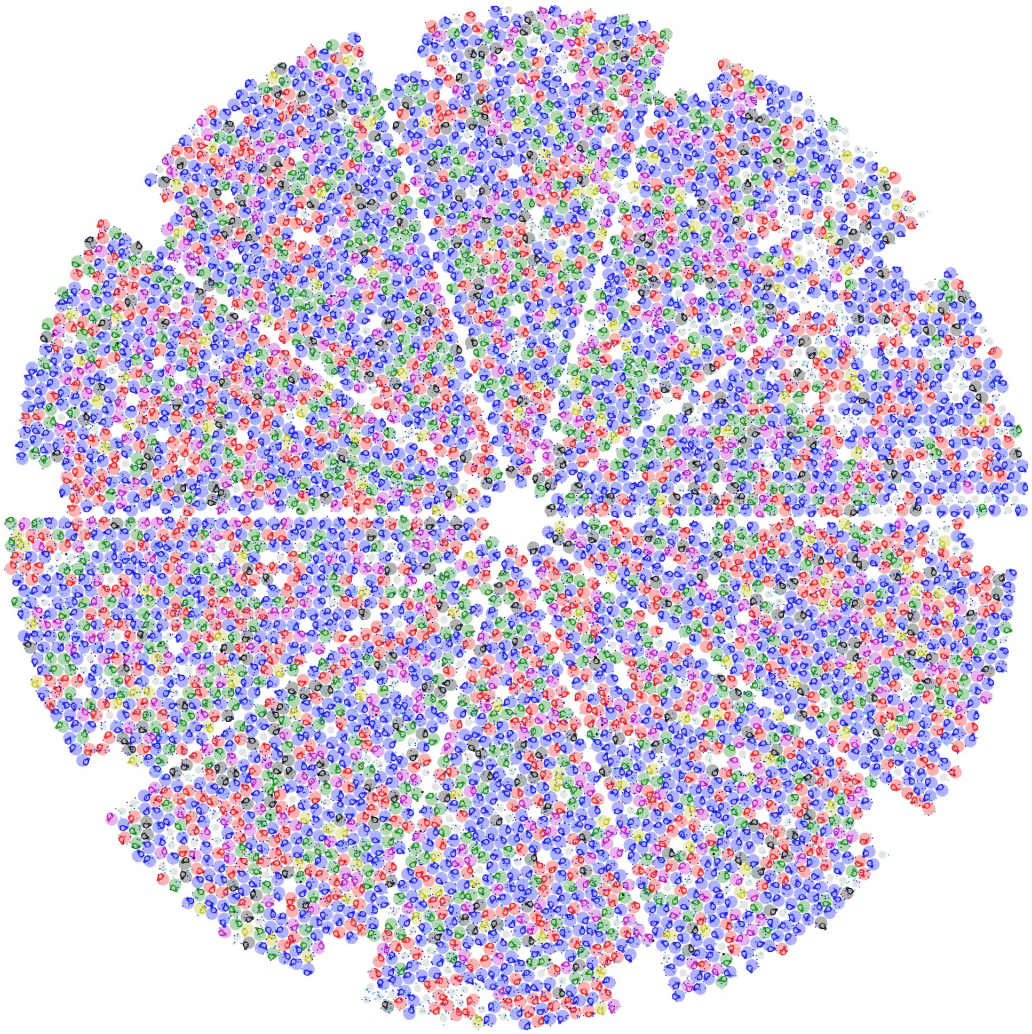
Description of fiber assignment code for Mocks in DESI experiment

Robert N. Cahn*[1] and Louis Garrigue†[2]

[1]Department of Cosmological Physics, LBNL, Berkeley
[2]Departement de physique, Ecole normale superieure, Paris

22 juin 2015



*rncahn@lbl.gov
†louis.garrigue@ens.fr

1 Introduction

Martin White started to develop a C++ code for fiber assignment, modified by Lado Samushia, then by Bob Cahn and finally by Bob and Louis Garrigue, intern for 5 months. We included the improvement, redistribution and update steps, to increase the number and the quality of the assignment, and at the same time to have a realistic process, in a way the code can be used. It provides a library of functions which can be easily adapted, with a main producing the assignment. It is well "pipelined" and someone can use it easily. The code can be found on DESI git repository with a short description of how to run it.

2 In a nutshell

2.1 Input files

They are found in the NERSC repository (see features.txt for addresses). The produced executable (to compile with `make all in src repository)` is `assign`. The calling sequence will look like : `./assign features.txt`. An example of how runing it on NERSC is provided in the `run` script, then call it with `qsub run`.

- Parameters : all parameters used in the fiber assignment are written in this file, including addresses of other input files
- Target DB : information, before the study, on all possible targets : (NERSC rep) /projects/projectdirs/desi/mocks/preliminary/objects_ss_sf0.rdzipn, created by `make_catalog`. The samples are taken from Martin's mocks and stored on NERSC at /project/projectdirs/desi/mocks/preliminary/. From these files we create a single file containing the appropriate mix of ELG, LRG, QSO, SS (Standard Stars) and SF (Sky Fibers) using the python script in git fiberassign/bin/make_catalog_starsandsky.py. In the same place there is python code to produce a mixture of galaxies without any correlations, but with the correct dn/dz
- Obs DB : database constructed from the ongoing DESI observations after the data has been processed by the Spectroscopic pipeline. This DB has not been designed yet
- Survey tiles : file containing the positions of all the tiles to be observed in /project/projectdirs/desi/software/edison/desimodel/0.3.1/data/footprint/desi-tiles.par
- Fiber positions : locations of the positioners in the focal plane in /project/projectdirs/desi/software/edison/desimodel/0.3.1/data/focalplane/fiberpos.txt

To change the location of other input files, one can simply change it in the features file.

2.2 Output files

They are produced in a directory defined in features.txt, in the format `tile54.fits` for example for the 54th tile. There are therefore 10666 such binary files. They consist in 5000 lines (fibers) with the following columns :

- fiber : [0-4999]
- positioner : [0-4999] (not provided yet)
- number of available objects
- ID of available objects
- objtype : ELG, LRG, QSO, SKY, STDSTAR, GAL, OTHER (actually QSOLy-a, or QSOFake etc for now) (-1 if the fiber isn't assigned)
- targetid : unique target identifier to get back to target selection info
- desi target0 : 64 bit mask of targeting info (not yet)
- ra : degrees [0-360]
- dec : degrees [-90 - +90]
- xfocal : mm from center in positioner coordinate system
- yfocal : mm from center in positioner coordinate system

3 Source files

Source files are file.h and file.cpp and are in this increasing dependency order :

- misc : a home-made library of structures (and functions on them) needed to manipulate concerned data, but independent of them. There are pair (of int), List (of int), Table, Cube, and timing, printing, string conversion, error report items.
- collision : used to compute collision checking and build polygons representing positioners

— features : carries all useful parameters
— structs : structures of the manipulated data and their members
— global : main high-level functions and algorithms used in the program to collect information, assign fibers, print statistics and output. Important ones are described further
— main : neat and quickly understandable code that sums up all the steps of the assignment

4 Parameters

Here are some features on input galaxies simulated catalog :

Kind	Id	Priority	Nobs	Density ($obj \cdot deg^{-2}$)
QSO Ly-α	0	1	5	50
QSO Tracer	1	1	1	120
LRG	2	3	2	300
ELG	3	4	1	2400
Fake QSO	4	1	1	90
Fake LRG	5	3	1	50
Standard Star	6	5	1	140
Sky Fiber	7	6	1	1400

TABLE 1 – Characteristics of galaxy samples as set in make_catalog_rnc.py

A number of parameters are defined in features.txt.
— input and output directories
— Output = true, whether you want to release the output
— Randomize = false randomize order of plates in making plans
— Pacman = false selects only spectrometers 0, 1, 2, 7, 8, 9 of the pacman
— Npass = 5 number of passes
— MaxSS = 10 ; MaxSF = 40 number of fibers assigned to SS and SF on a petal
— PlateRadius = 1.65° radius of the plate
— InterPlate = 0 minimal number of plates between two observations of the same galaxy
— Analysis = 0, tile distance for getting the information from previously observed tiles
— InfDens = false, simulate infinite density of SS and SF
— TotalArea = 15789.0 deg^2total area of the sky considered
— invFibArea = 700 inverse of area in deg^2accessible to a fiber (fiber density for a deg^2)
— moduloGal = 1, if 2 for instance, reads only one object over two in galFile
— moduloFiber = 1, same for fiber
— PatrolRad = 6.0 mm maximum distance, on plate coordinates, that allows a fiber to a galaxy
— Collision = true when we want to allow collisions, to compute the collision rate for instance, and very practical when one wants to make quick tests, to run the fiber assignment faster
— Exact = true whether we want exact collision checking (exact geometry of components) or just circles
— AvCollide = 3.2 mm in case of no exact collision checking, limit distance between two galaxies for their positioners to collide (so that we have the same collision rate than with exact geometry)
— Collide = 1.98 mm minimum distance allowed on plate projection of two assigned galaxies on the same fiber (optimizes collision checking)
— NoCollide = 7 mm maximum distance between two galaxies for the collision of their corresponding galaxies (optimizes collision checking)
— NeighborRad = 14.0 mm maximum distance to consider that two fibers are neighbors
— PlotObsTime, etc, whether we want to plot some information into output files
— Verif = false, whether we verificate that the assignment is sane (no collision, sane mapping, etc...)

