Draft version August 9, 2017
Preprint typeset using LaTeX style AASTeX6 v. 1.0

AN OBJECT ORIENTED FEATURE-BASED SCHEDULER

Peter Yoachim, Elahesadat Naghib

1. FEATURES

Features form the foundation of the scheduler. Features can take many forms, but most commonly they are HEALpixel maps that describe the current state of the sky and telescope or progress of the telescope. Some example features include:

- A map of the number of completed observations
- A map of the desired number of observations at each point in the sky
- The currently loaded filter
- The currently observable HEALpixels
- The sky brightness at each HEALpixel in each filter

Example feature maps are show in Figure 1

2. BASIS FUNCTIONS

Basis functions are built by combining features and reflect the relative reward of observing different regions of the sky.

3. REWARD AND DECISION FUNCTIONS

The final reward function (also commonly called a "cost function") is calculated as a linear combination of the basis functions. The free parameters of the scheduler are thus the weights assigned to each basis function. Once the final reward function is computed, a decision function selects observation(s) to attempt. A simple greedy algorithm would select the position with the highest reward. One can also smooth the map and select a large region to observe. Because most features change on long timescales, the resulting reward function should also be slowly varying, allowing the decision function to queue up several observations.

4. PERFORMANCE FUNCTION AND OPTIMIZATION

A performance function can be written that analyzes the performance of the survey, e.g., combining the total number of visits, fraction of recovered NEOs, and number of good supernova lightcurves into a single scaler. The scheduler free parameters can then be optimized to maximize the performance function.

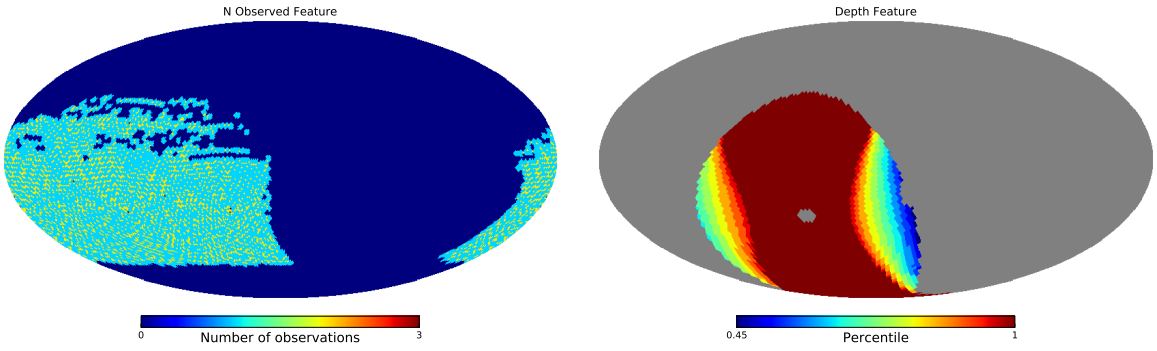

Figure 1. Example features from a simple 5-day simulation. Left show the number of times each HEALpixel has been observed. Right shows the current five-sigma limiting depth percentile with masks for zenith, high airmass, and near the moon.

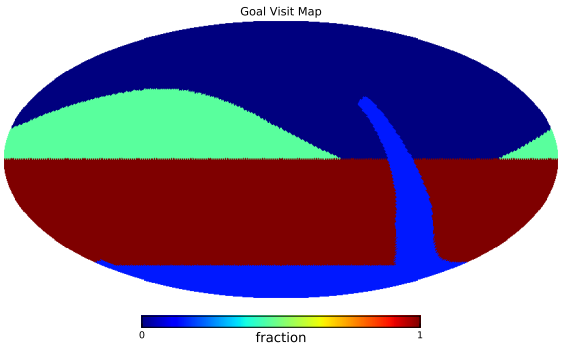

Figure 2. A typical goal map for how LSST observations should be distributed.

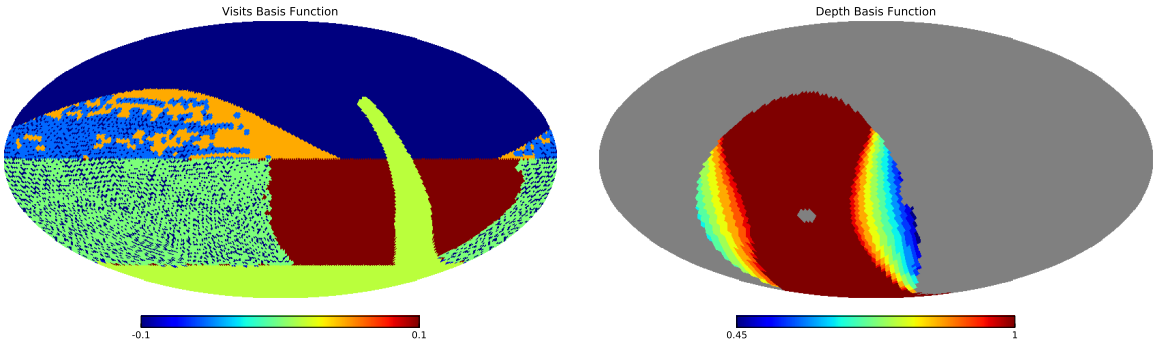

Figure 3. The basis functions for the simulation after 5 days. Left shows how the number of observations made is combined with the goal map to generate a basis function that prioritizes under-observed regions and penalizes over-observed areas. Right shows the depth basis function, which is the unmodified depth percentile feature.

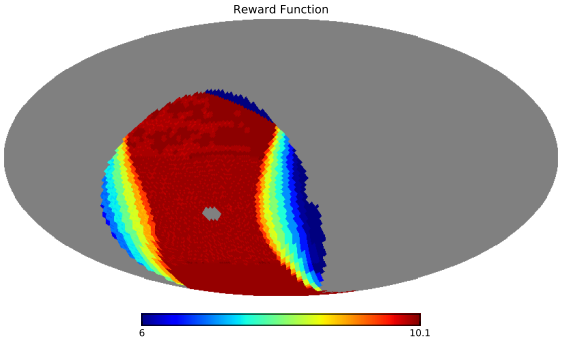

Figure 4. The final reward function generated by a linear combination of the basis functions in Figure 3. In this particular simulation we gave the observing history a weight of 1 and the depth basis function a weight of 10.

5. DISCUSSION

In the sims_featureScheduler package, we have implemented features and basis functions as python classes. We have then added a survey class which takes a list of basis functions and weights to compute a reward function. Each filter can then be it's own survey.

Some advantages of a feature-based scheduler:

- Object oriented nature of the code makes it possible to include other arbitrary schedulers (e.g., pre-scheduled deep drilling field observations).
- Users have direct access to the reward function, making small or large changes to the observing strategy possible.
- Easy to develop and debug new observing strategies as all the inputs to the reward function can be easily accessed

and visualized.

- High spatial resolution features make it possible to easily include dithering experiments in the scheduling algorithm.
- By using high spatial resolution, the feature-based scheduler can incorporate any arbitrary camera geometry, such as a single raft commissioning camera.
- HEALpixel based features make it easy to compute the cost function efficiently with numpy
- All our data structures are picklable, so we can use python multiproccessing to get a factor of 6 speedup computing reward functions by running each filter on it's own processor.