wise timetable

Generating a Timetable

Preparation, parameters, polishing
and the improvement loop

. . .

WISE TIMETABLE · NOVEMBER 2026

Contents

Introduction
Who this book is for
The ten-minute version

Part 1 – How generation thinks
1.1 What generating does
1.2 Hard rules and wishes
1.3 What is a hole
1.4 Why time matters
1.5 Where the settings live

Part 2 – Preparing the data
2.1 The two-minute checklist
2.2 The data mistakes we see most often
2.3 Five decisions to make on paper

Part 3 – Setting the stage
3.1 Decision: what does a school day look like?
3.2 Decision: how much is too much?
3.3 Decision: how much effort for the first draft?
3.4 Decision: how should lessons be arranged?
3.5 Warnings

Part 4 – The first generation
4.1 Press Generate
4.2 If some lessons are unplaced
4.3 Look before you polish

Part 5 – Polishing with the gap solver
5.1 What it is
5.2 The left column – the rhythm of your school
5.3 How long and how hard
5.4 The middle and right columns – the weights

Part 6 – The improvement loop
6.1 The loop in five steps
6.2 How long should each run be?
6.3 What each symptom tells you to change
6.4 Worked example – one night on a real school
6.5 When to stop
6.6 Keep a diary

Part 7 – Changing a timetable that is already published

Part 8 – Reading the numbers

Part 9 – When something goes wrong
9.1 Lessons are not placed
9.2 Too many holes after polishing
9.3 The solver seems to be stuck
9.4 Teachers complain

9.5 Pupils or parents complain
9.6 Strange or unexpected results
9.7 If nothing helps

Part 10 – Recipes
10.1 A small school or one department (up to about 200 lessons)
10.2 A secondary school with a morning rhythm and a short Friday
10.3 A university with many rooms and buildings
10.4 An evening or part-time programme
10.5 A school with two shifts

Appendix A – Keyboard shortcuts

Appendix B – Glossary

Appendix C – The program's own help texts
Run gap optimisation / Duration (min)
Afternoon starts at hour
Afternoon days per class and week
Min teacher hours per day
Max teacher hours per day (soft)
CPU threads (0 = all but one)
Value of a teacher's day of presence (hole-hours)
Minimum lessons per day for pupils (0 = off)
Penalty for a short day
Penalty for a linked pair placed apart
Teacher's home room – weight
Electives at the edge of the day
Kick depth (0 = off)
Kick after (seconds without improvement)
Kick frequency (1 in N tries)

Appendix D – Switches in wtt.scheme

Introduction

Who this book is for

You are a timetabler. Perhaps you are the person at a faculty who has built the timetable by hand for twenty years, perhaps you have just inherited the job in a secondary school, perhaps you run a training centre with a dozen rooms. You do not need to know anything about algorithms. You need to know what the program can do for you, which knobs to turn, and what to do when the result is not what you hoped for.

This book follows the order in which a real timetable comes into existence:

1. **How generation thinks** – five minutes of ideas that make everything else easier.
2. **Preparing the data** – the part that decides most of the quality.
3. **Setting the stage** – the parameters, arranged by the decision they express, not alphabetically.
4. **The first generation** – what to expect and how to read it.
5. **Polishing with the gap solver** – the patient part of the program.
6. **The improvement loop** – how to work by trial and error *efficiently*: how long to run, what to change, when to stop.
7. **Changing a timetable that is already published.**
8. **Reading the numbers.**
9. **When something goes wrong** – an extensive troubleshooting guide.
10. **Recipes** for typical schools.

You do not have to read it in order. If your timetable exists and you only want it better, jump to Part 6. If something is broken, go to Part 9.

The ten-minute version

If you have ten minutes and a school that is ready:

1. Open **Settings → Miscellaneous** (F2), tab **Generate**, tick **Run gap optimisation** and set **Duration (min)** to 10.
2. Press **Gap optimisation settings ...** and fill in the left column: *Afternoon starts at hour*, *Afternoon days per class and week*, *Min teacher hours per day*, *Max teacher hours per day (soft)*. Confirm both windows.
3. Press the **Generate** button (the alarm-clock icon in the toolbar), answer **Yes**, press **Start**. The generator places the week, then the gap solver polishes it for ten minutes.
4. Press **Ctrl + B** and read the result.

Everything else in this book is about doing that with more skill, and about knowing what to do when it is not enough.

Part 1 – How generation thinks

1.1 What generating does

You enter the facts of your school once: courses, who teaches them, which groups attend them, which rooms exist. Wise Timetable then **places every lesson on a day and an hour**, so that nobody is in two places at once, every room is large and well equipped enough, and your rules hold.

It does this in three stages, and you may stop after any of them.

Stage	What it does	Takes	Where
1. Generator	Places every lesson. Fast; obeys the hard rules of the Generate tab.	seconds to minutes	the **Generate** button in the toolbar
2. Gap solver (polishing)	Moves lessons to remove *holes* and to fulfil your wishes.	minutes to a whole night	**Gap optimisation settings …**
3. Publish	Makes the result visible to students and teachers.	seconds	Publish

Think of packing a suitcase. The generator gets everything in. The gap solver then folds it again, patiently, until it is tidy.

The golden rule. The gap solver never breaks a rule that the generator kept. It only moves lessons to improve things, and it always remembers the best week it has found. If you press **Stop**, you lose nothing.

1.2 Hard rules and wishes

Everything you can set belongs to one of two kinds, and knowing which is half of using the program well.

- A **hard rule** is never broken. A teacher cannot be in two rooms; a lesson cannot be placed after the last hour of the day; a class cannot have more afternoon days than you allow; a lesson must have a room. If no placement satisfies the hard rules, the lesson stays *unplaced* and is listed. It is never smuggled in.
- A **wish** (soft rule) may be broken, but at a price. “Teachers should not come to school for a single lesson” is a wish: each missing hour costs the same as ten hours of waiting. The solver weighs the prices against each other and chooses the cheapest timetable.

The prices are measured in one common unit, the **hole-hour**: one hour that a teacher waits between two lessons. A hole of one hour for the *pupils* is counted a thousand times heavier. The solver will sacrifice a great deal of teacher comfort before it leaves pupils waiting.

1.3 What is a hole

A **hole** (gap) is an empty hour between two lessons of the same class or the same teacher on the same day. A class that has lessons at 8:00 and 10:00 and nothing at 9:00 has one hole.

Free hours at the beginning or the end of the day are not holes; they are simply the start or the end of the day.

1.4 Why time matters

Timetabling is a search among an astronomical number of possibilities. Nobody – no human, no program – can promise the perfect timetable. What a good program can do is find a *very good* one quickly and then keep finding better ones for as long as you let it. This is why the book spends so much space on **how long to run and what to change**: that is the skill.

1.5 Where the settings live

Where	What is there
Settings → Miscellaneous (F2) → Generate	the Generator: working hours, limits, attempts, behaviour tick-boxes, warnings – and the switch **Run gap optimisation** with its **Duration (min)** (Parts 3 and 5)
Generate tab → "Gap optimisation settings …"	the gap solver: three columns – *Generator settings* (the rules of your school), *Gap solver settings* (the weights and the search), *Advanced gap solver settings* (Part 5)
? next to a field	a long explanation, with the valid range and the measured default
Hovering over a field	a one-line tooltip
Ctrl + B	the week statistics report (Part 8)

Part 2 – Preparing the data

The program can only be as good as what you give it. Most disappointing timetables are not a failure of the solver; they are a data problem that the solver faithfully optimised.

2.1 The two-minute checklist

1. **Every lesson that must be placed exists:** course → turn → group → teacher.
2. **Teachers and groups** have the right *hours per day* (see 3.2).
3. **Rooms** have the right number of seats and equipment. A lesson needs a room that has every property the course asks for.
4. If a teacher should always teach in "their" room, it is set as that teacher's **default room**.
5. **Elective subjects** are marked as electives, so that they can sit at the edge of the day.
6. **Linked turns:** lessons that must be taught at the same time are linked.
7. **Blocked times** – meetings, days off, unavailable rooms – are entered as reservations, because no generator can guess them.
8. **Save a copy** of the file. Generating changes a lot.

2.2 The data mistakes we see most often

Mistake	What the generator does with it
A course has no teacher assigned	The lesson stays unplaced, or is placed without anyone to teach it.
A room lacks a property the course requires (for example "computer lab")	That lesson cannot find a room and is counted as not placed.
Two lessons of the same group are accidentally not linked although they run together	They are placed at different times; the pupils then have two lessons that were meant to be simultaneous.
Teachers' unavailable days are not entered	The generator happily gives the teacher lessons on a day he is not at school.
The group size is larger than every room	No room fits. Split the group or add a bigger room.
Electives are not marked as electives	They land in the middle of the day and leave the other pupils waiting.
The hours per week of a course do not match the program	The timetable is "perfect" for the wrong curriculum.

2.3 Five decisions to make on paper

Before opening a single setting, answer five questions. Each one maps to a setting later.