5 Classes and structures

Classes and structures are built to be independent of each other, flexible, quickly understandable, logical, and with no redundant information as much as possible.

Structure name	Meaning	Description
element	Element of a geometric figure	Either a set of juxtaposed segments, either a circle
polygon	Polygon of segments + circles	Set of elements
Feat	Features/Parameters	Carries all needed parameters defined in features.txt and some other computed in the input reading process
PP	Plate Parameters	Carries locations of positioners on the plate, spectrometer correspondence, and neighboring fibers information
onplate	Plate coordinates	Used for coordinates in the focal plane in mm
plate	A plate	Locations in the sky of the tile in terms of a unitary vector derived from RA and DEC. Carries also the Id of the tile, its pass, and the available galaxies it is able to reach. Plates is vector of plate, and carries all information on tiles
galaxy	A galaxy	Information on a galaxy : kind which corresponds to Table 1, a position in the sky (in two differents ways), and the available tile-fibers that can observe it. Gals is vector of galaxy and carries all information on galaxies
Assignment	An entire assignment	Carries mapping of tile-fibers to galaxies on a table, and other redundant information for optimization

TABLE 2 – Classes and structures

6 Functions

In algorithms, j stands for a plate, k for a fiber, p for a petal, g for a galaxy. They are integers in the code.

6.1 Improvement functions ideas

Here we present the three ideas of improvement of this first assignment, which increase the number of used fibers and the quality of the assignment. It represents the body, the main purpose of our work. They need to be somewhere "orthogonal", so that they don't loose efficiency when we call one of them after an other. Those three ideas are coded into the three following functions :

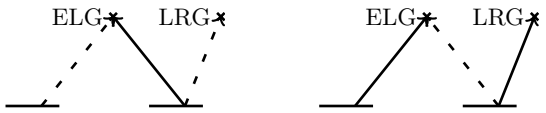


FIGURE 1 – Improve

The idea is to take an unassigned fiber and to look at the galaxies it can reach and that are already assigned. We then look at the second fiber, which is observing this galaxy and try to reassign it to an other galaxy, and assign the first fiber to the first object. It is powerful because we use it when making a plan, and so the two fibers can (and are almost always) from different plates, which can be arbitrarily separated.

FIGURE 2 – Redistribute

In redistribute, we look at an assigned fiber, and try to make the observed object observed by and other fiber, which can come from a different plate. It "anneals" the assignment. In this way, it doesn't improves anything, but we then

apply the improve function again, which is more efficient. The idea is the improve, then redistribute, then improve again, redistribute etc. If there is no redistribution between a new improve, the improve is almost useless. Furthermore, among possible fibers, we assign it to the least used petal.

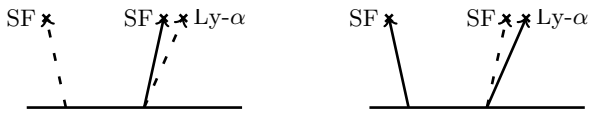


FIGURE 3 – Improve from kind

The third idea is to take an assigned fiber that is able to reach a SF (or SS), to assign it, and then we reassign a fiber assigned to a SF. As the number of SF per petal has to remain the same as previously (40) we search the second fiber among the same petal of the first fiber. It's not used anymore since it becomes useless when we first use new_assign.

6.2 Some inportant functions in structs.cpp

`plate_dist` turns radians into mm on the focal plane, i.e. it is the plate scale as a function of angle.

`projection` combines a galaxy and a particular plate to give the coordinates the galaxy will have in the focal plane when observed as with this tile. It's a rotation in angular coordinates

`find_collision` returns the fiber number of a fiber that conflicts with tiblefiber (j,k), or -1 if there is no collision

6.3 Collecting

`collect_galaxies_for_all` is multithreaded, and for each fiber of each tile, collects reachable galaxies. It uses kdTree and htmTree libraries written by Martin White. It's absolutely necessary in order to do computations in a reasonable time with supercomputers

`collect_available_tilefibers` computes, using the previous work, available tile-fibers for each galaxy (inverse map)

6.4 Useful sub-functions for global functions

`ok_assign_g_to_jk(g,j,k)` checks to see if we can assign g to the tile-fiber (j,k), according to some assigning rules described further (g has to be a QSO, LRG or ELG, only ELG in last pass, and no collision with neighbors)

`find_best(j,k)` finds the best reachable galaxy for this fiber, according to assignment rules (not observed in two tiles separated by less than Interplate, choose the one with the higher priority, then if several ones compete, the least observed)

`assign_fiber(j,k)` tries to assign this fiber using find_best

`improve_fiber(j0,n,j,k)` if this fiber is unused, tries first to simply assign it, and if it doesn't work, tries to reassign some used one (jp,kp) where $j0 \leq jp \leq j0 + n$. Before : (jp,kp) - g ; (j,k) & gp free. After : (j,k) - g & (jp,kp) - gp. The power of this function lies in the fact that j and jp can correspond to different passes. Among all galaxies of same priority and same number of observation, it takes the one belonging to the most unused petal, to distribute ELGs

Algorithm 1 Improve_fiber(j0,n,j,k)

if k is not assigned **then**
  try to assign running assign_fiber(j,k)
  if k couldn't be assigned this way **then**
    initialize a set of variables jp, kp, g, gp
    for each galaxy g available to (j,k) **do**
      if it's possible to assign g with k **then**
        for each chosen tile-fibers (jp,kp) which chose g (where $j0 \leq jp \leq j0 + n$) **do**
          $gp \leftarrow find\_best(jp, kp)$
          memorize jp, kp, g, gp if it's a better set (gp is more worthy) than previous one
    if $gp \neq -1$ **then**
      Unassign $(jp, kp) \longleftrightarrow g$
      Assign $(j, k) \longleftrightarrow g$
      Assign $(jp, kp) \longleftrightarrow gp$

6.5 Assigning making a plan

In the simulated catalog, we know all information on objects (if a QSO is a real one for example). During the real study, one won't have access to this information prior to the observation plus analysis time. Thus, in the code, we have to simulate that we have this information only when we have at least once observed the object, and when the analysis of received data tells which exact type it is (fake or not for ELG for example). Algorithms do a plan with only types of objects (QSO, LRG, ELG) available in the catalog. A plan consists of an assignment on a set of consecutive tiles (from 0 to 2000 for instance). The assignment works better when we do only one plan.

The argument "next" (integer) in following functions means that we treat all next "next" plates in the plan we make. If it is -1, it will deal with all left plates. For example, if $next_plate = 10$ and one launch a function with $next = 100$, this function is going to do his job on plates from 10 to 110.

`simple_assign` makes a first simple assignment plan : for each fiber assign to the best available galaxy

Algorithm 2 Simple_assign(j0,n)

for each plate j from j0 to j0+n **do**
  for each fiber k in a random order **do**
    try to assign_fiber(j,k)

`new_assign` makes a first assignment plan trying to assign first QSO, then LRG then ELG (neither SS nor SF). This way, a QSO can't be lost because of a collision, among other things.

Algorithm 3 New_assign_fibers(j0,n)

for each plate j from j0 to j0+n **do**
  for each petal p of this plate, in a random order **do**
    for each fiber k of this petal, in a random order **do**
      Try assign_fiber(j,k) only allowing QSO
    for each fiber k of this petal, in a random order **do**
      Try assign_fiber(j,k) only allowing LRG
    for each fiber k of this petal, in a random order **do**
      Try assign_fiber(j,k) only allowing ELG

`improve` improves the plan applying improve_fiber to all of the fibers involved in the plan

`improve_from_kind (kind)` for every concerned petal, tries to assign unassigned fibers to objects of kind "kind" (useful when it's SS or SF), to release an other fiber (formerly assigned to a SS or SF) that would be reassigned to a regular galaxy (kind is SS or SF when we call it). There is also the function `improve_fiber_from_kind` that does the same but on only a fiber. We don't use it anymore, since we use new_assign, which already includes potential improvement brought by this function.

