Guide to DESI Fiber Assignment Code

R. Cahn, April 25, 2017
Slightly amended for hack day at DESI conference, Angela Burden, 22 June 2017

1) Purpose

At the heart of DESI is the focal plane with its 5000 robotically controlled positioners capable of placing each optical fibers within a few microns of the desired location inside a radius of 6 mm. The positioners are arrayed in 10 petals shaped like pie slices. With about 10,000 pointings, the system can reach about 50 million targets.

The instrumented area of the focal plane is 7.5 square degrees and given the anticipated density of targets, about 25,000 targets fall on the focal plane for each pointing (generally called a tile or plate). With a total coverage of about 14,000 square degrees, the average coverage of each point in the footprint is about 75/14=5.35.

For each tile, choices must be made among the potential targets. Priorities are set in accordance with the scientific value of each target class. These classes include QSOs, LRGs, and ELGs. In addition, fibers must be assigned to measure the sky background ("sky fibers") and for calibration on standard stars. Both Ly-alpha QSOs and LRGs require multiple observations.

The fiber assignment code receives a variety of inputs. A fixed input is the list of the locations of the positioners in the focal plane. The survey plan is a list of pointings of the telescope. The actual decisions about which pointings to use on a particular night will be made on the spot, but these will be drawn from a list giving the basic outline of the survey. The targets are provided in a file mtl.fits. The standard stars and sky fibers are provided by stdstars.fits and sky.fits.
After the fiber assignment code has run, it writes a file for each tile not yet observed. Each of the files gives the assignment for each of the 5000 fibers. These files have names like tile 18924.fits. These are passed to DOS, where choices are made for observation. The results of observations inform the next version of mtl.fits to be passed to fiber assignment. Fiber assignment could be run every day during observing periods, or it might be run less frequently.

2) Context

The fiber assignment code is written in C++, but it is surrounded by python code.

For the purposes of testing the code and running simulations, packages have been written, primarily by Jaime Feroro-Romero, to provide all the necessary inputs for fiber assignment.

The relevant packages are **desisim** and **desitarget**. The latter provides the targets. The former simulates the survey, presenting it as a series of epochs. Each epoch results in a single instance of fiber assignment.

After each epoch, the status of the target list is updated to reflect new information. Some targets that have been observed will need additional observations and those are retained. Others are dropped.

The updated mtl.fits is passed back to fiber assignment along with the list of tiles in the next epoch. The results of the observations are collected in zcat.fits.

At the end, some summary statistics can be displayed. The python code wrapping fiber assignment is allowed to know the truth about each target: its true type and its true redshift. This information is in truth.fits, but is not accessible to fiber assignment since we want the fiber assignment code to really be what would be used ultimately in the experiment.

3) History

The existing fiber assignment code was initiated by Martin White and expanded by Lile Wang, who became a grad student at Princeton. The code was further developed by Arthur Stril, then at Ecole Normale Superieure in Paris, and by successor interns from ENS, Cyrille Doux, Aldo Riello, Louis Garrigue, and Lucas Pinol. Ted Kisner and Jaime Forero-Romero have contributed importantly.

4) Structure of the Fiber Assignment Code

Notation

Throughout, we indicate tiles by j, fibers by k, spectrometers or petals by p.

fiberassign.cpp

The main portion of the code contains several parts:

- **Read input files for targets, sky fibers, and standard stars.**
  The location of the files is specified through F.Targfile, F.SStarsfile, F.Skyfile. If fiberassign is run through the python code in desisim, the location is specified in the calling sequence for quicksurvey, which specifies the location of a “features” file. Reading the features file is actually the first action and sets out many of the parameters (F.xxx) for the code . If fiberassign is run directly, the target file must be provided. If fiberassign is run from the python wrapper in desisim, an mtl file is provided by make mtl.py in desitarget.

- **Combine these into a structure of type MTL.**
  Each galaxy carries a number of characteristics as specified in structs.cpp,

including RA, dec, and priority. The complete list read by fiberassign: TARGETID, RA, DEC, DESI TARGET, MWS TARGET, BGS TARGET, OBSCON- DITIONS, BRICKNAME, SUBPRIORITY, PRIORITY. The fiberas- sign code doesn't know the true nature or redshifts of the targets. TARGETID enables us to relate a single galaxy appearing in several files.

