# Fri. 17 May 2024

* We are in very preliminary stages, but Welch is thinking about a possible algorithm that would partition the domain of the objective function only when necessary. That is, if the objective function is easily optimized with few observations, then you don’t need fancier tricks like partitioning. But if it’s proving difficult to optimize, maybe because the predictive variance is staying high, or you are needing so many observations that fitting the new GP model is getting really slow (e.g. thousands of observations), then you might need to employ some fancier tricks like partitioning.
* Main takeaways:
  + pick one or two objective functions that have two-dimensional domains, but are complicated, e.g, with many local minima. An example would be the Ackley function
    - check the sonja surjanovic page under “many local minima”
  + Use MaxProLHD package to generate your initial observations (maybe 20)
  + do standard BO with those initial observations, i.e. over the whole domain of the objective function, using maybe 100 or 200 observations
  + plot the points that BO is visiting over the course of the observations. like, plot them on a grid, with numbers to indicate the order in which the BO visited those points.
    - and plot the location of the global optimum too, i guess
  + then split the domain of the objective function in half, and do standard BO in each half. So then I guess you get 10 initial observations in each half, and if you did 100 (additional) observations over the whole domain, then you would only do 50 additional observations in each half.
    - again produce the same plots for each half
  + the point is to try to compare how well BO does over the whole region, versus splitting up the region and doing BO in each half (with the same number of total observations)
    - so i guess you want plots of the best objective function value observed so far, as well
  + maybe do the above for a second objective function, and maybe do it for each objective function twice (with different starting seeds, to account for some randomness)
    - and maybe with 2 different sets of starting points (generated by MaxProLHD)
* Other things:
  + try to reproduce the bug I got when running the EGO code, and share it with him
  + try to repeat my Digital Research Alliance (i.e., Compute Canada) account renewal request

# Fri. 24 May 2024

* Welch does not think that any of the test functions I’ve used so far (Ackley, Rastrigin, Griewank, Cross-in-Tray, all in 2 dimensions) are likely to be ideal tests for what we’re trying to do
* He is interested in test functions where the optimum point is clearly not in most of the space, maybe something like the corner peak function
* things that I should do in time for the next meeting:
  + the corner peak function is available in Sonia’s library. Try standard BO with it, in say 2, 5, and 10 dimensions, maybe 100-400 observations, and generate the plots and see how well regular BO does
  + see if it’s possible to modify corner peak to have (lower) peaks in the other 3 corners
    - maybe ask on StackExchange if anyone has any idea how this might be done
  + more generally, can we find test functions where the argmin is clearly *not* in most of the space? the idea is that we want our method to be able to partition the space and then ignore most of the regions,because the argmin is obviously not in those regions
  + lastly, i should share a modified version of my EGO.R file with Welch. Try to get rid of the superfluous stuff and make it self-contained, so he can just run it and generate the errors I was getting. Then I guess attach it to an email and write a brief explanation for how to generate the errors

# Fri. 31 May 2024

* we are still in very preliminary stages, obviously
* i think my job is to just try to keep doing standard BO runs with different acquisition functions and in different dimensions (say 2, 5, and 10), and with different total observation numbers (say 100, 200, … 500), for 10 runs at a time, and just try to see how and when standard BO struggles.
* like maybe i could go back and do the ackley, cit, grie, and rastr again
* and maybe try what welch said about modifying corner peak to put peaks in other corners as well
* and i could also try also some of the other corner-peak-like functions that i found (check my handwritten notes)
* and i need to get set up again on compute canada (digital research alliance) because i should try to send my compute jobs there instead of doing them on the macbook pro. ideally i could just send multiple jobs, say with different acquisition functions.