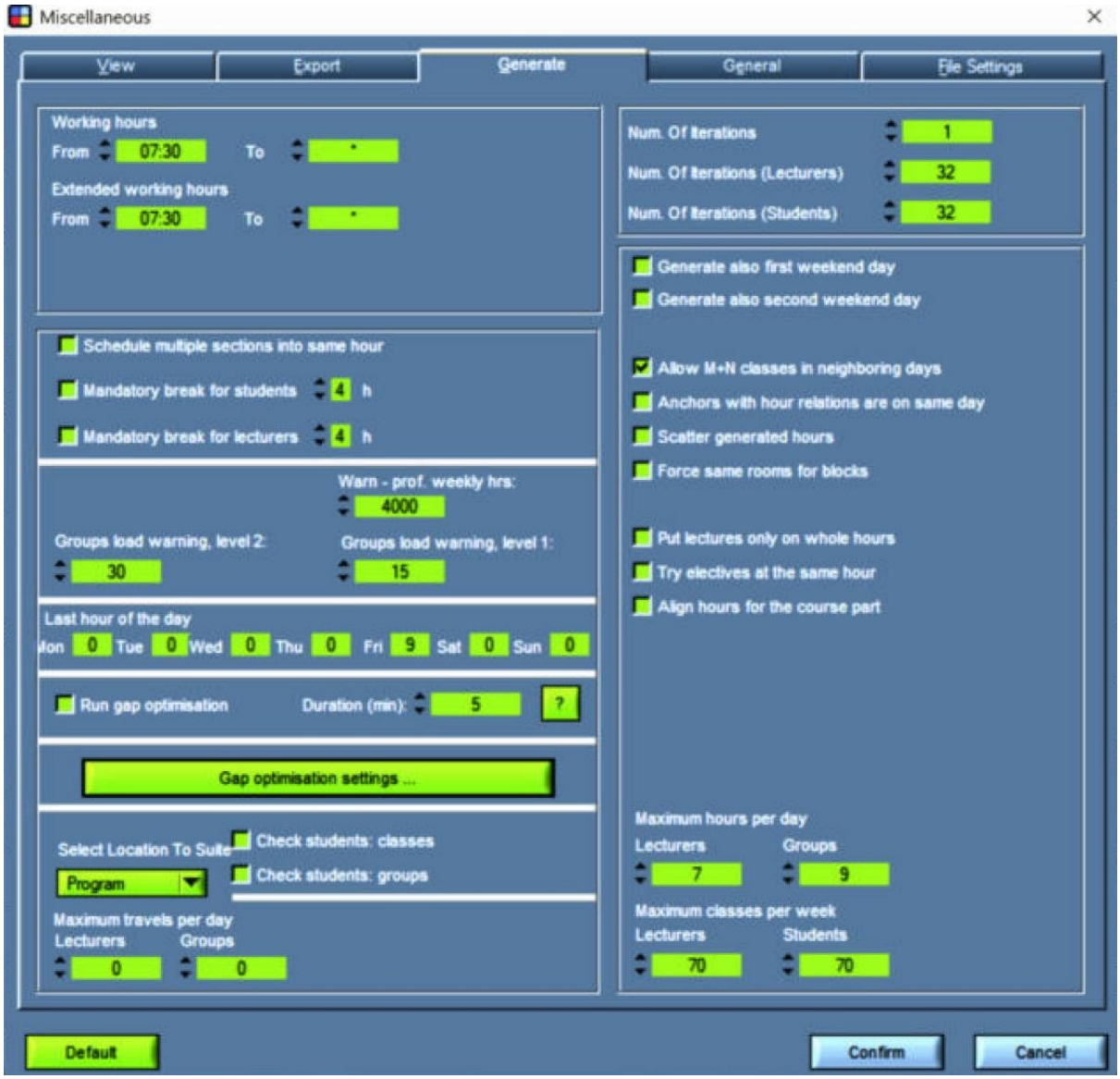
Question	Setting
When does the school day start and end?	Working hours
Is there a day that ends earlier (a short Friday)?	Last hour of the day
From which hour is it "afternoon", and how many afternoons may a class have?	Afternoon starts at hour / Afternoon days per class
How short may a teacher's day be?	Min teacher hours per day

Question	Setting
Do teachers teach in their own rooms?	Teacher’s home room – weight

Part 3 – Setting the stage

The settings of the Generator are on **Settings → Miscellaneous (F2) → Generate**. They are **saved with the timetable**, so everybody who opens the file gets the same behaviour. Only the three warnings are personal to your Windows user.

We go through them as decisions, not alphabetically.



3.1 Decision: what does a school day look like?

The **Working hours** are your normal day, for example 07:30–15:10. The generator tries to put every lesson there first. The **Extended working hours** are an *extension* used only for lessons that did not fit, for example hours up to 19:35 for the afternoon. Be stingy with the extension: every hour you add invites afternoon lessons, and afternoon lessons are what most schools do not want.

For a school whose days differ, there is **Last hour of the day** – one field per weekday. The number is a **school hour counted from the start of the working hours**, not a clock time: 1 is the first hour of the day, 9 the ninth. It names the last hour that may still hold a lesson;

nothing on that day ends after it. 0 means no limit beyond the extended working hours. A school whose Friday ends after the 9th hour (14:25–15:10 when the day starts at 07:30) sets Friday to 9 and the other days to 14. It is a **hard rule** of the gap solver: whatever the generator placed beyond it is moved back inside, and no later polishing ever breaks it. The generator itself does not read these fields yet, so a school that relies on them keeps *Run gap optimisation* ticked.

Schools with a strict hourly rhythm switch on *Put lectures only on whole hours*; universities that start at 08:15 or 08:30 leave it off. The **school edition** of the program shows a few more controls on this tab: the **Second shift – From / To** window for schools that run two shifts (sections of the second shift are ticked on the Sections screen), *Allow one hour before the second shift*, and four preferences about the shape of a pupil's day. They are listed in the table with the mark *(school edition)*; if you do not see them, your edition does not need them.

Setting	In practice
Working hours – From	Set this to the first lesson of your school day (07:30 in a secondary school, 08:00 at many faculties). Nothing generated starts earlier. *What the program says: Time from which automatic generation starts placing lessons each day. It first fits every lesson inside the working hours and uses the extended hours only for lessons that do not fit. Saved with the timetable, so everyone who opens it gets the same.*
Working hours – To	The end of the normal school day, for example 15:10. A narrow window gives compact timetables; widen it only if lessons stay unplaced. *What the program says: Time by which a lesson must end in generation's first pass. The pick is an end time: 'To 16:00' means the last lesson finishes by 16:00. Lessons that do not fit may run on until the extended working hours end. Saved with the timetable.*
Extended working hours – From	Usually the same as Primary From. Make it earlier only if you want the gap optimiser to be allowed to pull lessons earlier than the main search would. *What the program says: Earliest time a lesson may start once the working hours are not enough. The main search still starts at the working-hours From, so an earlier time here is used mostly when generation moves lessons to close gaps. It is also the first hour column of the CSV exports for instructors, sections and classrooms. Saved with the timetable.*
Extended working hours – To	The hard end of the generated day. Every hour you add here invites afternoon lessons – add as few as you can bear. In a morning school the right value is often only one or two hours beyond Primary To. *What the program says: Latest time any generated lesson may end - the hard end of the generated day. Lessons that do not fit inside the working hours may run on until this time. Also the last hour column of the CSV exports for instructors, sections and classrooms. Saved with the timetable.*
Last hour of the day (Mon … Sun)	One field per weekday, a school hour counted from the start of the working hours (1 = first hour), not a clock time: the last

Setting	In practice
	hour that may still hold a lesson, 0 = no limit. A school that finishes on Friday after the 9th hour sets Friday to 9 and the other days to 14. A hard rule of the gap solver; the generator does not read it yet. *What the program says: Last hour of the day*
Second shift – From (school edition)	Only for schools that run two shifts. Sections ticked "Second shift" on the Sections screen are placed from this time. *What the program says: Start of the afternoon shift. Lessons of sections ticked 'Second shift' on the Sections screen are generated from this time on (one hour earlier if 'Allow forward hour in second shift' is ticked). Other sections use the working hours. Saved with the timetable.*
Second shift – To (school edition)	The end of the second shift. Also the last column of CSV exports. *What the program says: Time by which lessons of 'Second shift' sections must end in generation. The pick is an end time. It is also the last hour column of the CSV exports for instructors, sections and classrooms. Saved with the timetable.*
Allow one hour before the second shift (school edition)	Lets a second-shift lesson start one hour before the shift officially begins. Use it when the shift start is a convention, not a hard limit. *What the program says: Lets generation start lessons of 'Second shift' sections one hour before the second shift begins. Off: they start no earlier than the shift start. Saved with the timetable.*
Put lectures only on whole hours	Switch on for schools with a strict hourly rhythm (08:00, 09:00 …). Leave off for universities that start at 08:15 or 08:30. *What the program says: Generation starts lessons only on whole hours counted from the start of the working hours - with 08:00 that is 08:00, 09:00, 10:00 - never on the slots in between. Placing lessons by hand is not affected. Saved with the timetable.*

3.2 Decision: how much is too much?

Limits are hard rules: the generator never exceeds them.

0 does not mean "unlimited". For hour limits, 0 prevents the generator from placing lessons at all. Write a large number for "no limit".

The daily limits are defaults: a value typed on a teacher's own screen, or in a group's own screen, always wins. Be careful with the group limit – pressing Confirm after changing it writes the value into every group, replacing individual values. The weekly limits are ceilings that most schools never touch; 70 means practically none.

