geq 1$, any element of $\text{Aut}_e(X_{r+1})$ can be restricted to X_r . This restriction defines a homomorphism $\pi_r : \text{Aut}_e(X_{r+1}) \rightarrow \text{Aut}_e(X_r)$ for every $1 \leq r < n$.

By using induction on r , we prove that $\text{Aut}_e(X_r) \in \Gamma_{t-1}$ for every r . Simultaneously, we give an algorithm which provides a generating set of $\text{Aut}_e(X_r)$ with the assumption that a generating set of $\text{Aut}_e(X_{r-1})$ is already known. (This algorithm relies on the solution of the Colour Automorphism theorem.)

Both of these problems is trivial for $\text{Aut}_e(X_1) \simeq S_2$. Let us assume that we know that $\text{Aut}_e(X_r) \in \Gamma_{t-1}$ for some r and we know a generating set for $\text{Aut}_e(X_r)$, and we proceed with $\text{Aut}_e(X_{r+1})$. First, we define the set $\mathcal{F} = \{H \subset V(X_r) \mid 1 \leq |H| \leq t\}$ and we define a “set of fathers” function $f : V(X_{r+1}) \setminus V(X_r) \rightarrow \mathcal{F}$ such that $f(a) = \{b \in V(X_r) \mid (a, b) \in E(X)\}$.

First we examine the kernel of π_r . By definition, the restriction of any element of $\ker(\pi_r)$ to $V(X_r)$ is the identity map, so $\ker(\pi_r)$ is determined by its action on $V(X_{r+1}) \setminus V(X_r)$. Let us define a partition $\Delta_1 \cup \dots \cup \Delta_s$ of $V(X_{r+1}) \setminus V(X_r)$ such that $a, b \in V(X_{r+1})$ are in the same Δ_i if and only if $f(a) = f(b)$. Since X has valency $\leq t$, we get $|\Delta_i| \leq t - 1$ for every i . By the edge-preserving property of $\ker(\pi_r) \leq \text{Aut}_e(X_{r+1})$, we get each Δ_i is invariant under the action of $\ker(\pi_r)$, so $\ker(\pi_r) \leq \text{Sym}(\Delta_1) \times \dots \times \text{Sym}(\Delta_s)$. Moreover, X_{r+1} was defined such that any two vertices $a, b \in V(r+1) \setminus V(X_r)$ are non-adjacent in X_{r+1} , so any element of $\ker(\pi_r) \leq \text{Sym}(\Delta_1) \times \dots \times \text{Sym}(\Delta_s)$ extends to a graph isomorphism of X_{r+1} which acts trivially on X_r . Thus, we showed $\ker(\pi_r) = \text{Sym}(\Delta_1) \times \dots \times \text{Sym}(\Delta_s)$. It follows that $\ker(\pi_r) \in \Gamma_{t-1}$ and a generating set for $\ker(\pi_r)$ can be easily found.

Now we turn to the image of π_r . By the homomorphism theorem, $\text{Aut}_e(X_{r+1})/\ker(\pi_r) \simeq \text{Im}(\pi_r) \leq \text{Aut}_e(X_r)$. By our inductive assumption, $\text{Aut}_e(X_r) \in \Gamma_{t-1}$. Since Γ_{t-1} is closed both to taking subgroups and group extensions, we get $\text{Aut}_e(X_{r+1}) \in \Gamma_{t-1}$, as well. Our next goal is to find a generating set for $\text{Im}(\pi_r)$. For this purpose, we define a colouring on \mathcal{F} with $2t$ colours as follows. First, let $\mathcal{F}_s = \{H \in \mathcal{F} \mid |f^{-1}(H)| = s\}$ for every $0 \leq s \leq t - 1$. Furthermore, let $\mathcal{F}' = \{\{a, b\} \in \mathcal{F} \mid (a, b) \in E(X_{r+1}) \setminus E(X_r)\}$ the set of new edges in X_{r+1} connecting to vertices of $V(X_r)$. Then the subsets

$$\mathcal{F}_1 \cap \mathcal{F}', \mathcal{F}_2 \cap \mathcal{F}', \dots, \mathcal{F}_s \cap \mathcal{F}', \mathcal{F}_1 \setminus \mathcal{F}', \mathcal{F}_2 \setminus \mathcal{F}', \dots, \mathcal{F}_s \setminus \mathcal{F}'$$

form a partition of \mathcal{F} . By colouring the elements of this partition with different colours, we get a $2t$ -colouring of \mathcal{F} . Now, let us consider the induced action of $\text{Aut}_e(X_r)$ on \mathcal{F} . Then an easy argument shows that an element $g \in \text{Aut}_e(X_r)$ leaves this colouring invariant if and only if $g \text{Im}(\pi_r)$ holds, i.e. when g is extendable to X_{r+1} . So, if a polynomial-time algorithm is ready at hand, which gives a solution for the Colour Automorphism Problem with input $\text{Aut}_e(X_r)$ and the above colouring of \mathcal{F} , then we can find a generator set for $\text{Im}(\pi_r)$ in polynomial time. Any extension of the elements of this generating set to be elements of $\text{Aut}_e(X_{r+1})$ and the already found generating set of $\ker(\pi_r)$ together provides a generating set of $\text{Aut}_e(X_{r+1})$ in polynomial time. Applying this process at most n times, finally we get a generating set of $\text{Aut}_e(X) = \text{Aut}_e(X_n)$ in polynomial time.

3.2 The solution of the Colour Automorphism Problem for 2-groups.

A graph X is called *trivalent* if it is 3-regular, i.e. if every vertex of X has degree 3. By the previous chapter, if X is trivalent (or, more generally, if X has valency 3) and $e \in X$, then $\text{Aut}_e(X_r)$ is a 2-group for every $1 \leq r \leq n$ (where $n = |V(X)|$).

Therefore, as we saw in the last section, the Graph Isomorphism Problem for trivalent graphs is polynomially reducible to the Colour Automorphism Problem for 2-groups. In this section we show how this later problem can be showed in polynomial time with help of

In order to handle this problem in a recursive way, we consider a generalisation of this problem. In the following, if A is a coloured set and $a, b \in A$, then the relation “ a has the same colour as b ” will be denoted by $a \sim b$. Furthermore, for subsets $B \subseteq A$ and $K \subseteq \text{Sym}(A)$ let

$$\mathcal{C}_B(K) := \{\sigma \in K \mid b^\sigma \sim b, \forall b \in B\}.$$

Now, the problem we would like to solve is the following.

Problem 3.3. Let A be a coloured set, $G \leq \text{Sym}(A)$ be a 2-group given by a set of generators, $B \subseteq A$ a G -invariant subset and $\sigma \in \text{Sym}(A)$. Find $\mathcal{C}_B(G\sigma)$ in time which is polynomial in $n := |A|$.

Of course, our original colouring problem is just a special case of this with $B = A$ and $\sigma = 1$. It is easy to see that if $\mathcal{C}_B(G\sigma)$ is non-empty, then it is a right coset of $\mathcal{C}_B(G)$. This means that if $\mathcal{C}_B(G\sigma) \neq \emptyset$, then it can be given by a

representative $\sigma_0 \in \mathcal{C}_B(G\sigma)$ and a set of generators for $\mathcal{C}_B(G)$. We highlight two useful properties of $\mathcal{C}_B(K)$, which can be easily seen.

- For every $B \subseteq A$, $K, K' \subset \text{Sym}(A)$ we have $\mathcal{C}_B(K \cup K') = \mathcal{C}_B(K) \cup \mathcal{C}_B(K')$.
- For every $B, B' \subseteq A$, $K \subset \text{Sym}(A)$ we have $\mathcal{C}_{B \cup B'}(K) = \mathcal{C}_{B'}(\mathcal{C}_B(K))$.

Now, if we get (B, σ, G) as an input where B is G -invariant, then first we can check whether B is a G -orbit. If not, then we can find a proper partition $B = B_1 \cup B_2$ of B to G -invariant subsets B_1, B_2 , so $|B_1|, |B_2| < |B|$. Then we have $\mathcal{C}_B(G\sigma) = \mathcal{C}_{B_2}(\mathcal{C}_{B_1}(G\sigma))$. So we can apply a recursive procedure such that first we call the algorithm input (B_1, σ, G) to get $\sigma_1 \in \mathcal{C}_{B_1}(G\sigma)$ and $G_1 = \mathcal{C}_{B_1}(G)$ as output. Then we call it again with input (B_2, σ_1, G_1) to specify $\mathcal{C}_B(G\sigma) = \mathcal{C}_{B_2}(G_1\sigma_1)$.

Now, let us assume that G acts on B in a transitive way. Since G is a 2-group, every maximal subgroup of G has index 2. This implies that any non-trivial block system with maximal blocks must consists of two blocks. By using Atkinson's algorithm repeatedly, we can find in polynomial time (in n) such a system of blocks $\{B_1, B_2\}$ with $B = B_1 \cup B_2$, $|B_1| = |B_2| = |B|/2$. Let $H \leq G$ be the setwise stabiliser of B_1, B_2 and $\tau \in G \setminus H$. Then we can find such a τ and we can also find a generating set for H in polynomial time with help of the transporter algorithm and by calculating Schreier generators, respectively.

Then $G = H \cup H\tau$ where τ transpose the blocks B_1 and B_2 , so we get

$$\mathcal{C}_B(G\sigma) = \mathcal{C}_B(H\sigma \cup H\tau\sigma) = \mathcal{C}_B(H\sigma) \cup \mathcal{C}_B(H\tau\sigma) = \mathcal{C}_{B_2}(\mathcal{C}_{B_1}(H\sigma)) \cup \mathcal{C}_{B_2}(\mathcal{C}_{B_1}(H\tau\sigma)).$$

This equality means that Problem 3.3 for input (B, σ, G) can be reduced in polynomial time to four similar problems on sets B_1 and B_2 with $|B_1| = |B_2| = |B|/2$. In that way we get a recursive algorithm, which is able to calculate a set of generators for $\mathcal{C}_A(G)$. One can check that the total running time of this algorithm is still polynomial in $|A|$.

- [1] M. D. Atkinson. An algorithm for finding the blocks of a permutation group. *Math. Comp.*, **29**, (1975) 911–913
- [2] L. Babai, Graph Isomorphism in Quasipolynomial Time, arXiv:1512.03547
- [3] P. J. Cameron, *Permutation Groups*, Cambridge University Press (1999)
- [4] J. D. Dixon and B. Mortimer, *Permutation Groups*, Graduate Texts in Math., Springer-Verlag, Berlin (1996)
- [5] D. F. Holt, B. Eick and E. A. O’Brien, *Handbook of Computational Group Theory*, Discrete Mathematics and its Applications (2005).
- [6] A. Hulpke, Notes on Computational Group Theory, (2010)
- [7] E. M. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.* **25** (1982) 42–65.
- [8] T. Minkwitz, An algorithm for solving the factorization problem in permutation groups. *J. Symbolic Comput.* **26**, (1998) 89–95.
- [9] Á. Seress, *Permutation Group Algorithms*, Cambridge Tracts in Mathematics 152. Cambridge University Press, (2003)
- [10] C. C. Sims, Computational methods in the study of permutation groups. *Computational Problems in Abstract Algebra* (1970) 169–183

1 Introduction

Linear programming (LP) is one of the fundamental techniques for solving large-scale optimization problems. In an LP model, conditions are modeled by a system of linear inequalities, and the objective function is also linear. Although real-world problems tend to have non-linear characteristics (e.g., doubling the amount of resources used does not necessarily cost twice as much), the most efficient methods for solving optimization problems usually involve solving several linear subproblems derived from the original problem.

In addition to its role as a fundamental building block in optimization algorithms, linear programming, and in particular LP duality, is also very useful for analyzing the performance of purely combinatorial algorithms. In this lecture, we offer a glimpse into the huge variety of graph algorithms where LP duality is the underlying principle behind the performance guarantees. We start with well-known classics like Dijkstra's shortest path algorithm, explain the Hungarian Method for the assignment problem, and finally we describe a more recent result on online matching.

2 Linear programming duality – all we need to know

A *linear program* is a linear optimization problem given in the following form. (Note: the standard definition in the literature involves maximization instead of minimization, but here we use the equivalent definition as a minimization problem since it fits our purposes better.) We are given a matrix $A \in \mathbb{R}^{m \times n}$, a vector $b \in \mathbb{R}^m$, and a vector $c \in \mathbb{R}^n$. The goal is to find a vector $x \in \mathbb{R}^n$ satisfying $Ax \geq b$ and $x \geq 0$ (a *feasible solution*), for which $c^T x$ is as small as possible. A feasible solution that minimizes $c^T x$ is called an *optimal solution*.

LP duality states that for each linear program, there is an associated *dual LP* where the role of variables and conditions is exchanged, i.e. it has m variables, and n conditions. Furthermore, the dual is a linear maximization problem that has the same optimum value, provided that an optimal solution exists. This equality of optima is the *strong duality theorem*, whose proof is beyond the scope of this lecture (a good introduction into this and other aspects of linear programming is the book "Understanding and Using Linear Programming" by Matousek and Gärtner). We only prove the considerably easier result that the optimum of the dual maximization problem is at most the optimum of the minimization problem; this is called the *weak duality theorem*.

Definition 2.1. For a linear program $\min\{c^T x : Ax \geq b, x \geq 0\}$, the *dual LP* is $\max\{y^T b : y^T A \leq c^T, y \geq 0\}$. In the context of duality, the original linear program is called the *primal LP*.

Exercise 2.2. Prove that the dual of the dual LP is the primal LP.

Example 2.3. Consider the linear program $\min\{2x_1 + 3x_2 + 5x_3 : x_1 + 2x_3 \geq 1, x_2 + x_3 \geq 1, x_1 \geq 0, x_2 \geq 0, x_3 \geq 0\}$. This corresponds to the following input:

$$A = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, c = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}.$$

The dual LP is $\max\{y_1 + y_2 : y_1 \leq 2, y_2 \leq 3, 2y_1 + y_2 \leq 5, y_1 \geq 0, y_2 \geq 0\}$.

Theorem 2.4 (Weak Duality Theorem). If x is a feasible solution of the primal LP and $y \in \mathbb{R}^m$ satisfies $y^T A \leq c^T$ and $y \geq 0$, then $c^T x \geq y^T b$. Equality holds if and only if the following two conditions (called *complementary slackness conditions*) are satisfied:

- $x_j = 0$ whenever $(y^T A)_j < c_j$,
- $y_i = 0$ whenever $(Ax)_i > b_i$.

Proof. $c^T x \geq (y^T A)x = y^T (Ax) \geq y^T b$. If $c^T x = y^T b$, then equality must hold throughout, which gives the two conditions above.

Corollary 2.5. *If x is a feasible solution of the primal LP, y is a feasible solution of the dual LP, and $c^T x = y^T b$, then both x and y are optimal solutions of the respective linear programs.*

Exercise 2.6. Solve the LP in Example 2.3. Hint: first solve the dual problem by drawing it in the plane. Then use the complementary slackness conditions to obtain the optimal x .

There is another, equivalent version of the weak duality theorem where the primal LP has equations instead of inequalities.

Theorem 2.7 (Weak Duality Theorem – equality version). *If $x \in \mathbb{R}^n$ satisfies $Ax = b, x \geq 0$ and $y \in \mathbb{R}^m$ satisfies $y^T A \leq c^T$, then $c^T x \geq y^T b$. Equality holds if and only if $x_j = 0$ whenever $(y^T A)_j < c_j$.*

Proof. $c^T x \geq (y^T A)x = y^T (Ax) = y^T b$. If $c^T x = y^T b$, then $c^T x = (y^T A)x$ must hold, which is equivalent to the condition above.

3 Dijkstra's algorithm in light of LP duality

Let $D = (V, E)$ be a directed graph, $s \in V$ a node from which every node is reachable, and let $\ell \in \mathbb{R}_+^E$ be a non-negative length function on the arcs. Dijkstra's algorithm from 1959 finds the shortest path from s to v for every $v \in V - s$. In fact, it achieves a bit more: it finds an *arborescence rooted at s* (i.e., a spanning tree directed outwards from s) such that for every $v \in V - s$, the path from s to v in the arborescence is a shortest path.

We will prove the correctness of Dijkstra's algorithm for the problem using the weak duality theorem and Corollary 2.5. First, we introduce some notions relating digraphs to linear algebra.

- A vector $y \in \mathbb{R}^V$ is a *feasible potential* for D and ℓ if $y_s = 0$ and $y_v - y_u \leq \ell_{uv}$ for every $uv \in E$.
- The *incidence matrix* of D is the matrix with $|V|$ rows and $|E|$ columns, where the column corresponding to $uv \in E$ has a -1 in the row of u , a 1 in the row of v , and 0 in all other positions.
- The *characteristic vector* of an arc set $F \subseteq E$ is an $|E|$ -dimensional vector that has 1 in the positions corresponding to arcs in F , and 0 in all other positions.

The length of the shortest $s - t$ path is called the *distance* of t from s , and is denoted by $\text{dist}(s, t)$.

Lemma 3.1. *If y is a feasible potential with $y_s = 0$, and P is an $s - t$ path of length y_t , then P is a shortest $s - t$ path and $y_t = \text{dist}(s, t)$.*

Proof using Corollary 2.5. Let A be the incidence matrix of D , let $b \in \mathbb{R}^V$ be -1 on s , 1 on t , and 0 on every other node, and let $c = \ell$. If x is the characteristic vector of P , then x satisfies $Ax \geq b$ and $x \geq 0$, and $c^T x$ is the length of P . The feasible potential y satisfies $y^T A \leq c^T$, $y \geq 0$, and $c^T x = y^T b$. Thus x is an optimal feasible solution of the primal LP by Corollary 2.5. Since the characteristic vector of any $s - t$ path is a feasible solution, P is a shortest $s - t$ path.

```

input : digraph  $D = (V, E)$ ;  $s \in V$ ;  $\ell \in \mathbb{R}_+^E$ 
output: shortest path arborescence  $F$ , distances  $y \in \mathbb{R}_+^V$ 
initialization:  $y_s = 0$ ;  $y_v = \ell_{sv}$  if  $sv \in E$ ,  $y_v = \infty$  otherwise;  $U = \{s\}$ ;
while  $U \neq V$  do
    choose  $v \in V \setminus U$  with  $y_v$  minimal;
    for  $vw \in E$  do
        if  $w \notin U$  and  $y_w > y_v + \ell_{vw}$  then
             $y_w := y_v + \ell_{vw}$ 
        end
    end
    end
    choose an arc  $uv$  with  $u \in U$  and  $y_v - y_u = \ell_{uv}$ , and add it to  $F$ ;
    add  $v$  to  $U$ 
end
    
```

Algorithm 1: Dijkstra's algorithm

Dijkstra's algorithm can be seen as a way to successively compute a feasible potential y and an arborescence F rooted at s such that the paths in the arborescence satisfy the condition in the Lemma.

Claim 3.2. *The vector y is a feasible potential when the algorithm terminates.*

Proof. Consider an arc uv . If v is added to U earlier than u , then $y_v \leq y_u$, so $y_v - y_u \leq 0 \leq \ell_{uv}$. If v is added to U later than u , then $y_v - y_u \leq \ell_{uv}$ when the algorithm processes u ; after that, y_u does not change, while y_v can only decrease.

Claim 3.3. *If $uv \in F$, then $y_v - y_u = \ell_{uv}$ at the end of the algorithm, i.e. the complementary slackness conditions are satisfied.*

Proof. Equality holds when v is added to U , and neither y_u nor y_v is changed later.

