(If there are formatting issues on GitHub, I recommend viewing this document in Obsidian, or reading the identical PDF version instead.)

Program Usage

Creating New Tests

To get started, navigate to `./tests` and make a new directory with the name of your test case (no spaces). Then navigate back to the root and open `tester.py`, and add your new test case function. The following are to be supplied as return values:

1. `N`, the number of galaxies to be selected from.
2. `K`, the total number of fibers available per exposure.
3. `L`, the total number of exposures available.
4. `T_exp`, the duration of 1 per exposure.
5. `u_max`, an `ndarray` of size `(N,)` containing maximum utilities of observing a galaxy.
6. `T_target`, an `ndarray` of size `(N,)` containing the required target times to

To view the distributions of `u_max` and `T_target` generated, execute `python3 tester.py testname`. The histograms will be located under `tests/testname/hist_umax.png` and `tests/testname/hist_target.png.` #todo

Running the Optimizer

To run the optimizer, execute `python3 optimizer.py testname start stop` where `start` and `stop` are optional. (If you supply just one value, it will be interpreted as `start).

- If you do not supply `start` or `stop`, the optimizer will run all exposures `[1, L]`.
- If you supply `start`, the optimizer will run exposures `[start, L]`.
- If you supply `start` and `stop`, the optimizer will run exposures `[start, stop]`.

Loading Saved Results

If you choose to break up run of the optimizer, the program will by default store its results in `t.npy` and `obs.npy` to be continued later.

- `t.npy` contains an `ndarray` of shape `(N,L)` which stores integer values `0` or `1` representing whether galaxy `i` was allocated a fiber in exposure `l`. Initialized to zeros.
- `obs.npy` contains an `ndarray` of shape `(L,K)` which for each exposure `l` stores the `K` galaxies selected by the optimizer. Initialized to zeros.

Naturally if `t.npy` and `obs.npy` are modified and then reloaded into the program, it will accept the new values gracefully.

Viewing Outputs

The primary text-based description of a run of the optimizer may be found in `log.txt`. Test cases designed to evaluate performance of the optimizer also come with `desc.txt` describing their purpose.

Graphical files denoted `uhistxx.png` and `thistxx.png` are histograms of the maximum utility and target time for selected galaxies as the optimizer progresses, respectively. Note that if preloading occurs, *the optimizer will overwrite old images with new ones* to establish the progression of the histograms over time.

Finally, `sharp.png` is a time series plot of the attained sharp utility over time. (The sharp utility is defined as only being attained when the entire target time for a galaxy is reached. It is all-or-nothing.)

PFS Per-Exposure Targeting Optimization

Problem Description

We assume that we have $i = 1, \dots, N$ galaxies, each of which with a predefined utility u_i. The utilities can be parameterized as $u_i = \hat{u}_i\ \sigma((T_i - \hat{T}_i)/\delta T)$ with a maximum utility $\hat{u}$, which is modulated by a sigmoid function, to create the utility gains only after a desired integration time $\hat{T}$ has been reaching. For future reference, $\hat{u}$ and $\hat{T}$ may be grouped into classes of targets, not individual ones.

The observations are given by L exposures of equal length T_e in time on an instrument with $K = 2400$ fibers. We want to find per-exposure assignments t_{ikl} to

$$\begin{aligned}\text{maximize} \sum_{i=1}^{N} u_i &= \sum_i \hat{u}_i\ \sigma((\sum_{k,l} t_{ikl} - \hat{T}_i)/\delta T)\\ \text{subject to } \forall l : \sum_{k,i} t_{ikl} &= K\\ \forall k, l : \sum_i t_{ikl} &\le 1\\ \forall i, l : \sum_k t_{ikl} &\le 1\\ \forall i, k : t_{ikl} &\in \{0, 1\}\end{aligned}$$

This is combinatorially really hard and practically impossible when only next-exposure decision can be made. In this case, we have some $l \in [1, L]$, and we want to find the best assignments for the next exposure $l = 1, \dots, L$ such that the solutions approximates the global assignment/scheduling solution. We therefore introduce "intermediate" utilities u_{il} and then determine the targeting assignments s_{il} for the exposure l such that they

$$\begin{aligned}
\text{maximize} \sum_{i=1}^{N} u_{il} &= \sum_i \hat{u}_i \, s_{il} \\
\text{subject to} \sum_i s_{il} &= K \\
\forall i : s_{il} &\in \{0,1\}
\end{aligned}$$

But that doesn't have any aspect of long-term planning, so this approach goes after the high-utility targets whenever it can (not a terrible policy, but not very equitable). Let's see if we can find some more useful structure in lieu of the s_{il}.

