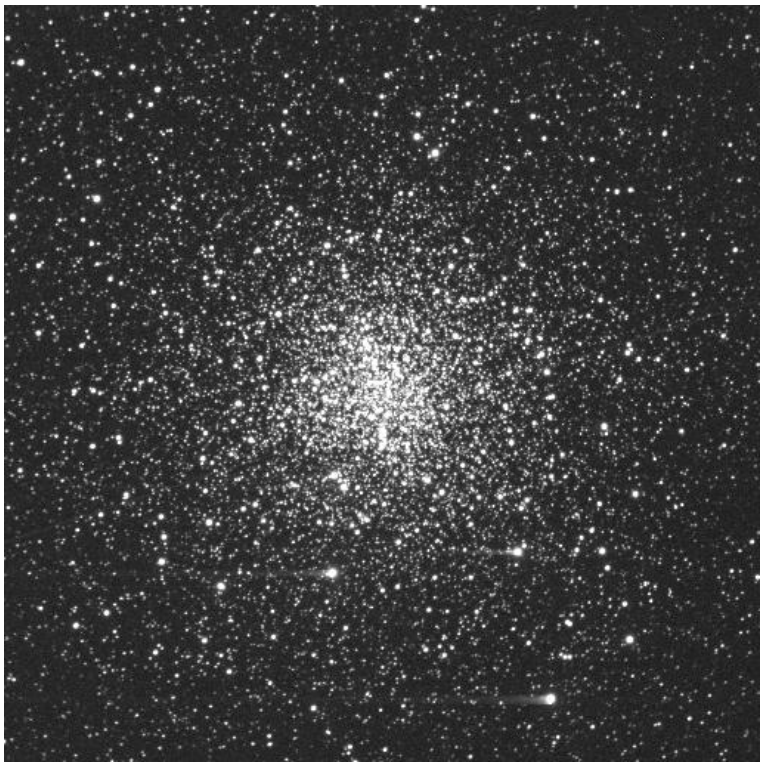
Description of fiber assignment code for Mocks in DESI experiment

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

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

30 avril 2015



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

1 Introduction

Martin White developed C++ code for fiber assignment over the full 14k deg^2footprint. We first modified Martin's code to incorporate features from Bob Cahn's python code, which ran on a restricted 480 deg^2... We included the improvement and redistribution steps, which switch fiber assignments to increase the number of galaxies observed. We then adapted the code to compute assignments not globally "knowing all information on galaxies yet to be observed" but plate after plate, as in the real experiment. The samples are taken from Martin's mocks in 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 $\frac{dn}{dz}$

The code needs to know the locations of the positioners in the focal plane. They are given in : $DESIMODEL/data/focalplane/fiberpos.txt. It also needs to know the locations of the centers of the fields in the sky, i.e. the plates, and their order. The original code written by Martin White provided the option of having the plate centers given in a binary file or an ASCII file. We are now using the ASCII option by defining ASCIICENTERS at the outset. The format is that of $DESIMODEL/data/footprint/desi-tiles.par, but an alternative ASCII file can be provided. If the executable is `assign` then the calling sequence will look like : `./assign galaxies.rdzipn desi-tiles.par fiberpos.txt assignment 1 1` Here the catalog of targets is in the NERSC directory : /projects/projectdirs/desi/mocks/ preliminary/objects_ss_sf0.rdzipn the binary file created by `make_catalog`. An option not now used is to write the actual assignments to a file here called `assignment`. The two last numbers stand respectively for the fraction of galaxies and fibers we take (if one write 2 5 for instance, the program will read only half of the galaxies and one fiber over two in the fiber locations file).

README.rst gives instructions for running on NERSC. We summarize here the various components of the code to facilitate their modification later. Here are some features on input galaxies simulated catalog :

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

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

2 Source files

Source files are file.h and file.cpp and are in this inscreasing dependency order :
- macros : set as global some parameters of the program
- misc : a home-made library of structures (and functions on them) needed to manipulate concerning datas, but independent of them. There are pair (of int), List (of int), Table, Cube, and timing, printing, string conversion, error report items.
- structs : structures of the manipulated datas and their members
- global : main high-level functions and algorithms used in the program to collect information, assign fibers and print statistics. Important ones are described further
- main : neat and quickly understandable code that sum up all steps

3 Parameters

A number of parameters are defined at the begining of the file main.
- parameters summed-up in the Table 1
- Npass = 5 number of passes
- MinUnused = 50 (not used anymore for the moment) minimum number of unused fibers on each petal
- MaxSS = 10 ; MaxSF = 40 maximum number of fibers assigned to SS and SF on a petal

— PlateRadius = 1.65° radius of the plate
— 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)
— Collide = 2.1 mm minimum distance allowed on plate projection of two assigned galaxies on the same plate
— NeighborRad = 11.0 mm maximum distance to consider that two fibers are neighbors
— PatrolRad = 6.0 mm maximum distance, on plate coordinates, that allows a fiber to a galaxy
— InterPlate = 200 minimal number of plates between two observations of the same galaxy
— Randomize = false randomize order of plates in making plans
— Pacman = false selects only spectrometers 0, 1, 2, 7, 8, 9 of the pacman
— Collision = true when we want to allow collisions, to compute the collision rate for instance

4 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
Feat	Features of galaxies	Initialized in the main function. Gives priority and number of observations desired for each galaxy type.
PP	Plate Parameters	Carries locations of fiber positions on the plate, spectrometer correspondence, and neighboring fibers information.
onplate	Plate coordinates	Used for coordinates in the focal plane in mm. The member `id` is used to give the identity of a galaxy. Onplates is vector of onplate.
plate	A plate	Locations in the sky of the tile in terms of a unit 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 has all information on tiles.
galaxy	A galaxy	Information on a galaxy : an `id` that 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, its inverse, galaxies to tile-fibers, and the cube (3d-matrix) of assigned fibers for a kind, a petal and a plate (useful for fast computations).

