Stelvio

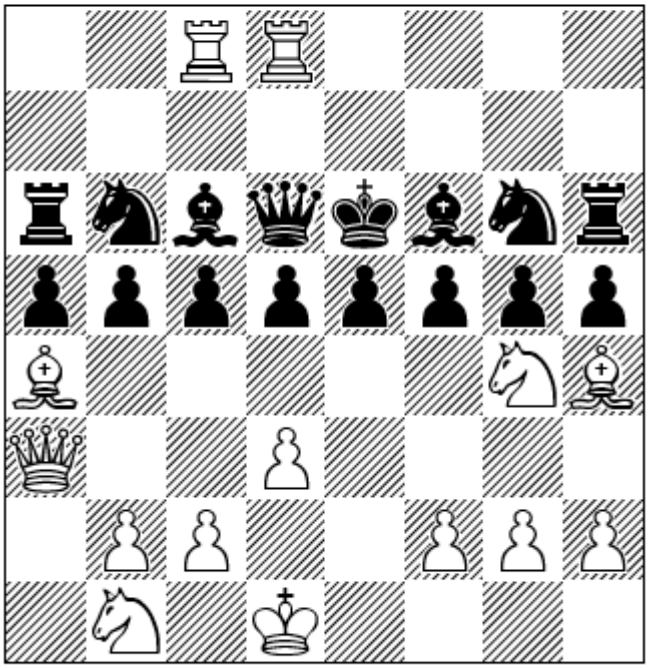
Introduction

Stelvio is a program devoted to solving orthodox SPGs (Shortest Proofgames). To make things bit clearer, its easiest to consider an example. Let us look at a masterpiece from Finland.

Unto Heinonen

Probleemblad 1998

1. Prize



SPG in 23.0 moves

This can be checked by Stelvio, and after around 1.5min, Stelvio comes to the conclusion, that there is exactly one way to reach this position in 23.0 moves:

The successful strategy and the unique solution can subsequently be displayed:

The name

Stelvio, with its 2757m of altitude above sea level, is one of the highest and one of the most beautiful mountain passes of the Alps. As a bike rider, I've been up there many times and have good memories of it. I needed a name, and I liked the way Stelvio sounds. This name breaks the

"tradition" of naming the program after a famous mathematician like Jacobi or Euclide. Let's say the idea of naming it Scholze was not very appealing…

Compatibility

Stelvio can run on Windows and also on Mac (Thierry LeGleuher has tested it on Mac - I personally do not own one). For the moment, the UI is broken on Linux (at least for my installation), something I should fix in a later release. It is possible though to run Stelvio on Linux with the provided stelvio10.sh file. The UI looks distorted, but at least the solving process is not affected, and the problems_out.txt file with the solving result is correctly generated.

Requirements

Solving SPGs is often memory-intensive, especially for SPGs with a lot of free moves. For that reason, it can be very beneficial to run Stelvio with most of the RAM that you have on your machine. You can find the amount of RAM you have in the System properties, common values for a notebook would be 8g-32g. I'd say 4g is bare minimum (can Tetris run with less?).

Btw: Running Stelvio with way more RAM than is needed for a particular SPG will likely increase the solving time for this SPG, as there are costs involved accessing large amounts of memory on hardware level. You can witness this e.g. in the required ramp-up time: When given a lot of memory, Stelvio needs much longer to get going. This is due to the fact that a large cache needs to be initialized. But for SPGs that take a long time to solve, and added 20 seconds or so up front do not matter. When giving Stelvio almost all or all of your RAM, it is advised to close other applications so that the memory is in fact free to use.

