Beginners' Guide to TSH

for UK organisers – by Stewart Holden

Version 1.3 (October 2007)

Main contacts for assistance:

Stewart Holden (UK): stewart@tilefish.co.uk, home 0115 841 5179 or mobile 07971 634098
Anand Buddhdev (Netherlands): arb@anand.org, home 00 31 20 345 2890 or mobile 00 31 623 925 983
John Chew (Canada): jjchew@math.toronto.edu, AIM username: poslmobile, mobile 001 416 876 7675 (NB: Toronto is 5 hours behind UK time).

Introduction

TSH is the tournament software developed by John Chew. It has been constantly modified and improved since 1999 based on user feedback and an ever-growing list of demands for various new features and it is widely recognised as the most powerful and versatile tournament pairing system for Scrabble events. To summarise its main advantages:

- nearly all details can be entered in the configuration file in advance of the tournament
- the default pairings system, Chew Pairings, will calculate the fairest and most competitive pairings for any division automatically. Options for round robin, king-of-the-hill, Swiss and manually-configured pairings are also included.
- the basic commands for showing pairings, standings and entering results are extremely simple
- every division can be run in the same copy of the program (even if using different pairings systems)
- mistakes are easy to correct
- standings and pairings printouts are very clear
- results are output in the best format for the ABSP Ratings Officer

This guide is divided into three sections: Installation, Configuration and Use. The Appendix gives some example configurations.

Installation

TSH will run under Windows, Linux, MacOS and other operating systems. The instructions here are for Windows users. TSH is written in the Perl programming language, so you'll need to install Perl on your computer. You can download an installer from here:

http://www.absp.org.uk/perl.msi

This file is 16mb so it may take a while to download. After downloading the installer, just run it, and accept the defaults to install a regular setup of Perl on your computer.

TSH can be downloaded from **http://tsh.poslfit.com**

It is distributed as a ZIP file, so you just have to unzip it to a convenient location on your hard disk. One suggestion is to unzip it to the root of the C: drive, so it is easy to locate and work with, or to place the folder on your desktop so it is easy to find.

IMPORTANT: TSH is constantly being updated and improved. You should update your copy of the program every time you intend to use it. To do this you simply need to be connected to the Internet. Run the program as usual and type **'update'**. The latest version will download automatically from John Chew's website and install itself into your existing copy.

Configuration

The main thing to know about running an event with TSH is that the vast majority of information is set up before the tournament in a plain text file named **config.tsh**. This plain text file (open with Notepad or WordPad if using Windows) contains every piece of information about your event except for the list of player names. Thus the file contains:

- The number of divisions
- The number of rounds

- The kind of pairing required in every round (Swiss, KOTH, Round Robin). If you do not specify a system then Chew Pairings will be used automatically; more on this below.
- Whether you want to enter the results as game scores or just using spreads
- The table numbers allocated to each division

There is an almost endless list of other features which can also be added to the config file, including but not limited to:

- reserving a table for one player throughout the whole event
- detailing how many prizes are available and thus allowing the program to work out automatically when Gibsonizing should be applied in the later stages (see explanation of Gibson situations later).

Setting up the config.tsh file

Stewart or Anand will be happy to set up any config file for any tournament given a few days notice. There are also a few examples (based on common tournament criteria) in the Appendix of this document. If you are using a **config.tsh** which someone else has already written you can skip straight to the Use section of this document.

Having installed Perl and TSH, you should now be ready to set up a dummy tournament. Locate the TSH folder and go into it. You will see some sample tournament folders already there named sample1, sample2, etc.. For the purpose of our example we will use the fictional Oxford event from 2006. Create a folder named 'oxford06' inside the TSH folder. Go into it and it will be empty. Now open a plain text editor such as Notepad or Wordpad or equivalent.

Firstly list all the divisions like this:

```
division a a.t
division b b.t
division c c.t
```

This tells the program that it will find details of Division A's players in a file named **a.t**, and so forth. We will create these files later on.

If you are giving ratings prizes then add the following lines:

```
classes a 4
classes b 4
classes c 4
```

This is to specify four classes (ABCD) of player, hence there would be three ratings prizes in each division. Adjust as necessary.