Mandatory break is for pupils and teachers separately. Switch it on, then say after how many lessons in a row the next hour must stay free. Institutions with several buildings limit **journeys between locations** and decide whose location – the instructor's, the program's or the section's – decides which rooms may be used.

Setting	In practice
Maximum hours per day – Lecturers	The default daily maximum for teachers. A value typed on a teacher's own screen wins. Typical: 8 for schools, 10 for faculties. *What the program says: Most hours of lessons generation may give an instructor on one day (default 10). An instructor's own 'Custom hours/day' on the Instructors screen overrides it. 0 here does not mean unlimited. Saved with the timetable.*
Maximum hours per day – Groups	The default daily maximum for classes and groups. Be careful: pressing Confirm after changing it writes the value into every group. *What the program says: Most hours of lessons generation may give a section on one day (default 10). A section's own 'Custom hours' on the Sections screen overrides it. Careful: if you change this number and press Confirm, it is written into every section as its own value, replacing any custom values. Saved with the timetable.*
Maximum classes per week – Lecturers	A weekly ceiling. 70 means practically no limit; set it to a real contract number (e.g. 22 teaching hours) if the generator must respect it. *What the program says: Most hours of lessons generation may give an instructor in one week (default 70). It counts hours, not lessons: a 3-hour lesson counts as 3. 0 does not mean unlimited - it stops generation placing their lessons. Saved with the timetable.*
Maximum classes per week – Students	A weekly ceiling for every group. Leave at 70 unless you have a legal weekly limit for pupils. *What the program says: Most hours of lessons generation may give any one section in one week (default 70). It counts hours, not lessons, and applies to each section of a lesson. 0 does not mean unlimited - it stops generation placing section lessons. Saved with the timetable.*
Mandatory break for students	Switch on if students must not have too many lessons in a row without a pause (for example a lunch break). *What the program says: Turns on the break rule for students: once a section has had as many hours in a row as set to the right, generation keeps the next hour free. Sections ticked 'Ignore mandatory break' on the Sections screen are exempt. The number is not the break's length. Saved with the timetable.*
Mandatory break for students – hours (h)	How many lessons in a row are allowed before the next hour must stay free. 4 is typical. *What the program says: Most hours of lessons a section may have in a row before generation must give it a break (default 4). After that many hours back to back the next hour is kept free - the break's length is an advanced setting, one hour by default. Needs 'Mandatory break for students'. 0 = off. Saved with the timetable.*
Mandatory break for lecturers	Same idea for teachers. Switch on if your collective agreement requires a break. *What the program says: Turns on the break rule for instructors: generation gives nobody more than the hours set to the right in a row, so a break has to come before their next lesson. The number is the teaching stretch, not the*

Setting	In practice
	break's length. Saved with the timetable.
Mandatory break for lecturers – hours (h)	The number of consecutive teaching hours before a break is required. *What the program says: Most hours an instructor may teach in a row before generation must give them a break (default 4). A longer unbroken run is refused, and a single lesson this long or longer needs a free slot on each side. It is not the break's length. Needs 'Mandatory break for instructors'. 0 = off. Saved with the timetable.*
Maximum travels per day – Lecturers	Relevant only for institutions with several buildings. 1 means a teacher may move between locations once a day. *What the program says: Most journeys between locations an instructor may make in one generated day (default 1). A journey is a move to a classroom at another location, counting the trip from the instructor's home location to the first lesson. Only classrooms with a location count. 0 = no journeys. Saved with the timetable.*
Maximum travels per day – Groups	The same for student groups. 0 forbids moving between locations within a day. *What the program says: Most journeys between locations a section may make in one generated day (default 1). A journey is a move to a classroom at another location, counting the trip from the section's home location to the first lesson. Only classrooms with a location count. 0 = no journeys. Saved with the timetable.*
Select Location To Suite	Decides whose location restricts the choice of rooms. Default: the section's. Switch to the instructor's if teachers are tied to a building. *What the program says: Whose location decides which classrooms generation may use: the instructor's, the division's or the section's. Classrooms at another location are skipped. A section with no location uses its division's. With no location set, or a classroom without one, any classroom may be used. Default Section. Saved with the timetable.*

3.3 Decision: how much effort for the first draft?

The generator builds the whole week from scratch several times and keeps the best attempt. The default is **2 iterations**. If your school is hard (many constraints, few rooms), 4–8 iterations give a better start, at the price of a longer first step. The other two numbers, *Num. Of Iterations (Lecturers)* and *Num. Of Iterations (Students)* (64 each), control a quick hole-closing that runs right after placement. For serious polishing you will use the gap solver instead (Part 5); leave these alone.

Setting	In practice
Num. Of Iterations	More iterations mean a better starting timetable but a longer first step. 2 is a good default; try 4–8 for a hard school and let it run while you take a break. 0 makes Generate do nothing. *What the program says: How many times generation builds the week from scratch (default 2). Each attempt clears all unlocked lessons, places them again and tidies gaps, and the*

Setting	In practice
	best attempt is kept: fewest unplaced lessons, then fewest gaps. More attempts take longer. 0 = Generate does nothing. Saved with the timetable.
Num. Of Iterations (Lecturers)	The built-in quick hole-closing for teachers. For serious polishing use the gap solver instead; keep this at 64. *What the program says: How many rounds the instructor-gap optimiser runs after lessons are placed (default 64). Each round moves lessons to close idle gaps in instructors' days while keeping sections' gaps in check. More rounds take longer and can leave fewer gaps. Values below 1 count as 1. Saved with the timetable.*
Num. Of Iterations (Students)	The built-in quick hole-closing for students; it runs before the instructor optimiser. Keep at 64. *What the program says: How many rounds the student-gap optimiser runs after lessons are placed (default 64). Each round moves lessons to close empty hours between a section's lessons. It runs before the instructor optimiser. More rounds take longer. Values below 1 count as 1. Saved with the timetable.*

3.4 Decision: how should lessons be arranged?

These tick-boxes change the *style* of the timetable. None is right for every school. A useful way to think about them is in four groups.

Where lessons may go. *Generate also first / second weekend day* open the weekend. *Teachers come every working day* (unticked) makes the generator fill the days a teacher already works first, so teachers can have whole days off. *Scatter generated hours* randomises the order of days, which spreads lessons evenly but makes days less compact – the opposite of what most schools want.

What belongs together. *Schedule multiple sections into same hour* keeps a lecture split among several sections at the same hour. *Force same rooms for blocks* keeps all parts of a block in one room. *Try electives at the same hour* gathers parallel electives into one common slot. *Align hours for the course part* gives part-time and evening courses a stable weekly rhythm. *Anchors with hour relations are on same day* keeps a lesson that is anchored "two hours after the lecture" on the same day as the lecture.

How days feel to pupils. *Allow M+N classes in neighboring days* decides whether a 2+2 course may fall on Monday and Tuesday; off, there is at least one free day between the parts. Four more preferences exist in the **school edition** of the program: *Allow a double lesson of the same course* lets two one-hour lessons of the same course meet on one day, back to back; *Keep the first hour filled* and *At least five hours a day* tidy the day for pupils; *Allow moving away from short days* lets the generator take lessons out of a short teacher day – read its explanation in Appendix C carefully before you switch it on.

How careful the generator is about students. *Check students: groups* and *Check students: classes* make the generator also avoid clashes for individual students, who may belong to several groups or choose individual courses. They are essential for programmes with many electives and they slow generation down.

Setting	In practice
Generate also first weekend day	Switch on only if your school teaches on Saturdays. Whether Saturday is shown at all is a separate setting on the Appearance tab. *What the program says: Lets automatic generation place lessons on the first weekend day (normally Saturday). Off: only iterations allowed on that day alone go there. Whether the day is shown is a separate View setting, and placing lessons by hand is not affected. Saved with the timetable.*
Generate also second weekend day	The same for the second weekend day. *What the program says: Lets automatic generation place lessons on the second weekend day (normally Sunday). Off: only iterations allowed on that day alone go there. Whether the day is shown is a separate View setting, and placing lessons by hand is not affected. Saved with the timetable.*
Schedule multiple sections into same hour	Keeps the parts of a split lecture at the same hour. Switch on when one lecture is given to several sections at once. *What the program says: When one iteration is split into parts for different sections (same lecture, same instructor), generation places all the parts at the same day and hour, each in its own classroom of similar size, and keeps them together. The conflict check then does not report the shared instructor. Saved with the timetable.*
Force same rooms for blocks	For block course types (*BL*): later parts stay in the room of the first part. Switch off if the room can change between blocks. *What the program says: For course types marked as blocks (BL, e.g. Lecture_BL_ then Tutorial_BL_), which generation places back to back, the later parts get the same classroom as the first part. Off: each part may get a different classroom. Saved with the timetable.*
Try electives at the same hour	Puts parallel electives into one common slot. Useful when students choose among electives of the same kind. *What the program says: When placing an elective, generation first tries the day and hour where another elective of the same type (lecture with lecture) is already placed, so parallel electives share one slot instead of spreading over the week. Saved with the timetable.*
Align hours for the course part	When generating week by week, reuse last week's slots. Gives a stable weekly rhythm for part-time and evening courses. *What the program says: When generating week by week, generation first tries to give a section's lessons of a course part the hours they had in earlier weeks, even when another iteration or instructor teaches them now, so the course part keeps one pattern. Off: only lessons of the same iteration reuse their hours. Saved with the timetable.*
Anchors with hour relations are on same day	For hour-based anchors ("2 hours after the lecture"), keeps the dependent lesson on the same day. A personal setting. *What the program says: For anchors set in hours (for example*

