MAGMA version 1.2.3

Methyl Assignment by Graph MAtching (MAGMA) allows you to take a protein structure and compare it to through-space solid and solution NMR data, to find all ways to map resonances to specific sites within the protein. The latest version and examples are available from http://magma.chem.ox.ac.uk. Please see Pritisanac et al. JACS 2017 (in press) for more details including applications to ligand binding and structure discrimination.

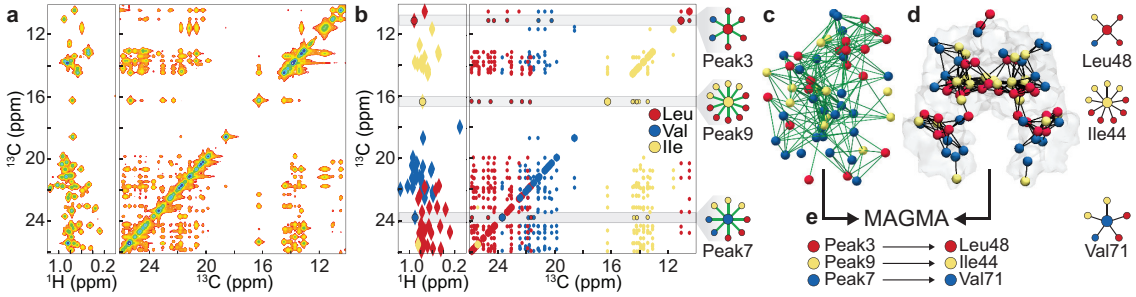

Figure 1: Automated methyl resonance assignment of solution-state NMR spectra by MAGMA. (**a**) Projections of a 3D (H)CCH HMQC NOE solution-state NMR spectrum of the R2 dimer of ATCase, recorded from a deuterated sample with [$^{13}CH_3$]-labelled Ile-, Leu- and Val- residues. Off-diagonal resonances reveal proximal pairs of methyl groups. These connectivities are used as distance restraints by MAGMA. (**b**) This is interpreted by MAGMA to include information about residue type. Analysis of regions associated with a single resonance (shaded) identifies the local network of connections for each methyl resonance. (**c**) The local networks can be combined to give a data graph. (**d**) Similarly, inter-methyl connections from a protein structure can be obtained at a predefined distance threshold and represented as a graph. (**e**) MAGMA compares all ways of mapping between the two graphs, to give assignments that explain the maximum amount of data possible. The workflow is summarised in figure 2.

Join our community!

We are keen to build a community of users to further improve the program, add to the benchmark and otherwise add new functionality. So please get in touch if you have questions, suggestions or bugs. Please email andrew.baldwin@chem.ox.ac.uk. Enjoy using MAGMA!

Philosophy

The philosophy of MAGMA is that even one erroneous confident assignment from real experimental data is a sufficient condition for failure. Your functional and biological conclusions in biomolecular studies will rest on your assignments, so it is desirable for there not to be any errors. The benchmark of data and results included with the package show you what to expect, and we include here recomendations for which data to use and how to run the program. If you catch MAGMA making errors (or successes!) please email and we would be delighted to look into it with you.

Operation

1. A PDB file is imported (provided it has relatively standard formatting). Specified methyl groups are extracted and a structure graph is created to a distance threshold (usually 10Å).

2. A distance restraint list is imported, where pairwise labels indicate that a through-space interaction has been measured in an appropriate NMR experiment. The formatting of the entry includes a numerical index and a letter indicating residue type.

3. Stereoisomeric methyl groups in leucine and valine residues in the PDB data are merged to make a pseudo-atom if desired (this is generally desirable!).

4. An assignment is defined here as all possible mappings between verticies in the data graph onto the structure graph that explain the maximum number of distance restraints. MAGMA obtains this by determining similarity between the two. This can be run by splitting the data into disconnected subgraphs (fast) or with all data networks together (slow) or a mixed approach where the disconnected results feed into a combined calculation. This final step is advantageous if the assignment results from the subgraphs heavily overlap. For example if a confident assignment from assigning one subgraph remains as a possibility when assigning one of the subsequent subgraphs.

5. The vf2 algorithm for subgraph isomorphism is run where possible owing to its speed, if the data graph can be exactly embedded into the structure graph.

6. If the two graphs cannot be directly embedded (eg, if there are erroneous distance restrains in the peaklist, or if the structure is not a perfect match) a modified version of the McGregor Maximum Common Edge Subgraph (MCES) algorithm is executed.

7. The MCES algorithm requires the order in which the search is performed to be carefully optimised. Our procedure has some randomness in it, so for long calculations, we recommend to run multiple instances of the program and compare the results. Allow the calculations to proceed that have high numbers of matching edges, and halt the others that are not doing so well.

8. The final assignment result provides all ways to map from the NMR data graph nodes to the structure graph nodes, that satisfy the maximum number of edges (distance restraints).

9. A pretty report can be generated if pdflatex and pymol are in the system path.

Running MAGMA

MAGMA comprises a python wrapper to parse the relevant information and prepare the graphs, a subgraph isomorphism algorithm from igraph, and a pre-compiled binary that executes the MCES algorithm. This has been compiled for mac and linux. Run:

```
RunMeFirst.sh
```

To have a script look for the required dependencies. Some of these are essential for MAGMA running, including the core binary, and some are needed for making graphical outputs. The former, including some non-standard python libraries are vital, whereas the latter are not neccessary. The results can still be deduced through reading the relevant

output files in the absence of graphical outputs. The program determines the operating system by comparing the result of 'uname -s' from the command line with os.uname()[0], called within python. If these two are equal to each other, and either 'Darwin' or 'Linux', then the autodetect should work perfectly. If in doubt, execute 'mcesCore_Darwin' and 'mcesCore_Linux'. If you see 'USAGE:' followed by some detail, you have the correct binary. You can hard wire this binary into the file 'mces.py' (search for the variable mcesPath and edit that) if the autodetection of operating system does not work.