The idea is to make this problem probabilistic and continuous and

$$\begin{aligned}
\text{maximize} \sum_{i=1}^{N} u_{il} &= \sum_i \hat{u}_i \, \Pr(\sum_{k,l'} t_{ikl'} = \hat{T}_i \mid \theta_{il}) \\
\text{subject to} \sum_i \theta_{il} &= K \\
\forall i : 0 \leq \theta_{il} &\leq 1
\end{aligned}$$

where $\Pr(\sum_{k,l'} t_{ikl'} = \hat{T}_i \mid \theta_{il}) = \text{Binom}(L-l, (\hat{T}_i - T_i')/T_e, \theta_{il})$ is the probability of getting the desired total integration by the end of the program. This term provides memory because it includes integration from earlier exposures $T_i' = T_e \sum_{l'=1}^{l} t_{ikl'}$ and future outlook because it counts the number of choices available to get the remaining exposures.

By optimizing the targeting probabilities θ_{il} we thus strike a balance between high-utility and feasible targets. Since we need to make targeting choices, we would rank-order the targets i by their contribution to eq. (9), and pick the top K.

What remains is the choice of the θ_{il}. If all targets are equally easy to get, the initial probabilities are $\forall i : \theta_{i1} = K/N$. This would pick all possible high-utility targets first, no optimization needed. We should test this as a baseline, but it is likely not close to optimal because it sacrifices all lower-utility targets even if there are so many to have that they have more aggregate utility.

One can think of several ways of determining "better" θ_{il}:

- Brute-force optimization possible given N is $\mathcal{O}(10^4)$, but likely to get stuck in local minima.
- Introduce classes such that $\theta_{il} \propto N_c K/N$ for $i \in \mathcal{C}_c$, which aims to get N_c total objects in class C_c after all exposures are done.
- Minimizing the variance of estimator, which prefers $\theta_{il} \in \{0,1\}$.
- Reinforcement learning: Simulate the policy to find q table as function of $\hat{u}_i, l, T_i'$ and potentially other terms for class-based utilities.

Separability Approach

The primary impetus behind this optimizer is to parallelize the problem to find better θ. In particular, instead of looking at

$$\begin{aligned} \text{minimize} \quad & -\sum_{i=1}^N \hat{u}_i \binom{L-l}{R_i} \theta_i^{R_i}(1-\theta_i)^{L-l-R_i} \\ \text{subject to} \quad & 0 \leq \theta_i \leq 1 \\ & \sum_{i=1}^N \theta_i = K \end{aligned}$$

which for the Subaru PFS is an $N = 10,000$-dimensional nonconvex (but smooth) optimization across $K = 2400$ fibers, instead we first formulate the Lagrangian

$$L(\theta_i, \lambda) = -\sum_{i=1}^N \hat{u}_i \binom{L-l}{R_i} \theta_i^{R_i}(1-\theta_i)^{L-l-R_i} + \lambda\left(\sum_{i=1}^N \theta_i - K\right)$$

Then we solve the problem

$$\begin{aligned} \text{minimize} \quad & -C_i\theta_i^{R_i}(1-\theta_i)^{S-R_i} + \lambda\theta_i \\ \text{subject to} \quad & 0_i \leq \theta \leq 1 \end{aligned}$$

which is smooth over compact domain and thus we simply need to solve a low-degree polynomial (since L is the number of exposures and thus the degree is not too large) by checking vertices and local extrema. Doing so allows us to construct a set of functions $\theta_i(\lambda)$ giving the optimal θ_i. Then to enforce $\sum_{i=1}^N \theta_i = K$, all we must do is solve

$$\Theta(\lambda) = \sum_{i=1}^N \theta_i(\lambda) = K$$

which can be done by bisection.