TABLE 2 – Classes and structures

5 Functions

In algorithms, j stands for a plate, k for a fiber, p for a petal, g for a galaxy.

5.1 Some 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.

`change_coords` 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. This ought to be rigorously checked.

`find_collision` returns the fiber number of a fiber that conflicts with tiblefiber (j,k). Conflict is defined by two observed galaxies being separated by less than `Collide`, currently set to 2.9 mm.

5.2 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, because it's absolutely necessary 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)

5.3 Useful sub-functions for global functions

`ok_assign_g_to_jk_nobs and ok_assign_tot` checks to see if we can assign g to the tile-fiber (j,k), according to assigning rules described further

`find_best (j,k)` finds the best reachable galaxy for this fiber, according to assignment rules

Algorithm 1 Find best(j,k)

```
Initialize a fictional galaxy with ID = −1, called best
for each galaxy available galaxy (for this fiber) g do
    if g is better (according to priority and number of observations) then
        best ← g
```

`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 reassigning 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.

Algorithm 2 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 ≤ jp ≤ j0 + n) do
                    gp ← find_best(jp, kp)
                    memorize jp, kp, g, gp if it's a better set (gp is more worthy) than previous one
        if gp ≠ −1 then
            Unassign (jp, kp) ⟷ g
            Assign (j, k) ⟷ g
            Assign (jp, kp) ⟷ gp
```

5.4 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. Thus, in the code, we have to simulate that we have this infotmation only when we have at least once observed the object.

Thus, algorithms do a plan with only information available by the catalog and by previously seen galaxies. A plan consists of an assignment on a set of consecutive tiles (from 0 to 2000 for instance). One can use algorithms on size-one plans if one wants to do an assignment plate after plate for example (which shouldn't be called a plan anymore).

The argument "next" (integer) in following functions means that we treat all next "next" plates (in the right order of tiles) in the plan we make. If it is -1, it will deal with all left plates. For example, if next plate in the program is 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 3 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_fibers` makes a first assignment plan trying to assign first QSO, then ELG and LRG, then trying to assign SF and SS replacing other kinds if there are not enough available fibers. This way, a QSO can't be lost because of a collision. With the current value of Collision, it's not very useful, but could be for bigger values.

Algorithm 4 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 ELG and LRG
        for each unassigned fiber k of this petal do
            Try assign_fiber(j,k) to only SS or SF
        if Number of SS ≤ 10 then
            Replace ELG (then LRG) to an SS until there are 10 SS
        if Number of SF ≤ 40 then
            Replace ELG (then LRG) to an SF until there are 40 SF
```

`improve` improves the plan applying improve_fiber to all of the fibers that concern it

`improve_from_kind (kind)` for every concerned petal, tries to assign unassigned fibers to 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.

Algorithm 5 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
```

`update_plan_from_one_obs` updates the plan formerly made so that if for example on the previous observation we've watched at a fake QSO, it won't be observed anymore in case it was planed to be observed once or several times again. Released tile-fibers will then be tryed to be reassigned with improve_fiber

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
    for each of those tile-fibers (jp,kp) do
        Unassign (jp, kp) ⟷ g
        run improve_fiber(j0+1,end-j0,jp,kp)
        if jp, kp hasn't been assigned then
            run improve_fiber_from_kind(SF,j0+1,end-j0,jp,kp)
        if jp, kp hasn't been assigned then
            run improve_fiber_from_kind(SS,j0+1,end-j0,jp,kp)
```

5.5 Displaying results

`results_on_inputs` displays some statistics on input files (recall input features, print statistics on the number of fibers with 0 galaxies within reach, 1 galaxy within reach etc. out to 19 galaxies within reach ; number of available tile-fibers for a galaxy, ...)

`display_results` writes some statistics and provides the tex-formatted results to make Table **??**

`print_free_fibers` print histograms on free fibers towards/regarding petals, for everything, for only SS and for only SF

6 Rules of assignment

— Of course, a tile-fiber is only assigned once
— two observations of a galaxy can't be separated by less than `InterPlate` (200) plates
— two observed galaxies can't be separated by less than `Collide` (set by find_collision)
— in last pass, only ELG, SS and SF are considered
— there can't be more than 10 fibers assigned to SS on a petal
— there can't be more than 40 fibers assigned to SF on a 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

In nature we only know the basic type (QSO, LRG, ELG, SS, SF) of the galaxies of the catalog. 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. There are two way of assignment : either observe tile after tile and foresee the assignment of the tile just before observing it ; otherwise, we can make a plan for a definite number of succeeding tiles. In such 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. In the code, a difficulty was optimization at this point : we could have replanned everything in the plan after an update of information, but we can't because that would take too much time. So we only try to reassign released tile-fibers.

The idea is to get most information possible in the first pass, assigning independently tile after tile, and then, after this first pass, make a plan for everything else, when we have information on most of the QSO and LRG (the only kinds we want to observe several times). The plan will be better that way, because we do less mistakes than if we would do it from the begining.