For a successful operation of MAGMA, we use python 2.7.x with the following packages, each of which are likely available through your favourite repository manager.

1. numpy (version 1.8.1) link
2. scipy (version 0.12.1) link
3. matplotlib (version 1.3.1) link
4. igraph (version 0.7.1) link
5. networkx (version 1.8.1) link
6. munkres (version 1.0.7) link

The relatively non standard modules can all be installed from the command line. If you have setup tools for python installed link, such that 'easy_install' is available on your system, you can get the other modules using pip link (script from Reid Alderson).

```
sudo easy_install pip
pip install munkres
pip install networkx
pip install python-igraph
```

Outputs are made during a MAGMA run using gnuplot. So gnuplot (link) needs to be in the system's path. Missing this binary will not affect the running of MAGMA, but you won't get pretty graphs showing its progress.

The report is generated using pdflatex (link) and rendering of the datagraph and proteins are made using pymol (link). So pdflatex and pymol also need to be in the system's path. Again, this is not essential, but you will have to read the raw files to get the MAGMA result.

If multiple instances of pymol are on your system, make sure the shell is pointing towards the correct one. The following alias has been identified to be useful, for example, to ensure that MacPymol works nicely (thanks to Bryn Fenwick).

```
export PATH=/Applications/MacPyMOL.app/Contents/MacOS:$PATH
ln -s /Applications/MacPyMOL.app/Contents/MacOS/MacPyMOL \
/Applications/MacPyMOL.app/Contents/MacOS/pymol
```

MAGMA is run from a command line as follows:

```
python `MAGMApath`/magma.py input_file.txt
```

The report is created using

```
python `MAGMApath`/report.py input_file.txt
```

doIt.sh and report.sh scripts are provided in each benchmark folder allowing you see how to run MAGMA and get reports without modifying either the system or the python path. The calculation should be executed in the directory where the input text file is located. Study the benchmark to see how this is done via examples.

Anatomy of an input file

The input script is relatively permissive. Lines starting with a hash are ignored. Any options that are not recognised by the software will cause the software to exit. Unexpected arguments will also trigger the software to exit, with the detail reports to the screen via stderr. Not all arguments need to be specified. The absence of critical arguments will trigger the software to exit, whereas less critical options will resort to default values.

```
#PDB
pdb_file ./pdbs/1D09.pdb  #PDB file
chains B D                      #chains to include
residues ILE VAL LEU             #methyls to label
short_distance_threshold  10     #distance threshold for pdb
long_distance_threshold   15     #long distance threshold (not used)
#Data
distance_restraints ./dat/ATCase_NOES.txt  #data
check_NOE_reciprocity on  #make sure cross peaks are reciprocated
min_size 1    #remove subgraphs of this size (ie, orphan peaks)
#Methyl Handling
merge_LV_label off  #combine Ls and Vs if not distinguished
merge_proR_S on   #merge proR/S for leu/vals in data/pdb
#Run Mode
subgraphMode on  #run with disconnected data subgraphs for speed?
polishMode on  #if subgraphMode on, repeat run with fixed conf assignments
finalMode on  #if subgraphMode on, repeat run with subgraphMode off
#MCES Optimisation
priority_iterations 50   #max number if initial interations
N_priority_MCES     3    #number of mces to collect in each test
maximum_run_time    1000 #max iterations on each search
reprioritise off    #reprioritise search list (last resport! default off)
#Extras
stripMode on       #introduces a speedup via an approximation (see notes)
craicMode on       #fix any assignments? (see atcase_fix)
craicFile fix.res  #fix these assignments (formated as combinedResults.res)
```

MAGMA outputs

MAGMA will create a subdirectory called 'results' in the directory where 'input_file.txt' file is located. A further subdirectory is created within results with the name of the PDB file, and a further '_full' added to the end if all graphs have been combined in the calculation. This enables multiple PDB files to be run, and the calculation tested in various modes if desired with each result saved in its own subdirectory.

After a successful calculation, the directory 'results/pdbname' will contain a number of files. The file combinedResults.res is the most important. The others report on how the calculation proceeded and inform on how to refine it if required.

combinedResult.res This is the final assignment file, pooling results from all data subgraphs examined. The first column of each row is the NMR resonance label. Following the ':' in the row are the labels for all PDB residues that are possible assignments.

conv.1.x.out . If running the MCES algorithm on subgraph x, it is necessary to first optimised the search tree for efficient performance. There will be a number of iterations of optimisation, specified by 'priority_iterations'. After each iteration (column1), the size of the largest overlapping network is saved (column2). The best assignments have the largest overlapping network. The earlier a large network is found in the calculation, the faster the overall runtime as this allows inferior solutions to be discarded early on. At the end of the iterations, the search tree that scored the highest overall will be selected for the 'main' MCES run that will be executed to go to exhaustive completion. For a long calculation it is desirable to adjust the number of iterations (priority_iterations), the maximum runtime of each run in the optimisation (maximum_run_time) and the number of MCES to acquire before finishing a run (N_priority_MCES). An optional re-prioritisation step (reprioritise) is also available, but this should be considered a last resort used only for long calculations. A large network size at the end of the optimisation is desirable. The settings in the benchmark provide a guide for sensible values to use. For most of the benchmark, the final result is obtained during the course of the optimisation, rendering it largely redundant. For the larger calculations ($\alpha 7 \alpha 7$ and ATCase) this is a hugely important part of the calculation. A plot of the output of this search is automatically produced (see figure 3).