Algorithm 4 Improve_from_kind (kind,j0,n)

```
for each plate j from j0 to j0+n do
    for each petal p of this plate, in a random order do
        for each unassigned fiber k do
            initialize a set of variables kp, g, gp
            for each available galaxies g of k do
                for each fiber kp of p assigned to a galaxy of kind do
                    if no conflict & g is worthier than previous one then
                        memorize kp, g, gp
            if g ≠ −1 then
                Reassign kp to g
                Assign k to a galaxy gp of kind kind
```

`redistribute_tf` "anneals" the distribution (so that improve is more efficient after) and in the same time redistribute in a way that the number of unused fibers per petal tends to be the same for every petal. It is very important to prepare the step of assigning SS and SF, in order to replace the least number of ELG and LRG for having the needed number of SS/SF

Algorithm 5 redistribute_tf(j0,n)

```
for each plate j from j0 to j0+n do
    for each assigned fiber k do
        get g, the galaxy to which (j,k) is assigned
        initialize a set of variables (jpb,kpb) to memorize values of (jp,kp)
        for each available tile-fibers of g do
            if (jp,kp) is better (j0 ≤ jpj0 + n, unassigned, it's ok to assign, and the petal of (jp,kp) is more unused than the one of (jpb,kpb)) then
                memorize (jp,kp) in (jpb,kpb)
        if jpb ≠ −1 then
            Reassign g to (jpb,kpb)
```

`update_plan_from_one_obs(j)` updates the plan formerly made so that if for example on the plate $j - Analysis$ plates (to simulate there is a delay between observation and result on analysis) we observed a QSO which the analysis reveals fake but is still planed to be observed again further in the plan, we will unassign the corresponding fiber and try to reassign it with improve_fiber. In the input catalog we only know the basic type (QSO, LRG, ELG, SS, SF) of the galaxies. The precise type is only infered by a simulation coded in python, but is generated with randomness. So in the algorithms, we can know the nature of the object only when it has been observed once at least and analysed (this delay of "Analysis" number of plates is inserted between those two events). In a plan, we try to optimize assignment, but of course, we can't know the precise type (fake, ...) so it is possible to foresee observing for example 4 times a fake QSO. We won't do it, so after having planned, when we begin to apply it and do the real observation, if we have just previously observed a fake QSO, we will remove further observations in this plan. We then try to assign the released tile-fibers.

Algorithm 6 Update_plan_from_one_observation(j0,end_plan)

```
get the list of galaxies observed by this plate that are discovered fake or target
for each of those galaxies g do
    get the list of tile-fibers that are supposed to observed this galaxy further in the plan j0 ≤ jp ≤ end_plan
    for each of those tile-fibers (jp,kp) do
        Unassign (jp, kp) ⟷ g
        run improve_fiber(j0+1,end-j0,jp,kp) (tries to reassign (jp,kp)
```

`assign_sf_ss(j)` just before the observation of a tile, we assign SF and SS, replacing ELG then LRG if necessary

Algorithm 7 assign_sf_ss

```
for each petal p of this plate, in a random order do
    try to assign unused fibers to 40 SF and 10 SS
    if not enough SS and/or SF then
        try to replace some ELG for SS
        try to replace some ELG for SF
    if not enough SS and SF then
        try to replace some LRG for SS
        try to replace some LRG for SF
```

`assign_left(j)` just before the observation of a tile, for all unassigned fibers, tries to assign them to a galaxy in reach that is supposed to be observed later, and remove the later observation. The goal is to observe and thus analyse objects as soon as possible, not waiting further in the plan.

6.6 Displaying results

Histograms are written in Tikz format.

`results_on_inputs` displays some statistics on input files (histogram of number of objects in range of a fiber, histogram of number of tile-fibers which can observe a given object, histogram of redshifts by type)

`display_results` provides the tex-formated results to make Table **??** and writes histograms one can see further in this document

`write_FAtile_ascii(j)` writes output of the assignment in ASCII format

`fa_write(j)` writes output of the assignment in FITS format

`pyplotTile(j)` writes a file tileX.py from which (executing `python tileX.py` one can produce the pdf graphic colored plot of the tile. In it, are also ploted objects requiring more observations. A LRG already observed twice won't appear for example

7 Rules of assignment

— a tile-fiber is only assigned once
— no collision between fiber positioners
— two observations of a QSO or an LRG can't be separated by less than `InterPlate` plates
— in last pass, only ELG, SS and SF are considered
— there are 10 and 40 fibers assigned to SS and SF per petal
— basically, we choose best galaxy from at the same time available, observed less than maxgoal(kind) times and unassigned ones. Among them, we take those with highest priority, and then the one which has been seen the least number of times
— between an already observed Ly-α and an unknown QSO, one chooses the already observed Ly-α
— the same SS or SF can be observed several times
— SS has priority over SF
— our policy is : while improving the assignment, never unassign an object without reassigning it right then

8 Algorithm

`main` proceeds by :
— loading input files : features of galaxies to make F, reading in the galaxies catalog to make G, the positions of the fibers on the plate to make pp, the plate centers and positioner locations to make P
— collect available galaxies for each fiber for each plate, and compute the inverse map (available fibers for each galaxies, useful for improvement functions)
— before begining the study, make a plan of assignment
— begin the study, and update the plan as we get information from objects, ongoing

8.1 Redistribution-Improvement step

Assigning only QSO, LRG and ELG, not SS and SF when making the plan leads to have a lot more degrees of freedom for the improvement function, which is then much more efficient. Redistributing unused fibers by petal permits to replace the least ELG possible when assigning SS and SF just before an observation.

When applying the plan, sometimes redistributing-improving the plan is very efficient, it updates the further plan (compared to the initial maked plan, just before begining the study), taking into account that some new fibers have been released and that we have new information on precise kind of objects.

Here is presented, in the Figure 4, histograms of unused fibers per petal, during redistribution/improvement process. The goal is to have 50 unused fibers per petal before begining observations.