Installation

As Stelvio is written in Java, there needs to be a corresponding Java Runtime Environment installed for it to run. You can get your OS-specific version from here: https://www.oracle.com/ch-de/java/technologies/downloads/. I have tested Stelvio with Java 11 and Java 17. E.g. for Windows, you can download Java 17 with this link: https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi. You then need to run the downloaded file and install it to a directory of your choice, recommended is "c:\java\java17" (this path should not contain spaces). Now download stelvio<version>.zip and unzip it to a directory of your choice, e.g. "c:\spg\stelvio". The zip file contains the following:

- bin folder containing some *.jar files (the code)
- doc folder containing PDFs (documentation)
- stelvio<version>.bat (the file to run)
- stelvio<version>.sh (the file to run on Linux)
- stelvioUI.ini (the parameter file)
- problems.txt (example input)

Adjust stelvio<version>.bat

You should open stelvio<version>.bat in a text editor and adjust the maximum memory settings. Per default, it says "-Xmx8g" in the file, which means Stelvio is allowed to use 8g of RAM. If you have more memory available, then give most of it to Stelvio, e.g. "-Xmx16g" for 16g of RAM. Also make sure that the path that points to the Java Runtime Environment is correct. The default is "c:\java\java17\", but you need to adjust this in case you opted to install Java at some other location.

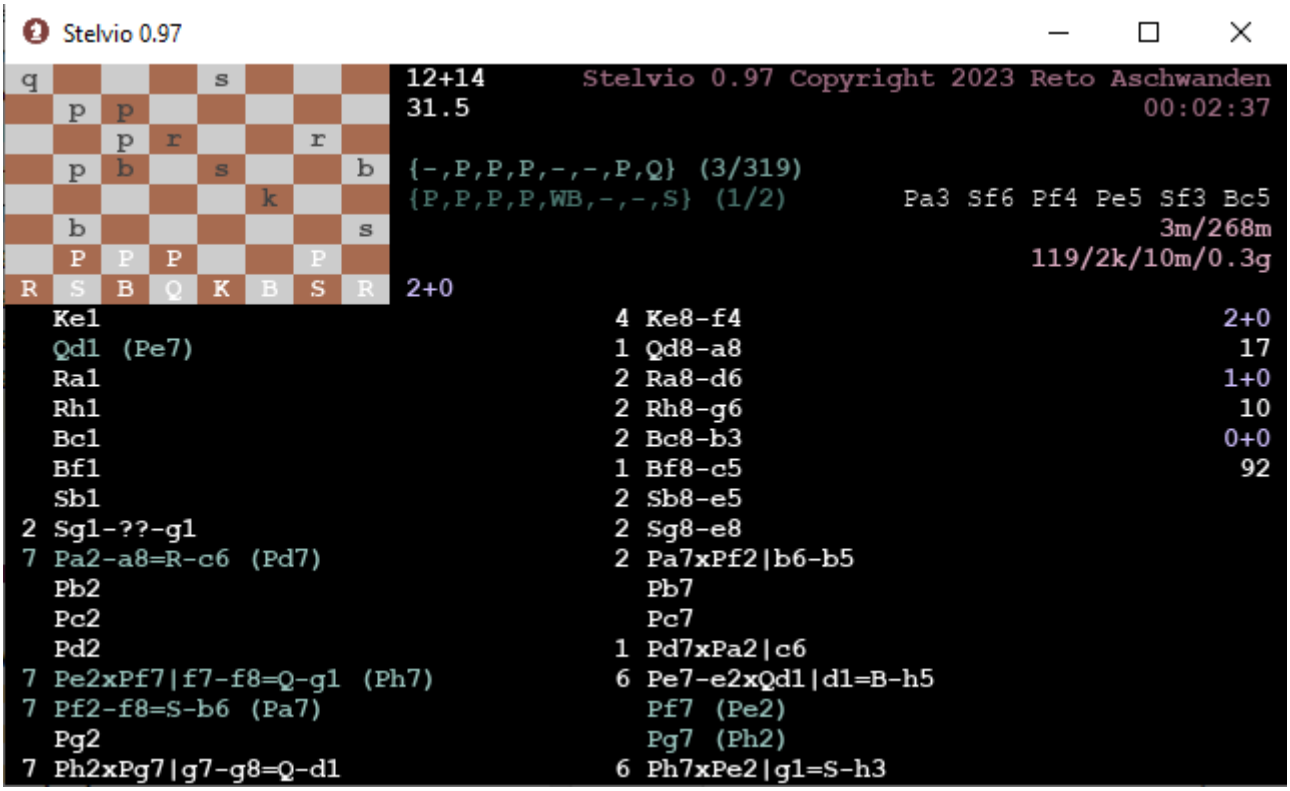
To test your installation, just double-click stelvio<version>.bat, which should run Stelvio with the provided problems.txt file.