conv.2.x.out Similar to the previous file but with more detail. The histogram of solutions is provided. So for each iteration of the optimisation (column1), the network size (column2) and the number of instances encountered (column3) are retained. A plot of this search is automatically produced (see figure 3).

mces_x.txt The core MCES binary reports all single assignment mappings that have an overlap size equal to the current best, or larger, as they are encountered in the search. These are individually written to this file. The .B and .G files monitor progress of the calculation. The current iteration number (column1) is shown with the node in the search tree under current consideration (column2) the node reached in the first branch of the calculation (column3) the current MCES size (coumn4) and the largest MCES size found (column5). The node reached in the first branch starts at the number of data nodes, and counts down to 0. A score of -1 here means the calculation is complete and so it shows you progress in the calculation. The .B file is added to when a solution equal to the current size or better is found. The .G file is added to every 10^6 iterations, allowing the current status to be seen. With gnuplot, the line: 'plot 'mces_x.txt.G' u 1:3:5 w li lc palette' allows you to see in real time how the calculation is doing, with the line starting at the total number of nodes, and decreasing with interactions towards -1 (completion). The total calculation time can be estimated from this plot. Extrapolate the 'lowest node reached' with number of MCES iterations using y=m log(iteration) +c to y=0 (or evaluate exp(-c/m)) for the total number of iterations expected before completion. Combine this with knowledge of how long 10^8 or 10^9 iterations took on your system, and you have an estimate. This is rather approximate however, as the program can find sudden new solutions that lead to a steeper gradient, and it can hit a lowest node with many unproductive options that will slow it down. mces files will appear with 'short' and 'long' in the file name. These are produced during the MCES optimisation step. The program deletes these as it progresses pased the initial optimisation. A final plot os made of the trajectory on completion (see figure 3).

vf2_x.txt The results from the xth subgraph from the vf2 algorithm. Files are added with 'new' in the filename during optimisation if 'stripMode' is turned on. These are used for optimisation and should be deleted during normal operation.

log Describes the progress of the calculation. Some of this is human readable. It parsed by report.py to summarise the key features of the calculation (see summary.pdf).

Practical notes

We have tested MAGMA with LV, ILV, and ILVA methyl labelled datasets. In the benchmark, the stereo-isomers in each case were merged into a pseudo-atom, taking advantage of additional experiments (e.g short mixing time NOESY or similar) to work out the pairings. This is really very useful information - the number of possible combinations goes with N!, so N!/(N/2)! is typically a large number. It is also highly desirable to distinguish amino acid types, so distinguishing Ls and Vs with a suitably labelled sample is going to significantly increase the number of successful assignments. So for solution NMR particularly, a recomended set of experiments/samples would be:

1. Label 1 methyl per LV, and record the full 4D HCCH NOESY.
2. For an LV labelled sample, label 2 methyls per LV and record a short mixing time NOESY (for linking stereo-isomeric pairs). Alternatively, label the residue with ^{13}Cs and execute a TOCSY.
3. Either L or V only labelled sample to unambiguously distinguish Ls and Vs.

On the benchmark, following this protocol we see no errors in any of the confident assignments, achieving 100% accuracy. Doubtless this will not be the case forever, and I would be delighted to receive your feedback with either successful, or unsuccessful analyses. In the absence of data to the contrary, you can safely assume MAGMA will 'fail-safe' and overestimate the ambiguity in a result, thus any confident assignments from analysis similar to those in the benchmark are worthy of your trust. If in doubt, or if trying a new approach, make a few mutations and double check.

From a data analysis perspective, MAGMA prefers a confident assignment of a distance restraint, so if there is extensive overlap in the spectrum meaning the pair of interacting residues cannot be clearly discerned, or if signal to noise is so low that you cannot see the cross peak being reciprocated in the relevant planes of the spectrum, we reccomend either leave it out, or treat the restraint with caution and try running the program multiple times and putting the restraint in each time with a different assignment. We recommmend 4D sparsely sampled datasets to overcome the this problem.

The number of confident assignments coming from MAGMA will depend on the sparsity of the input data and the shape of the networks involved. On average over the benchmark we get roughly 1 confident assignment per 3 unique distance restraints.

We have not tested the program with data from single amino acid labelled samples. Also we have not tested the program extensively without merging the L and V stereo pairs. Again, there is no reason it should not give a good answer, but our confidence for success comes primarily from the benchmark and the calculations / labelling schemes tested therein. We gain confidence in the success of the assignments from the benchmark, so sticking as close as possible to these data is recommended.

If there are multiple structures available, run with each of them and average the result as described in Pritisanac et al. The result from MAGMA fundamentally depend on the structure used. We show in the paper that this can be a good thing - a few assignments combined with the MAGMA results can be used to discriminate between possible structures. The ligand mode described in the paper also is available in the program. The interface for this hasn't been generalised. To use this in your project, please email.

Types of application

From the benchmark, with this version of MAGMA, most of the calculations are completed in seconds or minutes on a modern imac pro laptop (msrB, ubiquitin, ubiquitin_solid, ein, mbp, msg,atcase_fix). The ATCase calculation is more challenging and takes approximately 15 minutes. By contrast, the $\alpha 7\alpha 7$ dataset has not yet been seen to run to completion.