Now add the following lines:

```
config track_firsts = 1
config assign_firsts = 1
```

These are fairly self-explanatory, they tell TSH to keep a record of starts and replies and to dictate who should be starting each game.

```
config realm = 'absp'
```

This is a multi-purpose instruction which covers a number of ABSP-specific requirements. Including this line means that the default spread for byes will be 75 (as opposed to 50 in North America), that table numbers will be calculated automatically in sequential order, the names will be displayed in the format Name Surname, the ratings system used is the ABSP one and that you wish to enter spreads rather than actual game scores. If you do want to enter actual game scores then you can include the above `realm` line anyway but add the line `config entry = 'scores'` to specify this.

```
config director_name = 'Stewart Holden'
config event_date = '27-28 February 2006'
config event_name = 'Oxford 2006'
config html_top = '<p align=center>Oxford – 27-28 February 2006</p>'
```

The above lines are entirely optional and self-explanatory. The final line will make the words **Oxford – 27-28 February 2006** appear at the top of the standings and pairings printouts (providing that you use the .html rather than .doc files).

```
config max_rounds = 11
```

Set max_rounds to let tsh know when the tourney ends. This helps TSH detect Gibson situations (see explanation later), and also detect invalid round numbers. The value here isn't set in stone and extra rounds can be added later.

```
config html_in_event_directory = 1
```

This means the output files will appear in the oxford06 folder.

```
config reserved{'A'}[26] = 21
```

This will put Divison A’s player #26 on Table 21 for every round. This is useful for those with mobility issues.

At this point you should have a plain text file on your screen which reads something like this:

```
division a a.t
division b b.t
division c c.t
config track_firsts = 1
config assign_firsts = 1
config realm = ‘absp’
config director_name = 'Stewart Holden'
config event_date = '28-29 February 2006'
config event_name = 'Oxford 2006'
config html_top = '<p align=center>Oxford – 27-28 February 2006</p>'
config max_rounds = 11
config html_in_event_directory = 1
```

Save this file as ‘config.tsh’ in the oxford06 directory.

The final thing to include in this file is the pairing instructions. If you wish to use Chew Pairings (see below) then you do not need to write anything at all, since the program will use this system by default in the absence of any manually entered pairing instructions. Chew Pairings are perfect for most tournaments hence what follows is mainly for information purposes only.

Standard Swiss Pairing

If you wish to use Swiss pairing but not Chew Pairings, you will need to enter a separate line in the config file for each round. A standard set of Swiss Pairings for Div A of a 6-game tournament will read something like this:

```
autopair a 0 1 ns 0 0 a
autopair a 1 2 ns 0 1 a
autopair a 2 3 ns 0 2 a
autopair a 3 4 ns 0 3 a
autopair a 4 5 ns 0 4 a
autopair a 5 6 ns 0 5 a
```

The word **autopair** is followed by ‘a’ representing the division. Using the top line as an example, the 0 1 which come next indicate that you wish to use the standings from round 0 to create pairings for round 1. The ‘ns’ which follows stands for ‘new swiss’. The number which follows indicate how many repeat pairings are allowed (in this case 0 in every round).

The last two characters (0 a) are the actual instruction to TSH to create the pairings for that division based on that round (all characters preceding this have merely been clarifying *how* you want this instruction to be carried out).

Round robin (RR) groups

RR groups are so simple that it is not really worth specifying pairing instructions in the config file (although see next paragraph should you want to do this). Simply open up TSH on the day of the event and type **rr a** to pair Div A with RR pairings for the whole event. If the number of players and rounds do not create one or more perfect RR groups then ‘necklace’ or ‘oval’ pairings will be produced automatically. If you want a ‘twin round robin’ where every player plays an opponent twice in succession before moving on to the next person, use the command **rr 2 a**.

Remember that you can override the automatic pairings at any time so if, for example, you wanted to have RR pairings but then KOTH in the final round you can still use the **rr a** command but then before the final round instruct the program to produce KOTH pairings instead (the command is simply 'koth').

Chew Pairings will automatically begin with one or more round robins if there are enough rounds in the schedule to do so. Note that if you want to save your computer operator from having to type 'rr a' etc. on the day of the event you can include the line `autopair a 0 1 rr a` in the config file which will do the job for you.

In order to produce printouts listing every player's pairings for the whole event, use the command **sdsc a** (where **a** is the division). This command is an abbreviation of **S**how**D**ivision**S**core**C**ards. The program will automatically create .doc and .html files in the oxford06 folder which can then be printed and handed to the players. If using the .html files please note that they will print correctly with each player's scorecard beginning on a new page.