Troubleshooting

If Stelvio does not start when double-clicking on the stelvio<version>.bat file, then most likely the Java Runtime Environment cannot be found or its version is too old. Check if the path contained in stelvio<version>.bat pointing to java.exe is correct and adjust if needed.

UI

The simple terminal-based UI gives some impression where Stelvio stands in the solving process. Below is what you get in non-parallel solving mode.



What it all means:

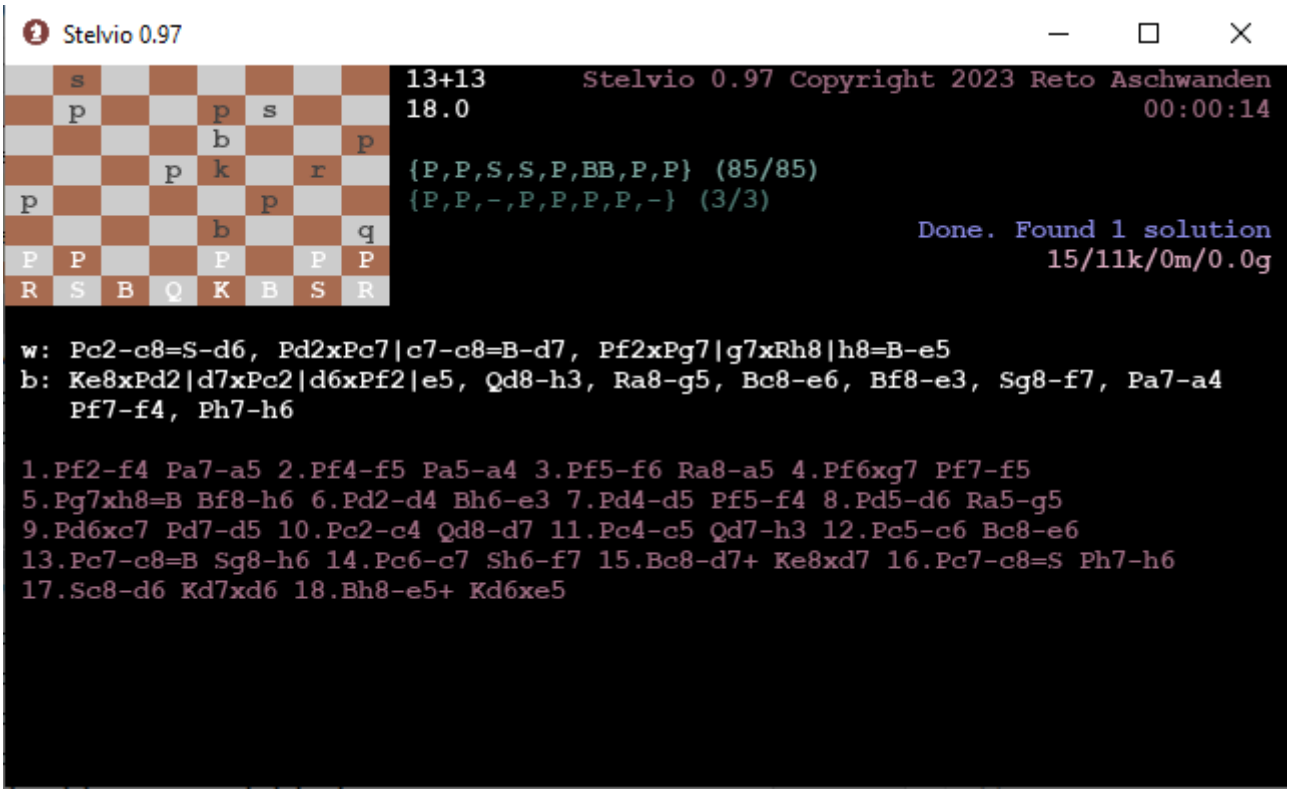
- Next to the board:
  - Number of pieces and number of moves are obvious.
  - The green lines: How the pawns are assigned to partitions currently in the strategy seeking process. Partitions are the types of pieces on the board that are mutually exclusive, we have partition values (K, Q, R, WB, BB, S, P). WB stands for white square bishop and BB for black square bishop. Top line is white assignment {-,P,P,P,-,-,P,Q}, lower line is black assignment {P,P,P,P,WB,-,-,S}. This means for white, that a/e/f pawns are captured, b/c/d/g pawns remain pawns and that the h-pawn is a queen in the diagram position. For black, a/b/c/d pawns remain pawns, the e-pawn is a white square bishop visible in the diagram, f/g pawns are captured and h-pawn promotes to knight and is visible in the diagram. At the outset of strategy seeking, all possible assignments of pawns to partitions are calculated, in the example we have 319 for white and 2 for black. During strategy seeking, these assignments are all gone through in a big loop, and we are currently at assignment 3 for white and 1 for black. This information gives you a hint how far strategy seeking and therefore solving already is.
  - 2+0: Number of free moves for the current strategy by color, so 2 for white and 0 for black.
- On the right:
  - Solving time so far (hh:mm:ss).
  - Current move path (not to be taken too seriously, this can be stale/inconsistent data for technical reasons. If garbage is displayed, this has no effect on the solving itself, only the display is invalid).
  - Cache/strategy metrics:
    - 3m/268m: Currently, the playing cache is basically empty (3 million). Cache can hold 268