- **Determine the number of different priorities**
  Create priority classes, collections of targets all of which have the same priority. For example, there might be just three for ELGs, LRGs, and QSOs.

- **Read the file containing the locations of the positioners in the focal plane.**
  Determine which fibers are neighbours and note which petal each fiber is on.

- **Read the RA and dec of the centers of each of the 10,000 or so plates (tiles)**
  In addition, read IN DESI, specifying whether a tile is in the footprint, TILEID, OBSCONDITIONS, specifying under which conditions the tile may be observed, obsconditions:

  [DARK, 0, "Moon is down"]
  [GRAY, 1, "Moon up; illum fraction< 0.6 and (illum*elev)< 30 deg"]
  [BRIGHT, 2, "Moon up and bright (not GRAY)"]
  [POOR, 3, "Very bad seeing, high extinction, or bright cloud cover"]
  [TWILIGHT12, 4, "Between sunset/sunrise and 12 degree twi- light"]
  [TWILIGHT18, 5, "Between 12 and 18 degree twilight"]
  [DAY, 6, "Daytime calibrations"]
  [CLOSED, 7, "Nighttime but dome is closed due to rain, wind, dew..."]

- **Establish the geometry of the positioners using file collision.cpp.**
  This is needed to make calculations identifying colliding positioners.

- **Use kd-tree in collect galaxies for all to detemine which targets are within reach of each fiber for each plate.** Code is in src/modules.

- **For each tile-fiber, determine which galaxies are within its reach**.
  P[j].av_gals[k] is that list.

- **For each galaxy, determine which tile-fibers can reach it:**
  M[g].av tfs is list of pairs (j,k).

- **Check to see how many targets are out of reach of the survey.**

- **Use simple assign to make a preliminary assignment of a galaxy to reach tile-fiber**. Assignment is A.TF[j][k].

- **Determine which of the tiles in the list aren't used by any of the targets.**
  A.inv order[j] gives the original tile number of the jth tile used.

- **Smooth out the distribution of galaxies.**
  The initial distribution will result in unused fibers accumulating at the end of the survey. In order to add in the required standard stars and sky fibers, we want unused fibers to be distributed throughout the survey. A combination of redistribute and improve is used for this. At present the number of iterations of these processes is hard-coded here, but probably should not be.

- **Assign sky fibers and standard stars.**
  The nominal requirement is 10 standard stars and 40 sky fibers for each petal, using altogether 10% of the fibers.

- Try to use unassigned fibers with **assign_unused** .

- Write some statistics.

- Write assignments of targets to each fiber for each tile, e.g. tile_18924.fits.

structs.cpp

This establishes many of the basic features of the code.

- **Read_MTLfile** reads a fits file.
  Extracting for each target, targetid, numobs (no of observations remaining), RA, dec, priority, subpriority.

- **Read_fiber_positions** (x,y coordinates in mm)
  fp_x and fp_y are the x and y coordinates in the focal plane in mm, the spectrometer number (0-9). Compute N[i] for fiber i, the list of all other fibers within a distance F.NeighborRad and the list fibers of sp[k] of fibers that go to spectrometer k (or petal) 0, 1,...9

- **Read_plate_centers**
  gets the tile-id, RA and dec and pass it is observed in.

- **A_mapping**, invert tile, between the true tileid and where it is in the list of tiles.

- **Assignment class:**
  - TF (tile-fiber) TF[j][k] is the galaxy assigned to tile j, fiber k.

  - GL GL[g] is the pair (j,k) to which g is assigned.

– assign assigns galaxy g to (j,k), updates GL, M[g], keeps track of the number of standard stars and sky fibers in each petal.

– unassign unassigns, updates just as assign does.

– is_assigned_jg checks whether galaxy g is already assigned on tile j.

– is_assign_tf returns true if already (j,k) is assigned.

– chosen_tfs list of all tile-fibers that have chosen g.

– unused_fbp gives number of unused fibers on a particular petal of a given plate.

– unused_f counts unused fibers on plates.