For the longer problems containing more edges (distance restraints), such as ATCase and $\alpha 7\alpha 7$, the initial optimisation parameters, (priority_iterations, N_priority_MCES, maximum_run_time) and the optional parameter of reprioritise (see the $\alpha 7\alpha 7$ input file) can be adjusted. At the end of the optimisation, (see conv files) the object is to have the largest possible overlapping edge score. If the calculation takes longer than a minute, then these parameters can affect the overall run time greatly. So if the calculation takes longer than a minute, it is desirable to pay attention to these values and look at how the optimisation is proceeding. Increase the number of priority iterations for example, allows the optimisation to search a greater proportion of sequence space. There is some randomness in the optimisation, so you can spawn multiple instances of the code, and after optimisation, keep only the calculation that scores the highest. The MCES algorithm itself is not particularly amenable to parallelisation and so the current version of MAGMA uses only a single processor. Use the benchmark to guide your choice of optimisation values for your problem.

In the event that the calculation is takes a day or longer, and a very large number of solutions are being continually generated, assuming you have spent time adjusting the optimisation protocol, the best course of action is probably to abort. This likely means the data is too sparse and the prospect for unambiguous assignments is not good, as there are far too many ways to explain the data. Go and assign a few residues via mutagenesis and have another go - fixing even a small number of assignments has drastic effects on the calculation. See atcase_fix for an example of how to fix assignments. The problem goes from N! to (N-1)! which means that if you have 100 nodes, 1 mutation corresponds to a 100 fold decrease in the problem's complexity.

The $\alpha 7\alpha 7$ example is an interesting case. The number of distance restraints in the network is particularly large here and we have not yet seen it run to completion despite running the calculation for a year with a previous version of MAGMA. With the current version which is the fastest yet, we estimate the time for completion to be on the order of hundreds of years. However - we find good solutions (edge overlap score of 137), which is the theoretical maximum, within minutes. After approximately 1 day we find another solution of this size, taking the result up to 2 solutions. Running the program past this point has not resulted in more solutions despite an extensive search to find more through adjusting the code and altering how the program runs. We will seek to address this in a future release. So in the case, where MAGMA reports a small number of 'large' solutions, but yet the calculation does not run to completion over a week, we would define your problem '$\alpha 7\alpha 7$-like', and recommend halting the calculation and take those results as your assignment solution. Alternatively, fix a few residues by mutagenesis and the runtime will decrease dramatically.

Included in this version of MAGMA is a mode referred to as 'stripMode'. This a fairly gentle approximation that looks for single restraints in the data network that are likely erroneous or exerting an unduly strong effect on the final result. This approximation gives a significant speed up to the calculation and we see identical results with and without this mode on the benchmark. You can try running with this mode on during testing, and off later in a project for a final validation.

It is generally desirable to run the calculation directly and combine all data into one graph. In practise, this is reasonable only for the proteins with small networks with few

edges (ubiqutin, msrB). For medium and large networks, additional steps are neccessary to 'funnel' the solution and get a reasonable calculation time. Here, run with subgraphs separated (subgraphMode on), for example Ein, MBP and MSG in the benchmark. Both $\alpha 7\alpha 7$ and ATCase data form one complete network so this option is irrelevant for these cases. In this mode, subgraphs are free to sit anywhere in the structure graph and there is no pressure to avoid overlap with each other, which is unphysical. This leads to an overly ambiguous result. There are two further modes that re-run MAGMA with the subgraphs recombined. As described for the input file, polishMode re-runs MAGMA where any confident assignments from the first run are removed as options for ambiguous residues. This mode increasese the number of confident assignments in MBP, MSG and Ein and reduces the ambiguity in other residues with little computational expense.

Finally, 'finalMode' then takes this result, and runs MAGMA calculation with all the subgraphs treated together (subgraphMode set to off), but restrained such that confident assignments from the previous run are fixed. This can be incredibly expensive, though it principle it should provide the least ambiguous result. In the case of MBP, 1536 solutions are obtained from polishMode, which finalMode reduces to 512 in a few minutes. The assignment ambiguity is reduced, but no new confident assignments are revealed. In the case of Ein and MSG, the number of solutions found by polishMode is greater than 10^{10}, so moving to the finalMode leads to very large pressures in terms of disk space and calculation time. For EIN it is possible to complete the calculation over several days - no additional confident assignments are returned in this case although several Terra-bytes of disk space are required. For MSG, we estimate that the finalMode calculation will take centuries to complete. Taken together, for medium and large proteins, we recommend subgraphMode and polishMode to bet set to on. If finalMode can complete in a reasonable amount of time this is desirable, but we have yet to see more confident assignments come from this setting so leaving it off is reasonable.

Multiple runs of MAGMA can be executed by a user, where 'craicMode' is used to read in an assignment file, for example the 'combinedResults.res' file from a previous run, that restricts residues to either single confident assignments, or to a specified set of options. These can be either from mutagenesis experiments or from other data. In this way, MAGMA can be used to complement an assignment with data from mutations. In the worst case, where MAGMA provides few confident assignments, it can help make an assignment performed via mutatgenesis more efficient.

Also included in this distribution is the module 'Rescore.py'. Execute this with:

```
python `MAGMApath`/Rescore.py input_file.txt
```

This will take the solutions from a MAGMA run, and rescore them against a longer distance threshold, 15Å. On the benchmark, this protocol results in a few extra confident assignments (MBP goes from 55 to 57, ATCase goes from 13 to 17, ATCase_fix goes from 29 to 31). Again, to date this mode run in this way increases the confident assignments maintaining the 100% accuracy. This analysis is experimental, and so it would be prudent to take the new results as highly likely rather than certain.

In principle it is desirable to rescore the MAGMA solutions against other experimental data such as residual dipolar couplings (RDCs), paramagnetic relaxation enhancements (PREs) or chemical shifts. As described in Pritisanac et al., we introduce erroneous assignments when we apply chemical shift information even in a very conservative fashion. We would be delighted to work with groups that have proposals or solutions to this so please get in touch. PREs and RDCs or similar are untested as filters in our hands, but previous work from the Kay and Clore groups strongly suggest that screening the MAGMA results by these experimental measures should be highly effective.