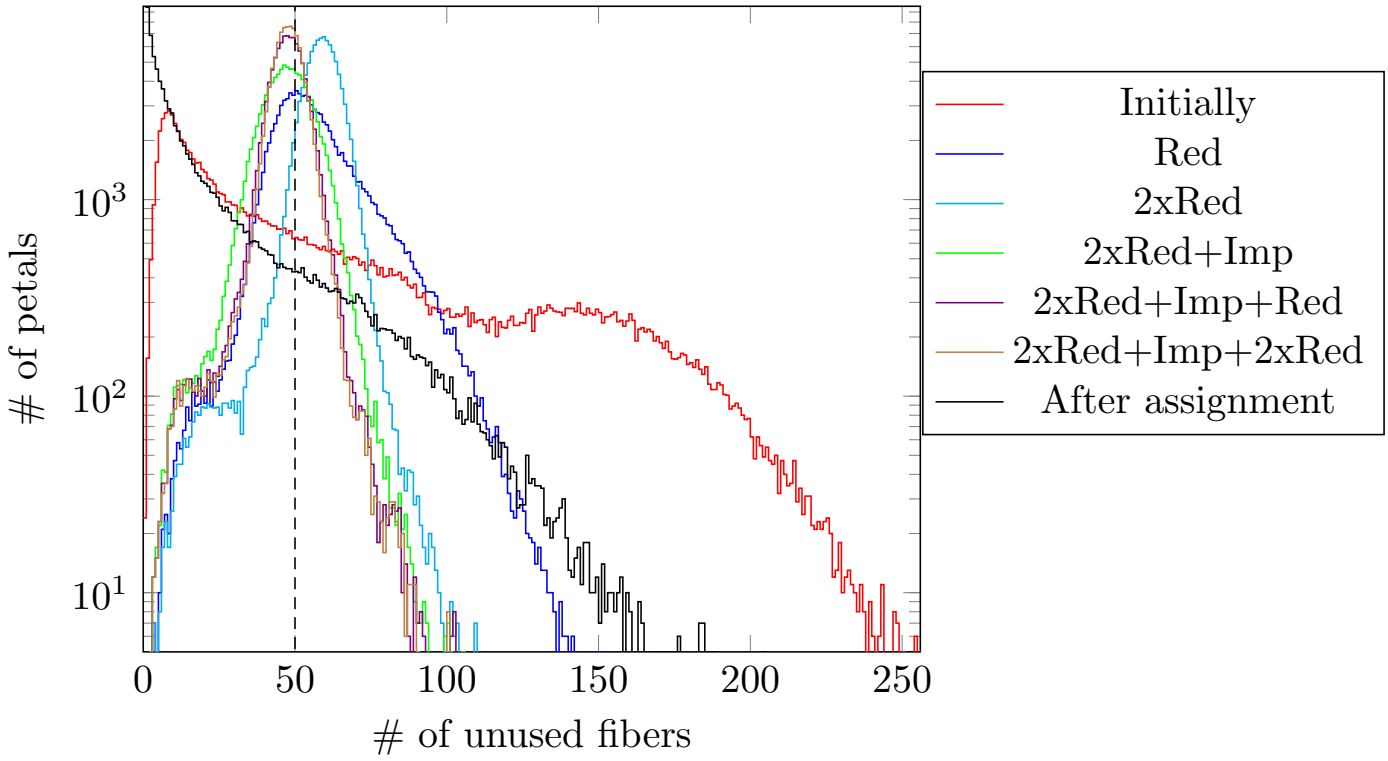


FIGURE 4 – Unused fibers, at different steps

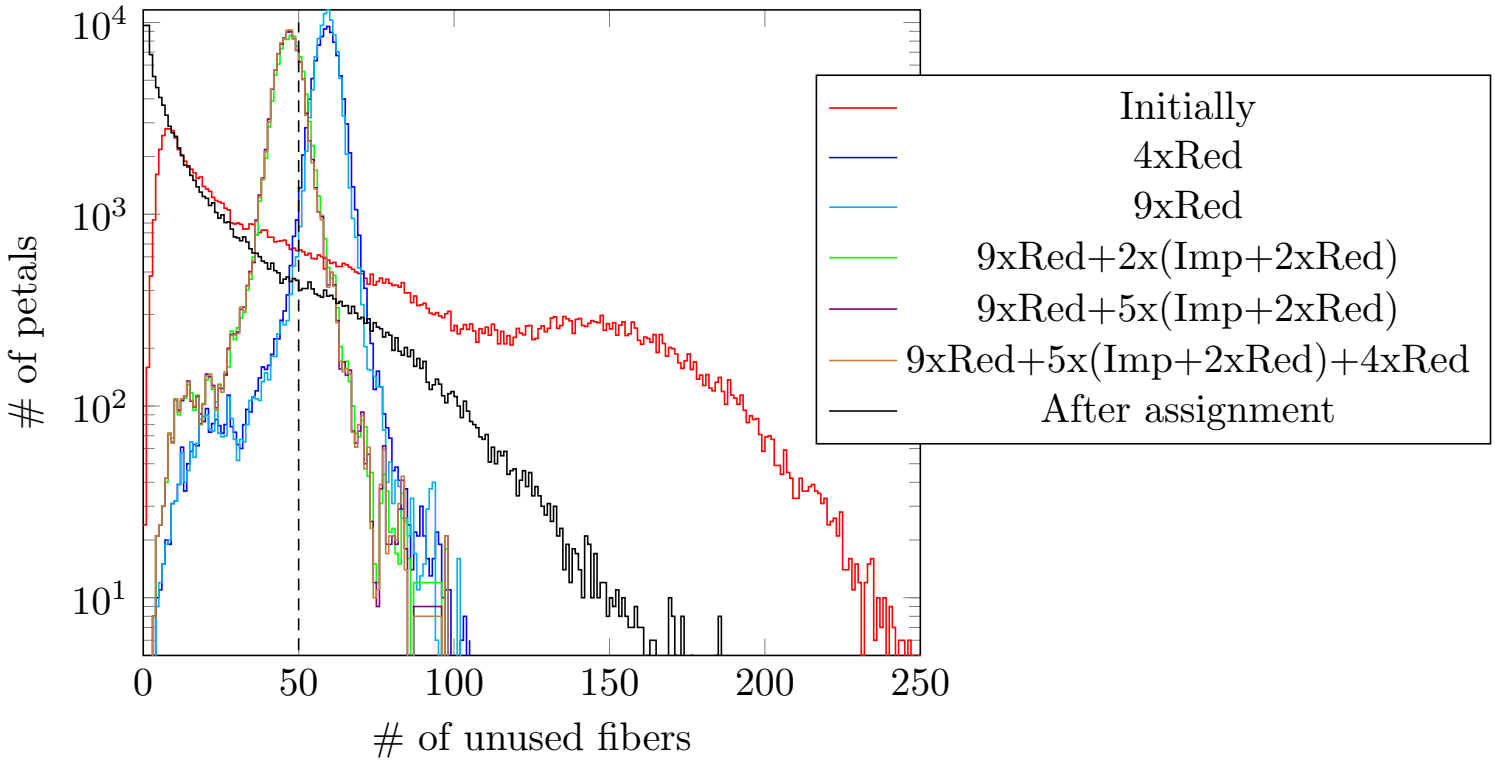


FIGURE 5 – Unused fibers, at different steps, with more improvements

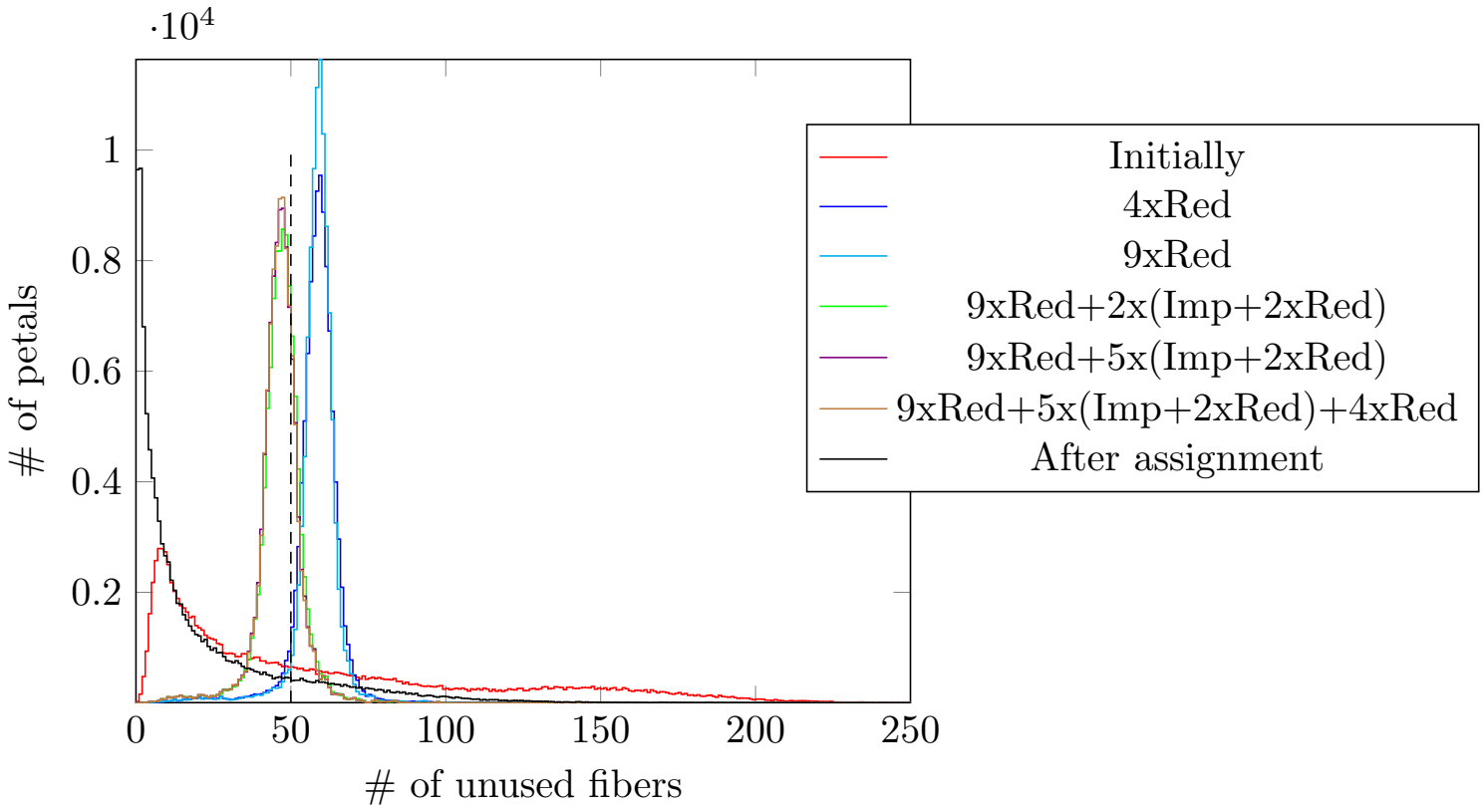


FIGURE 6 – Same, no in log scale

We also do improvement/redistribution process during the observation 5-years time.