By using an appropriate data structure, the running time of the algorithm can be shown to be $O(m + n \log n)$.

Exercise 3.4. Give an algorithm similar to Dijkstra's for the following problem. We are given a digraph $D = (V, E)$, nodes s and t , and costs $c \in \mathbb{R}^E$. Find an $s - t$ path for which the largest cost of an arc along the path is as small as possible.

4 Minimum cost arborescences

Dijkstra's algorithm constructs a shortest path arborescence rooted at s . Is it true that this is also the spanning arborescence that has the smallest total length? A simple counterexample shows that this is not necessarily so. Consider a digraph with nodes s, u, v and arcs (with length in parentheses) su (2), sv (2), uv (1). The shortest path arborescence is $\{su, sv\}$, while the shortest arborescence is $\{su, uv\}$.

In the following we give an algorithm, based on LP duality for a different LP, that find a cheapest arborescence rooted at s for any cost vector $c \in \mathbb{R}^E$. To define the LP, let A be the 0-1 matrix whose columns correspond to the arcs in E , and whose rows are the characteristic vectors of $\delta^{in}(Z)$ for each $Z \subseteq V - s$ (here $\delta^{in}(Z)$ denotes the set of arcs entering Z). Consider the LP $\min\{c^T x : Ax \geq \mathbf{1}, x \geq 0\}$. The dual LP is $\max\{y^T \mathbf{1} : y^T A \leq c^T, y \geq 0\}$. In other words, we have dual variables y_Z for every node set $Z \subseteq V - s$, and y is dual feasible if $\sum\{y_Z : u \notin Z, v \in Z\} \leq c_{uv}$ for every $uv \in E$.

Lemma 4.1. *Let F be an arborescence rooted at s . If there is a dual feasible y such that $c(F) = \sum\{y_Z : Z \subseteq V - s\}$, then F is a cheapest arborescence.*

Proof. If x is the characteristic vector of an arborescence F rooted at s , then x is a feasible solution of the LP, and $c^T x$ is the cost of F . Thus the weak duality theorem implies the statement.

Example 4.2. Consider the digraph mentioned above, with nodes s, u, v , and arcs (with cost in parentheses) su (2), sv (2), uv (1). Let $F = \{su, uv\}$, and $y_{\{u\}} = 2, y_{\{v\}} = 1$. This satisfies the conditions of the lemma, so F is a minimum cost arborescence.

The algorithm that we describe for this problem is a so-called two-phase greedy algorithm: first we find a dual feasible y , and then we build F such that $c(F) = \sum\{y_Z : Z \subseteq V - s\}$. During the algorithm, we use the notation $c_{uv}^y = c_{uv} - \sum\{y_Z : u \notin Z, v \in Z\}$. A vector y is dual feasible if and only if $c^y \geq 0$. An arc e is a 0-arc if $c_e^y = 0$.

```

input : digraph  $D = (V, E)$ ;  $s \in V$ ;  $c \in \mathbb{R}^E$ 
output: cheapest arborescence  $F$ 
begin phase 1
  initialization:  $y = \mathbf{0}$ ;
  while not all nodes are reachable from  $s$  on 0-arcs do
    choose inclusionwise minimal  $Z \subseteq V - s$  that is not entered by 0-arcs;
    increase  $y_Z$  by  $\min\{c_{uv}^y : u \notin Z, v \in Z\}$ 
  end
end
begin phase 2
  initialization:  $V(F) = \{s\}, F = \emptyset$ ;
  while  $V(F) \neq V$  do
    find 0-arc  $uv$  leaving  $V(F)$  that became 0-arc at the earliest point in phase 1;
    add  $uv$  to  $F$  and add  $v$  to  $V(F)$ 
  end
end

```

Algorithm 2: Cheapest arborescence algorithm

Proposition 4.3. At the end of phase 2, y and F satisfy the complementary slackness conditions, so $c(F) = \sum\{y_Z : Z \subseteq V - s\}$.

Proof. Let x be the characteristic vector of F . It is clear that $x_{uv} = 0$ whenever $c_{uv} > \sum\{y_Z : u \notin Z, v \in Z\}$, since F consists of 0-arcs. We have to show that if $y_Z > 0$, then Z is entered by exactly one arc in F . Consider the point in phase 1 when y_Z becomes positive. At that point, every nonempty proper subset of Z already has an entering 0-arc, so the induced 0-arcs in Z form a strongly connected graph. This means that in phase 2, after we first enter Z , we do not add another arc entering Z because we can instead extend the arborescence inside Z with an arc that became 0-arc earlier.

Exercise 4.4. Let $D = (V, E)$ be a digraph, let $s \in V$ be a node from which all other nodes are reachable, and let $\{Z_1, \dots, Z_k\}$ be a family of subsets of $V - s$. Give an algorithm that decides if there is an arborescence rooted at s that enters every Z_i exactly once. Hint: use the above algorithm with an appropriate cost function.

5 The Hungarian Method

In the assignment problem, there are n tasks and each has to be assigned to one of n employees. Not all employees are qualified to do all tasks; furthermore, even if employee j is qualified to do task i , assigning task i to employee j

has cost c_{ij} . How can we find the cheapest assignment of the tasks?

In mathematical terms, this is a bipartite minimum cost perfect matching problem: given a bipartite graph $G = (S, T; E)$ and costs $c \in \mathbb{R}^E$, find a cheapest perfect matching. The algorithm that solves this problem is based on the work of Kőnig and Egerváry, hence the name Hungarian Method. This name was given by Harold Kuhn, who first described the algorithm in 1955. In a surprising turn of events, it was discovered in 2006 that Carl Gustav Jacobi had already solved an equivalent problem in the 19th century, using essentially the same method. Here we describe a faster version developed by Edmonds and Karp (1970) and Tomizawa (1971).

To analyze the algorithm using LP duality, we define the *incidence matrix* of G as the matrix with $|S| + |T|$ rows and $|E|$ columns, where the column corresponding to $uv \in E$ has two 1s, in the rows of u and v .

Lemma 5.1. *Let M be a perfect matching, and let $y \in \mathbb{R}^{S \cup T}$ be a node weighting that satisfies $y_u + y_v \leq c_{uv}$ for every $uv \in E$. If $\sum_{v \in S \cup T} y_v = c(M)$ (or, equivalently, $y_u + y_v = c_{uv}$ for every $uv \in M$), then M is a cheapest perfect matching.*

Proof. Let $A \in \mathbb{R}^{2n \times m}$ be the incidence matrix of G . The characteristic vector of a perfect matching is a feasible solution of $Ax = \mathbf{1}$, $x \geq 0$. A node weighting $y \in \mathbb{R}^{S \cup T}$ satisfies $y^T A \leq c^T$ if and only if $y_u + y_v \leq c_{uv}$ for every $uv \in E$. The lemma thus follows from the equality version of the Weak Duality Theorem.

If M is a matching in G , then we denote by D_M the directed graph obtained from G by orienting the edges of $E \setminus M$ from S to T , and the edges of M from T to S . Let S_M (T_M) denote the set of nodes not covered by M in S (T). Let U_M denote the set of nodes reachable from S_M in D_M .

input : bipartite graph $G = (S, T; E)$ with $|S| = |T| = n$; costs $c \in \mathbb{R}^E$

output: cheapest perfect matching (or proof that no perfect matching exists)

initialization: $M = \emptyset$; $y \in \mathbb{R}^{S \cup T}$ satisfying $y_u + y_v \leq c_{uv}$ for every $uv \in E$;

while $S_M \neq \emptyset$ **do**

if there is a path in D_M from S_M to T_M **then**

 let $\ell_{uv} = c_{uv} - y_v - y_u$ for every $uv \in E$;

 run Dijkstra on D_M with length function ℓ to find shortest path P from S_M to T_M ;

 let $M := (M \setminus P) \cup (P \setminus M)$;

 let $y_u := y_u - \text{dist}(S_M, u)$ if $u \in S$;

 let $y_v := y_v + \text{dist}(S_M, v)$ if $v \in T$;

else

return “there is no perfect matching”

end

end

Algorithm 3: The Hungarian Method

Claim 5.2. *The node weighting y satisfies $y_u + y_v \leq c_{uv}$ for every $uv \in E$ and $y_u + y_v = c_{uv}$ for every $uv \in M$ throughout the algorithm.*

Proof. For $uv \in E$, let y'_u and y'_v be the new node weights after a step where the algorithm modifies y . Then $y'_u + y'_v - y_u - y_v = \text{dist}(S_M, v) - \text{dist}(S_M, u) \leq \ell_{uv} = c_{uv} - y_u - y_v$ for every $uv \in E$. Furthermore, $y'_u + y'_v - y_u - y_v = \text{dist}(S_M, v) - \text{dist}(S_M, u) = c_{uv} - y_u - y_v$ for every $uv \in P$, because P is a shortest path, and $y'_u + y'_v - y_u - y_v = \text{dist}(S_M, v) - \text{dist}(S_M, u) = 0 = c_{uv} - y_u - y_v$ for every $uv \in M$, since the only way to get to u in D_M is via the arc vu which has $\ell_{vu} = 0$.

It follows by Lemma 5.1 that M is a cheapest perfect matching at the end of the algorithm. The running time is $O(mn + n^2 \log n)$.

6 Vertex cover – A primal-dual approximation algorithm

In the next two sections, we examine how LP duality can help us to find approximately optimal solutions to problems which are too hard to solve optimally. We start by introducing the Vertex Cover Problem. Let $G = (V, E)$ be a graph, and let $c \in \mathbb{R}_+^V$ be a nonnegative cost function on the nodes. A *vertex cover* of G is a node set $X \subseteq V$ with the property that $|\{u, v\} \cap X| \geq 1$ for every $uv \in E$. The cost of a vertex cover X is $\sum_{v \in X} c(v)$.

Finding a cheapest vertex cover is an NP-hard problem (in fact, finding a minimum size vertex cover is already NP-hard). One approach to deal with NP-hard optimization problems is to try to find an approximately optimal solution. An α -approximation algorithm (for a minimization problem) is a polynomial-time algorithm with the property that for any instance, the output is a feasible solution whose cost is at most α times the optimal cost.

In the following, we give a 2-approximation algorithm for the vertex cover problem. This means that for any graph, our algorithm will output a vertex cover whose cost is at most twice that of the cheapest vertex cover. Interestingly, although this algorithm is very simple, no α -approximation algorithm for $\alpha < 2$ is known for the vertex cover problem. In fact, there is a conjecture, called the Unique Games Conjecture, whose truth would imply that such an approximation algorithm is impossible.

Let A be the transpose of the incidence matrix of G . We consider the linear programming problem $Ax \geq \mathbf{1}$, $x \geq 0$, $\min c^T x$. Notice that the characteristic vector of a vertex cover is a feasible solution to this problem. The dual problem is $y^T A \leq c^T$, $y \geq 0$, $\max y^T \mathbf{1}$. Thus we have a nonnegative variable y_e for each edge, and there is one inequality for each node v : $\sum_{uv \in E} y_e \leq c_v$. The objective is to maximize $\sum_{e \in E} y_e$. Our algorithm is a kind of dual greedy algorithm, where the dual solution is increased greedily, and nodes are chosen to be included in the vertex cover based on the dual solution.

```

input : graph  $G = (V; E)$ ; costs  $c \in \mathbb{R}_+^V$ 
output: vertex cover  $X$ 
initialization:  $X = \emptyset$ ;  $F = E$ ;  $y_e = 0$  for every  $e \in E$ ;
while  $F \neq \emptyset$  do
    choose arbitrary  $uv \in F$ ;
    increase  $y_{uv}$  by  $\min\{c_u - \sum_{uw \in E} y_{uw}, c_v - \sum_{vw \in E} y_{vw}\}$ ;
    if  $c_u = \sum_{uw \in E} y_{uw}$  then
        add  $u$  to  $X$ ;
        remove the edges incident to  $u$  from  $F$ 
    else
        add  $v$  to  $X$ ;
        remove the edges incident to  $v$  from  $F$ 
    end
end

```

Algorithm 4: 2-approximation algorithm for cheapest vertex cover

Theorem 6.1. *This is a 2-approximation algorithm for the cheapest vertex cover problem.*

Proof. To see that X is a vertex cover at the end of the algorithm, observe that edges are removed from F only if one of their endpoints is added to X . Thus X is a vertex cover when F becomes empty.

In order to prove that X is a 2-approximation, we resort to the weak duality theorem. The vector y constructed by the algorithm is a feasible solution of the dual problem, because it satisfies $y \geq 0$ and $\sum_{uv \in E} y_e \leq c_v$ after every step.

Therefore, by the weak duality theorem, any vertex cover has cost at least $\sum_{e \in E} y_e$. It is enough to prove that $\sum_{u \in X} c_u \leq 2 \sum_{e \in E} y_e$ at the end of the algorithm. When a node u is added to X , the value of $\sum_{uw \in E} y_{uw}$ is c_u , and it cannot decrease later in the algorithm, so $c_u \leq \sum_{uw \in E} y_{uw}$ at the end. Thus $\sum_{u \in X} c_u \leq \sum_{u \in X} \sum_{uw \in E} y_{uw} = 2 \sum_{e \in E} y_e$, as required.

Exercise 6.2. Consider the following alternative algorithm. Solve the LP $\min\{c^T x : x \geq 0, x_u + x_v \geq 1 \forall uv \in E\}$, and let x^* be the optimal solution. Let X be the set of nodes with $x_v^* \geq 1/2$. Show that this is also a 2-approximation algorithm. Implement both algorithms using LEMON, and compare them on some random graphs.

7 Online matching

Consider the following online version of the assignment problem. We have n employees and m tasks, but the latter are not known in advance. Instead, tasks arrive one by one, and the set of employees qualified to do a particular task is revealed only when the task arrives. Upon arrival, we immediately have to either assign it to a qualified employee, or refuse it; each employee can be assigned at most one task. The aim is to maximize the number of assigned tasks; no costs are involved.

This problem is called the Online Bipartite Matching Problem. In graph-theoretical terms, we have a bipartite graph $G = (S, T; E)$, where $|S| = m$ and $|T| = n$. This graph is not known in advance; instead, the nodes of S arrive one by one, and the neighbors of $s \in S$ are revealed when it arrives. The new node s is either immediately matched to a previously unmatched neighbor, or it remains unmatched. The aim is to maximize the size of the matching.

To analyze the performance of an algorithm, we compare the size of the obtained matching to the “offline” optimum, i.e. the size of the maximum matching in G . We say that an online algorithm has *competitive ratio* β if for any graph G , the matching returned by the algorithm has size at least the size of the maximum matching in G divided by β .

Unfortunately, this problem is trivial in some sense. First, it is easy to achieve competitive ratio 2: whenever a node s arrives, we match it to an arbitrary unmatched neighbor; if none exist, then s remains unmatched. The matching M obtained this way is an inclusionwise maximal matching in G ; indeed, if there is an edge st for which s is not covered by M , then t is necessarily covered by M , otherwise we would have matched s to t when it arrived. It is well-known that the size of an inclusionwise maximal matching is at least half the maximum size of a matching.

It is also easy to see that a competitive ratio better than 2 is impossible in general. Consider two graphs G_1 and G_2 , both on 4 nodes s_1, t_1, s_2, t_2 . G_1 has edges $s_1 t_1, s_1 t_2$, and $s_2 t_1$, while G_2 has $s_1 t_1, s_1 t_2$, and $s_2 t_2$. When s_1 arrives, our algorithm cannot know whether it processes G_1 or G_2 , so it makes the same decision for both graphs. This means that for one of the graphs it will return a matching of size 1, although the maximum size of a matching is 2 in both graphs.

How can we still improve on this? The answer is *randomization*. If the algorithm is allowed to make random decisions, then we can consider the *expected size* of the obtained matching. A randomized online algorithm has *expected competitive ratio* β if for any graph G , the expected size of the matching returned is at least the size of the maximum matching in G divided by β .

In the above example, suppose that the algorithm chooses both $s_1 t_1$ and $s_2 t_2$ with probability $1/2$. For both G_1 and G_2 , the expected size of the obtained matching is $3/2$, so it is the maximum size of a matching divided by $4/3$.

Can we achieve expected competitive ratio $4/3$? Not quite, but we will show a very simple randomized algorithm that has competitive ratio $\frac{e}{e-1}$. Surprisingly, this is best possible!

The algorithm is called the **ranking algorithm**. At the beginning, a rank $r_t \in [0, 1]$ is chosen uniformly at random,

independently for each node $t \in T$. When a node s arrives, we match it with the unmatched neighbor that has the smallest rank, if one exists. The following theorem was proven by Karp, Vazirani, and Vazirani in 1990. The proof that we present is by Devanur, Jain, and Kleinberg from 2012.

Theorem 7.1. *The ranking algorithm has expected competitive ratio $\frac{e}{e-1}$.*

We prove the theorem using LP duality. Let A be the incidence matrix of G . Consider the linear program $\max\{\mathbf{1}^T x : Ax \leq \mathbf{1}, x \geq 0\}$, and its dual: $\min\{y^T \mathbf{1} : y^T A \geq \mathbf{1}^T, y \geq 0\}$. Notice that the characteristic vector of a matching is a feasible solution to the first problem, while the dual is the vertex cover LP for the special case when all nodes have weight 1.

In the proof, a curious role is played by the function $h(z) = e^{z-1}$, a monotone increasing function that satisfies

$$1 - h(\alpha) + \int_0^\alpha h(z) dz = \frac{e-1}{e} \quad \text{for any } \alpha \in (0, 1). \quad (1)$$

To every particular run of the online algorithm, we associate a vector y : if s is matched to t , then let $y_s = \frac{e}{e-1}(1 - h(r_t))$ and let $y_t = \frac{e}{e-1}h(r_t)$. This clearly gives $y^T \mathbf{1} = \frac{e}{e-1}|M|$, where M is the matching returned by the algorithm. If y was a feasible dual solution, then we would be done. Unfortunately, y is not always feasible (can you show an example where it is not?). Nonetheless, we will prove that the expected value of y , denoted by \bar{y} , is a feasible solution of the dual, hence $\bar{y}^T \mathbf{1} \geq |M'|$ for any matching M' . Since $\bar{y}^T \mathbf{1} = \frac{e}{e-1}E(|M|)$, this proves that the algorithm has expected competitive ratio $\frac{e}{e-1}$.

Lemma 7.2. *The expected value of y , denoted by \bar{y} , satisfies $\bar{y}^T A \geq \mathbf{1}^T$.*

Proof. Consider a given edge $st \in E$; we have to prove that $\bar{y}_s + \bar{y}_t \geq 1$. The key to the proof is the behavior of the ranking algorithm on the graph $G - t$. Let r^* denote the rank of the partner of s if s is matched in the $(G - t)$ -algorithm, and let $r^* = 1$ if s is unmatched. Note that r^* is also a random variable, and it is determined by $\{r_v : v \in T - t\}$, since r_t does not influence the $(G - t)$ -algorithm.

Let us fix the values $\{r_v : v \in T - t\}$. This means that r^* is also fixed, while y_s and y_t are functions of r_t .

Claim 7.3. *If $r_t < r^*$, then t gets matched in the G -algorithm.*

Proof. If t is matched by the algorithm before s arrives, then we are done. Otherwise the $(G - t)$ -algorithm and the G -algorithm are identical until s arrives. Since s is matched to a node of rank r^* (or remains unmatched) in the $(G - t)$ -algorithm, and $r_t < r^*$, s is matched to t in the G -algorithm.