We would be delighted if you can send us any data, and please feel free to discuss your progress by email.

Usage suggestion

First, run the benchmark. Go to the benchmark directory, and execute:

```
RunBenchmark.sh
```

The calculation moves to each specified directory, and calls MAGMA with each benchmark member in turn. A summary file is supplied giving execution times. Summary and correct pdf files are produced in each folder graphically showing the results of the calculation. Confident (1 option) assignments are specified, and whether or not they are correct is indiciated.

On the benchmark, this accuracy is 100% for confident assignments. Either raw data or picked peak lists were supplied by a number of world-leading NMR labs detailed in benchmark credits, below. Note that the calculations for $\alpha 7 \alpha 7$ is very expensive. See the description above in Types of application.

These scripts serve as a basis from which a new script can be generated. If errors occur, please email.

Generating a report

Graphical reports (summary.pdf) can be generated (see report.sh scripts). These are made via a python script that takes advantage of pdflatex, pymol and gnuplot. A pymol script is generated and saved in a newly created 'report' folder. The automatic view of pymol is taken, and images are rendered (test1.png, test2.png, test3.png). If the script detects these image files, it will not recreate them in the subsequent runs. If you wish to create images with a different view of the molecule, then run the pymol script (pymol report/pymolscript.py or pymol report/pymolscript_noe.py), alter the view to something desirable, then render and save with 'ray' and 'png(testX.png)'. The data graph is visualised use a Monte-Carlo approach, so repeating this multiple times will give different results, some of which may be more pleasing to the eye than others. The report contains summarised information about the networks, the assignment result, and details about the specifics of the calculation. Each subgraph is annotated, and the algorithm used for its calculation is stated. Where the MCES routine has been used, graphs showing the progress of the final calculation, and of the early optimisation are also shown. See the benchmarkResults folder for reference reports generated internally.

Benchmark results

Table 1 shows the results from executing RunBenchmark.sh on an imac pro laptop. For each member of the benchmark, MAGMA is executed, a report is generated, and the result is assessed for accuracy. Two pdf summaries are created, summary.pdf and correct.pdf showing the report, and an overview of the assignments compared to the known result showing if the results are correct or not. A file correct.out is created detailing the confident assignments, and whether or not these are correct. The complete calculation takes approximately 20 minutes, of which 5 minutes is taken by pymol rendering make pretty pictures for the report (see table 1).

Credits

Dr. Iva Pritisanac (mail), University of Oxford

Dr. Matteo Degiacomi (mail), University of Oxford

BenchmarkID	PDBfile	Time(s)	Subgraphs	TotalDataEdges	ExplainedEdges	Peaks	Confident	Correct	Accuracy(%)
msrB	3e0o.pdb	0.011447	1	37	37	21	17	17	100.0
ubiquitin	1UBQ.pdb	0.004440	1	27	27	18	16	16	100.0
ubiquitin_solid	2l3z.pdb	0.003732	1	18	18	10	10	10	100.0
ein	1EZA.pdb	52.914104	6	145	144	84	42	42	100.0
mbp	1ez9.pdb	51.741046	4	144	141	70	55	55	100.0
msg	1Y8B.pdb	216.349999	12	230	229	141	60	60	100.0
atcase_fix	1D09.pdb	9.384543	1	91	83	34	29	29	100.0
atcase	1D09.pdb	1068.494664	1	91	83	34	13	13	100.0

Table 1: Results in bench.log from running RunBenchmark.sh on a quad core imac pro with two processors busy on other tasks. Total MAGMA calculation time is around 15 minutes, which rises to 20 including the pymol rendering required for the summary.pdf report files.

Prof. Andrew Baldwin (mail), University of Oxford

The problem of assigning resonances in NMR spectra is a very old one and even assigning methyl groups specifically can be traced back to early days of biological NMR. This problem has regained significant attention recently owing to the realisation of the Kay laboratory that specific $^{13}CH_3$ methyl labelling against a background of deuteration allows one to study high molecular weight machines using solution NMR. More recently the approach has also shown promise in solids NMR applications. It is typically effective, but expensive, slow and deeply tedious to assign methyl resonances through site directed mutagenesis. The origins of MAGMA stem from discussions over coffee in the Kay lab between Andrew Baldwin and Michael Latham, where the goal was defined to achieve not just one assignment, but all assignments consistent with experimental data. Such an approach can help guide the design of further experiments. This resulted in an early model taking advantage of properties of graph theory.

Developing this model further was the DPhil project of Iva Pritisanac, working in the group of Prof. Baldwin in Oxford. Iva realised that the assignment problem can be formulated in terms of exact search algorithms designed to solve MCES and subgraph isomorphism problems, and that the MCES search itself can be accelerated by pre-ordering the search tree using network approaches and the Hungarian algorithm. Her algorithm for adaptive graph matching sits at the centre of MAGMA. Matteo Degiacomi provided critical insight and mentoring in the python implementation, as well as having the liguistic capabilities to name the software MAGMA. The MAGMA logo was designed by Reid Alderson. The current version of the MCES optimisation, the core MCES algorithm and tweaks therein differ from that presented in the JACS article and offer enhanced performance, decreasing the calculation time, and increasing the number of confident assignments, while maintaining the accuracy. The details of this will be the subject of a future paper. These recent developments and the user reports are the work of Andrew Baldwin. MAGMA will continue to be actively developed.