7 Algorithm

The calling sequence is `assign galaxy plate_centers fiberpos assignment 1 1`. The last of these is the file to which we would write all the final galaxy assignments, but this is generally suppressed since we aren't using it yet.

`main` proceeds by taking important parameters, features of galaxies, reading in the galaxies to make G, the positions of the fibers on the plate to make pp, the plate centers and positioner locations to make P.

We then collect available galaxies for each fiber for each plate, and compute the inverse map.

We next, tile after tile, do the assignment of galaxies. We have tried several strategies : making several plans, use simple assignment, use complexe one, improve several times, in several ways, etc... Here is the one which gives the best results.

We present the strategy that produced the best results, which will be the strategy of reference, We tried a strategy where we make then apply a plan for the first 2000, then for the following 2000, then for the left, but it's not better.

Simple assign fibers and new assign, even if there structure are quite different, give almost the same results. Simple assign is a little bit quicker so we will use this one.

Algorithm 7 Assignment of reference in main program

Phase I - Make a plan for the first 2000 plates

1: Run, globally, on the list plates from 0 to 1999 :
2: Assign fibers
3: Improve SF

Phase II - Applying the plan

1: **for** each plate j of the plan, in order **do**
2: Real observation is here
3: Update information collected on previous just watched tile
4: Update_plan_from_one_observation(j)

Phase III - Make a plan for all left plates

1: Run, globally, on the list plates from 2000 to the end :
2: Assign fibers
3: Improve
4: Improve SF
5: Improve SS
6: Redistribute
7: Improve
8: Redistribute
9: Improve

Phase IV - Applying the plan

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

```
- Plate 1998 : 9 not as - 0 unas & 0 replaced
- Plate 1999 : 8 not as - 0 unas & 0 replaced
# ... took : 10.5 s
# Begin simple assignment :
37,635,165 assignments on all left next plates
# ... took : 2 mn 17.3 s
# Begin improve :
921,594 more assignments (2.449 % improvement)
# ... took : 42.1 s
# Begin improve SF :
1,386,726 more assignments (3.597 % improvement)
# ... took : 3 mn 13.8 s
# Begin improve SS :
194,603 more assignments (0.487 % improvement)
# ... took : 6.99 s
# Begin redistribute TF :
2,202,483 redistributions of couples of TF
# ... took : 16.8 s
# Begin improve :
167,146 more assignments (0.416 % improvement)
# ... took : 23.3 s
# Begin redistribute TF :
2,058,702 redistributions of couples of TF
# ... took : 16.8 s
# Begin improve :
76,484 more assignments (0.190 % improvement)
# ... took : 22.1 s
# Begin real time assignment at 11 mn 34.7 s
- Plate 2000 : SF-imp + 38 (+0.769 %) SS-imp + 1 (+0.020 %) 22 not as - 2630 unas & 1985 replaced
- Plate 2001 : SF-imp + 37 (+0.750 %) SS-imp + 1 (+0.020 %) 29 not as - 3227 unas & 2619 replaced
- Plate 2002 : SF-imp + 34 (+0.691 %) SS-imp + 2 (+0.040 %) 44 not as - 3296 unas & 2632 replaced
- Plate 2003 : SF-imp + 13 (+0.262 %) SS-imp + 1 (+0.020 %) 17 not as - 2797 unas & 2526 replaced
```

8 Results

8.1 Results on the input catalog

Here are some statistics on the input galaxies catalog :

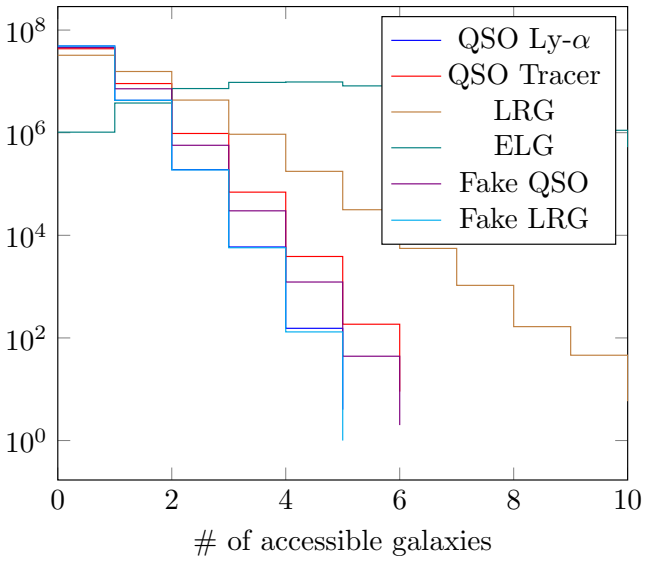


FIGURE 1 – Available galaxies (by kind) for a single fiber on a single tile

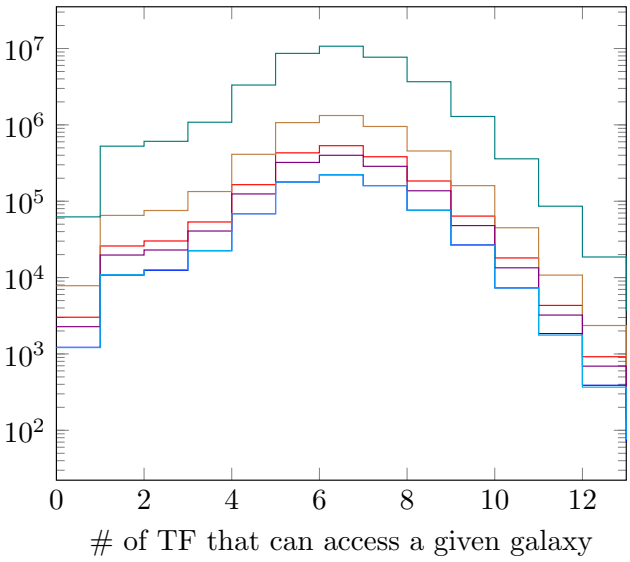


FIGURE 2 – Available tile-fibers for a galaxy (by kind)