8.2 Algorithm, more precisely

Algorithm 8 Assignment of reference in main program

Phase I - Make a plan for all plates

1: Run, globally, on the list plates from 0 to last :
2: "New" assign fibers
3: Redistribute (several times)
4: Improve + 2x Redistribute (several times)
5: Redistribute (several times)

Phase II - Observation time

1: Begining of the study
2: **for** each plate j of the plan, in order **do**
3: Assign_sf_ss
4: Assign_left
5: Possibly "pyplot" the planned observed tile
6: Real observation is here
7: Update information collected on the previously observed Analysis'th past tile with update_plan_from_one_obs
8: If we are in the plate 100, or 300, or 800, or... Redistribute + Improve + Redistribute (on all the plan of remaining tiles)

A standard output display of an execution looks like that (for assignment plan and applying) :

```
   # Read 71,998,144 galaxies from /project/projectdirs/desi/mocks/preliminary/objects_ss_sf0.rdzipn
# Read 10,666 plate centers from /project/projectdirs/desi/software/edison/desimodel/0.3.1/data/footprint/desi-
tiles.par and 5000 fibers from /project/projectdirs/desi/software/edison/desimodel/0.3.1/data/focalplane/fiber-
pos.txt
# Start building HTM tree at 13.8 s
# ... took : 25.5 s
# Begin collecting available galaxies
# ... took : 31.6 s
# Begin computing available tilefibers
# ... took : 1 mn 12.5 s
# Start assignment at : 2 mn 27 s
# Begin new assignment :
50,518,743 assignments on all left next plates
# ... took : 18 mn 38.3 s
# Begin improve :
565,801 more assignments (1.120 % improvement)
# ... took : 36.5 s
# Begin redistribute TF :
1,760,465 redistributions of couples of TF
# ... took : 23.3 s
# Begin improve :
206,948 more assignments (0.405 % improvement)
# ... took : 30.2 s
# Begin redistribute TF :
1,575,337 redistributions of couples of TF
# ... took : 23.1 s
# Begin improve :
92,508 more assignments (0.180 % improvement)
# ... took : 26 s
# Begin real time assignment at 23 mn 24.5 s
- Plate 0 : 550 not as - 3852 unas & 2772 replaced
- Plate 1 : 98 not as - 3784 unas & 2773 replaced
- Plate 2 : 104 not as - 3985 unas & 2844 replaced
- Plate 3 : 121 not as - 3602 unas & 2563 replaced
- Plate 4 : 105 not as - 3773 unas & 2685 replaced
- Plate 5 : 85 not as - 3797 unas & 2853 replaced
- Plate 6 : 145 not as - 3537 unas & 2514 replaced
- Plate 7 : 92 not as - 3964 unas & 2856 replaced
- Plate 8 : 123 not as - 3714 unas & 2578 replaced
- Plate 9 : 143 not as - 3543 unas & 2378 replaced
- Plate 10 : 148 not as - 3528 unas & 2472 replaced
- Plate 11 : 86 not as - 3507 unas & 2885 replaced
# Begin redistribute TF :
1,160,465 redistributions of couples of TF
# ... took : 23.3 s
# Begin improve :
106,948 more assignments (0.205 % improvement)
# ... took : 30.2 s
- Plate 12 : 48 not as - 3128 unas & 2475 replaced
- Plate 13 : 96 not as - 3407 unas & 2485 replaced
...
```

After an observed tile, we print the number of unassigned fibers on it, the number of further objects unassigned because they were analysed as fake, and the number of those tile-fibers that have been successfully reassigned

9 Results

9.1 Results on the input catalog

Here are some statistics on the input galaxies catalog :

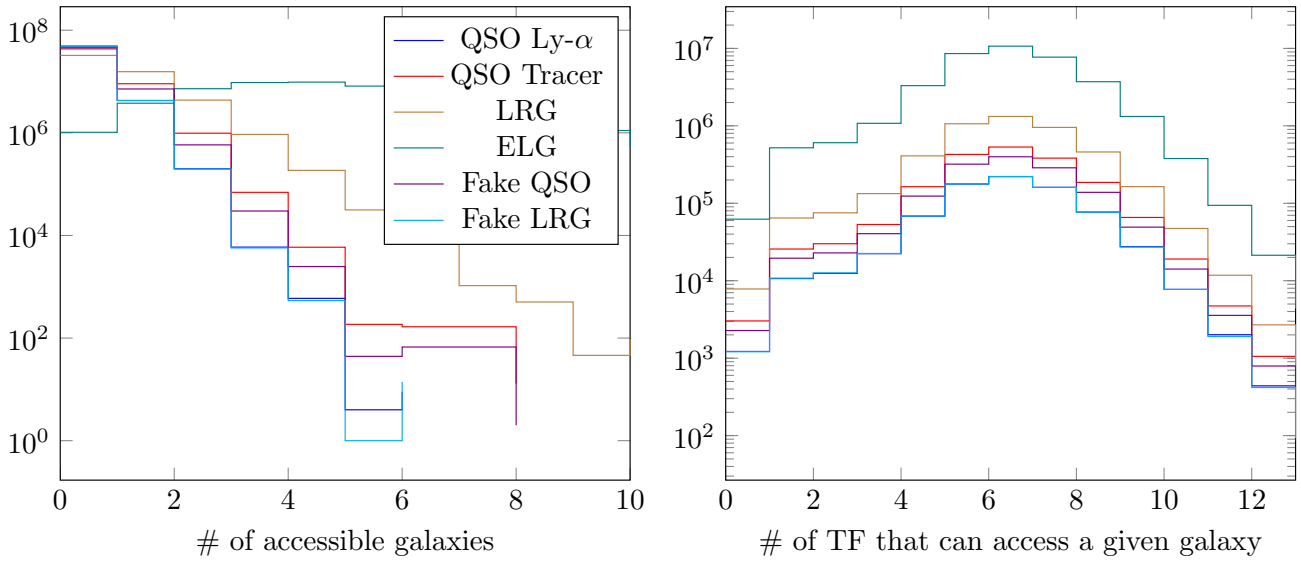


FIGURE 7 – Available galaxies for a fiber on a tile and available tile-fibers for a galaxy (by kind)

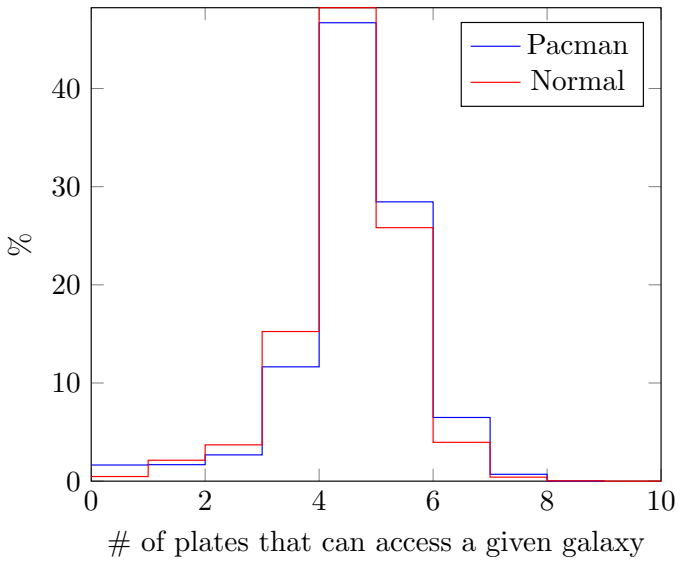


FIGURE 8 – Available plates for a galaxy (without 5th pass)

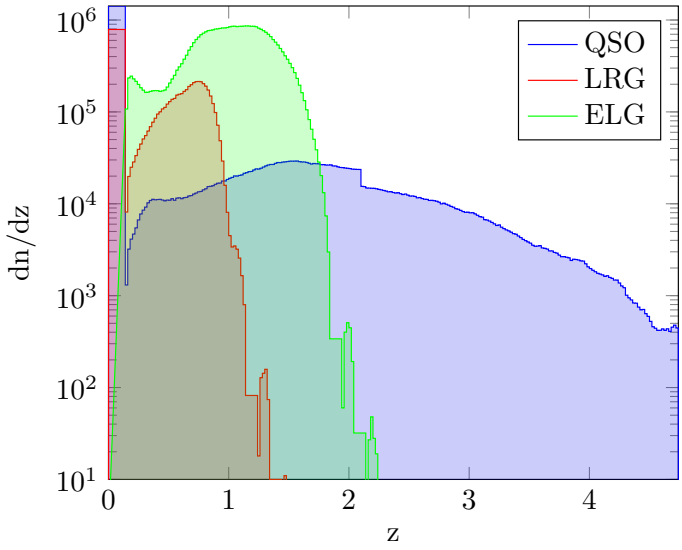


FIGURE 9 – Distribution of objects as a function of redshift

9.2 Results of the assignment

We run the program with all information (prior knowledge of information on fake, target, etc) to compare with realistic simulation, this result is interesting to see the effect of this prior unrealistic knowledge.

We have taken a reference strategy, which was a trade-off between computation time and quality of results. We use improvement functions in a way such that they are still efficient given the time they take (because the more improvement execution we launch, the less they are efficient). And from this reference strategy, we change parameters to see the effects of them on the program.

We tried to run it with an "ideal" infinite density of SS and SF, which only led to an 0.5% inscreasing of the number of observed ELG.

9.3 Sum-up of assigned galaxies

The weighted score of a certain kind is defined as : $Score(kind) = 100 \cdot \frac{\sum_{g \in kind} obs(g)}{\sum_{g \in kind} goal}$ where $obs(g)$ is the number of times g is observed.

A sum-up table of general results on assignment is provided on Table 3.