The software can be used under academic or commercial licence, the terms of which are detailed below and contained in the file 'LICENCE'. If you are interested in using the Software commercially, please contact Oxford University Innovation Limited to negotiate a licence. Contact details are enquiries@innovation.ox.ac.uk quoting reference RAu/12970.

Beta-Testers

Writing software designed to be used by other people, that both doesn't swear at users and runs on many systems is a twin challenge. We would like to thank Brynn Fenwick (Scripps) and Lucas Siemons (UCL) for providing testing of the current MAGMA distribution. We would be delighted to be informed of any errors encountered by future users so that we might fix them.

Benchmark credits

The benchmarking data supplied with MAGMA, comes either from donated spectra that were analysed in house (Reid Alderson and Iva Pritisanac), or pre-picked peak lists were kindly supplied for each protein. For more details on experimental settings and solution conditions please refer to the primary papers, and ask the relevant authors. Distance restraints for the data graph were obtained from either NOESY (solution) and DREAM (solid) NMR experiments. Without the donation of supporting data this work would not be possible and I would like to reiterate my thanks to the contributing authors. May their karma, NMR and otherwise, remain ever brimming. If you are happy to donate your data to the benchmark, we would be delighted, and your NMR karma will also increase to new heights.

```
ubiqutin:
   data source:  Gianluigi Veglia/Fa-An Chao
   NMR data: NOESY solution state
   Data analysed by (peak picked): University of Minnesota
   contact:    vegli001[at]umn.edu
NOTE: the peak list identical to that with download from FLAMEnGO
    Chao et al. J. Magn. Reson. 2014, 245, 17-23.

ubiquitin\_solid:
   data source:  Beat Meier/Matthias Huber
   NMR data: 4D DREAM solid-state
   Data analysed (peak picked): ETH Zurich
   contact:      beme[at]ethz.ch
NOTE: the list is contained  in NMR restraint file here:
https://files.rcsb.org/download/2L3Z.mr

msrB:
   data source:  Oliver Lange et al./ Northeast Structural Genomics Consortium
   NMR data: 3D NOESY solution-state
   Data analysed by (peak picked): Northeast Structural Genomics Consortium
   contact:      oliver.lange[at]tum.de
NOTE: the list is contained under # noe_constraints in the NMR restraint file:
https://files.rcsb.org/download/2KZN.mr

ein:
  data source:  Marius Clore, NIH / Vincenzo Venditti
  NMR data:     4D NOESY solution-state
  Data analysed by (peak picked): Oxford
  contact:      mariusc[at]mail.nih.gov / nicolas_fawzi[at]brown.edu

atcase:
  data source:  Lewis Kay/Algirdas Velyvis, University of Toronto
  NMR data:     3D NOESY
  Data analysed by (peak picked): Oxford
  contact:      kay[at]pound.med.utoronto.ca / avelyvis[at]pound.med.utoronto.ca
```

```
atcase_fix:
  data source:  Lewis Kay/Algirdas Velyvis, University of Toronto
  NMR data:     3D NOESY
  Data analysed by (peak picked): Oxford
  contact:      kay[at]pound.med.utoronto.ca / avelyvis[at]pound.med.utoronto.ca

msg:
  data source:  Lewis Kay/ Vitali Tugarinov, University of Toronto
  NMR data:     3D NOESY solution-state
  Data analysed by (peak picked): Toronto
  contact:      kay[at]pound.med.utoronto.ca / vitali.tugarinov[at]nih.gov

mbp:
   data source:  Oliver Lange et al./ Northeast Structural Genomics Consortium
   NMR data:     3D NOESY solution-state
   Data analysed by (peak picked): Northeast Structural Genomics Consortium
   contact:      oliver.lange[at]tum.de
NOTE: the list is contained under # noe_constraints in the NMR restraint file:
https://files.rcsb.org/download/2MV0.mr

a7a7:
  data source:  Lewis Kay, Remco Sprangers, University of Toronto
  NMR data:     3D NOESY solution-state
  Data analysed by (peak picked): Toronto
  contact:      kay[at]pound.med.utoronto.ca / remco.sprangers[at]tuebingen.mpg.de
```

Licence

```
Academic Use Licence

These licence terms apply to all licences granted by THE CHANCELLOR, MASTERS
AND SCHOLARS OF THE UNIVERSITY OF OXFORD whose administrative offices are
at University Offices, Wellington Square, Oxford OX1 2JD, United Kingdom
(the University) for use of MAGMA for automated assignment of methyl nuclear
magnetic resonances (the Software) through this website http://magma.chem.ox.ac.uk/
 (the Website).
By downloading the Software through the Website, you (the Licensee) are confirming
that you agree that your use of the Software is subject to these licence
terms.

PLEASE READ THESE LICENCE TERMS CAREFULLY BEFORE DOWNLOADING THE SOFTWARE
THROUGH THIS WEBSITE.  IF YOU DO NOT AGREE TO THESE LICENCE TERMS YOU SHOULD
NOT DOWNLOAD THE SOFTWARE.

THE SOFTWARE IS INTENDED FOR USE BY ACADEMICS CARRYING OUT RESEARCH AND NOT
FOR USE BY CONSUMERS OR COMMERCIAL BUSINESSES.

1. Academic Use Licence
1.1 The Licensee is granted a limited non-exclusive and non-transferable
royalty free licence to download and use the Software provided that the Licensee
will:
(a) limit their use of the Software to their own internal academic non-commercial
research which is undertaken for the purposes of education or other scholarly
use;
(b) not use the Software for or on behalf of any third party or to provide
a service or integrate all or part of the Software into a product for sale
```

or license to third parties;
(c) use the Software in accordance with the prevailing instructions and guidance for use given on the Website and comply with procedures on the Website for user identification, authentication and access;
(d) comply with all applicable laws and regulations with respect to their use of the Software; and
(e) ensure that the Copyright Notice Copyright 2016, University of Oxford appears prominently wherever the Software is reproduced and on any documents or other material created using the Software.
1.2 The Licensee may only reproduce, modify, transmit or transfer the Software where:
(a) such reproduction, modification, transmission or transfer is for academic, research or other scholarly use;
(b) the conditions of this Licence are imposed upon the receiver of the Software or any modified Software;
(c) all original and modified Source Code is included in any transmitted software program; and
(d) the Licensee grants the University an irrevocable, indefinite, royalty free, non-exclusive unlimited licence to use and sub-licence any modified Source Code as part of the Software.