Setting	In practice
	2 hours after its anchor lesson), keeps the lesson on the same day as the anchor. Off: it may also go on a later day (an earlier one for 'before'). Personal setting for your Windows user, so another planner may generate under a different rule.
Allow M+N classes in neighboring days	Allows 2+2 courses on Monday and Tuesday. Off: at least one free day between the parts, which is better for students. *What the program says: Lets the two parts of a course whose weekly hours are entered as M+N (for example 2+2) fall on neighbouring days, such as Monday and Tuesday. Off: generation leaves at least one day between them for the section. On by default. Saved with the timetable.*
Allow a double lesson of the same course (school edition)	Allows two one-hour lessons of the same course on one day as a double lesson. Off: at most one per day. *What the program says: Lets a section get two one-hour lessons of the same course and type on one day, but only back to back (a double lesson). Off: generation gives a section at most one such lesson per day. Saved with the timetable.*
Scatter generated hours	Randomises the order of days. Gives a more even spread over the week but less compact days. *What the program says: Generation tries the days in random order and, on about half of them, fills the day from its end instead of from the morning. Lessons spread over the week and the day instead of filling the same days and the morning hours first. Saved with the timetable.*
Keep the first hour filled (school edition)	A preference to keep the first hour of the day occupied for sections. Tidier for pupils, less flexible for the generator. *What the program says: Keeps the first hour of the day filled for sections: when generation tidies gaps it treats an empty first hour as a gap to close, and it will not move a lesson out of the first hour. A strong preference rather than a guarantee. Saved with the timetable.*
Allow moving away from short days (school edition)	Lets the generator move lessons away from a short teacher day. May leave a one-hour day behind – read the explanation carefully. *What the program says: Ticked, generation may move lessons away from an instructor's short days (under about 2.5 hours), which can leave a one-hour day. Unticked, it will not move lessons off such days - the opposite of what the label suggests. Saved with the timetable.*
At least five hours a day (school edition)	Tries to give every section five hours a day when it has lessons. A preference, not a guarantee. *What the program says: Generation tries to give each section at least five hours on every day it has lessons: it moves last lessons from longer days to days with four hours or fewer, and will not move a lesson out of the fifth hour. A preference, not a guarantee. Saved with the timetable.*
Check students: groups	Needed when students belong to several groups. Slows generation down. *What the program says: Generation also*

Setting	In practice
	looks at individual students' section memberships: two lessons of different sections that share a student are never placed at the same time. Use it when students belong to several sections. Needs students entered with their sections, and can slow generation. Saved with the timetable.
Check students: classes	Needed for elective-heavy programmes where each student has an individual set of courses. Slows generation down. *What the program says: Generation also looks at each student's own course enrolments: two lessons that any one student attends are never placed at the same time. Needs students entered with their courses on the Students screen, and can slow generation. Saved with the timetable.*

3.5 Warnings

Warnings are advice for the person typing; they do not limit generation. They are personal to your Windows user, so another planner keeps his own.

Setting	In practice
Warn – prof. weekly hrs	A personal warning when you assign a teacher too many hours. It does not limit generation. *What the program says: When you assign an instructor to a course, WTT warns if their lessons in any week would add up to more than this many hours (default 40). Over 100 iterations the warning off. A warning only - generation does not use it. Personal setting: applies to your Windows user only, other planners keep their own.*
Groups load warning, level 1	Weekly-hours warning for sections that carry the level-1 flag. *What the program says: Weekly hours limit for sections ticked 'Warning hrs/week, level 1' on the Sections screen. When you confirm a course that takes such a section over it in any week, you are asked whether to ignore it. A warning only - generation does not use it (default 15). Personal setting: applies to your Windows user only, other planners keep their own.*
Groups load warning, level 2	The same for the stricter level-2 flag. *What the program says: Weekly hours limit for sections ticked 'Warning hrs/week, level 2' on the Sections screen. When you confirm a course that takes such a section over it in any week, you are asked whether to ignore it. A warning only - generation does not use it (default 30). Personal setting: applies to your Windows user only, other planners keep their own.*

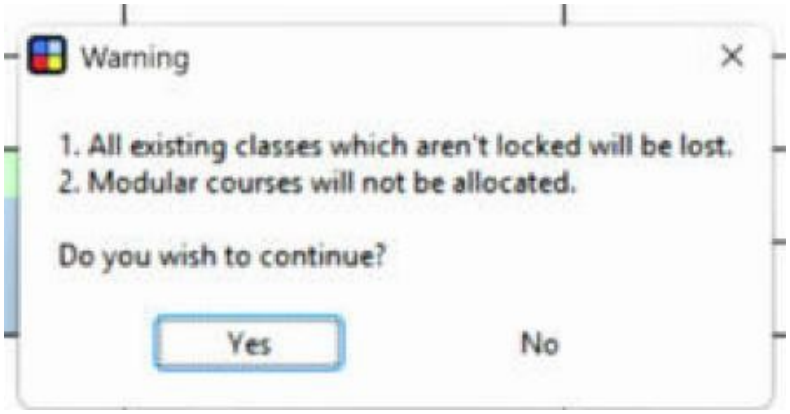
Part 4 – The first generation

4.1 Press Generate

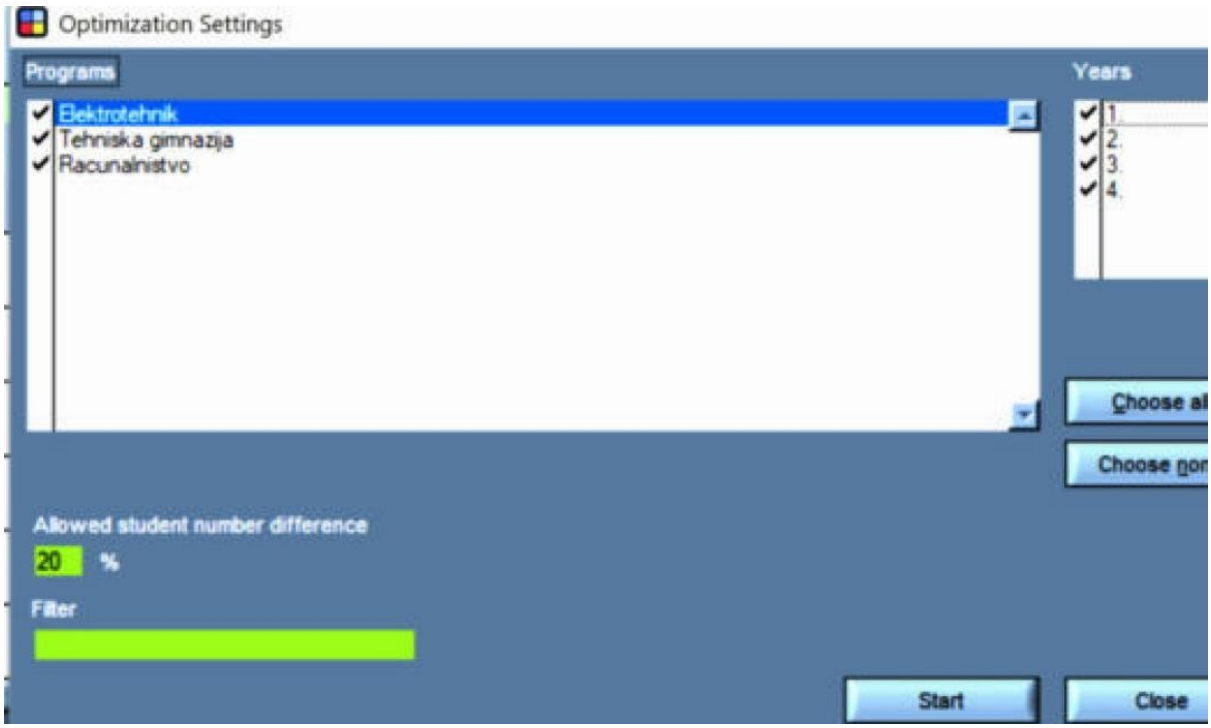
Generate is the alarm-clock button in the toolbar at the top right of the main window (hover over it and the tooltip says *Generate*).



Two things happen before anything is placed. First a warning: every lesson that is not **locked** will be removed and placed again, and modular (non-repeating) courses are never generated. If you have placed lessons by hand that must stay, lock them first and only then answer **Yes**.



Then the **Optimization Settings** window asks *what* to generate.



- **Programs** and **Years** – tick the divisions and grades whose lessons should be placed. Everything unticked keeps its lessons exactly as they are. This is how you

generate one grade at a time, or regenerate a single programme after a change. The ticks are saved with the timetable.

- **Allowed student number difference** – how much smaller than the section a classroom may be, in percent. At 20, a section of 50 may get a room with 40 seats. 0 means every room must seat the whole section.
- **Filter** – limits the run to some courses: part of a course-type name (for example *Lab*) places only lessons of those types; a week range such as *5-10* places only courses with an iteration in those weeks. Leave it empty for everything.
- **Start** begins the run; **Close** leaves without generating.