Chew Pairings

Chew Pairings are similar to Swiss pairings but with additional modifications implemented automatically. This can include a very limited number of repeat pairings if (and only if) this allows more players to remaining in contention for prize-winning positions than using regular Swiss pairings. The central principle is that no player should be excluded earlier than necessary from potentially finishing in a prize; this can sometimes happen accidentally if straight, unmodified Swiss pairings are used. Chew pairings are suitable for small one-day events as well as major operations such as the BMSC and UK Open. If you do not enter any 'autopair' or other pairing instructions when writing the **config.tsh** file then Chew Pairings will be generated automatically.

If using Chew Pairings you should specify the prize-winning positions in each division in config.tsh thus:

```
config prize_bands{'A'} = [2,3,4,5]
```

This specifies which final ranks are to be considered equivalent for pairings purposes: the numbers given mark the end of each band (range) of equivalent prizes (ranks). In the example, the top prize band goes from 1st place to 2nd place (presumably because the top two finalists qualify for a playoff); each of 3rd, 4th and 5th place is its own prize band (presumably awarding players cash prizes); and everything from 6th place on is the last prize band (presumably winning nothing).

The following more technically-worded explanation of how Chew Pairings create pairings is adapted from the TSH documentation:

Chew pairings draw on Swiss pairings and the two-victor pairing system developed for the 2003 Canadian NSC and refined at various major North American events since then. Tournament simulations determine which players are still in contention; the minimum number of repeats required to pair those players is computed; the contenders are split into leaders and nonleaders so as to minimize the number of leaders while not increasing the required number of repeats. Beginning at the top, each leader is paired with the lowest-ranked other leader who can catch up to him/her; the nonleaders are paired Swiss.

Chew (or similar) pairings should be used in tournaments where after a number of preliminary rounds two or more top players are selected to compete in final rounds. They are also sufficiently flexible that they may be used in any sort of tournament, and are therefore used as tsh's default pairing algorithm.

Note that Chew Pairings will occasionally give one or two repeat pairings for the top-ranked players in the final round of a six-game event, because there is a strong preference for pairing those who could potentially win the tournament against each other rather than against players who cannot. If you want to avoid this use the example autopair lines given in the Swiss Pairings section above.

The player files

As well as writing **config.tsh** you will need to create a separate plain text file for each division, containing the names of the players. These should have the names and ratings written in the following format: Surname, Firstname <space> rating. For example:

```
Smitheram, Brett 198
Allan, Paul 194
Martin, Ed 192
Kirk, Terry 185
```

Save the player name files as **a.t**, **b.t**, etc. as appropriate.

Tip: In the days before your event you should create one single player list (for everyone at the tournament) in one single text file, with names written in the above format in ratings order. Then if rejigging of divisions becomes necessary on the day (as it inevitably does), you can simply cut and paste the player names into the appropriate a.t, b.t, etc. files and save them accordingly.

If you are setting up a team event, use the following format:

```
Hawkins, Chris 174 ; ; team peterborough
Perry, Steve 174 ; ; team
Robinson, Jared 172 ; ; team nomads
```

The area between the two semicolons on each line will be filled in by TSH later with game scores. Note that Steve Perry does not belong to team in this example (this example fits the current UK NSC Regional events where some players are also competing in the National Club Knockout Qualifier, whilst others are playing as ‘unattached’ individuals).

What else?

The many other options available can be found by reading the full TSH documentation, which is included with every copy of the program and found in the ‘doc’ folder. In the days or weeks leading up to your event you should send your **config.tsh** file to one of the contacts listed at the start of this guide, who will be happy to read through it and check that everything makes sense. They will all be happy to advise on any improvements and raise any queries. This will enable you to go into the tournament room fully prepared and confident to use the program on the day.

Gibsonization

This term describes a situation which can occur late in a tournament, where one player is already guaranteed to win the top prize and cannot be overtaken. In events where the top two finishers qualify for a head-to-head final it can also describe a player who is guaranteed a place in this final no matter what happens in the remaining rounds. In this situation the player in this safety zone is not playing with the same motivation and under the same pressure as those in the top position who are still competing for places. Gibsonization (named after US player David Gibson to whom this first applied in the 1995 AllStars tournament) pairs this player against the highest-ranked player who cannot finish in a prize position, effectively removing the Gibsonized player from influencing the results of the players near him/her. TSH can be told to automatically detect when a player should be Gisonized; please refer to the full TSH documentation if you want to know the exact criteria applied by the program when working this out.