- **plate_dist** gives radial distance at angle theta. Needs to be checked.
- **change_coords** given the center of the plate and the location of target, both in RA and dec, returns x-y coordinates of image in focal plane. Needs to be checked!
- **collision** checks for collisions of positioners.
- **Assignment::find collision** finds if assigning galaxy g to (j,k) would cause a collision.
- **is_collision** returns true if the galaxy assigned to (j,k) is in collision.
- **colrate** returns percentage of collisions.
- **projection** returns x,y of galaxy g assigned to (j,k).
- **pyplot::plot_tile** fancy plot of tile with assignments.

global.cpp

- **collect_galaxies_for_all** provides list of galaxies available to each tile-fiber (j,k): P[j].av gals[k].
- **pairCompare** used to compare the subpriorities of two targets.
- **sort_by_sub_priority** does just that.

- **collect_available_tilefibers** for each galaxy, g, finds list of tile-fibers that can reach it: M[g].av tfs.

- **ok_assign** g to jk checks for collisions with other fibers.

- **ok_for_limit_SS_SF** makes sure we don't exceed requirements for SS or SF.

- **find_best** uses remaining observations, priority, and subpriority to find best target for (j,k)

- **assign_fiber** uses find best to pick target for given (j,k)

- **assign_galaxy** goes through list of available tile-fibers and selects unused ones and among these uses the one on the petal with the most free fibers.

- **improve_fiber** tries to assign a galaxy to unused (j,k). If this is im-mediately possible, it makes the assignment. If not, it looks at all the galaxies it can reach, sorts them by subpriority. For highest subpriority galaxy, it looks at what could be done with its current tile-fiber and takes the best opportunity, provided there is one. Otherwise, it goes to the second best subpriority of galaxies within reach of (j,k). This increases the number of measured galaxies by one, when it succeeds.

- **simple_assign** goes through every tile and fiber using assign fiber .

- **improve** goes through every tile and fiber using improve fiber .

- **new_replace** Put in standard stars and sky fibers as required.

- **assign_unused** not clear what the value is if it just exchanges one observation for another.

- **redistribute** move galaxy to petal with most unused fibers.

- **diagnostic** Not used in fiberassign.

- **display_result** as it says.

- **fa_write** writes file for each tiles giving assignment of each fiber, in fits format.

- **pyplotTile** Makes Louis' plot of fiber positioners and targets.

feat.cpp

Features file
This file covers the various inputs passed in the features file.
example section by section:

SStarsfile {targetdir}/stdstars.fits
SkyFfile {targetdir}/sky.fits
Secretfile {targetdir}/truth.fits
tileFile /project/projectdirs/desi/software/edison/desimodel/master/data/footprin
fibFile /project/projectdirs/desi/software/edison/desimodel/0.3.1/data/focalplane
outDir {inputdir}/fiberassign/
surveyFile {inputdir}/survey_list.txt

The values of targetdir and inputdir are derived from the calling sequence for quicksurvey. For example:
./quicksurvey -O /global/homes/r/rncahn/desisim/desisim/temp_out -T /project/proj
datachallenge/quicksurvey2016/input/dark/lite/ -f /global/homes/r/rncahn/garrigue
bin/./fiberassign -E ~/garrigue/fiberassign/test/two_epochs -t /project/projectdi
/rncahn/large_mock_test/input/template/lite_template_fiberassign.txt --n_epochs 2

- Targfile columns: ['TARGETID', 'RA', 'DEC', 'DESI TARGET', 'BGS TARGET', 'MWS TARGET', 'SUBPRIORITY', 'OBSCONDITIONS', 'BRICK-NAME', 'DECAM FLUX', 'SHAPEDEV R', 'SHAPEEXP R', 'DEPTH R', 'GALDEPTH R']

- SStarsfile, Skyfile, columns: ['TARGETID', 'RA', 'DEC', 'DESI TARGET', 'BGS TARGET', 'MWS TARGET', 'SUBPRIORITY', 'OBSCONDI- TIONS', 'BRICKNAME']

- Secretfile columns: ['TARGETID', 'RA', 'DEC', 'TRUEZ', 'TRUE- TYPE', 'SOURCETYPE', 'BRICKNAME', 'MOCKID', 'OIIFLUX']
  – 'TRUETYPE'= 'GALAXY', 'QSO', 'STAR' – 'SOURCETYPE'= 'ELG', 'LRG', 'QSO'

A true QSO has 'TRUETYPE'= 'QSO' and 'SOURCETYPE'= 'QSO', while a fake QSO has 'TRUETYPE'= STAR' and 'SOURCETYPE'= 'QSO'.