For a school of a thousand lesson hours the run takes seconds to a few minutes, with a progress window that can stop it. If **Run gap optimisation** is ticked on the Generate tab, the gap solver follows for the minutes you set there. When it all ends, read the message first: *"All placed"* is what you want.

4.2 If some lessons are unplaced

The program lists them. Do not panic: an unplaced lesson is information. Usually one of these is true.

- There is **no room** with the needed properties at the needed times.
- The **teacher or the group** has no free slot that also satisfies the hard rules (last hour of the day, afternoon limit, daily maximum).
- The **hours window** is too narrow for the amount of teaching.

Widen the Extended working hours a little, or add a room, and generate again. Part 9 has a step-by-step procedure.

4.3 Look before you polish

Before polishing, press **Ctrl + B** and write down the numbers (Part 8). You will compare them with the numbers after every improvement. Without a "before" you cannot know whether a change helped.

Part 5 – Polishing with the gap solver

5.1 What it is

The generator is fast but it knows little about comfort. The gap solver is the patient part. It works like a very experienced timetabler: it takes a handful of lessons out, tries to put them back better, keeps the change if the timetable is now cheaper, and repeats – tens of thousands of times in one run, on all your processor cores.

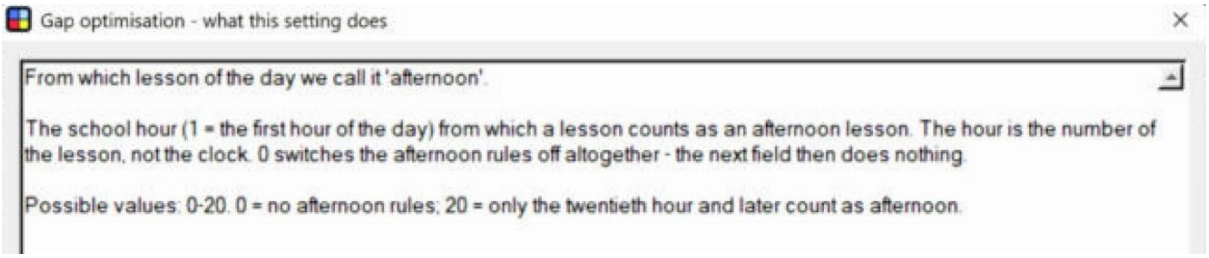
It is switched on with two controls on the Generate tab itself: the tick-box **Run gap optimisation** and **Duration (min)**, the number of minutes it may work after the generator has placed the week. With the box ticked, every **Generate** is followed by a polishing run of that length. The **?** next to the duration explains it.

Everything else about the solver is behind the button **Gap optimisation settings ...**. The window has three columns.



Column	Title in the window	Contents
Left	**Generator settings**	the rules of your school: afternoon, minimum and maximum teacher hours
Middle	**Gap solver settings**	the weights of a day, a short day and a linked pair; the kick; the number of CPU threads
Right	**Advanced gap solver settings**	whether teachers come every day, the home room, electives at the edge of the day

Every field has a **?** button. The long explanation behind it is in Appendix C.



5.2 The left column – the rhythm of your school

These are the settings a planner should always check. They describe how your school *feels*.

Afternoon starts at hour says from which school hour on a lesson counts as afternoon (the hour's number, where 1 is the first hour of the day). *Afternoon days per class and week* is a **hard rule**: how many days a week a class may have afternoon lessons. If the generator already violated it, the gap solver moves lessons until it holds. Two teacher settings are wishes: *Min teacher hours per day* (nobody should come to school for one lesson; each missing hour costs ten hole-hours) and *Max teacher hours per day (soft)* (a day longer than this is penalised, three hole-hours per extra hour; the hard daily maximum stays on the Generate tab).

A set of settings that works for a typical morning secondary school: *Afternoon starts at hour* 10, *Afternoon days per class and week* 1, *Min teacher hours per day* 2, *Max teacher hours per day (soft)* 5, *Last hour of the day* 9 on Friday and 14 on the other days.

Setting	In practice
Afternoon starts at hour	In a school with 14 lessons a day, 10 means "from the 10th lesson on it is afternoon". Zero switches the afternoon rules off. *What the program says: From this school hour on a lesson counts as afternoon (0 = no afternoon rule) (Range 0-20)*
Afternoon days per class and week	A hard rule. 1 gives every class at most one afternoon a week. If the generator already violated it, the gap solver moves lessons until the rule holds. *What the program says: A class may have afternoon lessons on at most this many days a week (0 = no limit) (Range 0-7)*
Min teacher hours per day	Set 2 so that nobody commutes for one lesson. A soft rule: each missing hour costs ten hole-hours. *What the program says: A teacher who comes in teaches at least this many hours that day (0 = off) (Range 0-8)*
Max teacher hours per day (soft)	Set 5 or 6 to discourage very long teacher days. Soft rule; the hard daily limit is on the Generate tab. *What the program says: Above this many hours a teacher's day is penalised; the hard limit stays on the Generate tab (0 = off) (Range 0-10)*
Teachers come every working day	Ticked (the default): a teacher is expected on every day he is not blocked, so an unused day costs and the generator spreads lessons over the week. Unticked: the generator first fills the days a teacher already works, so teachers can get whole free days, and the day value on the right decides how much the solver pays for one. This box replaced the old "allow empty

Setting	In practice
	days" switch of the Generate tab. *What the program says: On: a teacher is expected every day he is not blocked - a missing day costs; off: the day value above applies*

5.3 How long and how hard

Duration (min) on the Generate tab is how many minutes the solver works. Part 6 is about choosing it well. **CPU threads (0 = all but one)** at the bottom of the middle column is how many processor cores it may use; 0 means all but one, which is perfect for an overnight run. If you want to keep working on the same computer, enter 4 or 8.

Setting	In practice
Run gap optimisation / Duration (min)	Ten minutes for experiments, thirty to sixty for a timetable you will publish, several hours for the final one. Stopping never loses the best result. *What the program says: How long the gap solver works on the week after the generator has placed it. It never breaks a rule the generator kept; it only moves lessons to close holes, and it keeps the best week it found at every moment, so stopping it early (Prekini) loses nothing.*
CPU threads (0 = all but one)	0 uses every core but one – ideal for an overnight run. Enter 4 or 8 if you still want to work on the same computer. *What the program says: CPU threads for the solver (0 = every core but one) (Range 0-64)*

What you see while it works

While the solver runs, a progress line appears at the bottom of the window. It updates **once a minute**, calm enough to leave overnight, and shows how many holes remain for pupils and teachers, how many lessons are in the afternoon, and which of your rules are currently kept. **Stop** stays available the whole time.

Long runs are safe. The best week found so far is stored continuously. Stopping, closing the window or a power failure costs you at most the last few minutes.

5.4 The middle and right columns – the weights

Leave these alone until you have run Part 6 a few times. The defaults were measured on real schools: they are the values that gave the best timetables there. The four groups are worth understanding even if you never touch them.

What a day, a short day and a split pair cost. *Value of a teacher's day of presence (hole-hours)* is what a teacher's extra day at school is worth; high values squeeze hours into fewer days. *Minimum lessons per day for pupils* and *Penalty for a short day* keep classes from coming for one or two lessons. *Penalty for a linked pair placed apart* keeps two lessons that belong together – a class split into two workshops – starting at the same time.

How adventurous the search is. When the search stops improving, the solver may deliberately accept a slightly worse timetable to escape a dead end – a *kick*. *Kick depth* is how bold the step is, *Kick after* how many seconds without improvement it waits before trying, *Kick frequency* how often a stalled attempt is a kick. These are the only settings that change *how* the solver searches rather than *what* it wants.

Whether teachers come every day. *Teachers come every working day* (right column, ticked by default) tells the generator and the solver that a teacher is expected on every day he is not blocked, so an unused day costs. Untick it for a staff that values whole free days: the generator then fills the days a teacher already works first, and the day value above decides how many hole-hours one more day is worth.

What places care about. *Teacher's home room – weight* and *Electives at the edge of the day* are about comfort: teachers in their own classrooms, electives at the start or end of the class's day so that pupils without that elective do not wait.