Use

Once the config.tsh and division files have been created, you are now in a position to run the program for the first time. To do this, run the **tsh.pl** file found in the main TSH directory.

The program will automatically search everywhere within the TSH folder and start up using the most recently saved version of config.tsh it can find. Thus the config.tsh files in the sample folders will be ignored since oxford06/config.tsh will be more recent.

Basic commands for using the program:

sp	**Show Pairings:** This needs to be qualified with a round number and a division. Typing **sp 1 a** will give you the pairings for the first round in Div A. Typing **sp 3 c** will show you the pairings for the third round in Div C. If you are using anything other than round robin format, the latter example will not work until all results from the second round have been entered.
st	**Standings:** This needs to be qualified with a divison. **st a** will show you the standings in Div A, etc.

Every time you use either of the above commands, the output is displayed on the screen **and** automatically written to .doc and .html files in the oxford06 directory. There is no need to give the program separate commands for writing the pairings/standings to a file and then converting them to printable format, this is all done automatically in the oxford06 folder every time you use the **sp** or **st** commands. The files are automatically overwritten every time you use one of these commands, meaning that you can keep all relevant windows open and simply refresh the page to see the latest pairings/standings once they have been generated onscreen.

a	**Add Results:** This needs to be qualified by a round number and division. Once the results start to come in, type **a 1 a** if you wish to begin entering results for round 1 in Div A. You will be told how many players are still playing in the division. The format required if you are entering spreads is: <player 1> <player 2> <spread>. For example if player number 6 has beaten player number 2 by 134, you would enter **6 2 134**. If you are entering game scores the format required is <player 1><score><player 2><score>, for example **6 501 2 367**.
missing	This shows which games have not yet had results entered. It needs to be qualified by a round, e.g. **missing 3**.

When a result is added TSH will display: **Shin, Austin (4.0 +477) – Violett, Bob (2.0 -66).** The data in brackets here represents the player's current wins and spread in the event. The first name listed will be the winner in the result you have just entered. You now have the option of entering another result for the same division or pressing RETURN to go back to the command line. If you now want to enter results for Div C, press RETURN to stop entering results for Div A, type **a 1 c** and continue from there. To enter a draw the spread should be entered as 0. To enter a bye for player 6, enter **6 0**.

If one player has entered the wrong player number and you try to put this in as a result, the program will advise you that that the two players you have entered did not play each other in that round. If you do not have the correct player number on hand you can enter the player's name instead in the format **surname,forename**. You only need to enter enough characters of the surname and forename to differentiate them from any other player in the division. For example if Gareth Williams was the player in question and he had beaten player 11 by 202pts, entering the game result as **will,gar 11 202** would work just as effectively as using his player number.

Additional commands:

The complete list of commands can be found in the documentation but here are a few others you may find useful or interesting:

huh	If you aren't sure what an error message means, look to see if it has a code in [square brackets] at its end. If so, type **huh** followed by that code to ask for an explanation in plain English. If you omit the code, TSH will explain the most recent error message.
es	**Editscore:** If you make a mistake when entering results you can correct it by changing the scorecard of either of the players involved. The usage is **es division player round**, so to correct a score for the Div C's player 3 in the fifth round, type **es c 3 5**. Simply typing **es** will produce a prompt for you to enter these variables in the correct order. For help on how to edit scores, type **help es**. NB: If you have generated pairings for the next round and subsequently been notified that a result has been input wrongly then after using the **es** command to change it, you will then need to generate new pairings. To do this firstly use the **unpairround** command (**upr**), for example **upr 6 a** to delete the old pairings for Div A round 6. Then use **sp 6 a** as normal to generate new pairings based on the revised results.