A true ELG has 'TRUETYPE'= 'GALAXY' and 'SOURCE- TYPE'= 'ELG.' Typically, no fake ELGs are included in the mock catalogs.

A true LRG has 'TRUETYPE'= 'GALAXY' and 'SOURCE- TYPE'= 'LRG.' IF there are fake LRGs, they would have 'TRUE- TYPE'= 'STAR' and 'SOURCETYPE'= 'LRG.'

- Tilefile columns: ['TILEID', 'RA', 'DEC', 'PASS', 'IN DESI', 'EBV MED', 'AIRMASS', 'STAR DENSITY', 'EXPOSEFAC', 'PROGRAM', 'OB-SCONDITIONS']
- fibFile:
  #- Fiber to positioner mapping; x,y,z in mm on focal plane
  #- See doc/fiberpos.md for more details.
  #- Coordinates at zenith: +x = East = +RA; +y = South = -dec
  #- fiber positioner spectro x y z
  0 131 0 184.393230 95.818601 -4.671885
  1 284 0 299.120215 23.228125 -9.925561
  ...
- outFile specifies area into which to write results

- surveyFile is simply a list of the files used, specified by their position in the list given by tileFile.

The next section specifies whether to print output files. For the use of fiberassign, you must writeout put:

PrintFits true
This is necessary because fiberassign uses these output files to update its list of galaxies, indicating which galaxies are yet to be observed.

The next section is

Pacman false
Npass 5
MaxSS 10
MaxSF 40
PlateRadius 1.65

Pacmac is reduced layout of the focal plane. Npass is the number of layers, usually 5. MaxSS is the requirfed number of standard stars to be assigned to each of the ten petals for every tile. Similarly, MaxSF is the required number of sky fibers for each petal. PlateRadius is the radius of the focal plane in degrees.

Collision false
Exact true
AvCollide 3.2
Collide 1.98
NoCollide 7.0

PatrolRad 5.8
NeighborRad 14.05

- Collision is whether we allow collisions of positioners.
- Exact is whether we use the exact shape of the positioners.
- AvCollide is the distance (mm) between two galaxies where > we consider there isn't a collision, and < we consider that there is.
- Collide is the distance (mm) between two galaxies that guarantees a collision.
- NoCollide is the distance (mm) between two galaxies that guarantees there is not a collision.
- PatrolRad is the distance (mm) of allowed reach of a fiber.
- NeighborhoodRad is the distance (mm) between two positioners beyond which there are no collisions possible.

PlotObsTime false
PlotHistLya false
PlotDistLya false
PlotFreeFibHist false
PlotFreeFibTime false
PlotSeenDens false
PrintGalObs false
PlotPyplotTile true
PyplotInterval 50

These control plotting, graphics, etc.

MinDec -90.
MaxDec 90.
MinRa 0.
MaxRa 360.

These control the domain in which we consider targets.

misc.cpp

Miscellaneous tools for list, tables, etc.

collision.cpp

- orientation p-q-r is it clockwise or counter-clockwise?
  See 10th slides from following link for derivation of the formula http://www.dcs.gla.ac.uk/ pat/52233/slides/Geometry1x1.pdf

- intersect do two segments intersect?

- intersect seg circle do segment and circle intersect?

- element class provides tools for coloring, rotating, translating segments, circles

- polygon class provides tools for creating polygons, coloring, rotating them etc. In particular, there is create_fh for fiber holders, and create_cb for the central body of the positioner.

structs.h

• **FP Plate parameters:**
– **fp_x, fp_y** fiber positions (x,y) in mm.
– **spectrom**, All spectrometer assignments of fibers. – fibers of sp fibers associated with a spectrometer.
– **N** fibers that neighbour fiber k.
– **coords** coordinates (x,y) of fiber k

• galaxy contains truth information
– **z** redshift
– **targetid** unique identifier
– **category_true** category, hidden from fiberassign