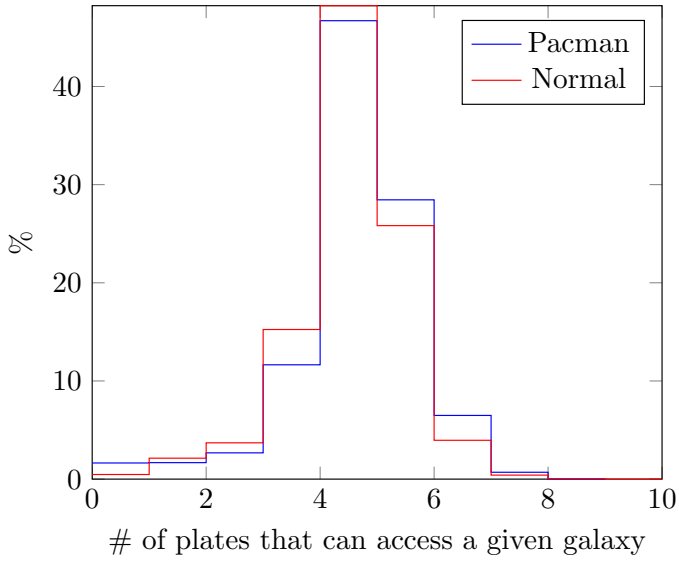


FIGURE 3 – Available plates for a galaxy (without 5th pass)

8.2 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.

The results are then written using `display_results`, and each assignment/improvement function display its own statistics.

We precise that we have indeed exactly 10 SS and 40 SF per petal. 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 efficent). And from this reference strategy, we change parameters to see the effects of them on the program. We also compare to other strategies.

The best strategy (discovered until now) is the one where we first look at 2000 tiles, tile after tile runing simple_assign on each tile without plan, and we then do a plan for all left tiles, before improving it only with an SF-improve.

In the second plan, we can experimentally see that after a first SF-improvement ($+ \sim 3.6\%$), even if we redistribute, a second one is almost useless ($+ \sim 0.1\%$), whereas a second simple-improve is still efficient after a redistribution ($+ \sim 0.4\%$), but useless without a redistribution step ($+ \sim 0.01\%$). An other kind of kind-improvement (instead of SF and SS ($+ \sim 0.5\%$)) is also almost useless. Furthermore, the computation time is far bigger for SF-improvement (5 mn) than for other.

The most results-sensible feature in it is the number of plan/applying and their plates sets. One could try to find a better one.

8.3 Sum-up of assigned galaxies

In the reference strategy, we have 50,095,513 assignments in total (93.54% of all fibers). 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 %
QSO Ly-a	0	2	4	14	25	1	49	167	49	98.434	67.032
QSO Tracer	1	118	0	0	0	0	119	118	118	98.454	98.454
LRG	16	37	244	0	0	0	298	526	281	94.376	88.069
ELG	537	1,874	0	0	0	0	2,411	1,874	1,874	77.721	77.721
Fake QSO	1	88	0	0	0	0	90	88	88	98.446	98.446
Fake LRG	2	47	0	0	0	0	50	47	47	95.063	95.063
SS	72	67	0	0	0	0	140	67	67	48.252	48.252
SF	1,129	270	0	0	0	0	1,400	270	270	19.300	19.300

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

8.4 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	28	0	49	169	48	97.889	67.703
QSO Tracer	2	117	0	0	0	0	119	117	117	97.864	97.864
LRG	16	31	250	0	0	0	298	532	282	94.41	89.205
ELG	487	1,924	0	0	0	0	2,411	1,924	1,924	79.794	79.794
Fake QSO	1	88	0	0	0	0	90	88	88	97.856	97.856
Fake LRG	2	47	0	0	0	0	50	47	47	94.977	94.977
SS	70	69	0	0	0	0	140	69	69	49.576	49.576
SF	1,122	277	0	0	0	0	1,400	277	277	19.830	19.830

TABLE 4 – Same than Table 3 but with the pacman (old version with collide = 2.1)

Furthermore, there are 50,938,132 assignments in total (92.9358 % of all fibers).

8.5 Free fibers

Here are the histogram of petals as a function of free fibers on Figure 4 and the number of free fibers for each plate, in inscreasing order on Figure 5.