Setting	In practice
Value of a teacher's day of presence (hole-hours)	At 5 the solver sacrifices up to five hole-hours to send a teacher home for a whole day. Raise it if teachers value free days more than compact days. *What the program says: Hole-hours one teacher-day is worth (0 = ignore days) (Range 0-20)*
Minimum lessons per day for pupils (0 = off)	Prevents classes from coming to school for one or two lessons. 4 is the measured default. *What the program says: Minimum lessons on a school day for a class (0 = off) (Range 0-8)*
Penalty for a short day	How painful each missing lesson below the minimum is. 20 is firm without being absolute. *What the program says: Penalty per lesson short of that minimum (Range 0-100)*
Penalty for a linked pair placed apart	Two lessons that belong together – a class split into two workshops – should start together. 200 keeps them together almost always. *What the program says: Penalty per linked pair placed apart (0 = links ignored) (Range 0-1000)*
Teacher's home room – weight	Switch on (e.g. 3) if teachers have their own classrooms. 0 ignores home rooms. *What the program says: How much a lesson outside its teacher's own room costs (0 = ignore home rooms) (Range 0-20)*
Electives at the edge of the day	Electives belong at the start or end of the day; in the middle they leave the other pupils waiting. 30 is a good value. *What the program says: How much an elective block in the middle of the class's day costs (0 = ignore) (Range 0-200)*
Kick depth (0 = off)	The depth of a deliberate step backwards. 4 gave the best timetables in measurements; larger values waste time on repairs. *What the program says: How much worse a move may be, to escape a dead end (0 = off) (Range 0-10)*
Kick after (seconds without improvement)	How long the solver waits without improvement before it kicks. 20 seconds is the measured default. *What the program says: Seconds without improvement before the solver starts kicking (Range 5-120)*
Kick frequency (1 in N tries)	2 means every second try of a stalled search is a kick. *What the program says: While stuck: one forced move in this many tries (Range 1-10)*

Part 6 – The improvement loop

A timetable is not generated; it is **grown**. The first draft takes seconds. The next 95 per cent of quality comes from a loop that you run several times: measure, change one thing, run a short polish, measure again. Done with a little discipline it takes an afternoon, and it is the difference between a timetable that is accepted and one that is loved.

6.1 The loop in five steps

1. **Measure.** Ctrl + B. Write down: pupil holes, teacher holes, afternoon lessons, one-hour days, lessons without room.
2. **Choose one thing to fix.** The most expensive problem first: lessons without a room, then pupil holes, then rule violations, then teacher holes, then comfort.
3. **Change one setting.** One. If you change three, you will not know which of them helped.
4. **Run a short polish** (see 6.2 for how short).
5. **Measure again and compare.** Better? Keep the change and save the file under a new name. Worse? Undo the setting. Then go to step 2.

6.2 How long should each run be?

The solver improves fast at first and slowly later, so the efficient strategy is to **climb a ladder of run times**. Use the shortest run that can answer your question.

Run time	Question it answers	When to use it
1–2 minutes	"Did this setting break something obvious?"	Every time you change a setting. If nothing is obviously worse, continue.
5 minutes	"Is this direction better than the last?"	The standard step of the loop.
10–20 minutes	"How good can this configuration get?"	When the 5-minute runs have stopped changing.
30–60 minutes	"Is this the timetable I will publish?"	A candidate for publication.
Several hours or overnight	"Squeeze out every last hole."	Only for the final version, with all settings fixed.

The rule of thumb. Never run longer than ten minutes while you are still changing settings. A long run with the wrong settings is a long wait for a wrong answer. Settle the settings with short runs, then spend the long run once.

6.3 What each symptom tells you to change

What you see	Change this	Why
Lessons without a room	Data first: room properties, seats. Then widen the Extended working hours.	The solver treats a missing room as a hard failure; it cannot invent a room.
Pupil holes remain	Run longer; lower *Afternoon days per class and week* only as a last resort	Pupil holes carry the biggest price; they normally disappear within a few

What you see	Change this	Why
		minutes.
Teacher holes remain high	Lower *Value of a teacher's day of presence*; run longer; check teachers' own limits	Teachers' holes are the slowest to shrink; the last ones are the hardest.
Too many afternoon lessons	Narrow Extended working hours; set *Afternoon days per class and week*	This is a hard rule, so it can be enforced exactly.
Teachers with one-hour days	Raise *Min teacher hours per day*, raise *Value of a teacher's day of presence*	The solver prefers to add an hour or to empty the day.
Teachers with very long days	Lower *Max teacher hours per day (soft)*	A wish; it costs three hole-hours per hour beyond.
Electives in the middle of the day	Raise *Electives at the edge of the day*	Pupils without the elective otherwise wait.
Teachers not in their own room	Raise *Teacher's home room – weight*	The room has to carry what the lesson needs, too.
The numbers have not moved for a long time	Stop, relax **one** rule, run again	The remaining cases are the hardest ones; it is normal that they take longer than all before.

6.4 Worked example – one night on a real school

This example comes from a secondary school with 976 lesson-hours a week in 622 turns, 82 teachers (78 of them teaching that week), 36 classes and 51 rooms, most of the rooms with short lists of what may be taught in them. It was generated under a morning rhythm with a short Friday (last hour 9), at most one afternoon day per class, teachers in school on every day they can work, two to five hours a day, electives at the edge of the day.

The table has two runs. **Run A** (rows A 00 to A 12) is the generator followed by four hours of polishing in twenty-minute steps, with the default weights. **Run B** (rows B 01 to B 28) continues overnight from the end of run A for another nine hours, with one change: a teacher's hole-hour weighs five instead of one. *Minutes* count from the start of run A; *Pupil holes* counts holes of whole classes.

Snap	Min	Pupil holes	Teacher holes	Teacher days	1-h days	> 5 h	Fri late	PM cap	PM hours	No room
A 00	0	431	891	362	20	51	39	60	223	0
A 01	19	0	80	361	2	26	0	0	7	0
A 02	39	0	47	361	1	25	0	0	7	0
A 03	59	0	36	361	1	22	0	0	7	0
A 05	99	0	34	361	1	23	0	0	6	0
A 07	139	0	32	361	1	23	0	0	6	0
A 08	159	0	36	361	1	23	0	0	6	0
A 12	239	0	35	361	1	23	0	0	6	0

Snap	Min	Pupil holes	Teacher holes	Teacher days	1-h days	> 5 h	Fri late	PM cap	PM hours	No room
B 01	260	0	17	361	0	26	0	0	9	0
B 06	360	0	16	361	0	27	0	0	9	0
B 10	441	0	13	361	0	27	0	0	6	0
B 11	461	0	12	361	0	27	0	0	6	0
B 15	541	0	12	361	0	27	0	0	6	0
B 28	802	0	12	361	0	27	0	0	6	0

PM cap: classes with afternoon lessons on more than one day; *PM hours*: lessons from the afternoon hour on.

What the table teaches:

- **The first twenty minutes do almost everything.** The generator's week had 431 pupil holes, 891 teacher hole-hours and 60 broken afternoon rules; twenty minutes later every hard rule held and the teachers were down to 80. Ninety-five per cent of all improvement happened in the first hour.
- **Then it plateaus.** Run A spent its last three hours between 32 and 36 teacher hole-hours, and at A 08 the teacher holes even rose from 32 to 36 – because the solver judged that filling one short pupil day was worth more. That is the weights speaking, not the solver being lost.
- **Changing a weight beats running longer.** Run B, with the teacher hole made five times dearer, took the teacher holes from 35 to 17 in its first twenty minutes and to 12 within four hours. It paid with four more teacher days above five hours and one afternoon lesson more – a trade a timetabler chooses, not one the program should choose silently.
- **Know when to stop.** From B 11 (about 460 minutes) nothing moved again. Two quiet twenty-minute runs in a row are the signal; the rest of that night was electricity.

6.5 When to stop

Stop when **two successive 20-minute runs improve nothing**, or when the remaining numbers are inside what you can live with. A timetable with three teacher holes is a good timetable. Do not tune a school towards zero holes at the price of a thousand clicks; the last holes are often caused by constraints in the data (a teacher who can only come on Monday and Thursday), not by the solver.

6.6 Keep a diary

For each run write one line: date, settings changed, run time, the numbers. Ten lines later you will know your school better than anyone. Save the file under a new name each time a run was better, so you can always go back.

Part 7 – Changing a timetable that is already published

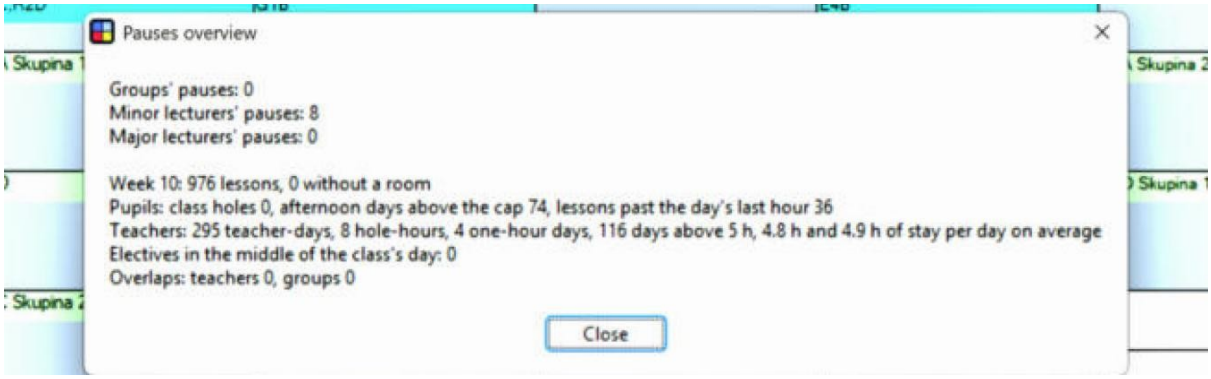
During the term things change: a teacher falls ill, a room is closed, a new group appears. You do not want a new timetable; you want the old one with one change.