# Fri. 14 June 2024

* we are interested in testing EGO/GaSP, Dice and BoTorch, perhaps with the 10-dim rastr function at least
  + it would be good to generate a set of space-filling design points via Initialize (say ten 40x10 tables), export them as .csv file(s), and then use these as the common init points for testing each implementation
  + so i should maybe make an implementation\_testing folder with different files for each implementation (one or EGO, Dice and BoTorch)
  + i would have to install and setup BoTorch
  + i should try to make the data generated by each implementation as similar as possible, so that my plot-making code can be re-used
* the other thing i need to do is look at the photo of the whiteboard and try to start writing this as code
  + what’s in the photo is only a sketch; there are details to be filled in
  + i guess the main questions are 1) when do you decide to split a region, 2) how do you split a region, and 3) how do you pick a region to “work on” (by which i guess i mean give to the explore\_sub function in the photo)
  + there are also issues to do with record-keeping, like how you keep track of the regions and record data
  + i should try to lay out what’s in the photo as code, and consider the outstanding issues and fill them to any extent that seems reasonable
  + and i should share a github link to the code with welch

# Fri. 28 June 2024

*4 July note: This meeting never happened, so these are just notes to myself.*

* i need to better understand how we decide which axis to split a region on

major questions:

1. how to decide which region to explore?
   1. each subregion has a y\_min, and an EI value (call it a\_max) from the last time you maximize an EI function in that region. we could use y\_min - a\_max as a measure of how promising a region is
      1. so if a\_max is big relative to y\_min, then the region is promising?
      2. what if you used norm of a\_max over norm of y\_min? then if | a\_max | / | y\_min | is large, the region is promising?
2. given that you are exploring a region (i.e. you are doing standard BO or something like it in a specific region), how do you decide to *stop* doing standard BO in that region?
   1. you could decide the region is not promising, so you want to stop exploring it
      1. this is where we write “a\_max < tol”
         1. **but doesn’t this condition have to somehow reflect information from the other regions in our list of potential regions? i.e. if we have decided to stop exploring a region, that is because we have decided the region is not promising… relative to the other regions we could be exploring, right?**
   2. or you could decide that the region is still worth exploring, but you’ve accumulated too many observations in the region, so the BO process will be too slow, so you want to split the region
3. given that you have decided to split a region, how do you split it?
   1. we could split a given axis at the midpoint, or at the median of the coordinate values of the data points in the region (the latter would give us close to a 50/50 split of points in each of the two new sub-regions)
   2. **i don’t see how it works to split based in y\_min, because splits on each axis will produce a new sub-region containing y\_min, so you have no way of preferring one split over another**
   3. **what if you chose the split that led to the most promising new sub-region? like if your measure of the value of a region was evaluated for each potential new sub-region, and then you chose the split that led to the creation of a new sub-region with the highest possible region value?**
   4. once you split a region into two sub-regions, you may want to return to your list of candidate regions and pick the most promising one (i.e. not immediately focus on exploring either of the two new sub-regions)
   5. another thought i had about splitting: it may be that an (overall) promising region has a promising sub-region and a lousy complement, in which case we would want to somehow discard the lousy sub-region
      1. by which i mean that we don’t necessarily want to prioritize each of the two new subregions equally

general issues:

1. i did modify EGO to return the EI value at the chosen point, but I don’t know how useful that is
   1. i am not sure how to get access to the EI values of all points considered in the region during the optimization of the EI function
      1. would i have to modify genoud to get access to this information?
2. should the “tolerance” be the highest EI value of all regions? like say I have three regions which have EI values of 3, 4, 5, and then I am evaluating the third region, and I get EI values of 5, 4.5, 3 in that region; would i then break and decide to evaluate the region with the EI value of 4?
   1. welch seems to think that the tolerance is somehow relative to what we think the optimum value might be (absolute vs. relative tolerance, so on…)
3. **more generally, it would be conceptually simple and attractive if we came up with a way to measure the value of a region, and then we used this metric repeatedly. like when choosing which region to explore, you choose the region with the highest value. when deciding to stop exploring a region, it’s because another region has higher value. when choosing how to split a region, you choose the split that leads to a new sub-region with the highest region value.**
   1. **If this is worthwhile, the question is, what is your measure of the value of a region?**