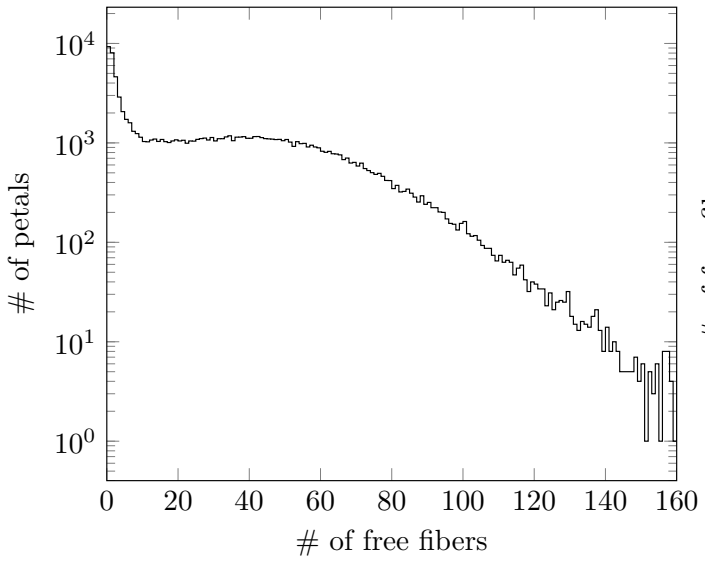


FIGURE 4 – # of petals with that many free fibers

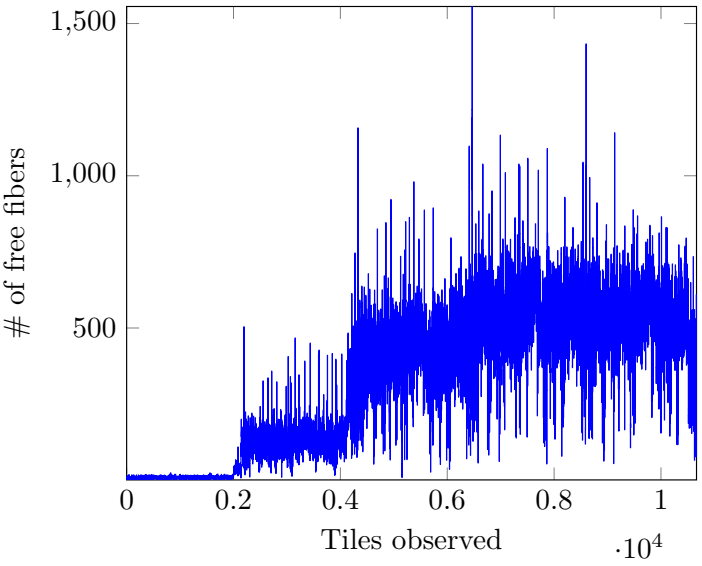


FIGURE 5 – Free fibers as a function of time (plates)

On the Figure 6, 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 1 in mind while reading this one.

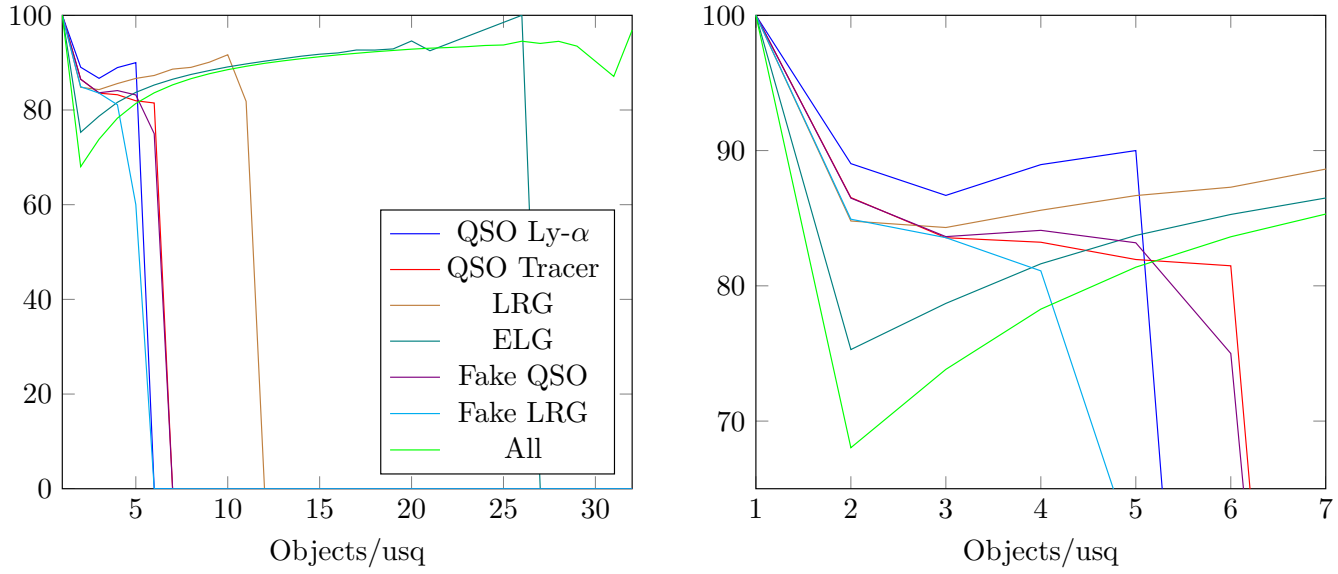


FIGURE 6 – % of observed galaxies as a function of objects density, wide and zoom

8.6 Redistribution-Improvement step

We see, in the second plan, what is the effect of several redistribution-improvement executions. Here is presented, in the Table 5, the number of redistributions, number of additionnal assignments, with the improvement in % and the total time taken for until this step.

Step	# red	# +as	+ %	Time
1	2,204,977	167,741	.418	40s
2	2,060,851	76,509	.190	1mn 20s
3	1,965,994	19,273	0.048	2mn
4	1,945,418	10,738	0.027	2mn 40s
5	1,929,917	7,999	0.020	3mn 20s
6	1,921,331	5,745	0.014	4mn

TABLE 5 – Results of several redistribution-improvement steps

8.7 Results on Ly-α particularly

The figure 7 gives the histogram of proportion of assigned Ly-α and the number of observations as a function of available TF.