	Times observed										
	0	1	2	3	4	5	Total	Fiber used	Once observed	observed %	weighted %
QSOLy-a	0	1	5	12	19	10	49	180	49	99.151	72.206
QSOTracer	1	118	0	0	0	0	119	118	118	99.141	99.141
LRG	13	42	243	0	0	0	298	528	285	95.505	88.434
ELG	480	1,930	0	0	0	0	2,411	1,930	1,930	80.054	80.054
FakeQSO	0	89	0	0	0	0	90	89	89	99.139	99.139
FakeLRG	2	47	0	0	0	0	50	47	47	95.792	95.792

TABLE 3 – Densities (objects/deg^2) as as a function of # of observations (with total), and % observed, once and weighted

Furthermore, there are 51,044,452 assignments in total (95.7143 % of all fibers).

9.4 Pacman plate

On the Table 4, we present results of the assignment taking for input the fiber locations files of the pacman (plate deprived of 4 petals) used in case of run out of money. Every parameter is the same. As one can notice, they are quite similar to regular ones.

	Times observed										
	0	1	2	3	4	5	Total	Fiber used	Once observed	observed %	weighted %
QSO Ly-a	1	1	4	14	27	0	49	166	48	97.593	66.818
QSO Tracer	2	117	0	0	0	0	119	117	117	97.575	97.575
LRG	17	33	247	0	0	0	298	528	280	93.987	88.411
ELG	493	1,918	0	0	0	0	2,411	1,918	1,918	79.540	79.540
Fake QSO	2	87	0	0	0	0	90	87	87	97.579	97.579
Fake LRG	2	47	0	0	0	0	50	47	47	94.608	94.608

TABLE 4 – Same than Table 3 but with the pacman

9.5 Free fibers

Here are the histogram of petals as a function of free fibers on Figure **??** and the number of free fibers for each plate, in inscreasing order on Figure 10.

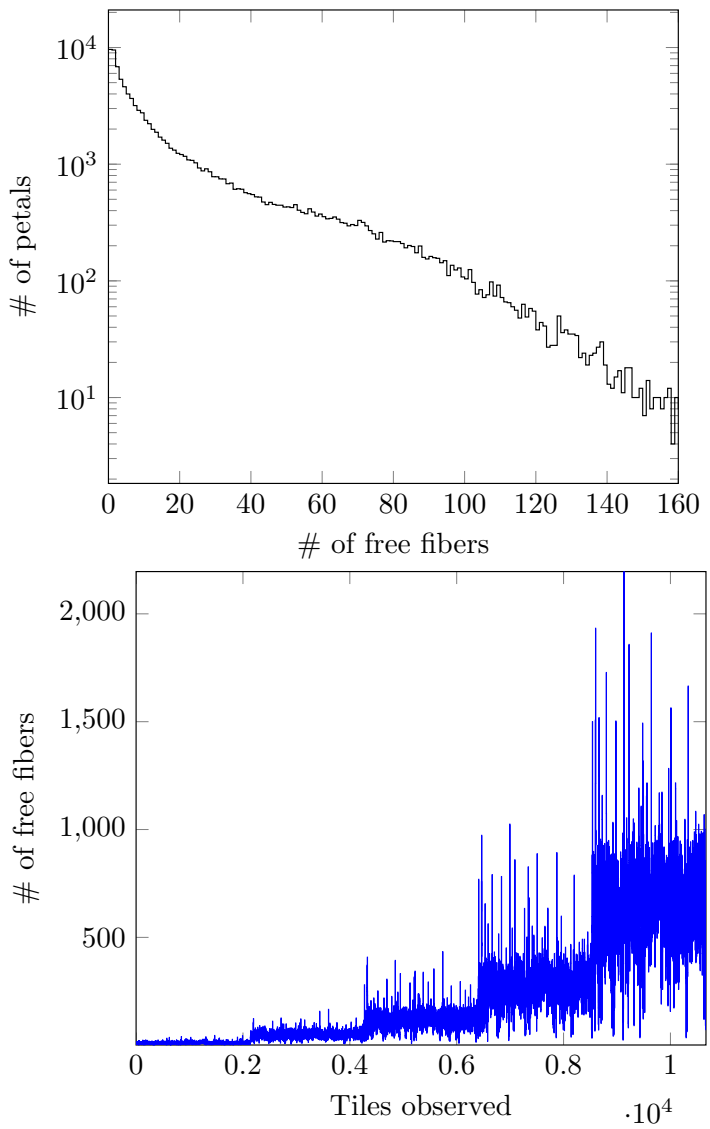


FIGURE 10 – # of petals with that many free fibers and free fibers as a function of time (plates)

On the Figure 11, one can seen the proportion of observed objects as a function of their density. "usq" stands for unity of square degrees, its the area of the sky reachable by a single fiber. Keep Figure **??** in mind while reading this one.

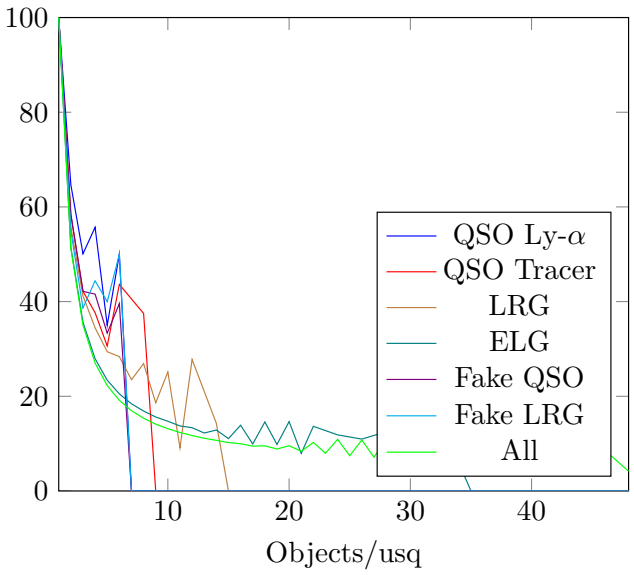


FIGURE 11 – % of observed galaxies as a function of objects density

9.6 Results on Ly-α particularly

The figure 12 gives the histogram of proportion of assigned Ly-α and the number of observations as a function of available TF.

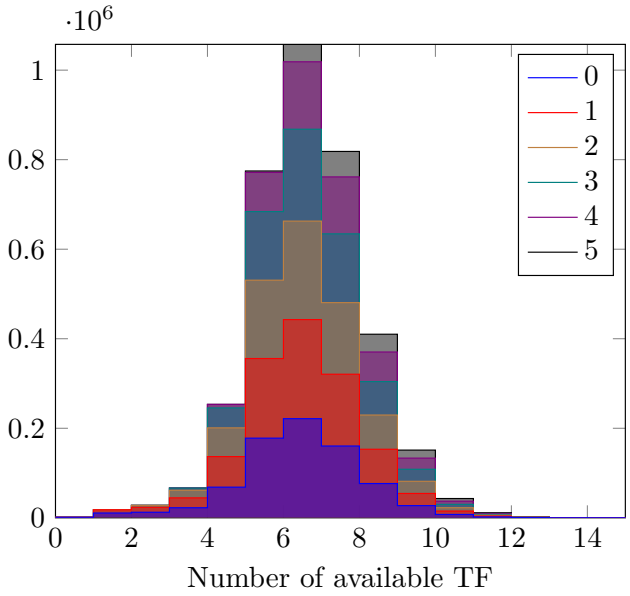


FIGURE 12 – # of QSO Ly-α (with their number of observation) as a function of available tile-fibers

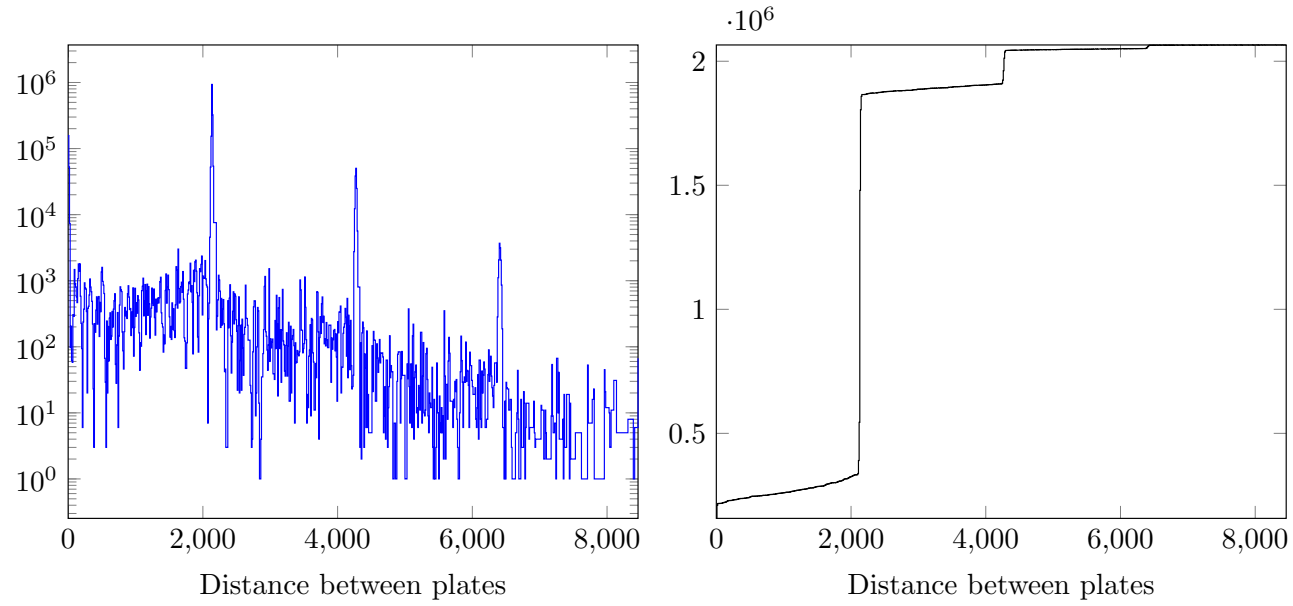


FIGURE 13 – Histogram of distances (in number of plates) between two consecutive observed QSO Ly-α and integral