It follows from the claim that

$$E(y_t) \geq \frac{e}{e-1} \int_0^{r^*} h(z) dz. \quad (2)$$

To bound y_s from below, we again consider the case when the values $\{r_v : v \in T - t\}$ are fixed.

Claim 7.4. *For any value of r_t , we have $y_s \geq \frac{e}{e-1}(1 - h(r^*))$.*

Proof. We execute the algorithm for G and $G - t$ in parallel. First, observe that at any point, the set of unmatched nodes in T in the $(G - t)$ -algorithm is a subset of the set of unmatched nodes in the G -algorithm. Indeed, at the first point where the G -algorithm would match a node $u \in S$ to a node $v \in T - t$ that the $(G - t)$ -algorithm does not match, the $(G - t)$ -algorithm has to match u to a node of smaller rank, but the same node is also available in the G -algorithm, a contradiction.

When s arrives, it has an unmatched neighbor of rank r^* in the $(G - t)$ -algorithm and, by the above observation, the same neighbor is unmatched in the G -algorithm. Thus s is assigned a partner of rank at most r^* , hence $y_s \geq \frac{e}{e-1}(1 - h(r^*))$.

Combining this claim with equation (2), and then applying property (1) of h , we get

$$y_s + E(y_t) \geq \frac{e}{e-1} \left(1 - h(r^*) + \int_0^{r^*} h(z) dz \right) = 1$$

for any fixed values $\{r_v : v \in T - t\}$. Thus $\bar{y}_s + \bar{y}_t \geq 1$, as required.

1 Introduction

Usually class P is considered as a synonym for efficiently solvable problems. Deciding whether a graph has a complete subgraph on one thousand vertices can be solved in $O(n^{1000})$ – that is in polynomial time – however this is not considered as an effective solution. To dissolve this contradiction the concept of class FPT (Fixed-parameter tractable problems) was introduced in 1992 by Downey and Fellows. Here problems (besides the input) have a natural parameter (usually denoted by k), and in the running time the exponent of the input length *must not depend* on k . The typical running time is usually something like $O(2^k \cdot n^2)$ or $O(4^k \cdot n^2)$ or $O(k^k + n^2)$, for large inputs these are much more practical than $O(n^k)$.

Example 1.1. Suppose we have algorithms A1-A4 with the following running times (here we only count $n = |V|$ in the bounds).

A1: n^k ; A2: $2^k \cdot n^2$; A3: $k^k + n^2$; A4: $4^k \cdot n^2$.

Let G be a relatively small graph with 1000 vertices, and suppose $k = 10$. When running these algorithms on your laptop the estimated running time for these algorithms is the following.

A1: $3 \cdot 10^{15}$ years; A2: 1.7 min; A3: 16.7 min; A4: 1.2 day.

(The age of the universe is less than $1.4 \cdot 10^{10}$ years.)

In the last twelve years 4 books and more than one thousand papers were published on this topic. Subsequently an enormous number of new algorithms were designed for fixed-parameter problems, together with developing new concepts and new techniques as well as new analyzing methods. The purpose of this talk is to give an introduction to this topic and to explain some basic methods developed.

Prerequisites: Basics of complexity theory (big O notation, classes P and NP, NP-hard problems), elementary graph theory, fundamental algorithms (sorting, BFS, DFS, dynamic programming, Dijkstra's algorithm for shortest paths, maximum matching in bipartite graphs)

2 Definitions

In a parametric problem the input consists of a pair (x, k) , where $x \in \{0, 1\}^N$ and k is a positive integer. A (decision) problem \mathcal{P} is a subset of all possible inputs, we call this subset the set of yes-instances. An algorithm solves \mathcal{P} if it stops on every input, and answers YES if and only if the input is a yes-instance.

If x is intended to describe a graph $G = (V, E)$, then is more convenient to use $n = |V|$ and $m = |E|$ instead of the length in bits. We assume the graph is given by edge lists, so $N = O((n + m) \log n)$ is the true bit complexity, however, we consider $n + m$ as the input length and we assume we can handle (e.g., read) the name of a vertex in one step.

Example 2.1. Problem VERTEX-COVER. The input consists of an undirected simple graph $G = (V, E)$ and the parameter k . It is a yes-instance if and only if G has a vertex cover T with $|T| \leq k$. ($T \subseteq V$ is a *vertex cover* if it contains at least one end-vertex of every edge.)

We say that problem \mathcal{P} is in the class XP (or that \mathcal{P} is slice-wise polynomial) if there is an algorithm solving it for every input (G, k) in time $O(f(k) \cdot N^{g(k)})$ for some (computable) functions f, g . This is equivalent to requiring a solver that

runs in polynomial time for every fixed constant k .

We say that problem \mathcal{P} is in the class FPT (or that \mathcal{P} is fixed parameter tractable) if there is an algorithm solving it for every input (G, k) in time $O(f(k) \cdot N^c)$ for some fixed constant c and (computable) function f . (Now the exponent of N must not depend on k . For example $O(k! \cdot N^3)$ is an acceptable running time.) If such an algorithm exists, we call it an FPT-algorithm.

A *reduction rule* is a function Φ mapping from the instances to the instances, such that if $\Phi(G, k) = (G', k')$, then either both (G, k) and (G', k') are yes-instances or both are no-instances (we will call such a pair *equivalent instances*.) A reduction rule is polynomial if it is computable in time $O((n + m + k)^c)$ for an absolute constant c .

A polynomial reduction rule is a *kernel* if $|V(G')| \leq f'(k)$ for a computable function f' and moreover $k' \leq k$.

The idea behind this is the following: after polynomial preprocessing we make a small equivalent instance. For a graph with $f'(k)$ vertices every usual decision problem can be decided in time $O(f(k))$ for another function f , usually $f(k) = c^{f'(k)}$ is sufficient.

In the first part we describe some basic techniques worked out for the problem VERTEX-COVER.

3 Main tool: Kernelization

In order to make a kernel for VERTEX-COVER we give a polynomial algorithm which makes an equivalent instance (G', k') with $k' \leq k$ and $n' := |V(G')| \leq k(k + 1)$. The algorithm is given via reduction rules; always the first rule must be used whose condition is met.

Algorithm 3.1. Initialize $T = \emptyset$, $G' = G$ and $k' = k$. Use one of the following rule (in order) until they applies.

- i) If $k' = 0$ and G' is empty, then return YES and T as a cover. If $k' = 0$ and G' has some edges, then return NO.
- ii) If G' has an isolated vertex, then simply delete it.
- iii) If G' has a vertex v with degree strictly more than k' , then $T := T + v$, delete v and $k' := k' - 1$.
- iv) If none of the rules above applies, and either $n' > k'(k' + 1)$ or $m' > k'^2$, then return NO; otherwise return (G', k', T) .

Theorem 3.2. *If the algorithm stops with an answer, then this answer is correct. Otherwise, when it finishes, the current graph G' and the current k' describe an equivalent instance, and if T' is a vertex cover of G' , then $T \cup T'$ is a vertex cover of G . Moreover Algorithm 3.1 runs in polynomial time, $k' \leq k$ and when it returns (G', k') , then $n' \leq k^2 + k$ and $m' \leq k^2$.*

Proof. By definition, either v or its neighborhood $N(v)$ must be in the cover. If $d(v) > k$ and there exists a cover of size at most k , then v must be in the cover.

If $k' = 0$, then $|T| = k$ and it consist of vertices that must be in the cover, so either T is a cover (and the answer is YES) or there are edges remaining not covered by T , in this case the right answer is NO.

When neither of the first 3 rules can be applied, every vertex has degree $1 \leq d(v) \leq k'$. If $n' > k'(k' + 1)$, then no cover in G' of size k' can exist because such a cover would have at most k'^2 neighbors and every vertex is either in the cover or has a neighbor in the cover. Similarly if $m' > k'^2$, then no cover in G' of size k' can exist because such a cover can

be incident with at most $k' \cdot k'$ edges, and as it is a cover, it must be incident with every edge. As we never increase k' , clearly $k' \leq k$ (and so $k'^2 \leq k^2$). \square

Exercise 3.3. Show that with appropriate data structures Algorithm 3.1 runs in time $O(n + m)$.

As for G' we can check every possible set of size k' whether it is a cover in time $O(2^{k'(k'+1)} \cdot n^2) = O(2^{k(k+1)} \cdot k^4)$, we get the following.

Corollary 3.4. VERTEX-COVER is in FPT.

4 Second tool: bounded search tree

Algorithm 4.1. Initialize $T = \emptyset$, $G' = G$ and $k' = k$.

Call recursive algorithm $\text{VC}(G', k', T)$.

$\text{VC}(G', k', T)$: /* if there is vertex cover T' of size k' in G' we want to
return YES and $T \cup T'$

Let v be a vertex of G' with maximum degree. If $d'(v) = 0$ and $k' = 0$, then return YES and T ; if $k' = 0$ and $d'(v) > 0$, then return NO.

If $d'(v) = 1$, then let X consists of one end-vertex of each edge in E' , and if $|X| \leq k'$, then return YES and $T \cup X$, otherwise return NO.

From now on $d'(v) \geq 2$. Call $\text{VC}(G'-v, k'-1, T+v)$, if it returned YES we also return YES and the set we got as a cover. If it returned NO, then we also call $\text{VC}(G'-v-N'(v), k'-d'(v), T \cup N'(v))$ if $k' \geq d'(v)$; otherwise, if $k' < d'(v)$, then we return NO.

If this second call returned NO, then we also return NO, otherwise we return YES and the set we got as a cover from this second call.

Theorem 4.2. Algorithm 4.1 solves VERTEX-COVER. The number of calls is bounded by $2 \cdot 1.6181^k$, so the total running time is $O(1.6181^k \cdot (n + m))$.

Proof. The first statement is obvious. A run of the algorithm can be described by a binary tree, where the nodes are labeled by the recursive calls, especially the leaves are labeled by such calls where further recursion is unnecessary.

A binary tree having L leaves may have at most $2L$ nodes. At a non-leaf vertex labeled by (G', k') , one of its children is labeled by $(G'-v, k'-1)$, the other child is labeled by $(G'-v-N'(v), k'-d'(v))$ where $d'(v) \geq 2$. Let $\phi = \frac{\sqrt{5}+1}{2} < 1.6181$. It is easy to prove by induction that the number of leaves is bounded by F_k , the k -th Fibonacci number, which is known to be less than ϕ^k .

The running time is bounded by the number of calls multiplied by the number s of steps inside one call. It is easy to see that $s = O(n + m)$. \square

Exercise 4.3. a). If the maximum degree is two in a graph, then a minimum cover can be computed in $O(n)$.

b). Use this fact for modifying Algorithm 4.1, and prove that the new version has running time $O(1.466^k \cdot (n + m))$.

Now we turn to another crucial point. The two method discussed so far can be successfully applied together. First we make the kernel of the previous section in time $O(n + m)$, this kernel has size $n' + m' = O(k^2)$. We apply the above

mentioned bounded search tree method not to the original graph but to the kernel! Doing so the joint running time of the two parts is $O(n + m + 1.466^k \cdot k^2)$ (using the bound stated in Exercise 4.3).

5 Smaller kernel via crown reduction

The kernel we made may have more than k^2 vertices. Can we make a smaller one? Though we cannot make the bound on the number of edges in the kernel below k^2 , it is much nicer to get the smaller instance with only a linear number of vertices.

A crown decomposition is a partition of the vertex set into $V = C \cup H \cup B$ with the following properties.

- i) C is an independent set.
- ii) No edge connects C to B .
- iii) In the bipartite graph with classes H and C , there is matching covering H .

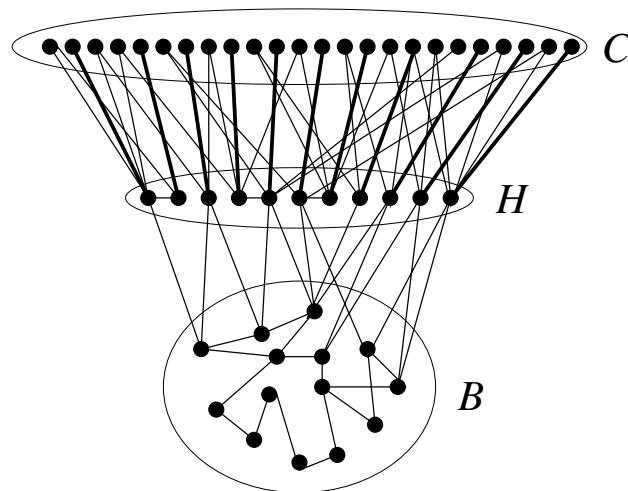


Figure 10: Crown decomposition

Statement 5.1. *There exist a k -element vertex cover in G if and only if there exist a $(k - |H|)$ -element vertex cover in $G[B] = G - H - C$.*

Proof. Let T be a vertex cover in $G[B]$. As C is independent, $T \cup H$ is a vertex cover in G .

Now let T be a minimum vertex cover in G . By Property iii) we have a matching M between H and C with size $|H|$. Its edges are covered by T , consequently $|T \cap (H \cup C)| \geq |H|$. The edges inside B are covered by $T \cap B$ and all the other edges can be covered by H . Consequently $T' = (T \cap B) \cup H$ is also a minimum vertex cover in G , so $|T \cap B| = |T| - |H|$. \square

We make another kernel for VERTEX-COVER where $n' \leq 3k$.

Algorithm 5.2. Initialize $T = \emptyset$, $G' = G$ and $k' = k$.

while $n' > 3k'$

Delete the isolated vertices


```

Let  $M$  be an inclusion-wise maximal matching in  $G'$ 
if  $|M| > k'$  then return(NO)

Let  $X$  consists of the  $M$ -covered vertices and  $I := V' - X$ 

Calculate the maximum matching  $M'$  in the bipartite graph  $G_b$ 
    containing the edges between  $I$  and  $X$ ;
    and a minimum vertex cover  $T'$  for  $G_b$ 

if  $|M'| > k'$  then return(NO)

if  $T' \cap X \neq \emptyset$  then
     $H := T' \cap X$ ;  $C := I - T'$ ;  $B := V' - H - C$ 
     $G' := G'[B]$ ;  $k' := k - |H|$ ;  $T := T \cup H$ 

return( $G', k', T$ )

```

Theorem 5.3. *If the algorithm stops with a NO answer, then this answer is correct. Otherwise, when it finishes, the current graph G' and the current k' describe an equivalent instance, and if T' is a vertex cover of G' , then $T \cup T'$ is a vertex cover of G . Moreover Algorithm 5.2 runs in polynomial time, $k' \leq k$ and when it returns (G', k') , then $n' \leq 3k$.*

Proof. Clearly any vertex cover must have size at least the size of any matching. As M is maximal, I is independent. If we did not stop after calculating M , then $|X| \leq 2k'$.

We calculate M' by the augmenting-path method of König, it also gives a minimum cover T' with the property $|M'| = |T'|$.

In the case $T' \cap X \neq \emptyset$ we claim that the partition given in the algorithm is a crown decomposition. C is independent as $C \subseteq I$. As T' is a vertex cover of G_b (and I is independent in G), edges from C must go into H . Finally as $|M'| = |T'|$, matching M' covers H , and the other end-vertices cannot be in T' , so they are in C .

When $T' \cap X = \emptyset$, we have $T' = I$ as we deleted all isolated vertices. Consequently $|I| = |M'| \leq k'$ and we deduced that $|X| \leq 2k'$, so $n' = |V'| = |I \cup X| \leq 3k' \leq 3k$.

With a small trick we can make Algorithm 5.2 to run in $O(n + k^2m)$. For this observe first that – as k' decreases – we have at most k iterations of the **while** loop. When we run König's augmentation algorithm we can stop after $k' + 1$ augmentations because at this point we already have $|M'| > k'$. \square

Exercise* 5.4. Make an even smaller kernel, namely of size $n' \leq 2k$.

Hint. A half-integral matching is a function $\mu : E \rightarrow \{0, 0.5, 1\}$ where at every vertex the sum of the μ -values on incident edges is at most one. Its size is $\sum_{e \in E} \mu(e)$. First show that the size of any half-integral matching is a lower bound for the minimum vertex cover. Then show that we can calculate the maximum size of a half-integral matching using König's algorithm for an appropriate bipartite graph, in time $O(km)$ if we may stop when the size grew above k (and this algorithm also calculates a half-integral vertex cover with the same size). Using this cover we can make the kernel required by just one crown reduction.

The current best algorithm for VERTEX-COVER runs in time $O(n + km + 1.274^k)$.

6 Another parameter for VERTEX-COVER

Is the bound on the size of the vertex cover is the right parameter? Usually, given a graph G , we are looking for the smallest vertex cover whose size is denoted by $\tau(G)$. Unfortunately, for a typical graph $\tau(G) = c \cdot n$, thus it is not a small value.

A much more practical parameter is the following. Let $\nu(G)$ denote the size of the maximum matching in G , we have already seen that $\tau(G) \geq \nu(G)$. The parametric problem VERTEX-COVER-ABOVE-MATCHING asks for a vertex cover with size at most $\nu(G) + k$.

Example 6.1. If G is a cycle on 1001 vertices, then $\tau(G) = 501$, thus VERTEX-COVER finds a cover if we run it with parameter 501. On the other hand $\nu(G) = 500$, so an algorithm solving VERTEX-COVER-ABOVE-MATCHING finds it with parameter 1.

Without any details we mention the following

Theorem 6.2. VERTEX-COVER-ABOVE-MATCHING can be solved in time $O(4^k \cdot n^c)$.

7 Longest path, randomization and color coding

Given a graph G and a vertex s , it is well-known that computing the longest path starting from s is an NP-hard problem. An essentially equivalent (see the remark at the end of this section) parametric problem is EXACT-PATH, when a parameter k is also given, and we ask for a path starting at s and have length exactly k .

Randomization helps a lot in many cases of algorithm design, we see now an example for EXACT-PATH. This method is called *color coding*.

We color vertices in $V - s$ randomly and independently by colors $C := \{1, 2, \dots, k\}$. We call a path (starting from s) C -colorful if its vertices (except s) have distinct colors and every color in C acts as a color of a vertex in the path. Note that if we have a C -colorful path, then its length is exactly k .

Statement 7.1. If we make $M = 185 \cdot e^k$ independent colorings of the graph, and neither admits a C -colorful path, then the probability of having a path starting at s with length exactly k is at most e^{-185} .

Proof. Suppose P is a path starting at s and have length exactly k . In a random coloring the probability that P is C -colorful is $k!/k^k$. So the probability that every coloring is bad is $(1 - k!/k^k)^M < (1 - e^{-k})^M < e^{-185}$. Remark: there are less than e^{185} particles in the (observable) universe. \square

Colorful paths in a colored graph can be find by dynamic programming.

Algorithm 7.2. The subproblems to solve: for every vertex $v \in V - s$ and for every color set $C' \subseteq C$ we decide whether there is C' -colorful path from s to v (and if the answer is YES, then we remember the last vertex before v).

So $\text{Cf}(v, C') = \bigvee_{u \in N(v)} \text{Cf}(u, C' - c(v))$ if $c(v)$, the color of v is an element of C' , otherwise $\text{Cf}(v, C') = \text{FALSE}$.

Algorithm 7.2 runs in time $O(2^k m)$, so with the repetition described in Statement 7.1 we get a randomized algorithm running in time $O((2e)^k m)$ with the following property. If we have a no-instance of EXACT-PATH, then the algorithm surely

answers NO, otherwise it returns a path of length k with probability at least $1 - e^{-185}$. (And, with probability at most e^{-185} , it answers erroneously NO).