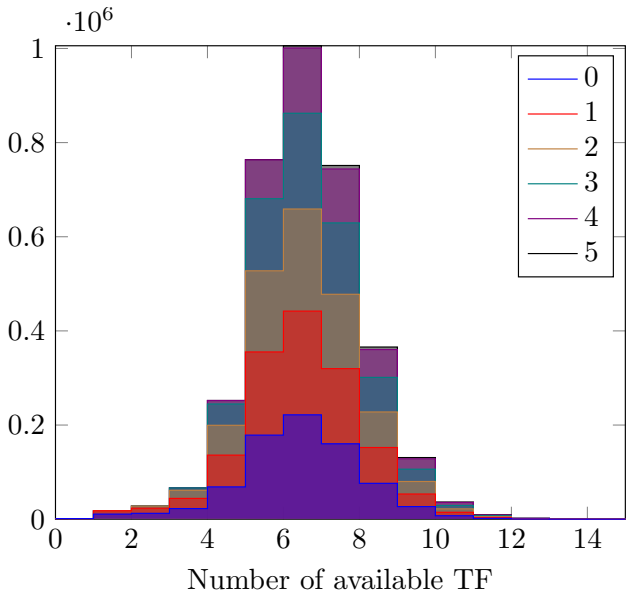


FIGURE 7 – # of QSO Ly-α (with their number of observation) as a function of available tile-fibers

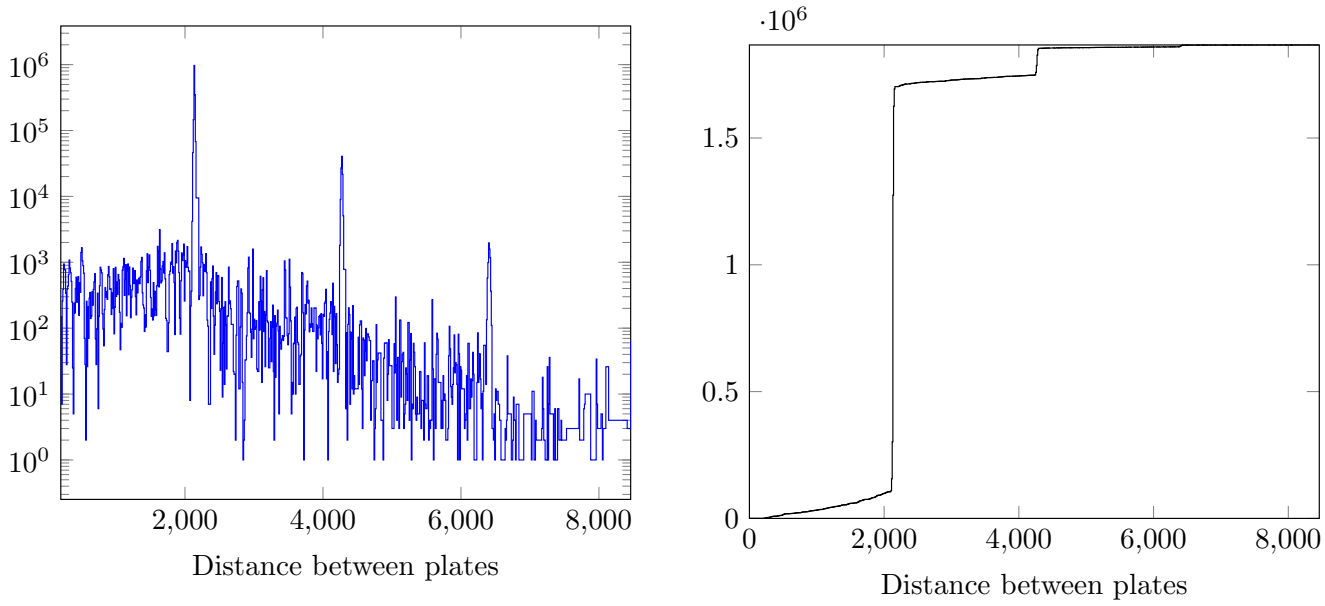


FIGURE 8 – Histogram of distances (in number of plates) between two consecutive observed QSO Ly-α and integral