1. **Lock** every lesson you do not want to move. Locked lessons are never touched by the generator or the gap solver.
2. Make the change by hand, or free the lessons that must move, and press **Generate** to place only the free ones.
3. Run a **5-minute polish**. It tidies the neighbourhood of the change and leaves the rest.
4. Compare Ctrl + B before and after; publish.

The more you lock, the faster and safer the change; the less you lock, the more the solver can improve. A good habit is to lock everything outside the affected week and the affected class.

Part 8 – Reading the numbers

Press **Ctrl + B** for the week statistics report (its window is called *Pauses overview*). The screenshot shows the report of a freshly generated week, before any polishing.



The first three lines are the classic pause counts of the program: *Groups' pauses* (holes of the sections), *Minor lecturers' pauses* (hours a teacher waits between two lessons) and *Major lecturers' pauses* (the same, weighted for long waits). Below them is the block that the gap solver works on, line by line.

Line	Meaning	Goal
lessons, without a room	how many lessons the week holds and how many of them have no room; a lesson without a room counts as **not placed**	**0 without a room**
class holes	empty hours *between* a class's lessons on one day	**0**
afternoon days above the cap	class-days with afternoon lessons beyond *Afternoon days per class and week*	**0**
lessons past the day's last hour	lessons ending after *Last hour of the day*	**0**
teacher-days	how many (teacher, day) pairs there are	fewer = more days off
hole-hours	the hours teachers wait between lessons, summed	as low as possible; a few are almost always unavoidable
one-hour days	teachers who come in for a single lesson	as few as possible
days above 5 h	long teacher days	as few as possible
stay per day on average	hours taught and hours present, averaged over teacher-days	the two numbers close together
Electives in the middle of the class's day	elective blocks with other lessons of the class before and after	**0**
Overlaps	a teacher or a group twice at	**0**, always

Line	Meaning	Goal
	the same time	

A measured example. On one secondary school (about 620 lesson turns), the previous timetable contained 14 pupil holes and 187 teacher hole-hours when measured by the new rules. The Wise result reached 0 pupil holes with fewer teacher holes. Your numbers will differ: compare **before and after on your own data**.

Part 9 – When something goes wrong

9.1 Lessons are not placed

Possible cause	How to check	What to do
No room with the required properties	Open the unplaced lesson; read which properties the course asks for; look for a room that has them.	Add the property to a room or remove it from the course if it was unnecessary.
Rooms too small	Compare the group size with the seats of the rooms.	Split the group, or add a bigger room.
All rooms of the right kind are taken at all hours	Count the lesson-hours that need that room type against the rooms and hours available.	Add a room, widen the Extended working hours, or move some lessons to another room type.
Teacher without free slots	Look at the teacher's weekly hours, the daily maximum, the unavailable days.	Raise the daily maximum for that teacher, or free a day.
The hours window is too narrow	Sum of the lesson-hours of the busiest class against the available hours.	Widen the Extended working hours a little.
The last hour of the day is too early	Compare with the hours that the class needs on that day.	Allow one more hour on that day.
Afternoon days limited to 0 or 1 and the day is too short	Check *Afternoon days per class and week* and the hours window.	Allow an afternoon day, or widen the Working hours.

9.2 Too many holes after polishing

First check that the solver had time: a 5-minute run on a large school is a sketch. Then check for *structural* causes.

- **A very strict combination of rules.** No afternoons, no holes, a short Friday, teachers on few days: each of these is reasonable, together they may be impossible. The solver then shows the best compromise. Relax one rule at a time and see how many holes it buys.
- **Teachers with few available days.** A teacher who can come on Monday and Thursday only, and who teaches many classes, is the origin of many holes for those classes. Check the teacher's availability.
- **Linked turns that stretch.** Two linked turns of different length force the class to wait.
- **Electives.** A block of elective in the middle of the day leaves the others waiting; mark them as electives.

9.3 The solver seems to be stuck

It is rarely stuck; it is working on the hardest cases. If the progress line shows no change for a long time, wait for a further 20 minutes before concluding anything. If you still see no change, stop. The best result is saved. Change one setting and run again.

9.4 Teachers complain

Complaint	Where to look
"I have a one-hour day."	*Min teacher hours per day*; *Value of a teacher's day of presence*; check if the teacher has fewer hours than the minimum.
"My day is too long."	*Max teacher hours per day (soft)*; the teacher's own *Allowed hours per day*.
"I am always in a different room."	Set the teacher's default room; raise *Teacher's home room – weight*.
"I wait too long between lessons."	Run the gap solver longer; check the teacher's hours spread over the week.
"I have no free day."	*Teachers come every working day* (unticked); raise *Value of a teacher's day of presence*.

9.5 Pupils or parents complain

Complaint	Where to look
"My class has a hole."	Pupil holes – run longer; check linked lessons and electives.
"We have lessons in the afternoon too often."	*Afternoon days per class and week*; narrow Extended working hours.
"Our Friday goes on too long."	*Last hour of the day* for Friday.
"Elective in the middle of the day."	Mark as elective; raise *Electives at the edge of the day*.
"Two hours of the same subject in a row."	*Allow a double lesson of the same course*.

9.6 Strange or unexpected results

Symptom	Likely reason
The timetable is perfect but for the wrong lessons	The data does not match the curriculum – check hours per week of the courses.
A setting seems to have no effect	Some settings (for example *Check all weeks*) no longer have an effect; the others only take effect in a new run. Generate or polish again after changing.
The result looks different every time	The solver uses randomness. Equal quality, different arrangement. Save the version you like.
It got worse after a long run	It cannot: the best version found is kept. Check whether you loaded the correct file

Symptom	Likely reason
	or changed settings in between.
Generate does nothing	*Num. Of Iterations* is 0.
A day appears empty for a teacher who “should” teach	The teacher may be blocked that day; check reservations and the teacher’s availability.

9.7 If nothing helps

Save the file, write down the settings and the Ctrl + B numbers, and send them to support. With those four things – the file, the settings, the numbers and a description of what you expected – the problem is usually solved in one exchange.

Part 10 – Recipes

10.1 A small school or one department (up to about 200 lessons)

Generator with the defaults, then the gap solver for 10 minutes, then check Ctrl + B and publish.

10.2 A secondary school with a morning rhythm and a short Friday

1. In Misc → Generate set **Working hours** to the normal day, **Extended working hours** only as wide as needed, and **Last hour of the day** for each weekday.
2. In the gap solver set *Afternoon starts at hour* **10**, *Afternoon days per class and week* **1**, *Min teacher hours per day* **2**, *Max teacher hours per day (soft)* **5**.
3. Follow the loop of Part 6: short runs while settling the settings, then 30–60 minutes, then overnight for the final version.

10.3 A university with many rooms and buildings

Check room equipment first. Switch on *Force same rooms for blocks*, give the teachers their home rooms and set *Teacher's home room – weight*, restrict journeys between locations, and give the solver at least 30 minutes per round.

10.4 An evening or part-time programme

Switch on *Align hours for the course part*. Narrow the Working hours to the evening window. Avoid *Scatter generated hours*.

10.5 A school with two shifts

Mark the second-shift sections on the Sections screen, set the Second shift hours, and generate. Check that rooms are not double-used at the shift change.

Appendix A – Keyboard shortcuts

Shortcut	Action
Ctrl + B	week statistics report
Ctrl + Shift + X	export to the EAS system

Appendix B – Glossary

- **Turn** – one teaching activity of a course (for example “Maths, group 2A, teacher X”).
- **Hole / gap** – an empty hour between two lessons of the same class or teacher on the same day.
- **Hard rule** – never broken: clashes, last hour of the day, afternoon-day limit, a room present.
- **Soft rule / wish** – may be broken at a price: minimum teacher hours, home room, electives at the edge.
- **Polish** – one run of the gap solver.
- **Working / extended working hours** – the preferred window and its extension.
- **Elective** – a subject that only part of a class attends.
- **Hole-hour** – the common unit of cost: one hour a teacher waits.
- **Kick** – a deliberate step backwards the solver takes to escape a dead end.

Appendix C – The program’s own help texts

The texts below are the ones the program shows when you hover over a field or press **?**. They are reproduced here word for word so that this book and the program always say the same.

Run gap optimisation / Duration (min)

How long the gap solver works on the week after the generator has placed it. It never breaks a rule the generator kept; it only moves lessons to close holes, and it keeps the best week it found at every moment, so stopping it early (Prekini) loses nothing.

What to expect: the first minutes bring the big gains - on a school of 1000 lessons the pupil holes usually fall to zero within 2-5 minutes and the teacher holes drop fast. After that the gains come slower and in smaller steps; the last few teacher holes can take as long as everything before them. 10 minutes is a good everyday setting; 30-60 minutes for the timetable you will publish; overnight (several hours) when every hole counts - the solver keeps improving as long as it runs, only more slowly.

When it stalls: on a plateau the solver kicks itself (see Depth of the kick) and goes on; if the progress line does not change for 5-10 minutes, more time will rarely help - the remaining holes are forced by the data (a teacher’s availability, a room, a fixed lesson). Look at the teachers who keep their holes and at what is fixed around them.

How to follow it: the progress screen writes one line a minute at most - pupil holes, teacher hole-hours, teacher-days, lessons without a room, rules broken, and the seconds. More CPU threads (gap settings) shorten the run roughly in proportion; the machine stays usable meanwhile.