9.7 Study on evolution over time

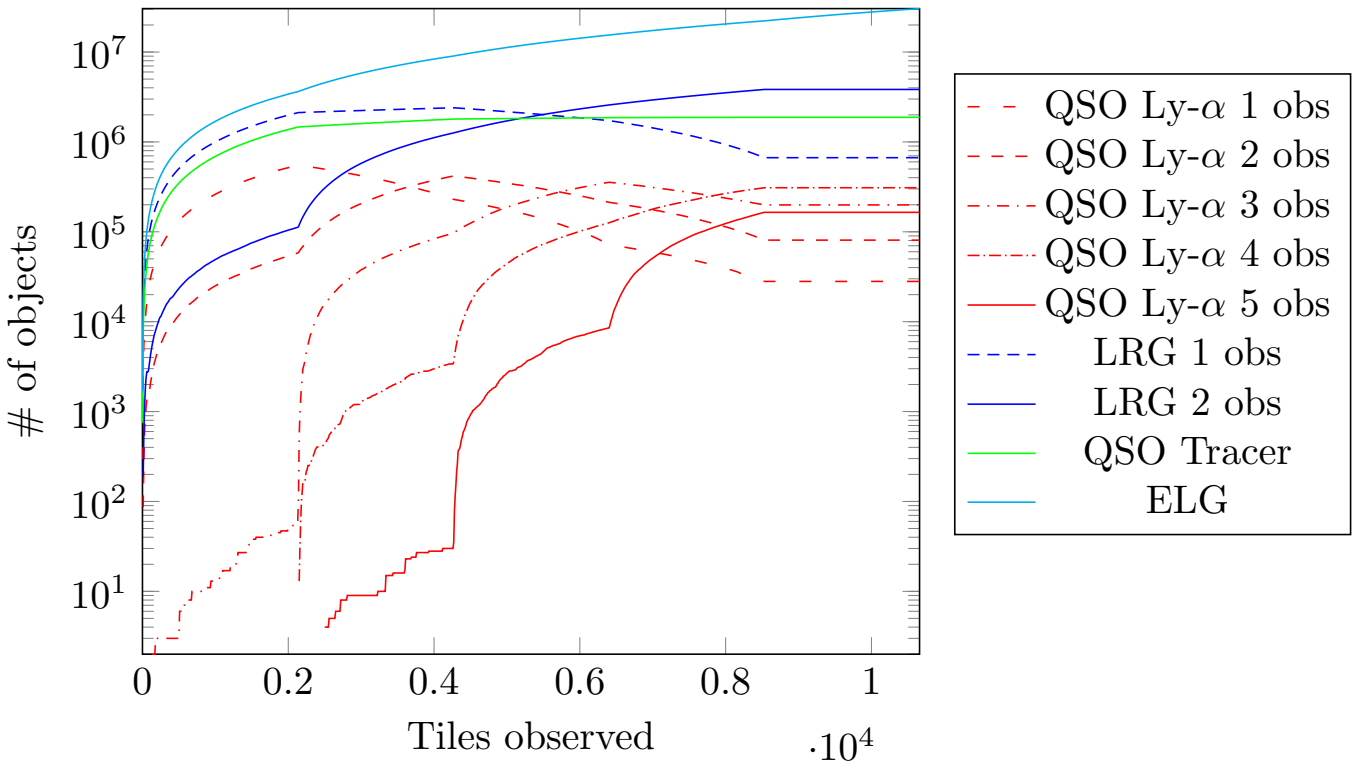


FIGURE 14 – Observed galaxy kind as a function of time (plates seen)

9.8 Choice by density

The assignment could include the idea well described in (**?**) which consists in, when you have the choice of several fibers to observe an object, you choose the one which has the least density of (weighted) remaining available objects in reach. This way, "busy" fibers are more available. Of course, it will introduce a bias in the survey, and since it doesn't improve results (it's likely because redistribution/improvement functions already make this optimization) we can avoid it. An illustration taken from (**?**) is in Figure 15.

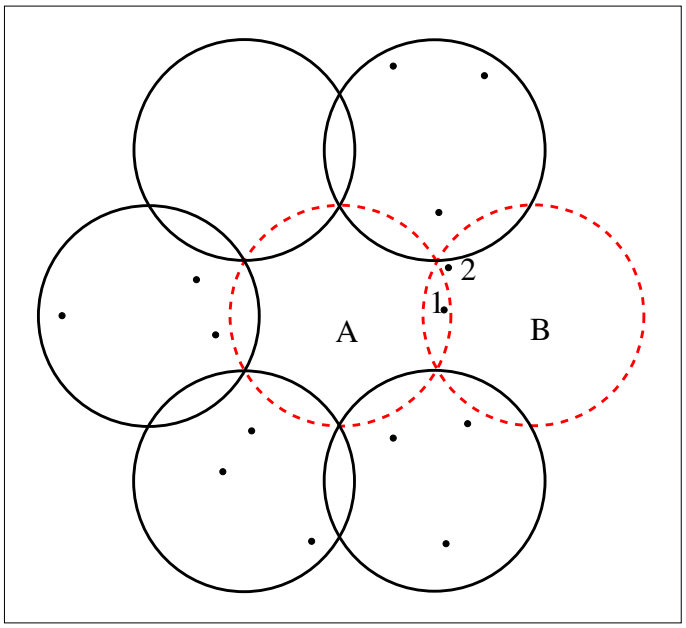


FIGURE 15 – Choice by density : A is chosen to observe 1

9.9 Tiling strategy

The tiling strategy is important and should separate overlapping tiles for more than "Analysis". This way, the assignment can be updated in order to avoid observing twice the same object. The results are then improved in a non negligible rate.

10 Collision problem

Fiber geometry can be found at this address, which is a work of Joe Silber. The exact geometry as built in our program can be seen on Figure 16, which consists in two circles and several segments. The black dot point is the center of the positioner, and other ones are objects in reach.

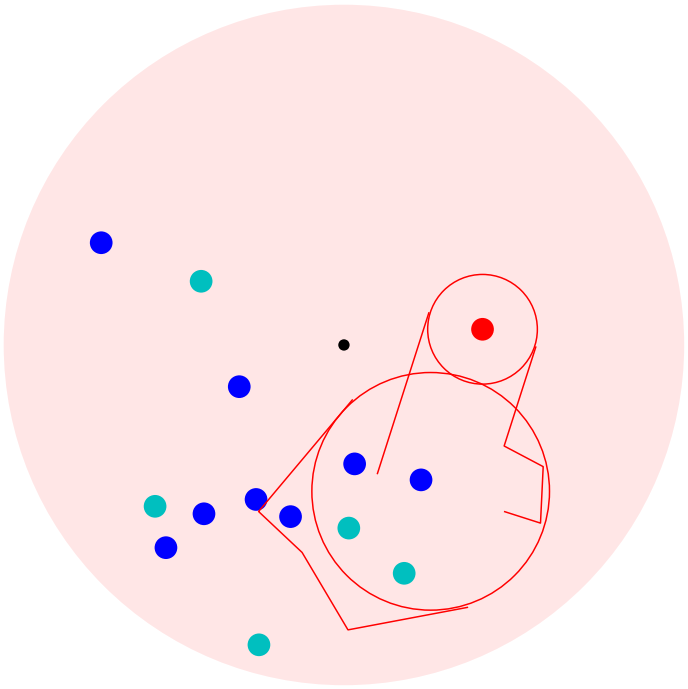


FIGURE 16 – Geometry of a fiber positioner, with fiber holder and central body oriented

In the features file the "Exact" boolean indicates whether we choose the exact geometry of the fiber positioners. If not, collisions are computed with only circles at the position of the fiber holder, of radius $AvCollide$, which we calibrate such that we get the same collision rate than with the exact geometry ($\sim 10.8\%$ of fibers collide). If the exact geometry is on, the modules collision.h and collision.cpp are used to check. They then build sets of circles or segments, and check collisions between those belonging to the first fiber positioner to the ones belonging to the second one. Computationnally cheap methods were used for segment-segment and circle-segment checking, because checking functions are called a lot of time, and optimization of those methods lead to considerably decrease total computation time. That is why we use circle and segments, not only segments : creating an arc with segments needs a lot of them, but only one circle.