Analytical Form of Polynomial

To solve a problem of form

$$\begin{aligned} \text{minimize} \quad & f(\theta) = -C\theta^R(1-\theta)^{S-R} + \lambda\theta \\ \text{subject to} \quad & 0 \leq \theta \leq 1 \end{aligned}$$

we must analytically express the gradient as a polynomial: for $S \geq 2$ and $1 \leq R \leq S-1$,

$$\begin{aligned} 0 = \frac{df}{d\theta} &= C\left[-R\theta^{R-1}(1-\theta)^{S-R} + (S-R)\theta^R(1-\theta)^{S-R-1}\right] + \lambda \\ &= C\theta^{R-1}(1-\theta)^{S-R-1}\left[-R(1-\theta) + (S-R)\theta\right] + \lambda \\ &= C\theta^{R-1}(1-\theta)^{S-R-1}(S\theta - R) + \lambda \end{aligned}$$

Expanding binomial coefficients,

$$\begin{aligned}
0 &= C\theta^{R-1}(S\theta - R)\left[\sum_{k=0}^{S-R-1}(-1)^k\binom{S-R-1}{k}\theta^k\right] + \lambda \\
&= C\theta^{R-1}\left[S\sum_{k=0}^{S-R-1}(-1)^k\binom{S-R-1}{k}\theta^{k+1} - R\sum_{k=0}^{S-R-1}(-1)^k\binom{S-R-1}{k}\theta^k\right] + \lambda \\
&= C\theta^{R-1}\left[\sum_{k=1}^{S-R}(-1)^{k-1}\binom{S-R-1}{k-1}\theta^k - R\sum_{k=0}^{S-R-1}(-1)^k\binom{S-R-1}{k}\theta^k\right] + \lambda \\
&= C\theta^{R-1}\left[S(-1)^{S-R-1}\theta^{S-R} + \sum_{k=1}^{S-R-1}(-1)^{k-1}\left(S\binom{S-R-1}{k-1} + R\binom{S-R-1}{k}\right)\theta^k - R\right] + \lambda
\end{aligned}$$

from which the optimal value θ^* may be numerically solved (for any given λ).

Meanwhile if $S = 1$ (recall $S \neq 0$ since it is the number of remaining exposures), then the objective for $X \sim \mathrm{Binom}(S, \theta)$ is just

$$\begin{cases} -\hat{u}(1-\theta) + \lambda & R = 0 \\ -\hat{u}\theta + \lambda & R = 1 \\ \lambda & \text{otherwise} \end{cases}$$

Of course since the minimization occurs over $[0, 1]$ (and looking forward to the sum constraint on θ) this is solved by

$$\theta^* = \begin{cases} 1 & R = 1 \\ 0 & \text{otherwise} \end{cases}$$

Finally consider $S > 1$, but $R = 0$ or $R = S$. If $R = 0$, then again $\theta^* = 0$. If $R = S$, then

$$\begin{aligned} \text{minimize} \quad & -C\theta^S + \lambda\theta \\ \text{subject to} \quad & 0 \leq \theta \leq 1 \end{aligned}$$

reduces to testing $\theta \in \{0, 1, \sqrt[S-1]{\lambda/CS}\}$ from first-order conditions.

Comparison to Other Optimizers

Comparative Advantages

Because of the high-dimensional nature of the problem, methods such as differential evolution (and other genetic algorithms), simulated annealing, etc. are too costly in terms of runtime by several orders of magnitude. On the contrary, the parallelized nature of this problem makes it extremely computationally efficient, as it reduces to solving a large number of one-dimensional polynomial optimizations.

One of the most desirable characteristics of this optimizer is that it has the capability to make next-exposure decisions, while still planning ahead for long-term optimality. This property makes it robust to external disruptions, such as weather or equipment interference, which prevents may prevent Subaru PFS from actually making all of the suggested observations recommended by the optimizer.

An additional desirable trait is the emphasis on high-utility, long-timeframe targets first, which is a policy this program empirically follows. This adds additional robustness to disruption as it gives the PFS more chances to observe vulnerable high-value targets.

Potential Drawbacks

Because of the bracketing involved in finding an optimal Lagrange multiplier at each exposure, in some rare occasions the optimizer will be forced to evaluate $\sqrt[S-1]{\lambda/CS}$ for $\lambda < 0$. The result is a complex value whose imaginary part the optimizer is forced to discard. Since this arises if and only if $R = S > 1$ (R is the number of necessary exposures for a galaxy, S is the remaining exposures left in the program) it is limited in scope and empirically does not affect the optimizer's success, even when it does occur. However, a more robust treatment could potentially help in some limiting cases where this phenomenon could make a difference.