Afternoon starts at hour

From which lesson of the day we call it ‘afternoon’.

The school hour (1 = the first hour of the day) from which a lesson counts as an afternoon lesson. The hour is the number of the lesson, not the clock. 0 switches the afternoon rules off altogether - the next field then does nothing.

Possible values: 0-20. 0 = no afternoon rules; 20 = only the twentieth hour and later count as afternoon.

Afternoon days per class and week

How many afternoons a week a class may have at most.

How many days a week a class may have lessons in the afternoon, counted with the hour above. It is a hard rule: the solver never places a class on more afternoon days than this, and where the generator already did, it moves lessons until the rule holds. 0 means no limit.

Possible values: 0-7. 0 = no limit; 1 = a class has afternoon lessons on one day a week at most.

Min teacher hours per day

Nobody should come to school for a single lesson.

When a teacher comes to school for a day, that day should hold at least this many lessons. Shorter days are not forbidden, they cost: every missing hour weighs ten hole-hours, so the solver prefers to add a lesson to the short day or empty it altogether.

Possible values: 0-8. 0 = any day length is fine; 8 = a teacher’s day with fewer than 8 lessons is penalised.

Max teacher hours per day (soft)

How long a teacher’s day may be before it starts to count against the timetable.

A teacher’s day longer than this is penalised, three hole-hours for every hour above. It is a wish, not a rule - the hard limit per day is still ‘Max hours per day - lecturers’ on the Generate tab, which the solver never breaks.

Possible values: 0-10. 0 = off; 10 = only days above 10 lessons are penalised.

CPU threads (0 = all but one)

How much of the computer the solver may take while it runs.

How many CPU threads the solver may use.

0 = every core but one, right for a dedicated run. Give a smaller number to keep working on this computer while the polish runs. More threads find better timetables in the same time.

Possible values: 0-64. 0 = automatic (every core but one); otherwise exactly this many threads.

Value of a teacher's day of presence (hole-hours)

How much we want teachers to come to school on fewer days, even if that means a little more waiting between their lessons.

How much one day of a teacher's presence is worth, measured in hole-hours.

With 5, the solver gives up to 5 hole-hours of waiting to send a teacher home for a whole day (or not call them in at all). 0 = the solver does not care how many days a teacher comes in. A higher value packs each teacher's lessons into fewer days at the price of more waiting between them.

Measured default: 5.

Possible values: 0-20. 0 = days do not matter at all; 20 = a day at home is worth 20 waiting hours, so lessons are squeezed into as few days as possible.

Minimum lessons per day for pupils (0 = off)

Pupils should not come to school for just one or two lessons.

The smallest number of lessons a class (or half-class) should have on a day it comes to school.

A day with fewer lessons is penalised, so the solver rather frees that day entirely or fills it up. 0 switches this off. Default: 4.

Possible values: 0-8. 0 = any length of school day is fine; 8 = a day with fewer than 8 lessons is penalised.

Penalty for a short day

How much we dislike a day that is too short for the pupils.

The penalty for each lesson short of the minimum above, in hole-hour points.

20 means one missing lesson weighs like 20 teacher hole-hours. Only used when the minimum is above 0.

Possible values: 0-100. 0 = a short day costs nothing; 100 = one missing lesson weighs like 100 teacher waiting hours.

Penalty for a linked pair placed apart

Two lessons that belong together (the class split into two rooms at the same time) should stay together.

Lessons marked as linked - two subjects of one class that must run at the same time, for example a class split into two workshops - should start together.

This is the penalty, in hole-hour points, for every linked pair placed apart. 0 = the link is ignored. Default: 200.

Possible values: 0-1000. 0 = linked lessons may be placed anywhere; 1000 = keeping them together comes before almost everything else.

Teacher's home room – weight

How much we care that teachers teach in their own classroom.

Every teacher may have an own room (the default room on the teacher's screen). When a lesson could be held there - the room has what the lesson needs - but is given another room, it costs this many hole-hours. The solver then prefers the teacher's own room whenever it is free. 0 ignores home rooms.

Possible values: 0-20. 0 = home rooms are ignored; 20 = a lesson away from the teacher's own room weighs as much as 20 hours of waiting.

Electives at the edge of the day

How much we care that pupils who do not take an elective are not left waiting for it in the middle of the day.

Elective courses (the 'selectable' flag on the course) are not attended by every pupil of the class, so they belong at the start or the end of the class's day; in the middle, the pupils who do not take them wait. An elective block with class lessons both right before and right after it costs this many hole-hours.

Possible values: 0-200. 0 = electives may be anywhere; 200 = an elective in the middle of the day weighs more than almost anything.

Kick depth (0 = off)

How boldly the solver may take a step backwards to find a better way forward.

When the search stops improving, it may deliberately accept a slightly worse timetable to escape a dead end. This is the depth of that step, in objective points (one teacher hole-hour = 1 point, one pupil hole = 1000).

0 switches the kick off. Measured: 4 gave the best timetables; larger values waste time repairing the damage.

Possible values: 0-10. 0 = never kick (the search stops at the first dead end); 10 = very aggressive, much time goes into repairs.

Kick after (seconds without improvement)

How long the solver keeps trying normally before it dares a step backwards.

How many seconds without any improvement the solver waits before it starts kicking.

Shorter means more kicks and more churn; longer means more patience on a plateau. Measured default: 20 s.

Possible values: 5-120. 5 = kicks almost at once; 120 = waits two minutes before trying a kick.

Kick frequency (1 in N tries)

How often, once it is stuck, a try is a deliberate step backwards.

While the search is stuck, one try in this many is a forced move.

1 = every try, 2 = every second try (measured default), 10 = rarely.

Possible values: 1-10. 1 = every try is a kick while stuck; 10 = only one try in ten.

Appendix D – Switches in wtt.scheme

Next to the program lies a small text file, **wtt.scheme**. Its `[Advanced]` section holds switches that are read when the program starts and that apply to **this installation**, not to one timetable. They exist for special cases that one school once needed; none of them is required for a good timetable, and most installations should leave the file as it is. They are listed here so that a line in the file is never a mystery. Change a value only after a backup of the file, and restart the program afterwards.

A line is `Name=value`; a line starting with an apostrophe is a comment and is ignored. `1` means on, `0` means off.

MaxClassesPerDayForSomeGenerating
A hard rule for the generator: at most this many lessons of one course for one section on a day (a multi-hour lesson counts once). Default 3; 0 switches the rule off. *Lower it to 1 if a course must never meet twice on one day; the course's own 'Different days' tick does the same for a single course.*

GenBigNumberOfStudents
The number of students from which the generator treats a lesson as "big" and places it in an early pass, before the small ones. Default 70. *Lower it for a small school where 40 students is already a hall-sized lesson.*

AlternativeGenerateMethod
1 switches to an older placing order: a seminar or tutorial is tried right next to the lecture of the same course before anywhere else, rooms are chosen by how many lectures they already hold, and the phase that moves existing lessons is skipped. *Leave 0 unless support asks for it.*

GenerateByTutor
1 makes the generator walk the week teacher by teacher in a single pass instead of lesson by lesson in repeated passes. Not available in the school edition. *Leave 0.*

SpecialCaseGenDuration
1 turns on special handling of lesson lengths written for one institution's duration scheme (long lessons pinned to days, different thresholds for what counts as a long lesson). *Leave 0.*

AvoidAllDaysSixHours
School edition only. 1 stops the generator from giving a teacher the sixth hour on every day of the week. *Set it if teachers complain about "always the sixth hour".*

GenerateOverlapPrograms
Changes how the sections of programmes you did *not* tick in the Optimization Settings window are treated when only some programmes are generated. *Leave as it is.*

MandatoryPauseSize · AllowBlockPauseSize · AddPauseAfterEachHour
The lengths, in timetable lines, of the mandatory break (default 2 lines = one school hour) and of an empty line the generator may be asked to keep before and after every lesson of a section. *Leave as they are; the Generate tab's 'Mandatory break' settings are the normal way to ask for breaks.*

PercentageDiffForMoreRooms · PutGroupsInMoreRoomsRes
For lessons that book *additional* classrooms: how much smaller or larger than the main room an additional room may be (percent, default 30, never less than 6 seats), and whether the booking carries the sections. *Only for institutions that use additional rooms.*

AutoAdjustGroupSizes
1 recomputes the sizes of sections from their divisions when a timetable is loaded. *Leave 0 unless sizes are maintained in the divisions.*

AutoCalculateHrs
How a lesson of one and a half hours is counted in school hours in the obligation tables. *Reporting only; no effect on generation.*

FindStudentConflicts · IgnoreConflictIfEnoughHours · FindConflictsScheduleReservation · FindConflictsReservationReservation
What the *Conflicts* screen checks: student clashes, lessons that have enough hours, lessons against reservations, reservations against each other. *Checking only; no effect on generation.*

ResetCourseHistoryWeeks
Clears the record of where a course's lessons stood in earlier weeks, which *Align hours for the course part* uses. *Set it for one start, then remove it.*

GenerateLectureExcerSequence · StopCheckingPrevSchedules
Read by the program but no longer used by anything. *Harmless; may be deleted.*

The rest of the `[Advanced]` section – colours, row heights, report options, import formats, web publishing – has nothing to do with generation and is described in the administrator guide.