Remark 7.3. Suppose in the input graph G the longest path starting from s has length K . We run the algorithm above successively for $k = 1, 2, \dots$ until we get a NO answer. The total running time will be

$$c \cdot m \cdot \sum_{k=1}^{K+1} (2e)^k < c \cdot m \cdot \frac{(2e)^{K+2}}{2e-1}.$$

8 Feedback vertex set and iterative compression

The last design method we discuss is *iterative compression*.

We show an example for this method through a new problem. In a graph G , a subset X of the vertices is called a *feedback vertex set* if $G-X$ is a forest (i.e., X blocks every cycle).

The problem FEEDBACK-VERTEX-SET asks whether – given G and k – a feedback vertex set of size at most k exists in G .

The basic idea of the method “iterative compression” is the following. Let $V = \{v_1, v_2, \dots, v_n\}$ and let V_i denote $\{v_1, v_2, \dots, v_i\}$ and $G_i = G[V_i]$. We are going to construct feedback vertex sets X_1, \dots, X_n for G_1, \dots, G_n of size at most k consecutively. For $i \leq k$ it is easy: $X_i = V_i$ suffices. (If for any i we have no such set, then we return NO.)

In a general step we are given X_{i-1} , a feedback vertex set of size at most k for G_{i-1} . Clearly $Z := X_{i-1} + v_i$ is a feedback vertex set of size at most $k+1$ for G_i . Starting with this as a hint, we are going to construct X_i . If $|Z| \leq k$, then we are done by defining $X_i := Z$, so we may assume that $|Z| = k+1$.

First we define a related problem DISJOINT-FEEDBACK-VERTEX-SET. Here G, k and a feedback vertex set $W \subseteq V$ is given, and the objective is to find a feedback vertex set $X \subseteq V - W$ of size at most k , or correctly conclude that no such feedback vertex set exists. We give an algorithm for DISJOINT-FEEDBACK-VERTEX-SET running in time $O(4^k \cdot n^c)$.

Denote $G - W$ by H ; now H is a forest (as W is a feedback vertex set), and if $G[W]$ is *not* a forest, then we may stop with answer NO (as the whole $V(H)$ is not a feedback vertex set).

First apply some basic reduction rules. Execute on of the following rules until none of them applies.

- i) If there exists $v \in V$ with $d(v) \leq 1$, then delete it.
- ii) If there exists a vertex v of H such that $G[W + v]$ contains a cycle, then include v in the solution, and let the new instance be $(G - v, W, k - 1)$.
- iii) If there is a vertex v of H of degree 2 in G with neighbors u and w , and u is a vertex of H , then delete v and add a new edge uw if $w \in V(H)$; otherwise (if $w \in W$) we have two possibilities. If uw is an edge, then include u in the solution, and let the new instance be $(G - u - v, W, k - 1)$. If uw is not an edge, then delete v and add a new edge uw .

Exercise 8.1. We made an equivalent instance.

From now on let (G, k, W) denote a reduced instance (none of the rules above applies). If $k = 0$, then we return YES if G is a forest, and NO otherwise.

Let x be a vertex of H with at most one neighbor in H (as H is a forest, such a vertex exists). Vertex x must have at least two neighbors in W (as rule i) and iii) were not applicable) but cannot have two neighbors in the same component of $G[W]$ because rule ii) was not applicable. Now we call recursively for $(G-x, W, k-1)$ and also for $(G, W+x, k)$. If both calls answer NO, then we also answer NO. If one of them answers YES, then we answer YES, and if this answer was given by the first call, then we also add x to the solution given by it.

As if there is a feedback vertex set required, then either x is inside it or not, thus this algorithm is correct.

Exercise* 8.2. Prove the running time $O(4^k \cdot n^c)$ claimed.

Hint. Take the value $c(G[W]) + k$ where $c(G[W])$ denotes the number of components. Prove that when we branch, this value strictly decreases for both the two new instances we create.

Now we remained to show how to use this algorithm for our original problem FEEDBACK-VERTEX-SET. We are looking for a feedback vertex set X_i of size at most k , given feedback vertex set Z of size exactly $k+1$. We branch on possible sets $Y = Z \cap X_i$ ($2^{k+1} - 1$ branches, as $Y = Z$ is not good as it has size $k+1$). Let $W := Z - Y$ and call DISJOINT-FEEDBACK-VERTEX-SET with $(G-Y, W, k-|Y|)$. If on any branch it returns YES and a feedback vertex set $X \subseteq V - Y - W$ with size at most $k - |Y|$, then we found the suitable $X_i := X \cup Y$. Otherwise, if on every branch we got NO, then we also answer NO.

Exercise* 8.3. Show that the total running time is $O(5^k n^{c+1})$.

Hint. Use Newton's binomial theorem.

9 Suggested reading

M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk and S. Saurabh: *Parameterized Algorithms*, Springer, 2015.

R.G. Downey and M.R. Fellows: *Parameterized complexity*, Springer, 1999.

J. Flum and M. Grohe: *Parameterized Complexity Theory*, Texts in Theoretical Computer Science (EATCS Series), Springer, 2006.

R. Niedermeier: *Invitation to fixed-parameter algorithms*, Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, 2006.

1 Introduction

Consider a workshop which produces gears for the automotive industry. In the workshop there are a number of production lines that are fed with cylindrical metal bars and cut out gears from them. At a high level, each production line is just one machine, and the shop manager's task is to assign customer orders to these lines such that the due dates of the orders are satisfied to the greatest extent. This is an example for scheduling a manufacturing shop. There is a finite set of tasks (obtained from the customer orders), each task has to be allocated to precisely one machine from a finite set of machines, and the tasks allocated to the same machine have to be ordered. An allocation of tasks to machines, and an ordering of tasks on each machine is sought such that the violation of due-dates is minimized. This is a rather complicated optimization problem.

As a further example, consider the scheduling of tasks on parallel computers: given an array of computers and computational tasks each specifying the number of processing units needed for their execution along with the expected running time. An assignment of tasks to processors is sought such that each task gets the required number of processors throughout its execution, and a timing of each task such that no processor has to process two tasks concurrently, and the maximum task completion time is minimized.

Yet another example is school timetabling, where teachers have to be matched to classes. Of course, no teacher can lecture in two classes simultaneously, and no class may be taught by two teachers concurrently. Each lecture is 45 minutes long. Each class has a list of teachers that have to give a lecture to it in some order. A time slot for each lecture is sought respecting the above constraints, and minimizing the maximum completion time of the lectures.

The above examples suggest that when we specify a scheduling problem, we have to define *resources*, and *jobs*, and also some constraints on how to assign resources to jobs in a feasible manner. In addition, some objective function to be minimized or sometimes maximized has to be given as well. The solution to such a problem is a *schedule* that assigns a resource or several resources to each job meeting the constraints, and which minimizes (or maximizes) the given objective function. The jobs will be indexed by j , the schedules will be denoted by \mathcal{S} , and the job completion times by C_j . Further notation will be defined at the first place of occurrence.

In all the scheduling models considered below we assume that each machine (or processor) can process at most one job (or task) at a time, and on each job at most one machine (processor) can work at a time. So, in the course of scheduling, jobs or fractions of jobs (in case of preemptive models) have to be ordered on each machine such that there is no overlap between the entities processed consecutively.

Scheduling problems are often denoted by the so-called three-field notation: $\alpha|\beta|\gamma$, where α denotes the processing environment, β some restrictions, and γ the objective function. For instance, α can be 1 denoting that we have a single machine only, or P representing a set of identical parallel machines. In the β field there may be a large variety of constraints, e.g., r_j indicates that the jobs have release dates (earliest date when a job can be started), or *pmtn* (preemption) stipulates that the processing of jobs can be stopped in the middle and resumed later. In the β field several constraints can be listed. In the γ field the objective function is usually a function of the job completion times, e.g., $C_{\max} = \max C_j$ is the maximum job completion time, and this is to be minimized as a rule of thumb, or $\sum C_j$ is the sum of job completion times, which is to be minimized as well. We will see several examples for the three-field notation when defining the scheduling problems in the following sections.

2 Single machine scheduling problems

One of the simplest scheduling problem is $1||C_{\max}$, i.e., minimize the maximum job completion time on a single machine.

INPUT: set of n tasks with processing times $p_1, \dots, p_n \in \mathbb{N}$.

OUTPUT: an ordering of tasks on the machine minimizing the maximum job completion time.

Since we have no reason to leave idle times between consecutive tasks on the machine, in any ordering of the tasks, the last task finishes at time $\sum_{i=1}^n p_i$. So, this scheduling problem admits an obvious algorithm: schedule the tasks in any order on the machine without idle times. The algorithm always delivers an optimal schedule.

2.1 Problem $1||\sum w_j C_j$

In the problem of this section, jobs have not only processing times, but weights as well, and we want to minimize the weighted sum of job completion times, i.e., solve the problem $1||\sum w_j C_j$.

INPUT: set of n tasks with processing times $p_1, \dots, p_n \in \mathbb{N}$, and weights $w_1, \dots, w_n \in \mathbb{N}$.

OUTPUT: an ordering of tasks on the machine minimizing the weighted sum of job completion times.

Surprisingly, this problem can be solved optimally by a scheduling the tasks in a non-increasing order of the w_j/p_j values without leaving idle times between the tasks. This simple rule is called WSPT in the scheduling literature.

Theorem 2.1. *If the tasks are scheduled in WSPT order without idle times, then we get an optimal schedule for $1||\sum w_j C_j$.*

Proof. Consider an optimal schedule \mathcal{S}^{opt} , and suppose it does not satisfy the ordering, i.e., there exist job indices j and k such that j is scheduled before k , but $w_j/p_j < w_k/p_k$. Clearly, if such a pair of jobs exists, then there is one where j and k are scheduled such that k starts right after j finishes on the machine. Now we swap these two jobs, and argue that the objective function strictly improves, which leads to a contradiction. That is, suppose j starts at time t in \mathcal{S}^{opt} , and completes at $C_j^{opt} = (t + p_j)w_j$. Then k starts at time $t + p_j$ (since there are no idle times between the jobs), and completes at $C_k^{opt} = t + p_j + p_k$. If we swap the jobs, then in the resulting schedule \mathcal{S}' , k starts at time t and completes at $C'_k = t + p_k$, and j at time $t + p_k$, and completes at $C'_j = t + p_j + p_k$, and the starting times of all other jobs do not change. Hence, the objective function value changes as follows:

$$-w_j C_j^{opt} - w_k C_k^{opt} + w_k C'_k + w_j C'_j.$$

We argue that this quantity is negative. To see this, we substitute in the completion times:

$$-w_j(t + p_j) - w_k(t + p_j + p_k) + w_k(t + p_k) + w_j(t + p_k + p_j) = -w_k p_j + w_j p_k < 0,$$

where the first equation holds since all other terms cancel out, and the last inequality follows from the assumption that $w_j/p_j < w_k/p_k$. Hence, \mathcal{S}' is a feasible schedule with a smaller objective function value than an optimal schedule, which is impossible.

Therefore, the optimal schedule must respect the WSPT ordering up to permutation of jobs with the same w_j/p_j values. The above argument also shows that if in some feasible schedule, j and k are consecutive jobs with $w_j/p_j < w_k/p_k$, then this schedule cannot be optimal. Hence, the WSPT ordering the tasks yields an optimal schedule. \square

2.2 Problem $1||L_{\max}$

Now we consider another objective function, namely, the maximum lateness. In this problem, jobs have processing times and due dates d_j , and a feasible schedule minimizing the maximum $C_j - d_j$ value is sought, i.e., $1||L_{\max}$.

INPUT: set of n jobs with processing times $p_1, \dots, p_n \in \mathbb{N}$, and due dates $d_1, \dots, d_n \in \mathbb{N}$.

OUTPUT: an ordering of jobs such that $L_{\max} = \max_j(C_j - d_j)$ is minimized.

Again, there is no advantage of leaving idle times between the jobs, so a schedule is merely an ordering of the jobs. Further on, we will show that sorting the jobs in non-decreasing d_j order always gives an optimal schedule. We call such an ordering EDD (earliest due-date).

Theorem 2.2. *Scheduling the tasks in EDD order without idle times gives an optimal schedule for $1||L_{\max}$.*

Proof. First we argue that there exists an optimal schedule in which the jobs are scheduled in EDD order. Suppose that \mathcal{S}^{opt} is an optimal schedule in which there is a pair of jobs j and k such that j precedes k , but $d_j > d_k$. Clearly, if such a pair exists, then there is a pair j and k such that k is scheduled right after j , and $d_j > d_k$. Let t be the starting time of j in \mathcal{S}^{opt} . Then $C_j^{opt} = t + p_j$, $C_k^{opt} = t + p_j + p_k$. Now swap j and k and let \mathcal{S}' be the resulting schedule. Then $C'_j = t + p_j + p_k$, $C'_k = t + p_k$, and $C'_i = C_i^{opt}$ for all the jobs $i \notin \{j, k\}$. Clearly, only the lateness of j and k may be different in the schedules \mathcal{S}^{opt} and \mathcal{S}' . We claim that

$$\max\{C_j^{opt} - d_j, C_k^{opt} - d_k\} \geq \max\{C'_j - d_j, C'_k - d_k\}.$$

To see this, observe that (i) the completion time of job k decreases by the swap, i.e., $C'_k < C_k^{opt}$, and (ii) $C'_k - d_k = C'_j - d_k > C'_j - d_j$, where the first equation follows from the definitions, and the second from our assumption $d_j > d_k$. Therefore, the maximum lateness of \mathcal{S}' cannot be greater than that of \mathcal{S}^{opt} . Clearly, repeated job swaps lead to a schedule respecting the EDD order.

Finally, note that any schedule respecting the EDD order must be optimal, because two distinct schedules each respecting the EDD order may differ only in the permutation of jobs with equal due dates. \square

The striking about the optimal schedules for $1||L_{\max}$ is that job processing times play no role at all.

2.3 Problems $1|r_j, pmtn|\sum C_j$ and $1|r_j, pmtn|L_{\max}$

Now we consider preemptive models, where the processing of the jobs may be interrupted and resumed later. So, a preemptive schedule consists of time intervals $[t_i, t_{i+1})$ for some $t_0 < t_1 < \dots < t_s$, where $t_0 = 0$, $s \geq n$ being the number of intervals, in each interval $[t_i, t_{i+1})$ the machine processes the same job, and for each job j , the total length of those intervals in which the machine processes j is precisely p_j . Notice that we do not allow intervals of zero length. The completion time of some job j in a preemptive schedule is t_{i+1} , if $[t_i, t_{i+1})$ is the last interval of the schedule where j is processed. Consider the problem $1|r_j, pmtn|\sum C_j$, in which preemption is allowed, jobs have release dates r_j , and the total job completion is to be minimized.

INPUT: a set of n jobs with processing times $p_1, \dots, p_n \in \mathbb{N}$, and release dates $r_1, \dots, r_n \in \mathbb{N}$.

OUTPUT: a preemptive schedule in which no job is processed before its release date and the sum of job completion times is minimized.

In order to define a rule for solving our preemptive problem, we need a new definition. The *remaining processing time of job j after time moment t* in a preemptive schedule is p_j minus the total time spent on processing j before t . The

shortest remaining processing time (SRPT) rule schedules at each time moment t the job with the least remaining processing time among all the jobs with release date at most t . Let $p_j(t)$ denote the remaining processing time of job j at time moment t in a specific schedule (which will be clear from the context). According to this rule, the machine may switch from one job to another if time t reaches a release date of a job, or a job gets fully processed. Hence, the number of preemptions is at most $2n - 1$.

Theorem 2.3. *The SRPT rule provides an optimal schedule for $1|r_j, pmtn|\sum C_j$.*

Proof. Consider an optimal schedule \mathcal{S}^{opt} , and suppose that it violates the SRPT rule, i.e., there is a time interval $[t_i, t_{i+1})$ such that the machine processes some job j with $r_j \leq t_i$, while there is another job k with $r_k \leq t_i$ such that $p_j(t_i) > p_k(t_i)$. Suppose t_i is the earliest time moment with this property. Let I be the set of intervals in the optimal schedule after t_i in which either j or k is processed. Clearly, the total length of the intervals in I is $p_j(t_i) + p_k(t_i)$. Now we reschedule the parts of j and k scheduled after t_i in the optimal schedule using the intervals in I . We start with the remaining part of job k , and then continue with job j . Let \mathcal{S}' denote the resulting schedule. Let C_j^{opt} and C_k^{opt} denote the completion time of jobs j and k , respectively, in \mathcal{S}^{opt} , and C'_j and C'_k the same quantities in \mathcal{S}' . Then we clearly have $\max\{C_j^{opt}, C_k^{opt}\} = \max\{C'_j, C'_k\}$, and $C'_k < C_k^{opt}$, because $p_j(t_i) > p_k(t_i)$ in \mathcal{S}^{opt} , but the machine processes j instead of k in the optimal schedule \mathcal{S}^{opt} . Since \mathcal{S}' is feasible, this contradicts the optimality of \mathcal{S}^{opt} . \square

Clearly, if all the job release dates are equal, then the SRPT rule provides an optimal schedule with no preemptions.

The following theorem shows that in a sense, the above result is the best possible.

Theorem 2.4. *Problem $1|r_j, pmtn|\sum w_j C_j$ is NP-hard.*

To solve the preemptive problem $1|r_j, pmtn|L_{\max}$, consider the following generalization of the EDD rule: at any time moment t , schedule the job with earliest due-date among those jobs with release date not greater than t .

Exercise 2.5. Show that the modified EDD rule provides an optimal schedule for $1|r_j, pmtn|L_{\max}$.

3 Parallel machine scheduling problems

In this section we consider scheduling problems in which there are more than one machines, and jobs have to be assigned to machines in the course of scheduling.

3.1 Problem $P||C_{\max}$

Suppose we have m parallel machines, n jobs, and each job can be processed on any of the machines, and the processing time is independent from the choice of the machine. The objective is to minimize the maximum job completion time, or makespan.

INPUT: a set of n jobs with processing times $p_1, \dots, p_n \in \mathbb{N}$, and the number of parallel machines, m .

OUTPUT: an assignment of jobs to machines, such that the maximum of the sum of processing times of those jobs assigned to a single machine is minimized.

In case of a single machine, this problem is trivial, however, even if there are only two parallel machines, it becomes intractable.

Theorem 3.1. *$P2||C_{\max}$ is NP-hard.*

Proof. There is an easy reduction from the well-known PARTITION problem. \square

In order to get good schedules, we apply list scheduling. In a *list scheduling algorithm* jobs are placed on a list in some order, and then scheduled one-by-one, by always assigning the next job to the machine where it can start the earliest. List scheduling algorithms differ in the ordering of jobs on the list.

Theorem 3.2. *Any list scheduling algorithm provides a feasible schedule of makespan at most $2 - 1/m$ times the optimum makespan.*

Proof. Let \mathcal{S} be the schedule obtained by a list scheduling algorithm. Let j be the job with maximum completion time. Suppose it starts at time t . By the rules of list scheduling, no machine finishes processing before t , otherwise j could be started earlier. Hence, the total processing of all the jobs except job j is at least $t \cdot m$. Hence, we have

$$C_{\max}(\mathcal{S}) = t + p_j \leq \left(\sum_{i=1, i \neq j}^n p_i \right) / m + p_j = \left(\sum_{i=1}^n p_i \right) / m + p_j(1 - 1/m) \leq (2 - 1/m)C_{\max}^{opt},$$

where C_{\max}^{opt} denotes the optimum makespan, the first inequality follows from the fact that $tm \leq \left(\sum_{i=1, i \neq j}^n p_i \right)$, and the second from the fact that both p_j and $\left(\sum_{i=1}^n p_i \right) / m$ are lower bounds on C_{\max}^{opt} . \square

The LPT rule places the jobs in non-increasing processing time order on the list, and then apply list scheduling.