million positions. This cache is bigger if you have more RAM available, which can be very useful depending on the SPG in question.

- 119/2k/10m/0.3g:
  - 119: The strategy number of the current strategy being played.
  - 2k: The number of strategy found so far in thousands. All but 119 were subsequently filtered by strategy analysis.
  - 10m: Number of moves played for the current strategy in millions.
  - 0.3g: Total number of moves played for all strategies so far, in billions.

- Below the board:
  - The current strategy, i.e. what each piece does, white on the left and black on the right, with associated move count per piece.
  - The column on the far right: Histogram of already played strategies (top 8 entries, as there is limited space). So in the example, 17 strategies with 2+0 free moves have already been played, next to strategies with 1+0 and 0+0 free moves.

When a solution is found or at the end of solving, pressing "Space" displays the solution and the successful strategy:



When there are cooks, Stelvio counts the number of strategies that contribute to cooks, in the example 63. In case strategies which can be uniquely played are found, these are counted as valid solutions and displayed as well.

A brief note on what Stelvio counts as a cook/solution: In case a strategy can be played in multiple ways, then this counts as a cook strategy for Stelvio. This is the correct approach to take for 99.9% of the SPGs out there. In case a SPG requires an even number of half-moves, e.g. 22.0, then any solution/cook needs to have an even number of half-moves. Correspondingly for SPGs requiring an

odd number of half-moves. This seems a good approach to take, since who is at play is part of the position to reach. So in case in the example one could reach the diagram position in 21.5 moves, then this does not count for Stelvio. Any shorter move-path with same parity of half-moves does count though, so in the example, Stelvio would count solutions/cooks in {21.0, 20.0, 19.0, … 1.0} moves.

Histogram mode

In order to get a first impression if an SPG can be solved in a reasonable time, it can be useful to simply search for all the strategies without actually playing them. This is what the histogram mode is for. In this mode, strategies are added up and grouped by white/black free moves.

So in the example, there are 352 strategies found with 2+0 free moves, 612 with 1+0 and 1352 with 0+0 free moves. Strategy seeking only took 23 seconds, so computing the strategy histogram is much faster than solving the SPG (in this particular case). The SPG looks solvable in reasonable time given this histogram information. It is also possible to attain a partial histogram: You can tell Stelvio to only start adding up strategies after strategy number X. This can be useful if you want to know what is left in terms of strategies, if Stelvio already solved up to strategy X.