koth	Does exactly what the command suggests. Enter this before generating pairings for a round if you wish to force it to be a King of the Hill round. This will override any pre-determined pairing commands in the **config.tsh** file and/or override the default Chew Pairings. Note that if you are running a team event and have used the ‘exagony’ command to forbid team members from playing each other, using **koth** will still respect this necessity. It will also respect any players who have been Gibsonized and pair 2-3, 4-5, 6-7 etc. if necessary.
rand	This will enter random results for a whole division. Useful when experimenting with the program in order to avoid entering data manually. Needs to be qualified by division, e.g. **rand b**.
dryrun	Genereates **rand** results for a whole tournament.
abspgrid	At the end of event, use this command for every division (e.g. **abspgrid a**) to output .doc files into oxford06 for sending to the ABSP Ratings Officer.
truncaterounds	This will delete all scores and pairings back to whichever round you specify. Entering **truncaterounds 0 a** will remove all data for Division A allowing you to pair the first round again and proceed from there. This is useful if you want to do a ‘dummy run’ of your event but then delete the false data easily.
tuffluck	Will display the players with the smallest combined losing spreads over a specified number of games. Needs to be qualified the number of games to based it upon and the division. For example **tuffluck 4 a** will list the Div A players who have the lost four games by the smallest combined total, starting with the ‘unluckiest’. This information can be used for awarding ‘Tuff Luck’ spot prizes.
luckystiff	The opposite of TuffLuck, this specifies the N smallest victories.
hw, hl, etc.	Various commands like hw (high wins) and hl (high losses) produce the self-explanatory reports after the tournament. Refer to the TSH manual for the complete list of statistical outputs that can be produced.
stats	Generates some more statistics about your event, such as % of games the starting player won, etc.
help	Every single command in TSH has a help file associated. For example if you cannot remember how to edit scores correctly, type **help editscore** or **help es**.
doc	Typing “doc” brings up the full TSH manual in a browser window.

If you would like data to be put on the Centre Star website for others to examine after the event please email the config.tsh file and division files (a.t, b.t, etc.) to Stewart Holden (stewart@tilefish.co.uk) who will be happy to upload them and notify the uk-scrabble and world-scrabble mailing lists.

This guide is only an introduction to TSH. For a list of the many, varied features it offers and full explanation of all the commands you are strongly advised to read the accompanying documentation. John Chew welcomes questions and feedback on any aspect of the program and with sufficient notice can implement specific requirements into the latest version in time for your event. Stewart Holden and Anand Buddhdev are experienced users who will be happy to answer questions. There is also a TSH users group found at **http://www.groups.yahoo.com/group/tsh-users/** which is read by many organisers around the world.

The following Appendix gives some example configurations for past UK events.

Appendix

Examples of config.tsh files

1) **A standard one-day event.** 4 divisions, 6 games. Chew Pairings used so there are no ‘autopair’ instructions.

```
division a a.t
division b b.t
division c c.t
division d d.t

config track_firsts = 1
config assign_firsts = 1

config realm = ‘absp’

config director_name = 'Stewart Holden'
config event_date = '21 April 2007'
config event_name = 'Romford (Collier Row)'
config html_top = '<p align=center><b>Romford (Collier Row) - 21st April 2007</b></p>'

config max_rounds = 6
```

2) **Durham 2007.** One large Div A paired for each round based on the standings after the previous round, with KOTH in the final round. Div B-H were round robin groups paired by simply typing ‘rr b’, ‘rr c’ etc. on the day of the tournament and hence there are no autopair commands for these divisions. 11 rounds.

```
division a a.t
division b b.t
division c c.t
division d d.t
division e e.t
division f f.t
division g g.t
division h h.t

config track_firsts = 1
config assign_firsts = 1

config realm = 'absp'

config director_name = 'Stewart Holden'
config event_date = '14-15 April 2007'
config event_name = 'Durham 2007'

config max_rounds = 11

autopair a 0 1 ns 0 0 a
autopair a 1 2 ns 0 1 a
autopair a 2 3 ns 0 2 a
autopair a 3 4 ns 0 3 a
autopair a 4 5 ns 0 4 a
autopair a 5 6 ns 0 5 a
autopair a 6 7 ns 0 6 a
autopair a 7 8 ns 0 7 a
autopair a 8 9 ns 0 8 a
autopair a 9 10 ns 0 9 a
autopair a 10 11 koth 1 10 a
```

NB: In the final round’s autopair command the digit after ‘koth’ changes from 0 to 1, since this is the indicator of how many repeats are allowed.