# Fri. 5 July 2024

*Pre-meeting notes below (listing what I wanted to talk about):*

* push/update the code from cedar and pull it down to the macbook
* I should briefly describe the code that I’ve written
  + I mostly tried to implement what Prof. Welch sketched on the whiteboard in our last meeting. I can briefly explain this.
  + I think the main points to discuss are i) which region to explore, ii) how to set the tolerance value, iii) how to split a region, and iv) how to estimate the value of a region
* I can briefly run through the 2-dimensional rastr example on the Macbook in Downloads/cedar\_test\_output/research\_testing/bo\_partition\_test\_y\_min\_minus\_a\_max.txt
  + draw what’s happening on the whiteboard; look at the picture I made in my notes as a guide
* the next thing to discuss is the segmentation fault errors
  + the relevant files are on the Macbook in Downloads/cedar\_test\_output/35795722\_output
  + hopefully by this time I have updated GaSP and re-submitted the jobs asking for 4000M memory
  + mention whether i’ve heard back from Cedar tech support
  + maybe mention creating core files and using GDB (show him the alliancecan page about this), but it seems like you need an executable along with the core file; ask if I have an executable?
* we can also look at the plots from the 10-dim dice/ego rastr comparison
  + the relevant files are in Downloads/cedar\_test\_output/dice\_ego\_rastr\_10
* we should also discuss the time profiling that he wanted
  + i haven’t looked into this much yet
* i guess splitting and fitting on all d axes is something that’s ripe for parallelization

*Post-meeting notes:*

* we discussed how to decide how to split a region
  + we are thinking about maybe modifying EGO to return the EI function, and then evaluating that function at a bunch of points in the region you want to split, and then i guess using those points to compute the y\_min minus a\_max?
    - I need to talk to Welch more about this
* we also discussed using R profiling tools (package “rprofile”?) to assess how much time/memory some R code is using
  + but a big question is whether these are RStudio/GUI tools, or whether it can be done via R code only, i.e. on Cedar
* Prof. Welch also mentioned using the R debugger to try to figure out where the segmentation fault is happening
  + but we don’t even know if the segfault is reproducible or not, which matters
  + Prof. Welch also suggested putting in lots of print statements (in split\_and\_fit, I guess, and maybe in EGO as well)
  + I also need to look at the results from my 4 GB jobs to see how they turn out
  + It’s also a a question whether the R debugger is an RStudio/GUI thing, or whether it can be used on Cedar
    - maybe also think about interactive/GUI Cedar jobs, if that’s helpful at all
* But probably my top priority is to re-write my research code to use either EGO or Dice (with if statements and a dice/ego parameter in the shell script), because then can I re-run my code using the Dice functions
  + the point being we can then see if I get errors when using the Dice functions
    - if I do get segfaults when using Dice, then maybe the error is mine (in my shell script or R code)
    - if I don’t get segfaults when using Dice, then the error is in EGO/GaSP where, maybe/probably
    - so this is my top priority

# Fri. 12 July 2024

*Pre-meeting notes below (listing what I wanted to talk about):*

* I think I have my bo\_partition method running with both dice and ego
  + show the plots (original ones and the incomplete ones)
  + discuss the performance differences, and suspected reasons (i.e. EI vals are higher when using EGO, so method never rejects a region?)
* discuss how I got them both running
  + i added a next escape hatch in case a proposed region for splitting had essentially zero width (i think this happens when EI proposes many points for sampling on the same boundary of a region)
  + without this escape hatch, when using dice, I got an error in genoud
  + i have not investigated whether this is the reason for the segfaults when using ego
* to investigate the segfaults, i installed r and ego on my old desktop, compiled ego from source with no optimization, and ran a small 2-dim, 100 obs example with valgrind
  + show the text file output
* i have done a little bit of reading about profiling, but i havent attempted to do any profiling yet
  + does welch only want R code profiled, or R and the C code as well?
  + some links about profiling in R
    - <https://dept.stat.lsa.umich.edu/~jerrick/courses/stat506/syllabus_506_F23.html>
    - <https://adv-r.hadley.nz/perf-measure.html>
    - <https://rstudio.github.io/r-manuals/r-exts/Tidying-and-profiling-R-code.html>
    - <https://bookdown.org/rdpeng/rprogdatascience/profiling-r-code.html>
    - <https://cran.r-project.org/web/packages/proftools/index.html>
* also discuss the method i’m devising/implementing (pass the region model from explore to split, and then ust that model and maximize EI over the proposed subregions)