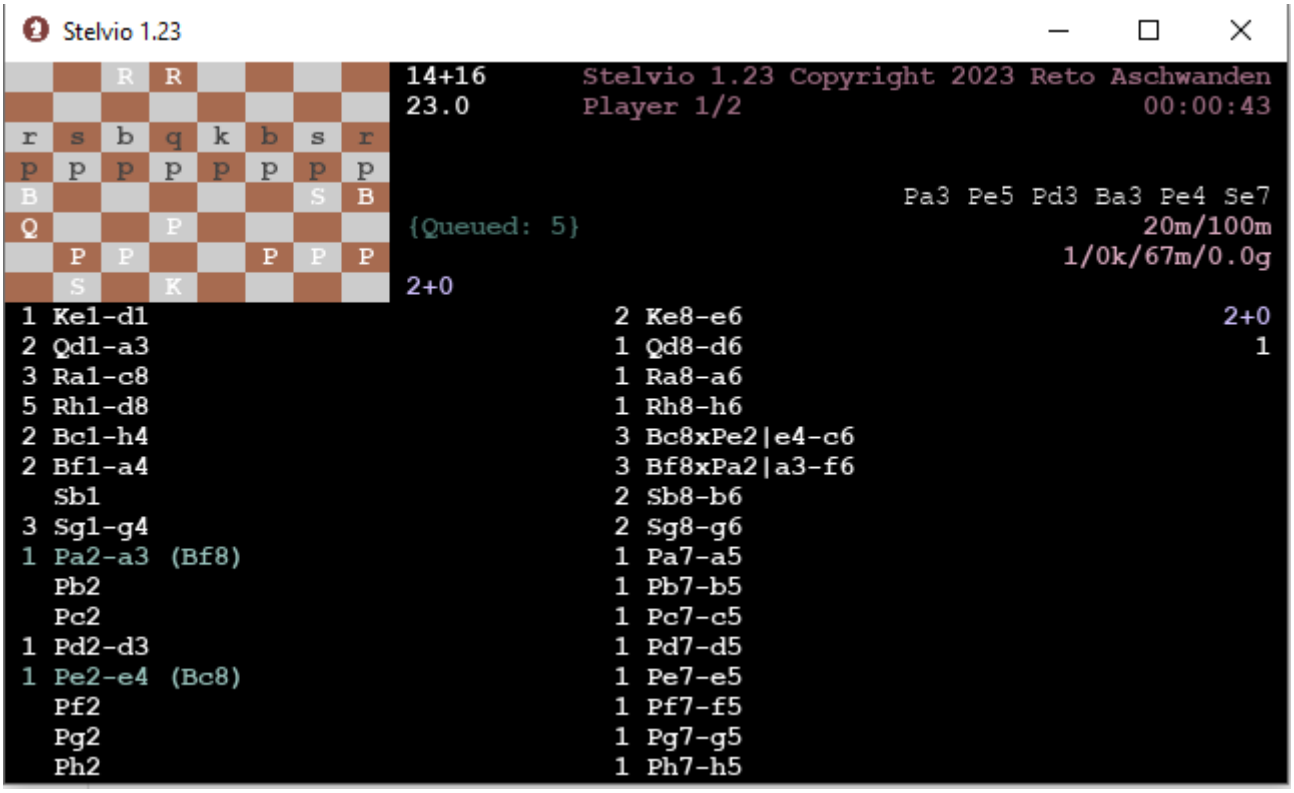
Parallel mode

Introduction to parallel solving

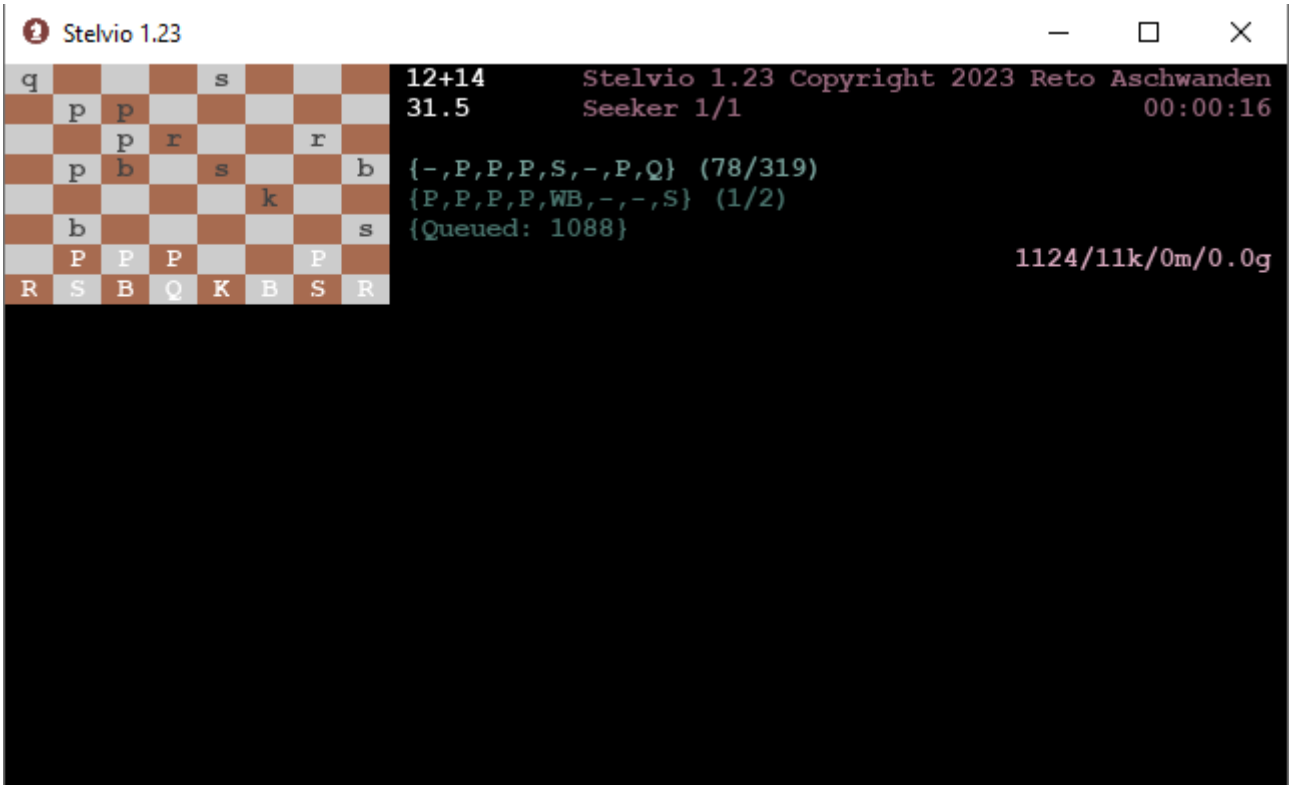
Both histogram and normal solving can be executed in parallel mode. Modern computers come with several CPUs, so calculating in parallel can be beneficial to performance. Parallel mode is actually the default, as it is usually faster.

The basic architecture is quite simple: There are x strategy seekers and y strategy players. The numbers x and y can be parametrized to suit the SPG at hand and your hardware. When running in parallel, the strategy seekers search for strategies and put the found strategies into a queue. The strategy players on the other hand consume strategies from this queue and try to play them. The granularity for the strategy seekers is a single pawn partition matching combination. This can be a bit coarse, and I might adjust this in the future. In the above example, there are 319 * 2 = 638 such combinations, which the seekers consume one by one until none are left.

In parallel mode, you can always see the strategy queue, in the example below we have 5 queued strategies that are waiting to be played. Each seeker/player has a separate UI page. Left-arrow/right-arrow switches between pages. Under the copyright notice, you can see which page is currently displayed.



The page for a strategy seeker shows what pawn partition matching this seeker is currently searching strategies for.