Theorem 3.3. *The LPT rule provides a feasible schedule of makespan at most $4/3$ times the optimum.*

Proof. Suppose job j is the first job scheduled by the LPT rule which has the maximum job completion time in the schedule \mathcal{S} for some instance of $P||C_{\max}$. We obtain a new instance by dropping all jobs following j in the list. Clearly, for the new instance, the LPT rule would deliver a schedule with the same makespan as for the original problem instance. We distinguish two cases:

- $p_j \leq C_{\max}^{opt}/3$. Then using the derivation of the previous theorem, we conclude that

$$C_{\max}(\mathcal{S}) \leq \left(\sum_{i=1}^n p_i \right) / m + p_j(1 - 1/m) \leq (1 + (1 - 1/m)/3)C_{\max}^{opt} \leq \frac{4}{3}C_{\max}^{opt}.$$

- $p_j > C_{\max}^{opt}/3$. Since p_j is the last element of the list for the modified problem instance, all job processing times are greater than $C_{\max}^{opt}/3$. Hence, there are at most $2m$ jobs, and in the optimal schedule in machine received at most two jobs. Moreover, the list scheduling algorithm constructs a schedule in which job $m + i$, if exists, is scheduled with job $m + 1 - i$. Observe that an optimal schedule is of the same structure, hence, the LPT rule delivers an optimal schedule. \square

Now we turn to other objective function, the total completion time.

Theorem 3.4. *The problem $P||\sum C_j$ can be solved in polynomial time.*

Proof. The algorithm for solving this problem is based on solving a bipartite matching problem to optimality. To this end, observe that in an optimal schedule, there is no idle time on any machine, and if, say, jobs 1 through k are assigned to a machine in this order, then the completion time of job j' is $\sum_{j=1}^{j'} p_j$, and thus

$$\sum_{j'=1}^k C_{j'} = \sum_{j'=1}^k \sum_{j=1}^{j'} p_j = \sum_{j=1}^k (k+1-j)p_j.$$

Therefore, the contribution of job k (this is the last job on the machine) to the objective function is p_k , job $k - 1$ contributes $2p_{k-1}$, etc.

Now, we define a bipartite graph $G = (V_1 \cup V_2, V_1 \times V_2)$, where V_1 consists of all the jobs, V_2 has a node for each pair (i, ℓ) with $1 \leq i \leq m$, and $1 \leq \ell \leq n$, and each job in V_1 , is connected to each node in V_2 . The weight of edge $(j, (i, \ell))$ is $(n + 1 - \ell)p_j$, which is the contribution of job j to the objective function if it was scheduled to position ℓ on machine i . Notice that we permit that some positions may be left unassigned.

We claim that an optimal schedule corresponds to a matching covering all the job nodes of minimum total weight, and conversely, any minimum weight matching of minimum total weight represents an optimal schedule. From this claim, we can conclude that an optimal schedule can be found in polynomial time by determining a minimum weight matching in G covering all the nodes in V_1 .

Now we verify our claim. Consider an optimal schedule, and defined the matching corresponding to the jobs scheduled on each machine in turn. For machine i , say, the job scheduled last is matched with the node $(i, n) \in V_2$, the penultimate job with node $(i, n - 1)$, etc. It is easy to verify that we obtain a matching covering each job node, and the weight of the matching equals the total completion time of the jobs in the optimal schedule. Conversely, suppose we have minimum weight matching covering all the job nodes. By the definition of edge weights, for each i , if node $(i, \ell) \in V_2$ is matched, then all the positions (i, ℓ') with $\ell \leq \ell' \leq n$ are matched as well. We construct a feasible schedule by putting job j matched with node $(i, n + 1 - \ell) \in V_2$ in position ℓ backward on machine i , i.e., $\ell = 1$ corresponds to the last position, $\ell = 2$ to the penultimate positions, etc. Again, it is easy to verify that the schedule obtained has the same objective function value as the weight of the matching. \square

As a by product, we have obtained a new method for solving the problem $1||\sum C_j$, which we have solved in Section 2.1. Further on, by the equivalence shown in the proof of Theorem 3.4, if we have a complete bipartite graph $K_{n,n}$ with edge weights equivalent to those derived from a scheduling problem $1||\sum C_j$, then the minimum weight perfect matching can be found by sorting.

The above result can be generalized to a more general scheduling problem in which the processing time of each job may depend on the machine assigned to it as well, i.e., in the input, the processing time p_{ij} of each job j on each machine i is given. Such a processing environment is called *unrelated machines*, and is denoted by R . It is easy to generalize Theorem 3.4 to the problem on unrelated machines:

Theorem 3.5. *The problem $R||\sum C_j$ can be solved in polynomial time.*

We leave the proof as an exercise to the reader.

4 Open shop scheduling

In an open shop, there are m machines / processors, and n jobs, each job being a set of m tasks $T_{1,j}, \dots, T_{m,j}$, where $T_{i,j}$ is to be processed on machine i . The tasks of the same job can be processed in arbitrary order, but concurrent processing of any two tasks of the same job is not allowed. Each task has a non-negative integer processing time $p_{i,j}$. Moreover, each machine can process only one task at a time. More formally, the makespan minimization problem can be defined as follows.

INPUT: number of machines m , number of jobs n , task processing times $p_{i,j} \in \mathbb{Z}_{\geq 0}$, $1 \leq i \leq m$, and $1 \leq j \leq n$.

OUTPUT: a schedule \mathcal{S} specifying the starting time $S_{i,j}$ of each task $T_{i,j}$ such that for any two tasks, say, $T_{i,j}$ and $T_{k,j}$ of the same job, $S_{i,j} + p_{i,j} \leq S_{k,j}$ or $S_{k,j} + p_{k,j} \leq S_{i,j}$, and for any two tasks, say, $T_{i,j}$ and $T_{i,j'}$ on the same machine $S_{i,j} + p_{i,j} \leq S_{i,j'}$ or $S_{i,j'} + p_{i,j'} \leq S_{i,j}$, and the maximum completion time over all tasks is minimized.

This problem is denoted by $O||C_{\max}$, where O stands for *open shop*.

Theorem 4.1. *Problem $O||C_{\max}$ is NP-hard.* □

First we consider a special case with $\{0, 1\}$ processing times. Notice that there is no advantage in delaying any tasks, so we consider only schedules in which no task may start earlier without violating feasibility. We call such a schedule *dense*. It follows that in a dense schedule each task starts at some integer time point, and at any moment of time between 0 and the maximum job completion time, at least one machine is working. We will represent schedules by means of edge colorings of a bipartite graph $G = (V_1, V_2, E)$, where $|V_1| = m$, $|V_2| = n$, and there is an edge connecting $i \in V_1$ and $j \in V_2$ if and only if $p_{i,j} = 1$, i.e., the processing time of $T_{i,j}$ is 1. An *edge coloring* is a set of matchings covering each edge exactly once.

Lemma 4.2. *Each feasible schedule of $O|p_{i,j} \in \{0, 1\}|C_{\max}$ can be represented by an edge coloring of G , and conversely, any edge coloring of G corresponds to a feasible schedule.*

Proof. Consider a schedule \mathcal{S} . Since each task starts at some integer time point, for each $t \in \{0, 1, \dots, C_{\max}(\mathcal{S}) - 1\}$, let M_t be the set of tasks starting at time t . Since each task is an edge of G , the M_t , $t = 0, \dots, C_{\max}(\mathcal{S}) - 1$, constitute a set of matchings covering each edge exactly once.

Conversely, consider an edge coloring of G consisting of the matching M_t , $t = 0, \dots, h - 1$. Then schedule all tasks of M_t concurrently, for $t = 0, \dots, h - 1$. Since each M_t is a matching of G , the resulting schedule is feasible and the last job completes at time h . □

As a consequence, to find an optimal schedule, it suffices to find an edge coloring of G with a minimum number of colors. How to find such an edge coloring?

Theorem 4.3. *Any bipartite graph admits an edge coloring with $\Delta(G)$ colors, where $\Delta(G)$ is the maximum degree of a node. Moreover, such a coloring can be found in polynomial time.*

Notice that $\Delta(G)$ is a lower bound on the number of colors of any edge coloring of G , since any matching may contain at most one edge adjacent to any node.

Proof. Firstly, notice that in a bipartite graph with maximum degree 2, there always exists an edge coloring with two colors (such a graph consists of a set of paths, and even-length cycles: from each cycle, and path pick every second edge, they constitute one matching, and the remaining edges the second matching).

Now suppose that $\Delta(G) \geq 3$. To prove the theorem, we construct a set of $\Delta(G)$ maximal matchings, M_t , $t = 0, \dots, \Delta(G) - 1$, of G in a greedy manner. Suppose some edge (i, j) of G is not contained in any M_t . If there were a matching M_t in this set which has no edge adjacent to i and no edge adjacent to j , then M_t would not be maximal, which contradicts the choice of M_t . Hence, there exists M_t with no edge adjacent to i , but having an edge adjacent to j , and another matching M_u with no edge adjacent to j , but having an edge adjacent to i . Consider the set of edges $\{(i, j)\} \cup M_t \cup M_u$. These edges determine a bipartite graph of maximum node degree 2. Hence, by our initial observation, it admits an edge coloring with two colors, say M'_t and M'_u . Replace M_t and M_u with M'_t and M'_u , and repeat this transformation until all edges get covered by some matching.

The running time of this algorithm is polynomial in the size of G , since the construction of the initial matching takes polynomial time, and then we need at most $|E|$ steps to obtain an edge coloring of G , each taking linear time. \square

As a byproduct, we obtain that $O\{p_{i,j} \in \{0,1\} \mid C_{\max}\}$ always admits an optimal solution with $C_{\max} = \max\{\max_j \sum_{i=1}^m p_{i,j}, \max_i \sum_{j=1}^n p_{i,j}\}$. Further on, the roles of machines and jobs can be exchanged without affecting the value of the optimal solution.

As an application to $O\{p_{i,j} \in \{0,1\} \mid C_{\max}\}$, consider school timetabling, where the machines/processors correspond to teachers, and the jobs to classes, and each edge (i,j) to a lecture given by teacher i to class j . Each lecture is 45 minutes long, and our goal is to minimize the number of time slots needed to schedule all the lectures.

Now we turn to $O \mid C_{\max}$. Since this problem is NP-hard, we provide a polynomial time approximation algorithm to solve it. Recall the definition of dense schedules.

Theorem 4.4. *Any dense schedule has a makespan of at most two times the optimum makespan.*

Proof. Suppose \mathcal{S} is a dense schedule, and let $T_{i,j}$ is a task with maximum completion time, i.e., $C_{\max}(\mathcal{S}) = S_{i,j} + p_{i,j}$. Since $T_{i,j}$ cannot be started earlier as the schedule is dense, at any moment of time before $S_{i,j}$, either machine i is processing some other task, or other task of job j is processed. Hence,

$$C_{\max}(\mathcal{S}) \leq \sum_{k=1}^m p_{k,j} + \sum_{j'=1}^n p_{i,j'} \leq 2C_{\max}^*,$$

where C_{\max}^* denotes the optimum makespan, and the second inequality follows from the fact that both terms are lower bounds on C_{\max}^* . \square

It is easy to construct a dense schedule. At time 0, pick a maximal set of tasks, each from a different job, and requiring a different machine, and schedule them concurrently. Then, whenever a task of a job completes, schedule another, yet unscheduled task of the same job at the earliest time on the required machine without modifying the schedule of the already scheduled tasks.

Finally, let us consider the preemptive version of this problem, $O \mid pmtn \mid C_{\max}$. In this problem, we assume that preemption of processing may occur only at integral time points, and all tasks may start only at integral time points.

Theorem 4.5. *Problem $O \mid pmtn \mid C_{\max}$ can be solved in polynomial time.*

Proof. We model the problem of a bipartite graph $G = (V_1, V_2, E)$, where V_1 has precisely one node for each job, V_2 has precisely one node for each machine, and E contains an edge for each machine-job pair (i,j) with $p_{i,j} > 0$. We also associate the weight $p_{i,j}$ with edge (i,j) . Let $\Delta_p(i) := \sum_{j=1}^m p_{i,j}$ and $\Delta_p(j) := \sum_{i=1}^n p_{i,j}$ denote the total processing time of those tasks requiring machine i , and the total processing time of job j , respectively. Let $\Delta_p^{\max} := \max\{\max_i \Delta_p(i), \max_j \Delta_p(j)\}$ be the maximum value of these quantities. A *weighted edge coloring* of G is a set of matchings M_1 through M_h along with multiplicities μ_1 through μ_h such that for each edge (i,j) , the total multiplicity of those matchings containing edge (i,j) is $p_{i,j}$, that is, $\sum_{t: (i,j) \in M_t} \mu_t = p_{i,j}$. We claim that there exists a weighted edge coloring with total multiplicity Δ_p^{\max} , which is a lower bound on the length of any feasible schedule (why?).

To prove our claim, we define an auxiliary bipartite graph H from G . Firstly, make a copy of G , call it $G' = (V'_1, V'_2, E')$. Let $H = (V_1 \cup V'_1, V_2 \cup V'_2, E \cup E' \cup \{(i,i') : i \in V_1, \Delta_p(i) < \Delta_p^{\max}\} \cup \{(j,j') : j \in V_2, \Delta_p(j) < \Delta_p^{\max}\})$, where the copy of $i \in V_1$ in V'_1 is i' , and the copy of $j \in V_2$ in V'_2 is j' . The weight of the new edge (i,i') is $p_{i,i'} := \Delta_p^{\max} - \Delta_p(i)$, and that of (j,j') is $p_{j,j'} := \Delta_p^{\max} - \Delta_p(j)$. Therefore, in graph H , the total weight of those edges adjacent to any node is Δ_p^{\max} . We call

such a graph *regular*. We argue that in regular graphs there always exists a perfect matching. To see this, we verify that H verifies the familiar Hall condition: let $S \subseteq V_1 \cup V_2'$, and $\Gamma(S) \subseteq V_2 \cup V_1'$ its neighbors in H . We have to show that $|\Gamma(S)| \geq |S|$. Suppose not. Then regularity implies

$$|S|\Delta_p^{\max} > |\Gamma(S)|\Delta_p^{\max} \geq \sum_{u \in S} \sum_{(u,v) \in E(H)} p_{u,v} = |S|\Delta_p^{\max},$$

where the first inequality follows from our indirect assumption, the second from the counting of edges between S and $\Gamma(S)$, and the third from regularity. Therefore, H admits a perfect matching \tilde{M}_1 . Let μ_1 be the minimum multiplicity of those edges in \tilde{M}_1 . Since H is regular, and \tilde{M}_1 is a perfect matching, $H - \mu_1 \tilde{M}_1$ is a regular graph again. so we repeat. In at most $|E(H)|$ steps, all edges vanish, and the procedure terminates. This gives a set of perfect matching \tilde{M}_1 through \tilde{M}_h of H along with multiplicities μ_1 through μ_h that constitute a weighted edge coloring of H . Then we define a weighted edge coloring of G by letting $M_t := \tilde{M}_t \cap E$ for $t = 1, \dots, h$. Since $\sum_{t=1}^h \mu_t = \Delta_p^{\max}$, we are done. \square

5 Recommended readings

1. Peter Brucker, *Scheduling theory*, Springer Verlag, 2007.
2. Jacek Blazewicz, Klaus Ecker, Erwin Pesch, Günter Schmidt, Jan Weglarz, *Handbook on Scheduling*, Springer Verlag, 2007.

The exact mathematical model of a network process, like epidemic propagation on a graph, can be formulated as a large system of linear ordinary differential equations. The mathematical model is given by a graph the nodes of which can be in different states, in the case of epidemic dynamics each node can be either susceptible or infected. The state of the network containing N nodes is given by an N -tuple of S and I symbols, i.e. there are 2^N states altogether. The transition rules determine how the state of the network evolves by infection from one node to its neighbours and by recovery, when an I node becomes S again. The system of master equations is formulated in terms of the probabilities of the states, i.e. the system consists of 2^N differential equations. Similar differential equations can be derived for modeling neural activity in a neural network (when each neurone, a node of the network, can be either active or inactive) or for the voter model describing the collective behaviour of voters.

Despite of the fact, that the mathematical model is relatively simple, the analytical or numerical study of the system can be carried out only for small graphs or graphs with many symmetries like the complete graph or the star graph. For real-world large graphs the master equations are beyond tractability, hence the system is approximated by simple non-linear differential equations, called mean-field equations. These models give accurate approximation for random graphs, like the Erdős-Rényi graph, regular random graphs, configuration random graphs with a given degree distribution and power-law random graphs. The mean-field equations are derived for these graphs and are studied by the methods of dynamical system theory.

1 Introduction

Many systems of interacting units are naturally represented and visualised as a network with nodes representing the units, and edges representing the interactions. This representation is conceptually simple and intuitive, leading to the use of the network paradigm across fields as diverse as social sciences, neuroscience, biology, physics, linguistics, finance, engineering, computer science, archaeology, and marketing.

The well-established mathematical field of graph theory, which is concerned with the rigorous analysis of structural properties of graphs, can be rightly considered to be a precursor of network science as it is known today. The textbook by Diestel [5] gives a good overview of graph theory, while the highly relevant branch of random graphs can be found in the monograph by Bollobás [1]. The description of large networks by using graph limits is presented in the book by Lovász [12].

However, graph theory and network science have different emphasis, even though the mathematical object studied is the same. Network science goes back only a few decades. Introductions to it can be found in the book by Newman [13], the monograph by Caldarelli [2], the textbook by Estrada [6] and the book by Cohen and Havlin [3].

The focus of this lecture notes is not on the structure of networks but the processes evolving on them. Several processes from epidemic and rumour spread to neuronal activity will be shown with emphasis on epidemic propagation. The exact system of master equations will be presented only for small graphs, because it contains exponentially many equations, so for large graphs it is not tractable even numerically. Then the focus will be on mean-field approximations that are low dimensional systems of ordinary differential equations. After introducing the mean-field and so-called pairwise approximations, they are studied by using the methods of dynamical systems. The steady states, their stability and the global behaviour of the systems are determined.

We note that more sophisticated models, describing processes on networks with degree heterogeneity, such as het-

erogeneous mean-field, heterogeneous pairwise and compact pairwise models are not dealt with in this lecture notes. These systems can be found in the review papers [4, 7, 15].

2 Networks

Real-world networks can be represented mathematically by graphs. A graph G is given by a pair (V, E) , where V is a set, the elements of which are called vertices (or nodes in the network context), and $E \subset V \times V$ is a set of pairs of vertices, called edges (or links in the network context). For example, a line graph with three vertices is given by $V = \{1, 2, 3\}$ and $E = \{(1, 2), (2, 1), (2, 3), (3, 2)\}$. The graph is represented visually by points (vertices) connected by segments (edges). In the case of the above line graph the pairs $(1, 2)$ and $(2, 1)$ are represented by a single segment connecting vertices 1 and 2. If each edge appears in both directions in the set E , then the graph is called un-directed, otherwise the graph is called directed. For example, the line graph given by $V = \{1, 2, 3\}$ and $E = \{(1, 2), (2, 3)\}$ is a directed graph.