1.3 The University reserves the right at any time and without liability or prior notice to the Licensee to revise, modify and replace the functionality and performance of the access to and operation of the Software.
1.4 The Licensee acknowledges and agrees that the University owns all intellectual property rights in the Software. The Licensee shall not have any right, title or interest in the Software.
1.5 This Licence will terminate immediately and the Licensee will no longer have any right to use the Software or exercise any of the rights granted to the Licensee upon any breach of the conditions in Section 1 of this Licence.

2. Indemnity and Liability
2.1 The Licensee shall defend, indemnify and hold harmless the University against any claims, actions, proceedings, losses, damages, expenses and costs (including without limitation court costs and reasonable legal fees) arising out of or in connection with the Licensee’s possession or use of the Software, or any breach of these terms by the Licensee.
2.2 The Software is provided on an as is basis and the Licensee uses the Software at their own risk. No representations, conditions, warranties or other terms of any kind are given in respect of the Software and all statutory warranties and conditions are excluded to the fullest extent permitted by law. Without affecting the generality of the previous sentences, the University gives no implied or express warranty and makes no representation that the Software or any part of the Software: (a) will enable specific results to be obtained; or (b) meets a particular specification or is comprehensive within its field or that it is error free or will operate without interruption; or (c) is suitable for any particular, or the Licensee’s specific purposes.
2.3 Except in relation to fraud, death or personal injury, the University’s liability to the Licensee for any use of the Software, in negligence or arising in any other way out of the subject matter of these licence terms, will not extend to any incidental or consequential damages or losses, or any loss of profits, loss of revenue, loss of data, loss of contracts or opportunity, whether direct or indirect.
2.4 The Licensee hereby irrevocably undertakes to the University not to make any claim against any employee, student, researcher or other individual engaged by the University, being a claim which seeks to enforce against any of them any liability whatsoever in connection with these licence terms or their subject-matter.

3. General
3.1 Severability – If any provision (or part of a provision) of these licence terms is found by any court or administrative body of competent jurisdiction to be invalid, unenforceable or illegal, the other provisions shall remain in force.
3.2 Entire Agreement – These licence terms constitute the whole agreement between the parties and supersede any previous arrangement, understanding or agreement between them relating to the Software.
3.3 Law and Jurisdiction – These licence terms and any disputes or claims arising out of or in connection with them shall be governed by, and construed in accordance with, the law of England. The Licensee irrevocably submits to the exclusive jurisdiction of the English courts for any dispute or claim that arises out of or in connection with these licence terms.

If you are interested in using the Software commercially, please contact Oxford University Innovation Limited to negotiate a licence. Contact details are enquiries@innovation.ox.ac.uk quoting reference RAu/12970.