8.8 Study on evolution over time

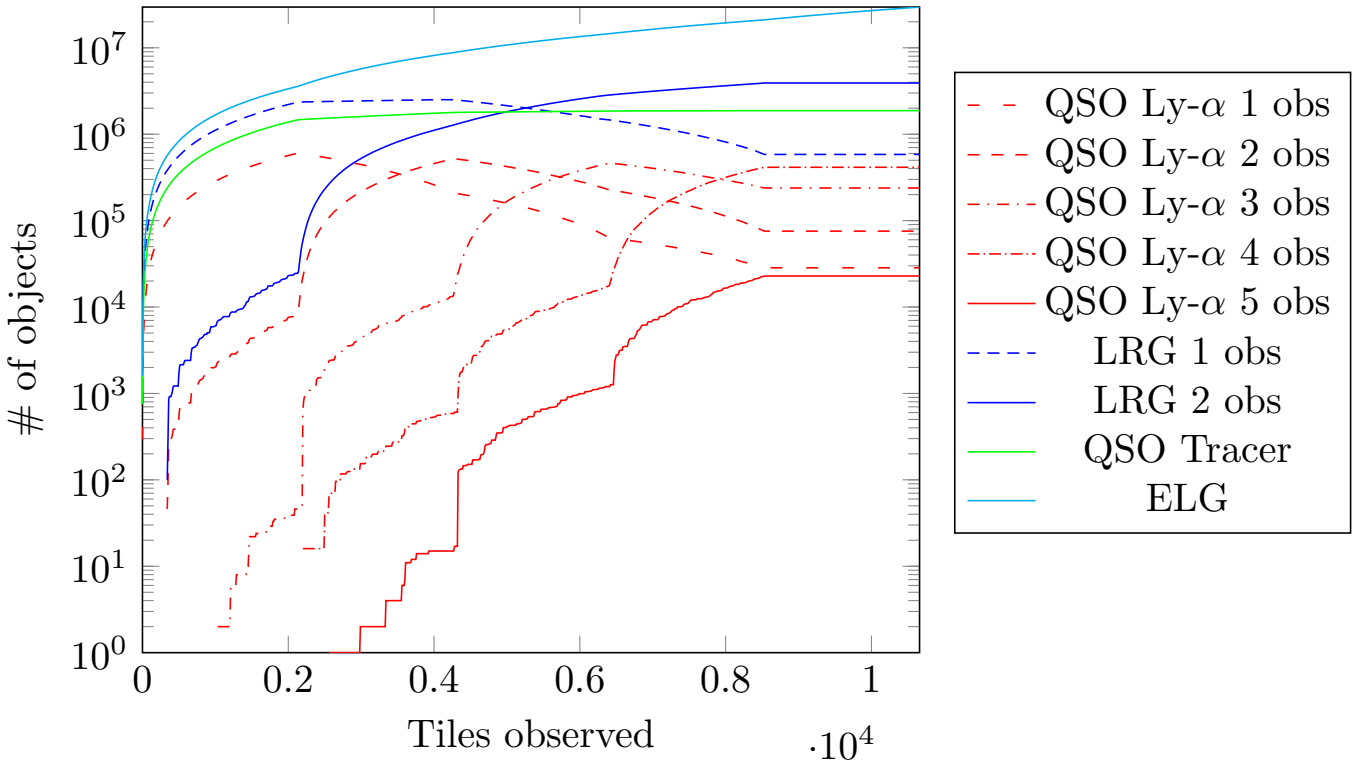


FIGURE 9 – Observed galaxy kind as a function of time (plates seen)

8.9 Shifting parameters

In the following, we change a little bit parameters around the references values to observe their influences on results.

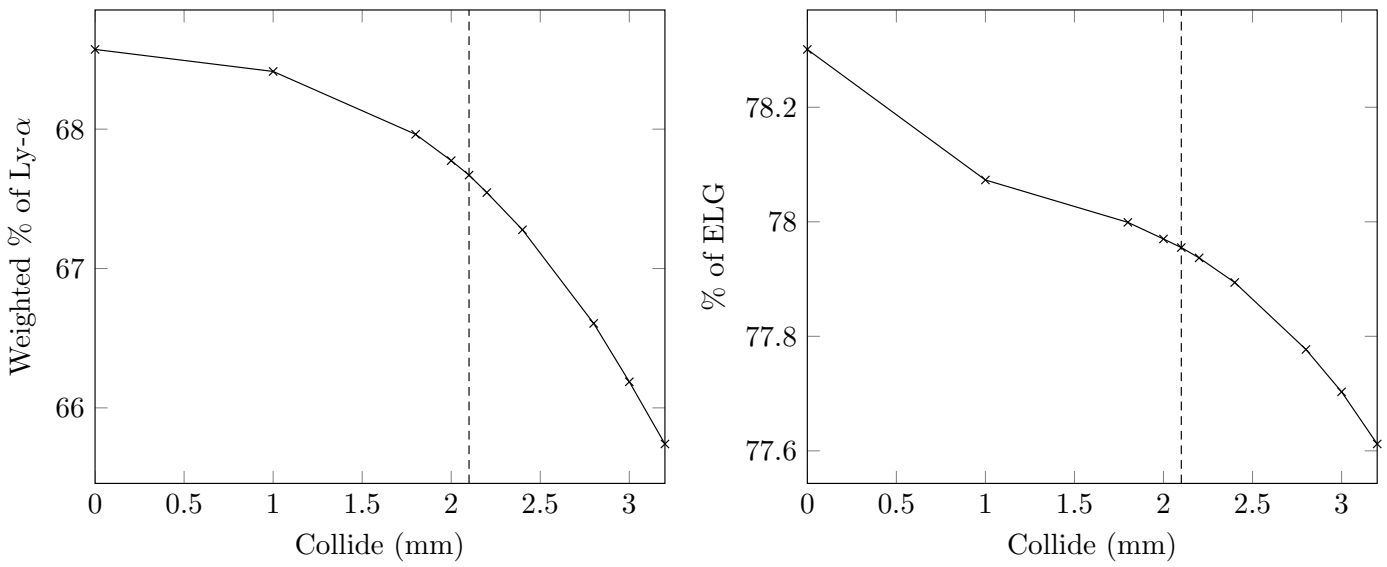


FIGURE 10 – Weighted % of Ly-α and % of ELG observed as a function of required stay-clear