3) **An NSC Regional event.** One division, six rounds. Top 10 places qualify for next round of NSC. Team members cannot play each other (this is known as ‘exagony’).

```
division a a.t

config track_firsts = 1
config assign_firsts = 1

config realm = 'absp'

config exagony = 1

config no_text_files = 1
config prize_bands{'A'} = [10]

config reserved{'A'}[26] = 21

config show_teams = 1

config director_name = 'Chris Hawkins'
config event_date = 'Saturday 9th June 2007'
config event_name = 'NSC/NSCT Midlands 2007'
config html_top = '<p align=center><b>NSC/NSCT Midlands Qualifier - Saturday 9th June 2007</b></p>'

config max_rounds = 6
```

With accompanying a.t file showing team names, as explained in main document. For an explanation of some of the above commands (e.g. exagony, prize_bands, no_text_files) please refer to the full TSH documentation included with the program.

4) **British Matchplay Scrabble Championship 2007.** Div A is Swiss paired with one repeat allowed after round 14, last round KOTH. Div B is Swiss paired with no repeats, then KOTH in the last round. Div C-F are round robins of 17 players with one player sitting out per round, then the 18th round KOTH. Prize positions are top five in Div A and top three elsewhere. Three ratings prizes in Div A and two ratings prizes in Div B, none in the other divisions. Board 3 in Div F reserved for player #9. Gibsonization allowed. Scores recorded rather than spreads.

```
division a a.t
division b b.t
division c c.t
division d d.t
division e e.t
division f f.t

classes a 4
classes b 2

config reserved{'F'}[9] = 3

config prize_bands{'A'} = [1,2,3,4,5]
config prize_bands{'B'} = [1,2,3]
config prize_bands{'C'} = [1,2,3]
config prize_bands{'D'} = [1,2,3]
config prize_bands{'E'} = [1,2,3]
config prize_bands{'F'} = [1,2,3]

config html_in_event_directory = 1

config realm = ‘absp’
config entry = ‘scores’

config track_firsts = 1
config assign_firsts = 1

config gibson = 1

config director_name = 'Ian Burn'
config event_date = '25-27 August 2007'
config event_name = 'BMSC 2007'
config html_top = '<p align=center><b>British Matchplay Scrabble Championship</b><br>Yarnfield Park, 25th-
27th August 2007</p>'

config max_rounds = 18

autopair a 0 1 ns 0 0 a
autopair a 1 2 ns 0 1 a
autopair a 2 3 ns 0 2 a
autopair a 3 4 ns 0 3 a
autopair a 4 5 ns 0 4 a
autopair a 5 6 ns 0 5 a
autopair a 6 7 ns 0 6 a
autopair a 7 8 ns 0 7 a
autopair a 8 9 ns 0 8 a
autopair a 9 10 ns 0 9 a
autopair a 10 11 ns 0 10 a
autopair a 11 12 ns 0 11 a
autopair a 12 13 ns 0 12 a
autopair a 13 14 ns 1 13 a
autopair a 14 15 ns 1 14 a
autopair a 15 16 ns 1 15 a
autopair a 16 17 ns 1 16 a
autopair a 17 18 koth 2 17 a
autopair b 0 1 ns 0 0 b
autopair b 1 2 ns 0 1 b
autopair b 2 3 ns 0 2 b
autopair b 3 4 ns 0 3 b
autopair b 4 5 ns 0 4 b
autopair b 5 6 ns 0 5 b
autopair b 6 7 ns 0 6 b
autopair b 7 8 ns 0 7 b
autopair b 8 9 ns 0 8 b
autopair b 9 10 ns 0 9 b
autopair b 10 11 ns 0 10 b
autopair b 11 12 ns 0 11 b
autopair b 12 13 ns 0 12 b
autopair b 13 14 ns 0 13 b
autopair b 14 15 ns 0 14 b
autopair b 15 16 ns 0 15 b
autopair b 16 17 ns 0 16 b
autopair b 17 18 koth 1 17 b

autopair c 0 1 rr c
autopair c 17 18 koth 1 17 c

autopair d 0 1 rr d
autopair d 17 18 koth 1 17 d

autopair e 0 1 rr e
autopair e 17 18 koth 1 17 e

autopair f 0 1 rr f
autopair f 17 18 koth 1 17 f
```

You will undoubtedly have more questions on finer points of using the program but hopefully this has provided a good starting point. Please contact one of the following who will be happy to anwer any queries.

Stewart Holden (UK): stewart@tilefish.co.uk, home 0115 841 5179 or mobile 07971 634098
Anand Buddhdev (Netherlands): arb@anand.org, home 00 31 20 345 2890 or mobile 00 31 623 925 983
John Chew (Canada): jjchew@math.toronto.edu, mobile 001 416 876 7675 (NB: Toronto is 5 hours behind UK time).