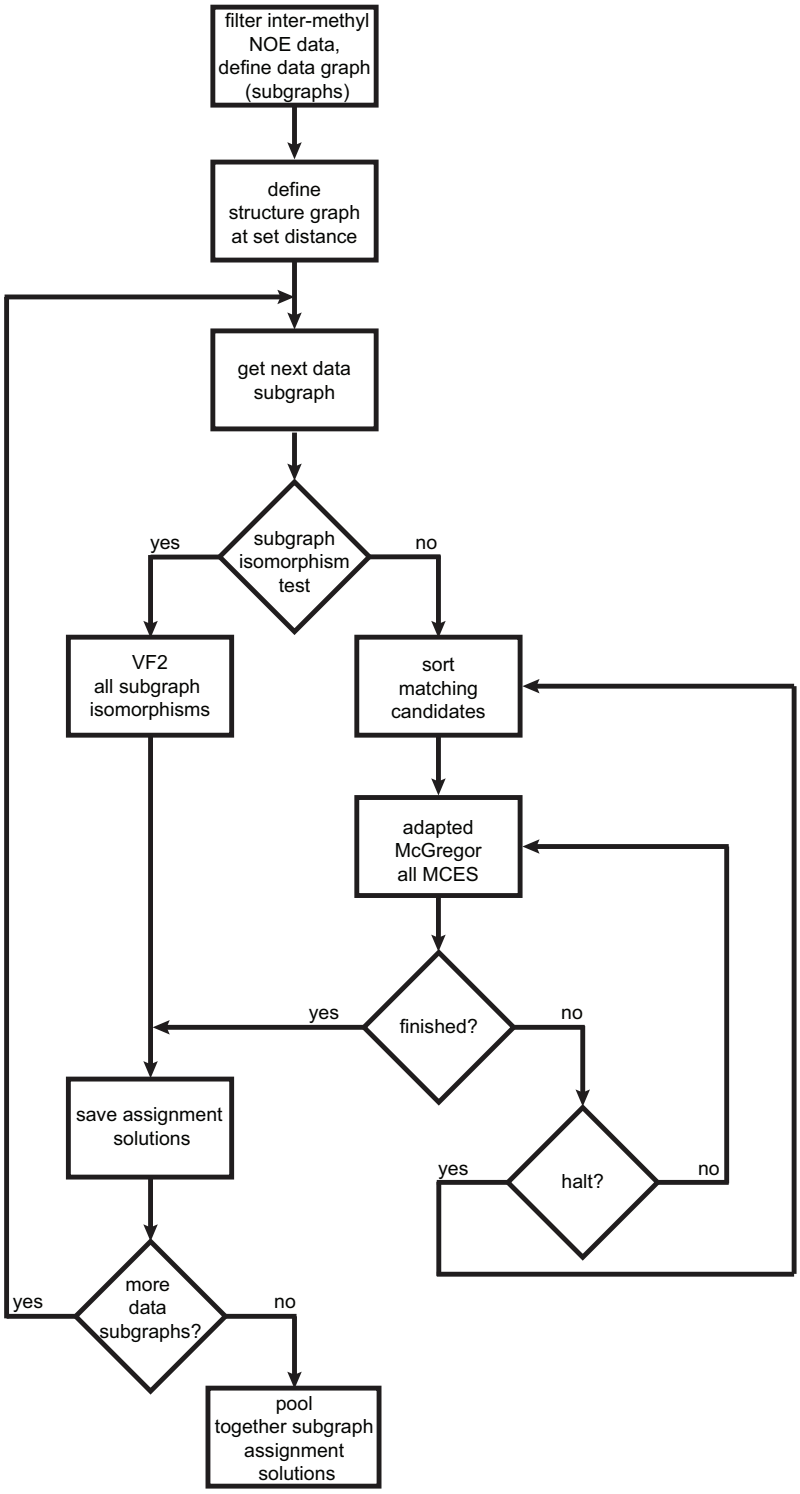

Figure 2: MAGMA workflow

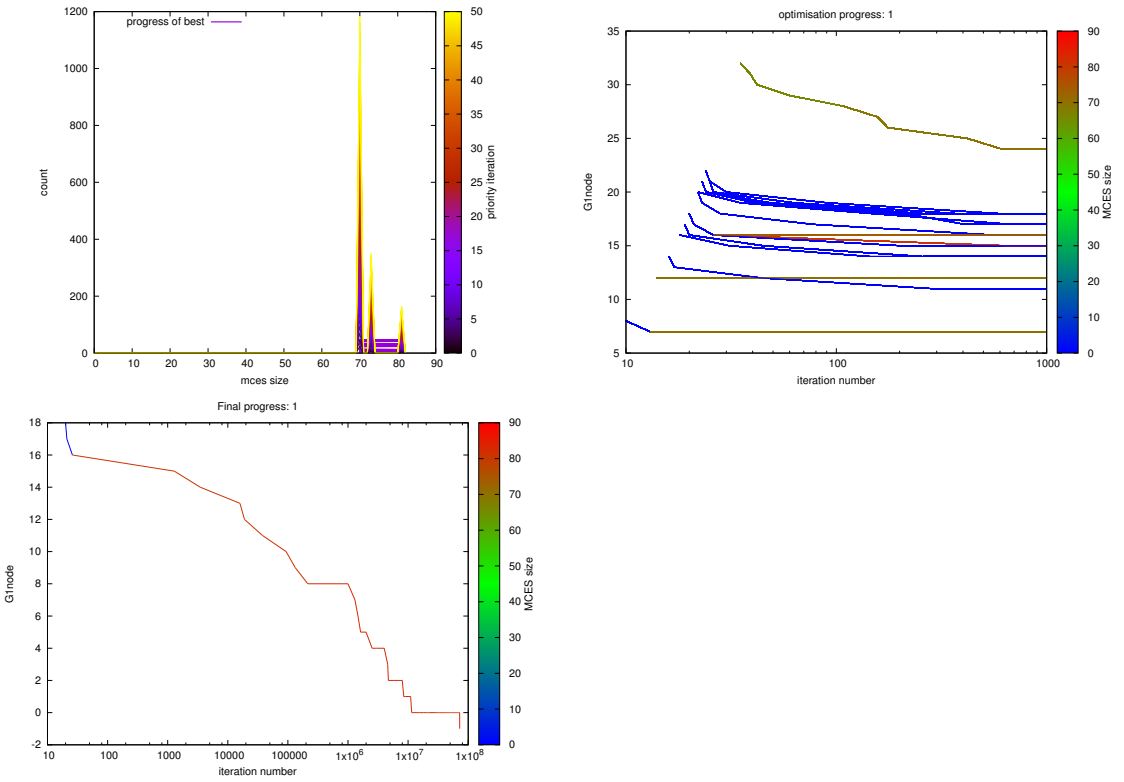

Figure 3: For a completed MCES run, up to 3 figures are generated by the report showing how the optimisation proceeded. The first two show the results of the MCES optimisation. For each optimisation iteraction, the distribution of solution sizes is shown. The indvidiual trajectories are shown in more detail, where the 'y' axis shows the progress down the search tree. The indicator starts at the maximum number of nodes and decreases to -1, indicating the search is complete. The colour shows the size of the largest solution encountered. At the end of optimisation, the solution that achieves the largest solution, and reaches the lowest node is selected for the final run. Finally, a plot showing the progress of the final run is shown. In cases where the winning solution was found during optimisation, the first two plots will not be created, with only the third being presented. The object of the optimisation is to have the largest possible solution, so for calcluations that take a long time, optimisating the parameters that affect this is desirable. In an extreame case, spawn multiple instances of the program and look at the trajectories of each, allowing only those with a large solution to proceed to completion. The figures shown here are for subgraph 1 of atcase. Solutions of size 81 were found during optimsiation. The best of these was selected for the main run, and the final set of solutions was obtained in under 10^9 MCES iteractions.