Parallel solving examples

Same SPG but different concurrency metrics

I give a few examples of the effect of parallelizing, when it is beneficial and when it is not. Let's take the 2023 Andernach TT winner by Michel. In non-parallel mode (the only mode available in Stelvio

1.2), this SPG is solved on my notebook in some 35 minutes:

Both strategy seeking and strategy playing take significant time for this SPG, so splitting up is beneficial. With 1 seeker and 1 player, the solving time drops to 27 minutes in Stelvio 1.3:

As no strategy requires a lot of memory, we can also split up strategy playing without harm. With 1 seeker and 2 players, the solving time drops to 20 minutes:

In the above configuration, the queue is sometimes empty. That means that the players have nothing to do at times, so an additional seeker will likely reduce the solving time. With 2 seekers and 2 players, the solving time drops to 18 minutes:

Just for fun, I tried with 4 seekers and 4 players, which still shaves a minute of solving time:

As my notebook now runs out of hardware, increasing the numbers further will have a negative effect, since in the end, all seekers/players compete for the same underlying hardware.

Different SPGs and the usefulness of parallelism for them

Histogram mode in parallel

As a rule of thumb, histogram mode can always be run in parallel. The worst thing that can happen is that it will not be faster than non-parallel mode. The seekers consume all pawn partition matching combinations one by one. In case there are only few of these, then parallelism is not very useful. E.g. in the following length record case (P1407171), there is only exactly one such pawn partition matching combination, so parallelism is useless, as only one seeker has work to do:

Much better is the following case (P1000601) with 9 * 3770 = 33930 pawn partition matching combinations. What usually happens though is that most of the strategy seeking time is required for one or very few such combinations. That has the negative effect that parallel seeking time does not proportionally decrease with the number of strategy seekers. In the worst case, 33929 pawn partition matching combinations could require basically no time and 1 such combination could require 99% of the time. Then parallelizing would not be very useful.

Solving in parallel

- Parallel solving can be detrimental under some circumstances, in case there are multiple players. The reason is that the players each need their own position cache, and this cache requires a lot of memory. So in case you have 2 players instead of 1, then each position cache is only half the size. Now in case while playing a strategy, the position cache is close to full, then solving performance for this strategy decreases dramatically. Therefore, it can be slower to play strategies in parallel than to play them serially. A case where this happens is the aforementioned length record problem (P1407171), which by the way still cannot be solved.
- In the Andernach TT winner above, parallel solving is pretty useful, cutting the solving time in half.
- Parallel solving is the most useful in case strategy seeking takes up most of the time and there are a lot of pawn partition combinations to comb through. As there is almost no limit for number of seekers (given enough available CPUs), you can go all-in in that respect. An order of magnitude can be shaved of the solving time in that way for P1386153.

Current limitations, possible future developments

Three things come to mind:

- Pawn partition matching as a unit of granularity is too coarse for strategy seekers. Finer grained work distribution would be beneficial to even out the workload.
- Parallel playing of a single strategy.
- Parallelize not only on one machine, but across a cluster of machines.

User interaction

The playing of the current strategy can be stopped by typing 's'. Stelvio will move on to the next strategy thereafter. Pressing Ctrl-C cancels the solving process. In parallel mode, you have different UI pages per seeker/player. You can switch between these pages using left-arrow/right-arrow.

Input / Output

A simple text file (by default problems.txt) serves as input. It needs to be in the same directory as stelvio<version>.jar. In problems.txt, the SPG needs to be given in FEN notation on the first line, and the number of half moves on the second line, something like:

1nbq4/ppk1p3/Rp5p/3npr2/R3P3/2br1B1P/PP2P2P/1NBQNK2
65

Pieces are denoted by:

- K/k = King
- Q/q = Queen
- R/r = rook
- B/b = bishop

- N/n/S/s = knight
- P/p = Pawn

The result is written into an output file (by default named problems_out.txt). There is a stelvioUI.ini file for parameters that can be adjusted by the user. If no such file is present, then default values are used. See also StelvioParameters.pdf.

Read/write strategies to file

Stelvio can be advised to save all found strategies in structured format to disk. In a second step, instead of searching for strategies, Stelvio can thereafter read these strategies from disk and try to play them. This is especially useful in case one uses histogram mode at first. Calculating the histogram sometimes requires a lot of time, only to find very few strategies. In order to subsequently play these strategies, they can now be read from disk in no time, instead of recalculating all them all over again.