Figure 17 is the histogram of distances between two galaxies, in a collision case, and its integral.

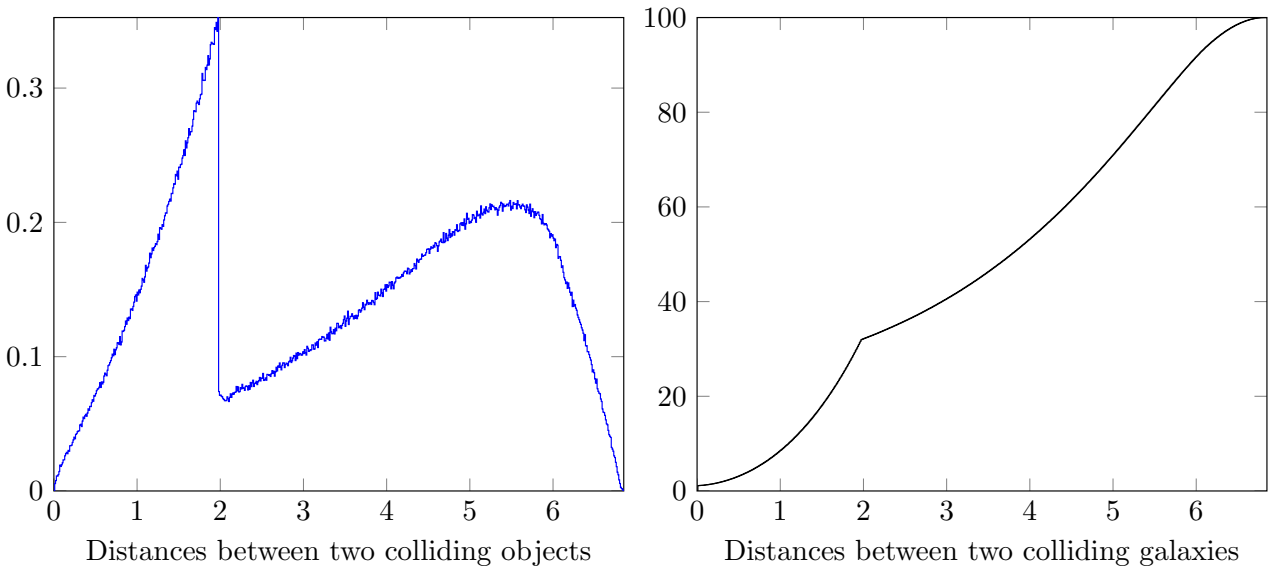


FIGURE 17 – Histogram of distances between galaxies in collision case, and integral

Here is an article on the problem to reach the final position for the positioner without colliding a neighboring one : (**?**)

11 Tile ploting

The function pyplotTile builds a tile.py file, which can be executed manually with python tile.py to create the pdf plot. Only objects that would need at least one further observation are projected on the plate and ploted. The function has to be called just before real observation time, otherwise, if it is called at the end for example, information on number of observations of an object would be the one that we have at the end, and not on the way. A black dot is ploted when there is an unassigned fiber at some place, not to mix up with fiducials (totally white), places where there is physically no fiber. Watching at a dozen of tile plots, one can convince hiself that the assignment is very close to the optimum.

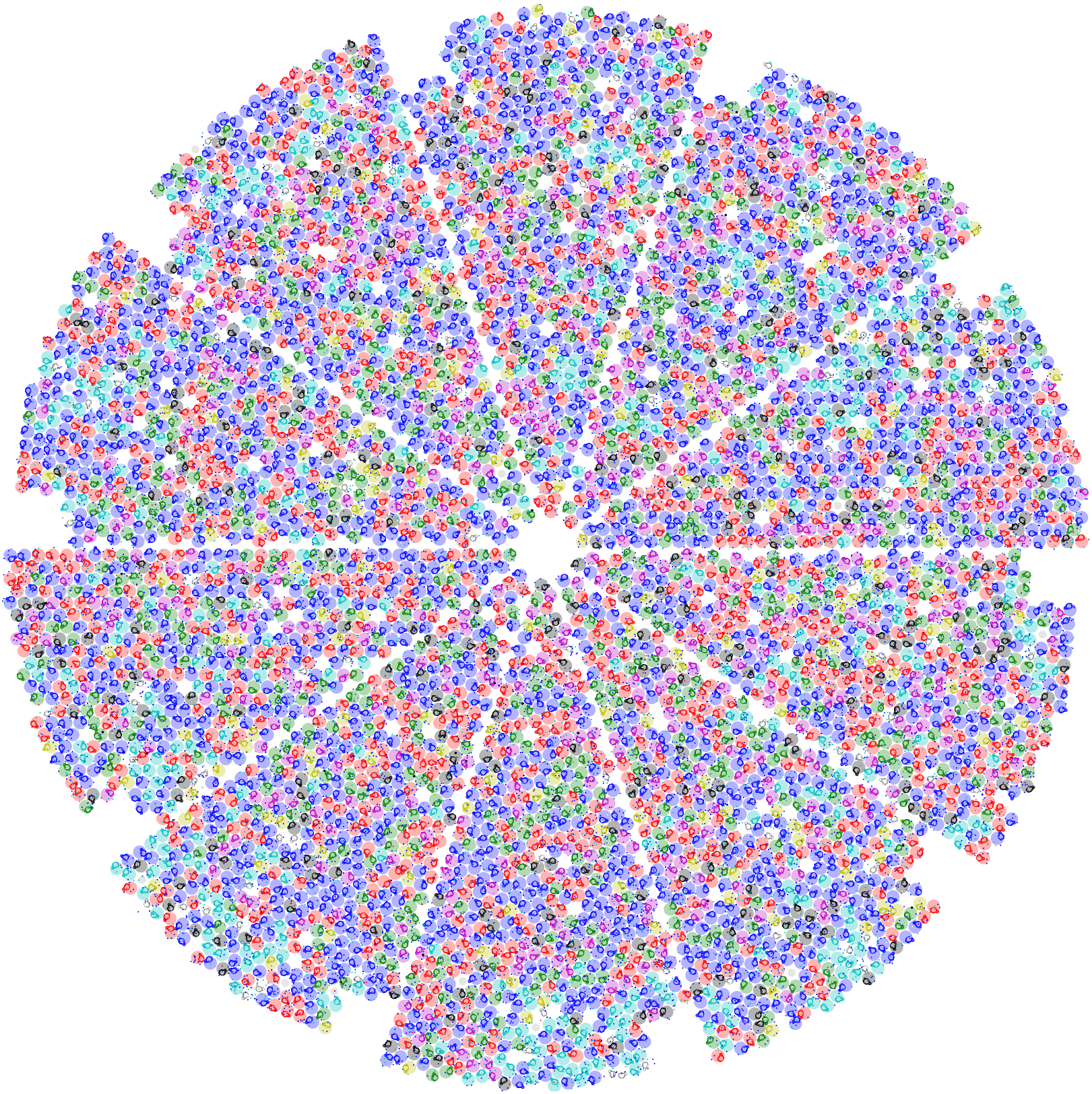


FIGURE 18 – Tile 1000

12 Other functions

In the library, functions are written to put data into .dat files, which can be compiled by the Latex module Tikz to create figures like histograms present here.

There is a function of verification which check several features of the assignment to know if it is sane (no collision, regular mapping, etc...).

13 Possible improvements

One could try to search for an "orthogonal" (of the 3 other ones) improvement function.
Nevertheless, it is likely that we are almost at the global optimum.
Other ideas from (**?**)

14 Effect on correlation function

One of the difficulties is that we have not to bring experimental bias the way we do the assignment. The fact that a lot of rules are used in the assignment make it more likely that a bias is created. We can do a study, comparing correlation functions given with this fiber assignment and ones with all galaxies.

Références

[1] Laleh Makarem, Jean-Paul Kneib, Denis Gillet, Hannes Bleuler, Mohamed Bouri, et al. Collision avoidance in next-generation fiber positioner robotic systems for large survey spectrographs. *Astron.Astrophys.*, 566 :A84, 2014.

[2] I. Morales, A. D. Montero-Dorta, M. Azzaro, F. Prada, J. Sánchez, and S. Becerril. Fiber assignment in next-generation wide-field spectrographs. *MNRAS*, 419 :1187–1196, January 2012.