For a finite graph with N vertices the vertex set is taken as $V = \{1, 2, \dots, N\}$. The set E can be encoded in the *adjacency matrix* whose entry g_{ij} is one if $(i, j) \in E$, otherwise it is zero. A finite graph is determined uniquely by its adjacency matrix. The matrix is symmetric if and only if the graph is un-directed.

Theoretical models of networks are often sampled from a set of networks that observe certain prescribed properties. These are called random graphs. One of the most well-known and widely studied random graph model is the Erdős-Rényi graph with N nodes and m edges. The set of all such graphs is $\{G_1, G_2, \dots, G_n\}$, where $n = \binom{M}{m}$ with $M = N(N-1)/2$, and the probability of picking each graph is the same, namely $1/n$. A closely related model which unfortunately is also usually referred to as the Erdős-Rényi model assigns each edge independently with probability p . Then every graph on N nodes is possible, but the probability of having a particular m -edge graph is $p^m(1-p)^{M-m}$. For “ k -regular” random graphs (random graphs in which all nodes have k neighbours), the set of graphs $\{G_1, G_2, \dots, G_l\}$ consists of all k -regular (labeled) graphs and the probability of picking each graph is the same. More generally, a random graph is a probability distribution over a certain set of graphs. Formally, a random graph is given by a set of graphs $\{G_1, G_2, \dots, G_n\}$ and a set of nonnegative numbers $\{p_1, p_2, \dots, p_n\}$, for which $p_1 + p_2 + \dots + p_n = 1$. A realisation of this random graph is G_i with probability p_i .

The exact representation of the graph by using its adjacency matrix is not feasible for large graphs, hence the network is characterised by some network metrics instead. A major goal in network science is to measure and understand how real-world networks emerge and parametrise these via some graph or network theoretical measures such as degree, degree distribution, preferential mixing, clustering etc. Below we provide a succinct overview of the main network metrics.

Degree and degree distribution: The most important characteristic is the degree distribution of the graph. The out-degree of a vertex is the number of edges starting from it. Expressed in terms of the adjacency matrix, $k_i^{out} = \sum_{j=1}^N g_{ij}$. The in-degree of a vertex is the number of edges going to it, i.e. $k_i^{in} = \sum_{j=1}^N g_{ji}$. For an undirected graph these are equal, this number is called the degree of the node and is denoted by k_i . The average degree of a graph is

$$\langle K \rangle = \frac{1}{N} \sum_{i=1}^N k_i = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^N g_{ij}.$$

If every vertex has the same degree then the graph is called regular.

If the degrees occurring in the graph are denoted by d_1, d_2, \dots, d_L and N_l is the number of vertices with degree d_l , then the degree distribution is $p_l = N_l/N$. For a regular graph $L = 1$ and $N_1 = N$, that is all nodes have the same degree d_1 .

For bimodal graphs there are two different degrees: d_1 and d_2 , i.e. $L = 2$. For an Erdős-Rényi graph every degree from 0 to $N - 1$ may occur, that is $L = N$, $d_l = l - 1$ and it can be easily shown that the distribution p_l is binomial. We note that in many cases each degree in the range 1 to M may occur as a degree in the graph. In that case $d_l = l$ for all $l = 1, 2, \dots, M$, hence the notation d_l is not needed. Instead, the degree of a general node is denoted by k , the number of nodes of degree k is denoted by N_k and the degree distribution is given by $p_k = N_k/N$, where $N = N_1 + N_2 + \dots + N_M$ is the total number of nodes.

Assortative or disassortative mixing: The next level of characterising the graph is to determine how the nodes with different degrees are connected to each other. This can be specified by an $L \times L$ matrix with entries n_{lj} yielding the total number of edges connecting nodes of degree l with nodes of degree j . These numbers should satisfy the compatibility conditions

$$\sum_{j=1}^L n_{lj} = d_l N_l, \quad \text{for all } l = 1, 2, \dots, L.$$

In the case of random mixing, when the edges of the nodes are connected randomly, we have

$$n_{lj} = \frac{d_l d_j N_l N_j}{\sum_{i=1}^M d_i N_i}.$$

If this relation does not hold then the mixing is called preferential.

Specifying the degree distribution p_l and the mixing coefficients n_{lj} does not determine the graph structure. This can be seen even on small graphs. For example, a 3-regular graph on $N = 6$ nodes can have two different manifestations, both of them are hexagons with three diagonals. In the first one the diagonals are $(1, 4), (2, 6), (3, 5)$. In the second one the diagonals are $(1, 4), (2, 5), (3, 6)$. It is easy to see that they are different (not isomorphic) graphs.

Clustering and higher order structure: The example above shows that the characterisation of the network requires further graph properties beyond p_k and n_{lj} . The most frequently used one is clustering, which is the ratio of triangles to triples. Heuristically, this measures the propensity that two neighbours of a randomly chosen node share an edge, and thus creating a triangle. It turns out that other higher order structures, over more than three nodes, may also be relevant. Hence, instead of triangles further subgraphs may be taken into consideration.

3 Processes

When processes on networks are modelled, individuals are represented by nodes and the contact pattern amongst these is encoded by the edges of the network. Births and deaths of nodes or edges are for the time being ignored. To set up the model, the status of each node needs to be defined. For most cases in this lecture notes we consider epidemic spread with the following statuses: susceptible S , infected and infectious I , and recovered/immune/removed R . Hence, at any time instant, the state of the whole network is given by the status of N nodes. This will of course change in time and the dynamics of the epidemic give the rates of change of the statuses of the nodes. Similar mathematical description can be used to model the spread of neuronal activity, when the nodes are the neurones, each of which can be quiescent or active, or modelling the behaviour of voters, when the nodes are individuals, the statuses of which are characterised by opinions.

In mathematical language, this means that the model is formulated in terms of the functions $X_i(t)$ yielding the status of node i at time t , for example, $X_i(t) = S$ or $X_i(t) = I$ when an SIS (susceptible-infected-susceptible) epidemic is considered. The goal of a mathematical description is to formulate models determining these functions. Such models take into account the structure of the network and the dynamics of the propagation process.

The aim of this section is to show how to formulate master equations for a process spreading on an arbitrary network, assuming that transitions of a node depend only on the statuses of the node and its neighbours. The network is given by the adjacency matrix of the corresponding graph with N nodes: $G = (g_{ij})_{i,j=1,2,\dots,N}$. Let $\{Q_1, Q_2, \dots, Q_m\}$ be the possible statuses nodes can take. Then a state of the network can be specified by an N -tuple (q_1, q_2, \dots, q_N) where $q_i \in \{Q_1, Q_2, \dots, Q_m\}$ is the status of node i , that is the number of all possible states of the network is m^N .

To aid the construction of our mathematical models, we make a few explicit assumptions:

1. time is considered to be continuous,
2. a node has only a finite number of possible statuses,
3. the process is stochastic,
4. the inter-event times are exponentially distributed,
5. the parameter of this exponential distribution may depend on the status of the node and on the number of neighbouring nodes of each status,
6. transitions at different nodes are independent.

Ideally we would like to know the probability the system has a given state at a given time, but this is likely to be impractical. Thus the aim of the investigation is to determine

- the probability that a node has a given status at a given time,
- the expected number of nodes having a given status at a given time.

These assumptions lead to a continuous time Markov chain with finite state space. This is a widely used and intensively studied field of mathematics with an extensive literature both from the theoretical and applied points of view. Our goal here is not to introduce Markov chains in general. For this, any of the following provides a good point of reference [8, 11]. Our topic is restricted to a special class of Markov chains which originate from exact propagation models on networks.

To motivate our approach and illustrate the processes we consider, we will analyse the state-space of an SIS type disease propagating on a triangle. We will derive equations governing the probabilities the system is in each state. Then we will derive a reduced system that captures the important details of the model. Several other dynamics will be shown in the remainder of the section.

Example 3.1. The state space of the Markov chain for an SIS disease in a triangle is

$$\{SSS, SSI, SIS, ISS, SII, ISI, IIS, III\},$$

where, for example SSI represents the state where the nodes 1 and 2 are susceptible and node 3 is infected. At most one transition can happen at a time (though the time between transitions can be arbitrarily small). We define X_{ABC} to be the probability that the state of the network is ABC , where A , B , and C may be S or I . If the transition corresponds to recovery of a single node, it occurs at rate γ , while if it corresponds to infection of a single node, it occurs at rate τ times the number of infected neighbours. For example, if the system is in state SII the rate at which it moves to SIS is γ . In contrast, the rate at which the system moves to III is 2τ . We note that the flux of probability from state ABC

to another is given by the rate at which that transition happens times X_{ABC} . Determining all possible transitions at all states carefully, yields

$$\begin{aligned}
 \dot{X}_{SSS} &= \gamma(X_{SSI} + X_{SIS} + X_{ISS}), \\
 \dot{X}_{SSI} &= \gamma(X_{SII} + X_{ISI}) - (2\tau + \gamma)X_{SSI}, \\
 \dot{X}_{SIS} &= \gamma(X_{SII} + X_{IIS}) - (2\tau + \gamma)X_{SIS}, \\
 \dot{X}_{ISS} &= \gamma(X_{ISI} + X_{IIS}) - (2\tau + \gamma)X_{ISS}, \\
 \dot{X}_{SII} &= \gamma X_{III} + \tau(X_{SSI} + X_{SIS}) - 2(\tau + \gamma)X_{SII}, \\
 \dot{X}_{ISI} &= \gamma X_{III} + \tau(X_{SSI} + X_{ISS}) - 2(\tau + \gamma)X_{ISI}, \\
 \dot{X}_{IIS} &= \gamma X_{III} + \tau(X_{SIS} + X_{ISS}) - 2(\tau + \gamma)X_{IIS}, \\
 \dot{X}_{III} &= -3\gamma X_{III} + 2\tau(X_{SII} + X_{ISI} + X_{IIS}).
 \end{aligned}$$

These equations are the *master equations* of the system and are also called *forward Kolmogorov equations*. A solution to the master equations gives the probability that the system is in each state at each given time.

The equations become simpler if we are willing to settle for just knowing the probability a given number of nodes have each status. Rather than calculating a given state's probability, we calculate the probability the system is in one of several states collected or grouped based on how many nodes are infected. Define Y_i to be the probability of having exactly i nodes with status I ; that is $Y_0 = X_{SSS}$, $Y_1 = X_{ISS} + X_{SIS} + X_{SSI}$, $Y_2 = X_{IIS} + X_{ISI} + X_{SII}$, and $Y_3 = X_{III}$; then these equations reduce to

$$\dot{Y}_0 = \gamma Y_1, \tag{1a}$$

$$\dot{Y}_1 = 2\gamma Y_2 - \gamma Y_1 - 2\tau Y_1, \tag{1b}$$

$$\dot{Y}_2 = 2\tau Y_1 - 2\gamma Y_2 - 2\tau Y_2 + 3\gamma Y_3, \tag{1c}$$

$$\dot{Y}_3 = 2\tau Y_2 - 3\gamma Y_3. \tag{1d}$$

For illustration we show several processes on networks: SIS; QAQ with a hyperbolic tangent transition rate and a voter-like model with two statuses, where both transitions depend on the status of the neighbours.

Example 3.2. Consider an SIS type disease propagating on a network. A node can be susceptible (S) or infected (I). Using the above notation let $Q_1 = S$ and $Q_2 = I$. There are two transitions: infection and recovery. In order to specify the dynamics the transition rates $f_{SI}(n_1, n_2)$ and $f_{IS}(n_1, n_2)$ have to be specified, yielding the rate at which a node with n_1 susceptible and n_2 infected neighbours changes its status from S to I and from I to S . In this case $f_{SI}(n_1, n_2) = \tau n_2$ with some parameter τ , which is called per-contact infection rate and $f_{IS}(n_1, n_2) = \gamma$ with some parameter γ , which is called recovery rate. That is, the infection rate depends linearly on the number of infected neighbours, while the recovery rate is independent of the statuses of the neighbours.

Example 3.3. Consider an SIR type disease propagating on a network. A node can be susceptible (S), infected (I) or recovered (R). Using the above notation let $Q_1 = S$, $Q_2 = I$ and $Q_3 = R$. There are two transitions: infection and recovery. In order to specify the dynamics the transition rates $f_{SI}(n_1, n_2, n_3)$ and $f_{IS}(n_1, n_2, n_3)$ have to be specified, yielding the rate at which a node with n_1 susceptible, n_2 infected and n_3 recovered neighbours changes its status from S to I and from I to R . In this case $f_{SI}(n_1, n_2, n_3) = \tau n_2$ with some parameter τ , which is called per-contact infection rate and $f_{IR}(n_1, n_2, n_3) = \gamma$ with some parameter γ , which is called recovery rate. That is, the infection rate depends linearly on the number of infected neighbours, while the recovery rate is independent of the statuses of the neighbours.

Example 3.4. A network of neurones is considered with purely excitatory connections. Within the network, neurones are considered to be either quiescent (Q) or active (A), that is using the above notation let $Q_1 = Q$ and $Q_2 = A$. There are

two transitions: a quiescent neurone with n_1 quiescent and n_2 active neighbours becomes active with rate $f_{QA}(n_1, n_2) = \omega \tanh(n_2)$, with some parameter ω called synaptic weight, while an active neurone with n_1 quiescent and n_2 active neighbours becomes quiescent with rate $f_{AQ}(n_1, n_2) = \alpha$ with some parameter α , which is called de-activation rate. That is, the activation rate depends in a non-linear way on the number of active neighbours, while the de-activation rate is independent from the statuses of the neighbours.

Example 3.5. Consider a voter-like model, where the nodes of the network represent the voters that can be of status A or B and the neighbours of a node can change its status as follows. A node of status A with n_1 and n_2 neighbours in status A and B , respectively, becomes B at rate $f_{AB}(n_1, n_2) = an_2$ with some parameter a . Similarly, a node of status B with n_1 and n_2 neighbours in status A and B , respectively, becomes A at rate $f_{BA}(n_1, n_2) = bn_1$ with some parameter b . Thus in this case both transitions depend on the statuses of the neighbours.

We note that an alternative rate function is widely used: $f_{AB}(n_1, n_2) = a \frac{n_2}{n_1+n_2}$. The corresponding $B \rightarrow A$ transition rate can be given by $f_{BA}(n_1, n_2) = b \frac{n_1}{n_1+n_2}$ expressing the fact that the transition rate depends on the ratio of the number of different neighbours.

Example 3.6. Consider rumour spread on a network. A node can be ignorant (X), spreader (Y) or stifler (Z). Using the above notation let $Q_1 = X$, $Q_2 = Y$ and $Q_3 = Z$. There are two transitions: spreading the rumour and forgetting it. In order to specify the dynamics the transition rates $f_{XY}(n_1, n_2, n_3)$ and $f_{YZ}(n_1, n_2, n_3)$ have to be specified, yielding the rate at which a node with n_1 ignorant, n_2 spreader and n_3 stifler neighbours changes its status from X to Y and from Y to Z . In this case $f_{XY}(n_1, n_2, n_3) = \tau n_2$ with some parameter τ , which is called per-contact spreading rate and $f_{YZ}(n_1, n_2, n_3) = \gamma + p(n_2 + n_3)$ with some parameters γ and p . That is, the spreading rate depends linearly on the number of spreader neighbours, while the forgetting rate is partly independent of the statuses of the neighbours but it increases linearly with the number of spreader and stifler neighbours.

Exercise 3.7. Based on the ideas explained in Example 3.1, write down the full system of master equations for SIS dynamics on a line graph with $N = 3$ nodes.

Exercise 3.8. Do the same for a line graph with $N = 4$ nodes.

Exercise 3.9. Do the same for a star graph with $N = 4$ nodes.

Exercise 3.10. Write down the system of master equations for the QAQ dynamics given in Example 3.4 on a line graph with $N = 3$ nodes.

Exercise 3.11. Write down the system of master equations for the voter-like model given in Example 3.5 on a line graph with $N = 3$ nodes.

Exercise 3.12. Write down the system of master equations for SIR epidemic given in Example 3.3 on a triangle.

Exercise 3.13. Write down the system of master equations for rumour spread given in Example 3.6 on a triangle.

4 Mean-field approximations for epidemic propagation

As already pointed out in the previous section, the high-dimensionality of exact mathematical models describing spreading processes on networks makes them neither tractable nor numerically solvable for networks of realistic size. We can avoid this fundamental difficulty by re-focusing our attention on expected population-scale quantities, such as

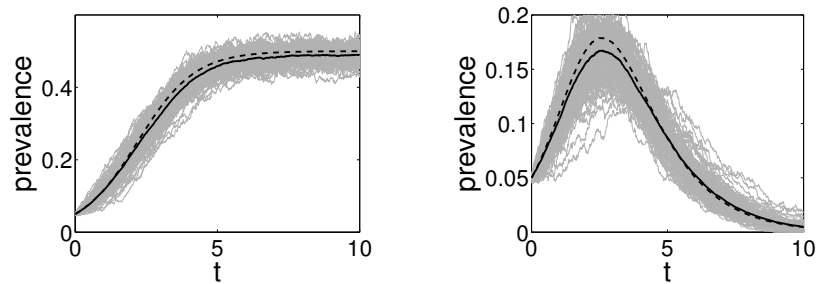


Figure 11: The time dependence of the expected number of infected nodes for an Erdős-Rényi random graph from 100 agent-based stochastic simulations (grey curves) and from the mean-field system closed at the level of pairs (black dashed curves) for (a) a SIS epidemic and (b) a SIR epidemic. The averages of the simulations are shown with black continuous curves. The parameter values are $N = 1000$, $\langle K \rangle = 30$, recovery rate $\gamma = 1$ and infection rate $\tau = 1/15$. We note that as the number of nodes increases the spread of simulation around the mean decreases.

the expected prevalence or the expected number of edges where one node is susceptible while the other is infectious. This opens up a range of possibilities to formulate so-called mean-field models, typically low-dimensional systems of ordinary differential equations (ODEs), that are widespread in the physics and mathematical biology literature and are used to approximate stochastic processes, with the potential to be exact in the large/fast system or thermodynamic limit, see for example [7, 9, 10, 14].

In Fig. 11 we show the typical outcome of a number of different realisations of SIS and SIR epidemics on an Erdős-Rényi network with an average degree $\langle K \rangle = 30$. The realisations of many stochastic simulations form the cloud plot, which highlights the stochasticity of the spreading processes. On the same figure, the average prevalence, based on all realisations, is also shown together with the prevalence obtained from the simplest mean-field approximation, system (9) for the SIS case and (10) for the SIR case.

The typical recipe for deriving a mean-field model is to first identify some quantities of practical interest and derive equations for how the average value of these quantities change in time. Often these equations rely on new variables. We iterate the process, deriving equations for the new variables. Occasionally this process terminates quickly, yielding a small, self-consistent system of equations. Sometimes we cannot derive the equations unless we make some approximating assumptions. Other times the iteration would continue forever; then we typically “close” the model by choosing some iteration and approximating the newly introduced variables in terms of already existing variables.

For models of infectious disease, the variables of interest are typically the average number of nodes in a given status. We denote these by $[S](t)$, $[I](t)$ and $[R](t)$ for susceptible, infected and recovered, respectively. The expected number of edges connecting a node of type A to a node of type B is denoted by $[AB](t)$ with $A, B \in \{S, I, R\}$. The expected number of triples, where a middle B node is connected to a node of type A and to a node of type C , is denoted by $[ABC](t)$. A formal definition of these quantities is given in the next subsection.