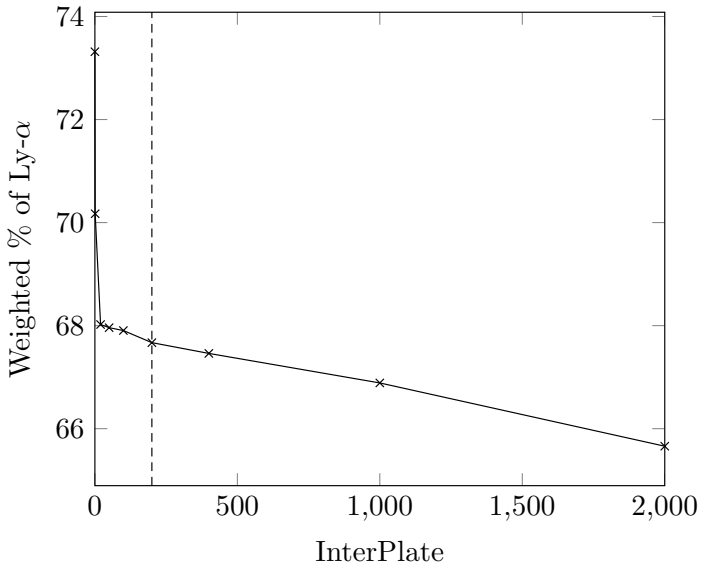


FIGURE 11 – % of Ly-α observed as a function of required minimum interval between plates observing the Ly-α

9 Possible improvements

We havn't tried all possible strategies, so it is possible that an other combination of runing of the functions could lead to better results. One could also try to search for an "orthogonal" (of the 3 other ones) improvement function. To make plans, we could use an automaton, which would be automatically able to use all kinds of assigment/improvement functions, and would go through all fibers at random to assign/improve randomly, and the controlable parameter would only be the time we want it to work. Nevertheless, a lot of efforts now lead to little improvements of the results. It is likely that we are almost at the global optimum. We only have a doubt on the number of plans/applying and their sets of plates that could improve significantly.

The problem of colliding fiber positioners is addressed in a simplified way (circles around a fiber) which can be precised.

We can adapt a little bit the code in order to do assignment and improvements without SS and SF, and only add SS and SF just before launching the observation, using the "replace" function in global.cpp. It's not likely to improve a lot because improve-SS and SF must already improve this way.

The update function might still be improved to lead to more reassignments, and an improvement execution on the current plan could be efficient after several hundred updatings, since some "redistribution" happened with updatings.

We could make plans with only Ly-α assigned, launching improve function, and then adding separately LRG then improving, and adding again ELG and improving. It's likely that it won't increase results because as we saw before, doing assignments separately in new_assign already didn't improved results. This kind of optimization must actually be already done by selecting with priorities and executing improvement functions.

10 Collision problem

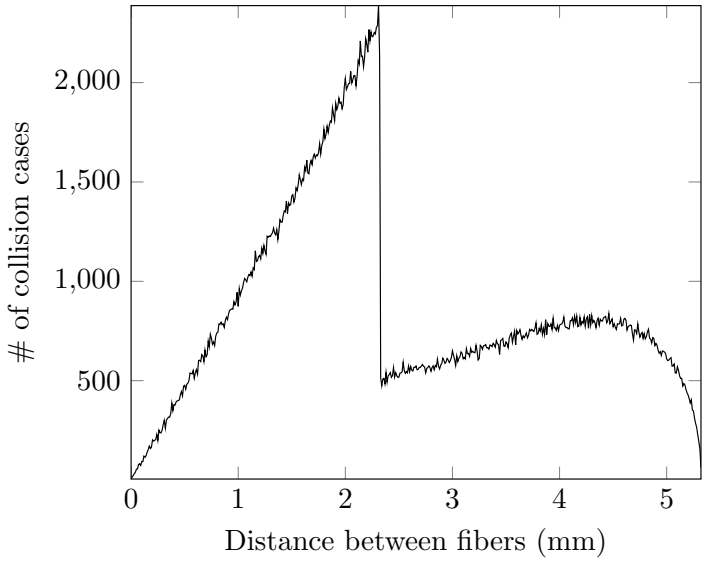


FIGURE 12 – Histogram of cases of collision

We will forbid distance between two galaxies to be less than 2 times the radius of what we will further call the first disk (FD), which is the disk of radius 0.990 mm that incorporates the hole for fiber ferrule. It is sure that it is a necessary condition (simple to convince oneself) and will avoid a good proportion of collisions (40% ?) using a very cheap computation. There is also 3-collisions problem possible, even with this constrain. 4-collision problem is possible without this constrain, but maybe impossible with the constrain.

At the moment of the study, we need to have a good precision of the final assignment efficacity. Joe Silber wrote a collision code with the exact geometry of positionners, and found out that the collision rate is 8%. To take it into account, we calibrate the *collide* parameter to get this same collision rate. Of course, this is not exact and eventually needs to be so, but by this way we get results very close to the final ones. We experimentally found out that we have to take $collide = 2.9mm$. Notice that with collisions allowed, we get 94.4 % of fibers assigned, instead of 93.9 % when they are not. A further improvement can be to do the assignment with collisions allowed and at the very end try to change the colliding ones, but it's likely to lead to the same results than now.

Collide (mm)	2.1	3.0	2.9
Collision rate at 2000 (%)	3.58	8.44	7.75
Collision rate (%)	3.93		8.27

TABLE 6 – Collision rate as a function of Collide

Note here that the collision rate is less when we only take the first 2000 plates, though on those plates more fibers are assigned, so one should get a greater percentage than when we take all the plates. Still to understand.

Références