• target contains only information that would be fed to fiberassign
– **id** long long immutable identifier.
– **nobs_remain** number of observations remaining for g
– **nobs_done** number of observations done for g
– **nhat** cartesian direction to g
– **ra,dec**
– **subpriority** number between 0 and 1 used in place of random number
– **desi_target**, mws target, bgs target (desi_target: 1->LRG, 2->ELG, 4->QSO)
– **SS** effectively boolean, yes for standard star
– **SF** effectively boolean, yes for sky fiber
– **priority_class** galaxies grouped into classes by t priority
– **t_priority** priority of galaxy a prior, e.g. QSO=3400, LRG=3200, ELG=3000. High number is high priority.
– **brickname**
– **av_tfs** for each target the available tile-files
– **obsconditions** observation conditions, 16-bit mask

• MTL vector of targets
– **priority_list** list of all priorities, e.g. 3000,3200,..
– **read_MTLfile**
– **assign_priority_class** group together targets with same priority

• onplate position in focal plane
– id
– pos pair giving x and y

• **Onplates** vector of on plates
• plate
  – tileid
  – tilera
  – tiledec
  – nhat three-vector pointing to center of tile
  – ipass which pass tile belongs to
  – av gals P[j].av gals[k] is list of galaxies accessible to fiber k on plate j
  – SS av gal standard stars available to some petal
  – SF av galsky fibers available to some petal
  – SS av gal fiberstandard stars available to some fiber
  – SF av gal fibersky fibers available to some fiber
  – SS in petal standard stars assigned to a petal
  – SF in petalsky fibers assigned to petal
  – obsconditions defines program, e.g. DARK, BRIGHT, GRAY

• **Plates** vector of plates
• **read_plate_centers**
• **read_save_av_gals** used only when we write initial results of kd-tree

• **Assignment**
  – TF_TF[j][k] is the target assigned to tile-fiber (j,k)
  – suborder:list of plates actually used in order of planned survey
  – inv_order: inverse of suborder
  – GL_GL[g] is list of tile-fibers to which target g is assigned
  – kinds_kinds[j][sp][id] is number of fibers in spectrometer sp, plate j, of kind id
  – unused_unused[j][p] is number of unused fibers on petal p – Methods
  – assign_assign g to (j,k)
  – unassign
  – find_collision
  – is_collision
  – is_assigned_jg
  – is_assigned_tf
  – nobs
  – chosen_tfs surveys sim only
  – nkind survey sim only
  – nobs how many more times object needs to be
  – unused_f total number of unused fibers
  – unused_fbp unused fibers by petal
  – colrate collision rate

– nobs time
– collision checks for collision between two fibers
– plate distplate scale calculation
– change coords change co-ordinates from RA and dec to x,y – projection projects a target onto a plate

5) Running the code

Compiling

Obtaining the code from the fiberassign section of desihub

```
git clone https:/github.com/desihub/fiberassign
```

Then set up the tools to run the code.
To do this run

```
source /project/projectdirs/desi/software/desi_environment.csh
```

There is a Makefile in ./fiberassign in github.
It calls a Makefile in fiberassign/src.
Go to your fiberassign directory and type

```
make install
```

The executable will appear in ./fiberassign/bin.

Running fiberassign

In practice, fiberassign will be invoked from python.

However, we will run it directly by pointing a collection of reduced size files, including both mocks and specifications of the layout of the focal plane.

The example we will use is provided in /project/projectdirs/desi/mocks/ fiberassign_example.

Next *important* make a subdirectory called outfiles in your fiberassign directory

copy the parameter file 'lite_template_fiberassign.txt' to your own directory
i.e.

```
cp   /project/projectdirs/desi/mocks/fiberassign_example/
lite_template_fiberassign.txt     "your_directory_path"
```

Change the line

outDir /project/projectdirs/desi/mocks/fiberassign_example/outfiles

to
outDir ‘your_path_to fiberassign’/outfiles

Now you can run the code
From the fiberassign directory, run the code

./bin/fiberassign “your_directory_path”/lite_template_fiberassign.txt

This should run quite quickly as it only runs on 200 sq. degrees of the survey. as the mtl.fits contains the targets for a restricted portion of the sky: 0<RA<10, -10<dec<10.
At completion, files are written to the subdirectory outfiles, one for each tile.