4.1 The variables of mean-field models: population level counts

To define the population level counts, we start from the random variable $X_i(t)$ that determines the type of node i at time t , i.e., for example $X_i(t) = I$ if node i is infected at time t . The above expected values can be defined formally as

$$[A](t) = \sum_{i=1}^N P(X_i(t) = A),$$

where $A \in \{S, I, R\}$. The expected number of (directed) edges in a given status can be defined similarly as follows,

$$[AB](t) = \sum_{i=1}^N \sum_{j=1}^N g_{ij} P(X_i(t) = A, X_j(t) = B)$$

with $A, B \in \{S, I, R\}$. It is important to note here that edges connecting two susceptible nodes contribute twice to $[SS]$, since the pairs (i, j) and (j, i) are both counted in the $[SS]$ class when i and j are susceptible nodes. As an example, consider the "toast graph", a square with a diagonal, with two infected and two susceptible nodes. The nodes are labeled clockwise from the top left with nodes one and two being infected and the diagonal connects node 2 and node 4. Let us for a moment assume that nodes are in the given statuses with probability one. Hence, $[I] = 2$ and $[S] = 2$. Counting the pairs we find that $(3, 2)$, $(4, 2)$ and $(4, 1)$ are of SI type and, hence $[SI] = 3$. The II pairs are $(1, 2)$ and $(2, 1)$, therefore $[II] = 2$. In a similar way we get $[IS] = 3$ and $[SS] = 2$, so the total number of pairs is 10. Let us finally turn to the counting of triples. The number of ABC triples is defined in general as

$$[ABC](t) = \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N g_{ij} g_{jk} P(X_i(t) = A, X_j(t) = B, X_k(t) = C),$$

with $A, B, C \in \{S, I, R\}$. Using this definition the number of SSI triples in the toast graph is $[SSI] = 3$, namely the SSI triples are $(3, 4, 1)$, $(3, 4, 2)$ and $(4, 3, 2)$. The number of other triples can be similarly obtained as $[ISS] = 3$, $[IIS] = 3$, $[SII] = 3$, $[ISI] = 2$ and $[SIS] = 2$, i.e., there are 16 triples altogether.

The expected values introduced above obey some conservation relations. The simplest one, for the SIS dynamics is based on the fact that $P(X_i(t) = S) + P(X_i(t) = I) = 1$ for any $i = 1, 2, \dots, N$, since any node is either susceptible or infected. This immediately implies that

$$[S](t) + [I](t) = N$$

for any time instant t . For the SIR epidemic the corresponding equation is $[S](t) + [I](t) + [R](t) = N$. The conservation relation for the pairs is based on the simple fact that for the SIS epidemic a pair can be in one of the following four statuses: SS , SI , IS or II . Hence $P(X_i(t) = S, X_j(t) = S) + P(X_i(t) = S, X_j(t) = I) + P(X_i(t) = I, X_j(t) = S) + P(X_i(t) = I, X_j(t) = I) = 1$, implying

$$[SS](t) + [SI](t) + [IS](t) + [II](t) = \sum_{i=1}^N \sum_{j=1}^N g_{ij} = Nn,$$

where $n = \langle K \rangle$ is the average degree defined as $\langle K \rangle = \frac{1}{N} \sum_{i=1}^N k_i$, where $k_i = \sum_{j=1}^N g_{ij}$ is the degree of node i . The definition of pairs immediately implies that $[SI] = [IS]$, hence the above relation can be formulated as

$$[SS](t) + 2[SI](t) + [II](t) = Nn.$$

Similar arguments lead to further pair conservation relations in the SIS case, namely

$$[SS](t) + [SI](t) = n_S(t)[S](t), \quad [SI](t) + [II](t) = n_I(t)[I](t),$$

where $n_S(t)$ and $n_I(t)$ denote the average degree of susceptible and infected nodes, respectively.

4.2 Exact differential equations for the singles and pairs

We use heuristic arguments here to derive the exact differential equations for the expected number of nodes and edges in given statuses. These can be derived from the system of master equations consisting of 2^N equations governing the full SIS dynamics [16]. We sacrifice having information about the precise states of each node to get a much smaller system of equations for the expected numbers of nodes in each state. The main parameters of the epidemic processes are the infection and recovery rates denoted by τ and γ respectively.

Theorem 4.1. For the SIS epidemic on an arbitrary network (undirected and not weighted) the expected values $[S]$ and $[I]$ satisfy the following system

$$[\dot{S}] = \gamma[I] - \tau[SI], \quad (2a)$$

$$[\dot{I}] = \tau[SI] - \gamma[I]. \quad (2b)$$

The differential equations can be obtained heuristically as follows. The rate of transmissions to S nodes is τ times the number of SI edges. The rate of recovery of infectious nodes back into a susceptible status is γ times the number of infectious nodes. Thus the rate of change of $[S]$ is $\gamma[I] - \tau[SI]$. We similarly find the $[I]$ equation.

For an SIR epidemic, the I nodes do not become susceptible again, instead those nodes move into the R class. The equations take a different form:

Theorem 4.2. For the SIR epidemic on an arbitrary network (undirected and not weighted) the expected values $[S]$, $[I]$ and $[R]$ satisfy the following system

$$[\dot{S}] = -\tau[SI], \quad (3a)$$

$$[\dot{I}] = \tau[SI] - \gamma[I], \quad (3b)$$

$$[\dot{R}] = \gamma[I]. \quad (3c)$$

We note that the variables in the SIS and SIR systems thus far are not independent because of conservation relations, taking the form $[S] + [I] = N$ in the SIS case and $[S] + [I] + [R] = N$ in the SIR case. Hence, for the exact systems one of the equations could be omitted in both cases. However, for closed systems the conservation laws do not hold automatically, their validity may depend on the choice of the closure. Therefore, differential equations for all singles and all pairs that are needed to get a self contained system will be considered. This will always be followed by investigating whether conservation laws hold as they enable us to reduce the system.

As illustrated in Eqs. (2), the dynamics of the expected number of susceptible ($[S]$) and infected nodes ($[I]$), i.e., singles, depends on the number of pairs ($[SI]$), hence the system depends on pairs, for which we need additional equations. Similarly, the number of pairs depends on the number of triples. For example, the number of $[SS]$ pairs decreases due to infection from outside the pair, i.e., it changes proportionally to the number of $[SSI]$ triples with rate τ . In an SI or IS pair the infected node can recover with rate γ hence the number of SS pairs increases proportionally to the number of SI and IS pairs. Since the numbers of SI and IS pairs are equal, this recovery process can be accounted for by a term such as $2\gamma[SI]$. Extending this simple heuristic reasoning to SI and II pairs and by accounting for all within- and outside-pair transitions leads to the following theorems.

Theorem 4.3. For the SIS epidemic on an arbitrary network (undirected and not weighted) the expected values of $[S]$, $[I]$, $[SI]$, $[II]$ and $[SS]$ satisfy the following system of differential equations

$$[\dot{S}] = \gamma[I] - \tau[SI], \quad (4a)$$

$$[\dot{I}] = \tau[SI] - \gamma[I], \quad (4b)$$

$$[\dot{SI}] = \gamma([II] - [SI]) + \tau([SSI] - [ISI] - [SI]), \quad (4c)$$

$$[\dot{SS}] = 2\gamma[SI] - 2\tau[SSI], \quad (4d)$$

$$[\dot{II}] = -2\gamma[II] + 2\tau([ISI] + [SI]). \quad (4e)$$

This result can also be derived directly from the master equations (see [17]). As before, for SIR epidemics the I nodes do not become susceptible again, hence the terms due to recoveries show up in different equations. We find:

Theorem 4.4. *For the SIR epidemic on an arbitrary network (undirected and not weighted) the expected values of $[S]$, $[I]$, $[SI]$ and $[SS]$ satisfy the following system of differential equations*

$$[\dot{S}] = -\tau[SI], \quad (5a)$$

$$[\dot{I}] = \tau[SI] - \gamma[I], \quad (5b)$$

$$[\dot{R}] = \gamma[I], \quad (5c)$$

$$[\dot{SI}] = -\gamma[SI] + \tau([SSI] - [ISI] - [SI]), \quad (5d)$$

$$[\dot{SS}] = -2\tau[SSI], \quad (5e)$$

We note that the differential equations for $[II]$, $[SR]$, $[IR]$, and $[RR]$ can be formulated similarly, but we leave them out because we can create a closed system of equations without them, and they are not generally of interest from an epidemiological viewpoint.

4.3 Closures at the pair and triple level and the resulting models

In this section the combinatorial ideas leading to the closures of the above four systems are presented.

Closures

We now assume the network is homogeneous, each node has the same degree n , i.e. a randomly chosen susceptible node has n neighbours. The number of infected nodes is $[I]$, hence the proportion of infected nodes in the population is $[I]/N$. Assuming that infected nodes are distributed randomly, a susceptible node has $n[I]/N$ infected neighbours. It is important to note that this assumption introduces the inexactness into the closed system since infected nodes are not distributed uniformly, they are more likely to be in contact with other infected nodes because of how infection propagates. Using this assumption, however, if a susceptible node has $n[I]/N$ infected neighbours, then the total number of SI edges is

$$[SI] \approx \frac{n}{N}[S][I]. \quad (6)$$

This relation is referred to as a closure since it leads to a closed system of equations. We escape the dependence on higher order moments in system (2). This closure is used also for the SIR epidemic in system (3). We note that sometimes $\frac{n}{N-1}$ is used instead of $\frac{n}{N}$, because choosing a susceptible node the remaining population contains only $N-1$ nodes.

To improve our accuracy we need a better accounting of the fact that infected nodes are not uniformly distributed. We take system (4) and close it by assuming an algebraic expression for $[SSI]$ and for $[ISI]$ in terms of the singles and pairs. To derive this we start again with a susceptible node and determine what proportion of the edges starting from this node lead to infected or susceptible nodes. The total number of edges starting from susceptible nodes is $n[S]$. The total number of SI edges is $[SI]$, hence a proportion $[SI]/n[S]$ of the edges starting from susceptible nodes lead to infected nodes. Similarly, the ratio of edges leading to susceptible nodes is $[SS]/n[S]$. Thus if we choose a susceptible node u and two neighbours v and w (arbitrarily calling v "first") the probability that v is susceptible and

w infected is $[SS][SI]/n^2[S]^2$. There are $n(n-1)$ ways to choose v and w . Thus the expected number of SSI triples is $n(n-1)[SS][SI]/n^2[S]^2 = (n-1)[SS][SI]/n[S]^2$. Assuming that these are uniformly distributed, we conclude that

$$[SSI] \approx \frac{n-1}{n} \frac{[SS][SI]}{[S]}. \quad (7)$$

Similar analysis leads to the closure

$$[ISI] \approx \frac{n-1}{n} \frac{[SI]^2}{[S]}. \quad (8)$$

As before this is only an approximation: for an SIS epidemic a previously infected, but recently recovered individual will tend to have more infected partners than other susceptible nodes. Thus the distribution of infected partners to susceptible nodes is not truly uniform. Further, for an SIR epidemic, if there are many short cycles, then one partner of a susceptible node u being infected is correlated with other partners of u being infected, so again the distribution is not uniform. If there are few short cycles, we will see that this closure is accurate for SIR epidemics.

Closed systems

We now write out the closed systems. We first apply closure (6) to system (2). Since the closure is only approximate the variables in the closed system are strictly speaking different from the original ones, so we use a different notation: $[S]_f$ and $[I]_f$ are the approximations given by the closed system. Applying the closure leads to the

SIS homogeneous mean-field model at single level

$$[\dot{S}]_f = \gamma[I]_f - \tau \frac{n}{N} [S]_f [I]_f, \quad (9a)$$

$$[\dot{I}]_f = \tau \frac{n}{N} [S]_f [I]_f - \gamma[I]_f. \quad (9b)$$

The subscript “ f ” refers to “first” since this can be considered as the first approximation when the number of pairs is expressed in terms of the number of singles. The system and variables resulting by applying the triple closure will be referred to as the second approximation. In the case of the SIR epidemic the simplest closed system takes the form below.

SIR homogeneous mean-field model at single level

$$[\dot{S}]_f = -\tau \frac{n}{N} [S]_f [I]_f, \quad (10a)$$

$$[\dot{I}]_f = \tau \frac{n}{N} [S]_f [I]_f - \gamma[I]_f, \quad (10b)$$

$$[\dot{R}]_f = \gamma[I]_f. \quad (10c)$$

It is worth noting that these closures lead to the usual SIS and SIR disease models. These equations are an approximation for epidemics on static networks. However, if we consider an alternate problem, in which individuals select a new set of random partners at each moment, our assumption that $[SI] \approx \frac{n}{N}[S][I]$ is correct. So these equations are correct if we consider a system in which individuals are quickly changing their partners. Thus the difference between these closed equations and the full equations is due to the duration of partnerships creating correlations between the status of neighbouring nodes.

We now apply the triple closures (7) and (8) to system (4). This is the second approximation, so we use the notation $[S]_s$, $[I]_s$, $[SI]_s$, $[SS]_s$ and $[II]_s$ yielding the

SIS homogeneous pairwise model

$$[\dot{S}]_s = \gamma[I]_s - \tau[SI]_s, \quad (11a)$$

$$[\dot{I}]_s = \tau[SI]_s - \gamma[I]_s, \quad (11b)$$

$$[\dot{SI}]_s = \gamma([II]_s - [SI]_s) + \tau \frac{n-1}{n} \frac{[SI]_s([SS]_s - [SI]_s)}{[S]_s} - \tau[SI]_s, \quad (11c)$$

$$[\dot{SS}]_s = 2\gamma[SI]_s - 2\tau \frac{n-1}{n} \frac{[SI]_s[SS]_s}{[S]_s}, \quad (11d)$$

$$[\dot{II}]_s = -2\gamma[II]_s + 2\tau \frac{n-1}{n} \frac{[SI]_s^2}{[S]_s} + 2\tau[SI]_s. \quad (11e)$$

We will show later that this system has conserved quantities, hence we do not need all of the differential equations to determine the number of susceptible and infected nodes. In the next section it will be shown that two differential equations are enough to determine all the unknown functions, because there is one conservation relation for singles and two for pairs.

For *SIR* epidemics the same closures are applied to system (5) leading to the

SIR homogeneous pairwise model

$$[\dot{S}]_s = -\tau[SI]_s, \quad (12a)$$

$$[\dot{I}]_s = \tau[SI]_s - \gamma[I]_s, \quad (12b)$$

$$[\dot{R}]_s = \gamma[I]_s, \quad (12c)$$

$$[\dot{SI}]_s = -\gamma[SI]_s + \tau \frac{n-1}{n} \frac{[SI]_s([SS]_s - [SI]_s)}{[S]_s} - \tau[SI]_s, \quad (12d)$$

$$[\dot{SS}]_s = -2\tau \frac{n-1}{n} \frac{[SI]_s[SS]_s}{[S]_s}, \quad (12e)$$

The system should be completed by further equations if one had reason to be interested in the values of pairs like $[II]$, $[SR]$, $[IR]$, or $[RR]$.

When the closed systems are solved, initial conditions are needed for all model variables. Typically, the initial number of susceptible, infected and recovered nodes are given, these can be used as initial conditions in the mean-field models at single level (9) and (10). In the case of pairwise models further initial conditions are needed for the initial number of pairs. Assuming that the different types of nodes are distributed randomly initially, the initial condition for AB pairs can be given as

$$[AB]_0 = \frac{n}{N} [A]_0 [B]_0,$$

where $[A]_0$ and $[B]_0$ denote the initial number of nodes of type A and B , respectively. Obviously, this is not the only choice for the initial condition of the pairs. For example, correlations in the initial position of infected nodes can be built in but these will dampen in time.

5 Analysis of the closed *SIS* systems

The ODE systems (9) and (11) are studied here by using the tools of dynamical system theory.

5.1 SIS homogeneous mean-field equations at single level

Consider first the SIS system closed at the level of pairs, i.e., system (9). Note that adding the two equations we get that $[S]_f(t) + [I]_f(t)$ is constant in time. If the initial condition satisfies $[S]_f(0) + [I]_f(0) = N$, then by using $[S]_f(t) = N - [I]_f(t)$ the system can be reduced to the single equation

$$[\dot{I}]_f = \tau \frac{n}{N} (N - [I]_f)[I]_f - \gamma [I]_f.$$

We can easily find the behaviour of the dynamical system given by this single equation. It has a disease-free steady state $[I]_f^{df} = 0$ and another fixed point at $[I]_f^e = N(1 - \frac{\gamma}{n\tau})$. This second point is only biologically meaningful if it is positive, i.e., $\gamma < n\tau$ in which case it gives an endemic equilibrium. Differentiating the right hand side of the differential equation with respect to $[I]_f$, i.e., linearising the right hand side, one obtains that for $\gamma > n\tau$ the disease-free state is stable, while for $\gamma < n\tau$ the endemic state is stable. Thus at $\gamma = n\tau$ a transcritical bifurcation occurs, the disease-free steady state loses its stability, and a new stable steady state appears. Moreover, these stabilities are global for $N > [I]_f > 0$. For $\gamma > n\tau$ the right hand side is negative, i.e., $[I]_f$ is decreasing in time and converges to zero starting from any meaningful initial condition. In the case $\gamma < n\tau$ the right hand side changes sign from positive to negative at the endemic steady state, which means that all solutions starting in the interval $(0, N]$ converge to the endemic steady state. Summarising, we have the following analytic results about the closed system (9).

Analytical results for SIS homogeneous mean-field equations at single level

- Conservation of singles: $[S]_f(t) + [I]_f(t) = N$.
- Reducibility to a single equation: $[\dot{I}]_f = \tau \frac{n}{N} (N - [I]_f)[I]_f - \gamma [I]_f$.
- Transcritical bifurcation at $\gamma = n\tau$: trivial steady state loses its stability and a stable endemic steady state appears.
- Global behaviour for $\gamma > n\tau$: all solutions converge to the disease-free steady state $[I]_f^{df} = 0$.
- Global behaviour for $\gamma < n\tau$: all solutions converge to the endemic steady state $[I]_f^e = N(1 - \frac{\gamma}{n\tau})$.

5.2 SIS homogeneous pairwise equations

Consider the system closed at the level of triples in the case of a SIS dynamics, i.e., system (11). Note that adding the first two equations we get that $[S]_s(t) + [I]_s(t)$ is constant in time. If the initial condition satisfies $[S]_s(0) + [I]_s(0) = N$, then

$$[S]_s(t) + [I]_s(t) = N.$$

Hence, the equation for $[\dot{S}]_s$ can be replaced with $[S]_s(t) = N - [I]_s(t)$, reducing the system to four differential equations. Alternatively one may choose to keep $[S]_s$ and dispose of the second equation. Moreover, the pair conservation holds in the closed system, meaning that the sum $2[SI]_s + [SS]_s + [II]_s$ is constant in time, where we keep in mind that $[SI]_s = [IS]_s$. The conservation relation can be easily verified by showing that the derivative of the sum is 0. This enables us to reduce the system further to three differential equations, namely one of the three variables $[SI]_s$, $[SS]_s$ or $[II]_s$ can be omitted. The total number of pairs can be expressed in terms of the average degree n as

$$2[SI]_s + [SS]_s + [II]_s = nN.$$

If this relation holds initially, it remains true for all time.

We have seen that in the exact system the pair conservation relation $[SS] + [SI] = n[S]$ also holds. We prove that this conservation relation is preserved if the simplest closures (7) and (8) are used.

Proposition 5.1. *In system (11) the relations*

$$[SS]_s + [SI]_s = n[S]_s, \quad [SI]_s + [II]_s = n[I]_s$$

hold.

Proof. In order to simplify our calculation, the following functions are introduced

$$A(t) = [SS]_s(t) + [SI]_s(t) - n[S]_s(t), \quad B(t) = [SI]_s(t) + [II]_s(t) - n[I]_s(t).$$

Differentiating these functions and using differential equations (11) leads to

$$\begin{aligned} \dot{A} &= \gamma B - \tau([SSI]_s + [ISI]_s - (n-1)[SI]_s), \\ \dot{B} &= -\gamma B + \tau([SSI]_s + [ISI]_s - (n-1)[SI]_s), \end{aligned}$$

where we used the short notations $[SSI]_s = \frac{n-1}{n} \frac{[SS]_s[SI]_s}{[S]_s}$ and $[ISI]_s = \frac{n-1}{n} \frac{[SI]_s^2}{[S]_s}$. It can be immediately seen that $[SSI]_s + [ISI]_s - (n-1)[SI]_s = -\frac{n-1}{n} \frac{[SI]_s}{[S]_s} A$. Introducing the notation $U = \frac{n-1}{n} \frac{[SI]_s}{[S]_s}$ the above system takes the form

$$\begin{aligned} \dot{A} &= \gamma B + \tau U A, \\ \dot{B} &= -\gamma B - \tau U A. \end{aligned}$$

Thus the pair (A, B) satisfies a system of homogeneous linear differential equations. The function U is continuous (assuming that $[S]_s \neq 0$), hence the solution of the system is unique. Constant zero functions are solutions of the homogeneous system, therefore if $A(0) = 0$ and $B(0) = 0$, then $A(t) = 0$ and $B(t) = 0$ for all t . This means that $[SS]_s(t) + [SI]_s(t) = n[S]_s(t)$ and $[SI]_s(t) + [II]_s(t) = n[I]_s(t)$ hold for all t if these are true at the initial time instant. \square

This proposition enables us a further reduction of the system to two differential equations, namely only one of the three variables $[SI]_s, [SS]_s, [II]_s$ is needed. From a technical point of view it is useful to use $[S]_s$ and $[SS]_s$ as independent variables, as this results in a system which is easier to analyse. Thus, let us consider the reduced system

$$\begin{aligned} \dot{[S]}_s &= \gamma[I]_s - \tau[SI]_s, \\ \dot{[SS]}_s &= 2\gamma[SI]_s - 2\tau \frac{n-1}{n} \frac{[SI]_s[SS]_s}{[S]_s}, \end{aligned}$$

with the algebraic relations below yielding the remaining variables as

$$[I]_s = N - [S]_s, \quad [SI]_s = n[S]_s - [SS]_s, \quad [II]_s = nN - 2[SI]_s - [SS]_s.$$

Substituting these into the differential equations yield

$$[\dot{S}]_s = \gamma N - (\gamma + n\tau)[S]_s + \tau[SS]_s, \tag{13}$$

$$[\dot{SS}]_s = 2(n[S]_s - [SS]_s) \left(\gamma - \tau(n-1) \frac{[SS]_s}{n[S]_s} \right). \tag{14}$$

This two-dimensional system can be studied by elementary phase plane analysis. First we prove the following proposition about the local behaviour at the steady states.

Proposition 5.2. *In system (11) a transcritical bifurcation occurs at $\gamma = \tau(n-1)$.*

- If $\tau(n-1) < \gamma$, then there is no endemic steady state and the disease-free steady state is asymptotically stable.

- If $\tau(n-1) > \gamma$, then the endemic steady state is asymptotically stable and the disease-free steady state is unstable.

Proof. First, we find the steady states. At the steady state, one of the factors in the right hand side of the second equation is equal to zero. If the first factor is zero, i.e., $n[S]_s = [SS]_s$, then we get the disease-free steady state, namely the first equation yields $[S]_s^{df} = N$, and then $[SS]_s^{df} = nN$. If the second factor is zero, that is $\gamma n[S]_s = \tau(n-1)[SS]_s$, then one obtains the endemic steady state, namely from the first equation $[S]_s^e = N \frac{\gamma(n-1)}{\tau n(n-1) - \gamma}$, and then $[SS]_s^e = \frac{\gamma n}{\tau(n-1)} [S]_s^e$. The number of infected nodes in the endemic steady state is $[I]_s^e = N - [S]_s^e = N n \frac{\tau(n-1) - \gamma}{\tau n(n-1) - \gamma}$. Hence for $\tau(n-1) < \gamma$ the endemic steady state does not exist for relevant values, since either $[I]_s^e$ or $[S]_s^e$ would be negative.

To investigate the stability we linearise the system at the steady states. The Jacobian matrix at the disease-free steady state is

$$J^{df} = \begin{pmatrix} -\gamma - \tau n & \tau \\ 2n(\gamma - (n-1)\tau) & -2(\gamma - (n-1)\tau) \end{pmatrix}.$$

The trace of the Jacobian is $\text{Tr}(J^{df}) = -3\gamma + (n-2)\tau$, its determinant is $\det(J^{df}) = 2\gamma(\gamma - (n-1)\tau)$. If $\tau(n-1) < \gamma$, then the trace is negative and the determinant is positive, hence the disease-free steady state is asymptotically stable. While for $\tau(n-1) > \gamma$ the determinant is negative, hence the disease-free steady state is unstable.

The Jacobian matrix at the endemic steady state is

$$J^e = \begin{pmatrix} -\gamma - \tau n & \tau \\ -\frac{2n\gamma}{(n-1)\tau}(\gamma - (n-1)\tau) & 2(\gamma - (n-1)\tau) \end{pmatrix}$$

The trace of the Jacobian is $\text{Tr}(J^e) = \gamma - (3n-2)\tau$, its determinant is $\det(J^e) = \frac{2}{n-1}(\gamma - n(n-1)\tau)(\gamma - (n-1)\tau)$. If $\tau(n-1) > \gamma$, then the trace is negative and the determinant is positive, hence the endemic steady state is asymptotically stable. For $\tau(n-1) < \gamma$ the endemic steady state does not exist. \square

Our analysis is now completed by showing that the stability results in the above proposition are global.

Proposition 5.3. *The global behaviour of system (11) changes at $\tau(n-1) = \gamma$.*

- If $\tau(n-1) < \gamma$, then all solutions converge to the disease-free steady state.
- If $\tau(n-1) > \gamma$, then all solutions converge to the endemic steady state.

Proof. The statement can be proved by investigating the direction field of the two-dimensional system (13)–(14). Observe first that the triangle with vertices $(0, 0)$, $(N, 0)$ and (N, nN) in the phase plane $([S]_s, [SS]_s)$ is positively invariant, i.e., trajectories cannot leave it. This can be seen by checking that on the boundary trajectories point inwards. For example, at the lower boundary where $[SS]_s = 0$ we have $[\dot{S}]_s = 2n\gamma[S]_s > 0$, i.e., at this boundary trajectories move upwards. The two other boundary segments can be checked similarly. Thus the global behaviour of the system has to be studied in this triangle, corresponding to biologically relevant values.

The direction field of the system is determined by the null clines $[\dot{S}]_s = 0$ and $[\dot{SS}]_s = 0$, shown with dashed lines in Figure 12. The first one is a straight line that passes through the top corner (N, nN) of the triangle and divides the triangle into two parts. In the left part trajectories move to the right and in the right part they move to the left. The null cline $[\dot{SS}]_s = 0$ consists of two straight lines passing through the origin. One of them coincides with the left boundary line of the triangle, namely the line given by $[SS]_s = n[S]_s$. The other line is given by $(n-1)\tau[SS]_s = \gamma n[S]_s$, which lies outside the triangle if $\tau(n-1) < \gamma$, i.e., when there is no endemic steady state.

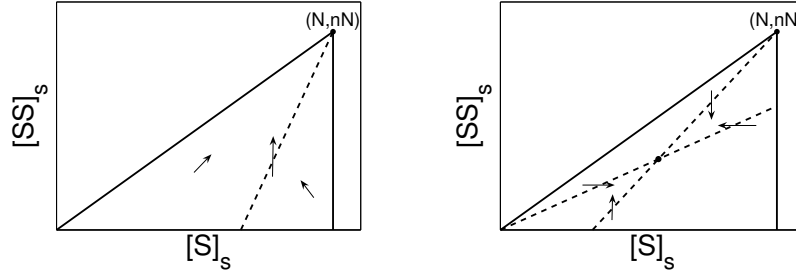


Figure 12: Direction field in the $([S], [SS])$ plane of the pairwise equation of an *SIS* epidemic for $\tau(n-1) < \gamma$ (left panel), and $\tau(n-1) > \gamma$ (right panel).

In Figure 12, the direction fields are shown in the two different cases. In the left panel $\tau(n-1) < \gamma$, hence $[\dot{SS}]_s > 0$ in the whole triangle, hence all trajectories tend to the disease-free steady state (N, nN) , which is the top vertex of the triangle. In the right panel, the case $\tau(n-1) > \gamma$ is shown. Then one of the straight lines given by $[\dot{SS}]_s = 0$ divides the triangle into two parts. In the top part $[\dot{SS}]_s < 0$, therefore trajectories move down there. In the bottom region $[\dot{SS}]_s > 0$, hence trajectories move up there. It is clear from the figure that there is no periodic orbit and there is a single stable steady state. Thus, all trajectories tend to the endemic steady state, which is inside the triangle. This elementary phase plane analysis completes the proof. \square

To conclude, the analytical results derived from the closed system (11) are summarised below.

Analytical results for *SIS* homogeneous pairwise equations

- Conservation of singles and pairs:

$$[S]_s + [I]_s = N, \quad 2[SI]_s + [SS]_s + [II]_s = nN,$$

$$[SS]_s + [SI]_s = n[S]_s, \quad [SI]_s + [II]_s = n[I]_s.$$

- Reducibility to the two-dimensional system (13)–(14).
- Transcritical bifurcation at $\gamma = \tau(n-1)$: the disease-free steady state loses its stability and a stable endemic steady state appears.
- Global behaviour for $\gamma > \tau(n-1)$: all solutions converge to the disease-free steady state

$$[I]_s^{df} = 0, \quad [S]_s^{df} = N, \quad [SI]_s^{df} = 0, \quad [SS]_s^{df} = nN, \quad [II]_s^{df} = 0.$$

- Global behaviour for $\gamma < \tau(n-1)$: all solutions converge to the endemic steady state

$$[I]_s^e = Nn \frac{\gamma - \tau(n-1)}{\gamma - \tau n(n-1)}, \quad [S]_s^e = N \frac{\gamma(n-1)}{\tau n(n-1) - \gamma}, \quad [SI]_s^e = \frac{\gamma}{\tau} [I]_s^e, \quad (15)$$

$$[SS]_s^e = \frac{\gamma n}{\tau(n-1)} [S]_s^e, \quad [II]_s^e = [I]_s^e \left(1 + \frac{\gamma(n-1)}{\tau n} \frac{[I]_s^e}{[S]_s^e} \right).$$

5.3 Comparison of the mean-field and pairwise approximations

The correlation $C_{AB} = \frac{N}{n} \frac{[AB]}{[A][B]}$ plays an important role in understanding the dynamic of the propagation process. The pairwise system (11) enables us to compute correlations of different types of nodes. The correlations given by this approximating system will be denoted by C_{AB}^s , where the superscript refers to the fact that these correlations are computed from the solutions of the second approximation.

Proposition 5.4. *The solution of system (11), subject to initial conditions satisfying the random mixing conditions*

$$[SI]_s(0) = \frac{n}{N}[S]_s(0)[I]_s(0), \quad [SS]_s(0) = \frac{n}{N}[S]_s^2(0), \quad [II]_s(0) = \frac{n}{N}[I]_s^2(0),$$

satisfies the following relations for any positive time

$$\frac{n}{N}[S]_s[I]_s - [SI]_s = [SS]_s - \frac{n}{N}[S]_s^2 = [II]_s - \frac{n}{N}[I]_s^2 > 0.$$

Proof. To make calculations simpler we introduce the time dependent functions

$$A = \frac{n}{N}[S]_s[I]_s - [SI]_s, \quad B = [SS]_s - \frac{n}{N}[S]_s^2, \quad C = [II]_s - \frac{n}{N}[I]_s^2.$$

Differentiating these functions and using differential equations (11) leads to

$$\begin{aligned} \dot{A} &= \gamma(A + C) + \tau([ISI]_s - [SSI]_s + [SI]_s + \frac{n}{N}[S]_s - \frac{n}{N}[I]_s), \\ \dot{B} &= -2\gamma A + \tau(2\frac{n}{N}[S]_s[SI]_s - 2[SSI]_s), \\ \dot{C} &= -2\gamma C + \tau(2[ISI]_s + 2[SI]_s - 2\frac{n}{N}[I]_s[SI]_s), \end{aligned}$$

where we used the short notations $[SSI]_s = \frac{n-1}{n} \frac{[SS]_s[SI]_s}{[S]_s}$ and $[ISI]_s = \frac{n-1}{n} \frac{[SI]_s^2}{[S]_s}$. Using Proposition 5.1, the definition of the closures yield $[SSI]_s + [ISI]_s = (n-1)[SI]_s$. Therefore the coefficients of τ in all of the three equations above can be simplified to $U = 2\frac{n}{N}[S]_s[SI]_s - 2[SSI]_s$. Thus, the above system takes the form

$$\begin{aligned} \dot{A} &= \gamma(A + C) + \tau U, \\ \dot{B} &= -2\gamma A + \tau U, \\ \dot{C} &= -2\gamma C + \tau U. \end{aligned}$$

Thus the triple (A, B, C) satisfies a system of linear differential equations, the solution of which is unique subject to a given initial condition. It is obvious that if A satisfies the differential equation $\dot{A} = -2\gamma A + \tau U$, then the triple given by $B = C = A$ satisfies the system. The assumptions on the initial conditions imply $A(0) = 0$, $B(0) = 0$ and $C(0) = 0$, that is $A(0) = B(0) = C(0)$ hold. Therefore, by uniqueness argument, the solution satisfies $A(t) = B(t) = C(t)$ for all t . Thus it remains to prove that $A(t) > 0$ for all positive t . Observe that using the closure, U can be written as

$$U = 2B \frac{[SI]_s}{[S]_s} + 2 \frac{[SI]_s[SS]_s}{n[S]_s}.$$

Using that $A = B$, the differential equation $\dot{A} = -2\gamma A + \tau U$ for A takes the form

$$\dot{A}(t) = p(t)A(t) + q(t),$$

with $p(t) = 2\tau \frac{[SI]_s(t)}{[S]_s(t)} - 2\gamma$ and $q(t) = 2\tau \frac{[SI]_s(t)[SS]_s(t)}{n[S]_s(t)}$. This linear differential equation can be solved as

$$A(t) = \int_0^t e^{P(s)-P(t)} q(s) ds,$$

where P is the integral of p , i.e., $P'(t) = p(t)$ and the initial condition $A(0) = 0$ was used. Since q is positive the function A is also positive, which completes the proof. \square

Thus for any positive finite time

$$\frac{n}{N}[S]_s[I]_s > [SI]_s, \quad [SS]_s > \frac{n}{N}[S]_s^2, \quad [II]_s > \frac{n}{N}[I]_s^2 \quad (16)$$

implying the following inequalities for the correlations

$$C_{SI}^s < 1, \quad C_{SS}^s > 1, \quad C_{II}^s > 1.$$

This means that susceptible nodes are more likely to be connected to other susceptible nodes and infected nodes are more likely to be connected to other infected nodes. This is due to the process itself: if newly infected nodes were selected randomly from all nodes we would see no correlation of neighbours, but because they are selected from those nodes with infected neighbours, a correlation develops.

The correlation inequalities in (16) enable us to prove that the prevalence obtained from the second approximation is always smaller than that given by the first approximation, that is $[I]_s < [I]_f$.

Proposition 5.5. *The prevalence $[I]_s$ obtained from system (11) and $[I]_f$ given by (9) satisfy the inequality $[I]_s \leq [I]_f$.*

Proof. Subtracting (11a) from (9a) yields

$$[\dot{I}]_f - [\dot{I}]_s = -\gamma([I]_f - [I]_s) + \tau \frac{n}{N} ([S]_f [I]_f - [S]_s [I]_s) + \tau \left(\frac{n}{N} [S]_s [I]_s - [SI]_s \right).$$

Applying the identities $[S]_f + [I]_f = N$ and $[S]_s + [I]_s = N$ in the middle term of the right hand side, this equation can be written as

$$[\dot{I}]_f - [\dot{I}]_s = d([I]_f - [I]_s) + A,$$

where $d = -\gamma + n\tau - \frac{n}{N}([I]_f + [I]_s)$ and $A = \frac{n}{N}[S]_s[I]_s - [SI]_s$. Integrating this linear differential equation and using the initial condition $[I]_f(0) = [I]_s(0)$ yields

$$[I]_f(t) - [I]_s(t) = \int_0^t e^{D(s)-D(t)} A(s) ds,$$

where D is the integral of d , i.e. $D'(t) = d(t)$. According to Proposition 5.4 $A(s) > 0$, hence the right hand side is positive, which completes the proof. \square

- [1] Bollobás, B., *Random graphs*, Cambridge University Press, Cambridge, 2001.
- [2] Caldarelli, G., *Scale-free networks: complex webs in nature and technology*, Oxford University Press, 2007.
- [3] Cohen, R., Havlin, S., *Complex networks: structure, robustness and function*, Cambridge University Press, Cambridge, 2010.
- [4] Danon, L. Ford, A.P., House, T., Jewell, C.P., Keeling, M.J., Roberts, G.O., Ross, J.V., Vernon, M.C., Networks and the Epidemiology of Infectious Disease, *Interdisciplinary Perspectives on Infectious Diseases* **2011** (2011) 1–28.
- [5] Diestel, R., *Graph theory*, Springer-Verlag, Heidelberg, New York, 2005.
- [6] Estrada, E., *The structure of complex networks: theory and applications*, Oxford University Press, 2011.
- [7] House, T., Keeling, M.J., Insights from unifying modern approximations to infections on networks, *Journal of The Royal Society Interface* **8** (2011) 67–73.
- [8] Jones, P. W., Smith, P., *Stochastic Processes: An Introduction*, CRC Press, 2012.
- [9] Keeling, M.J., The effects of local spatial structure on epidemiological invasions, *Proc. R. Soc. Lond. B* **266** (1999), 859-867.
- [10] Keeling, M.J., Eames, K.T.D., Networks and epidemic models, *J. Roy. Soc. Interface* **2** (2005), 295-307.
- [11] Kemeny, J.G., Snell, J.L., *Finite Markov chains*, Springer, New York, 1976.
- [12] Lovász, L., *Large networks and graph limits*, American Mathematical Soc., 2012.
- [13] Newman, M.E.J., *Networks: an introduction*, Oxford University Press, 2009.
- [14] Pastor-Satorras, R., Vespignani, A., Epidemic dynamics and endemic states in complex networks, *Phys Rev E* **63** (2001), 066117.
- [15] Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A., Epidemic processes in complex networks, *Rev. Mod. Phys.* **87** (2015), 925.
- [16] Simon, P.L., Taylor, M., Kiss, I.Z., Exact epidemic models on graphs using graph automorphism driven lumping, *J. Math. Biol.* **62** (2010), 479–508.
- [17] Taylor, M., Simon, P.L., Green, D. M. , House, T., Kiss, I.Z., From Markovian to pairwise epidemic models and the performance of moment closure approximations, *J. Math. Biol.* **64** (2012), 1021–1042.