*Post-meeting notes:*

* I should listen to the recording
* I think I should probably make new versions of my bo\_partition.R and the helper\_func file,
  + focus on just one R library (Dice, I think)
  + and one split criterion
  + just try to get it working reliably and bug-free, so we can iterate on it
* i think the idea is roughly as follows:
  + most of bo\_partition.R will work the same
  + explore\_region() will work mostly as before, but if we’re going to call split(), we fit a GP model for the region to split, and we then pass the region and its GP model to split()
  + then in split, we use random lhs to generate a bunch of points, say 1000, in the subregion
    - look at my picture (and the end of the recording) for how to scale and translate the unit cube points to our actual region
      * i think you just scale each dimension by the length of the region in that dimension, and then subtract the coordinates of the center point of the region
  + we evaluate the EI function at all those points (use apply?)
  + then we iterate over the dimensions to split
    - and we could maybe consider splitting at 0.25, median and 0.75 percentile on every axis
    - and we want the split leading to a new sub-region with the highest value
    - so maybe we will keep track of how many of the top 10 (say) EI values are obtained in a given region
    - and then we will choose the split that leads to a new sub-region with the largest number of high EI values (i am being vague here)
      * maybe if multiple regions contain the top 10 EI points, you could pick the one with the smallest volume?
    - then you have two new sub-regions, so we need to fit new GP models for each region and optimize EI for each
      * note that we probably don’t need to actually take observations from these new sub-regions; just bring them back to our list of regions and decide afresh which region to pick for evaluation
    - there is some question about what you do if one of your new sub-regions has very few points (do you try to augment that?), but maybe I should just ignore this issue at first
    - it might also be possible to explore regions in parallel
      * like let’s say you asked for 8 cores, then you could possibly run explore\_region up to on up to 8 regions in parallel (simultaneously)
      * but i probably shouldn’t worry about this yet
  + maybe work on 2-dim rastrigin first, just to keep things simple so i can follow along
* for code profiling, welch only cares about profiling the R functions (not any compiled code)
  + but this is not our highest priority (get something working decent first)

# Fri. 19 July 2024

*Pre-meeting notes below (listing what I wanted to talk about):*

* nserc email: i plan to change my start date from may to sept (to get >$6,000)
* briefly explain my method
  + maybe follow along in desmos, maybe the init region split
  + and then the 2dim test (also on desmos)
* discuss main 10dim test results.
  + the ultimate results (best so far and plot)
  + computing time, memory usage
    - sacct -j 37014323 for thursday’s results
    - sacct -j 36036977 for regular dice results
  + job memory usage
  + how the jobs ended, i.e. how many promising regions remained, and how many regions had been rejected
    - seed 1 prom. reg. 3, seed 2 prom. reg. 3, seed 3 prom. reg. 6, seed 4 prom. reg. 3, (not seed 5), seed 6 prom. reg. 3, (not seed 7), seed 8 prom. reg. 2, seed 9 prom. reg. 3, seed 10 prom. reg. 5 all seem to contain the argmin, so in principle, with further exploration, our method could find it
    - it does not seem that the method is rejecting regions prematurely
  + how many times we refused to consider a split because of narrowness (search for “consider” in the .out files)
    - i thought about considering more percentiles e.g. 0.10, 0.20, …, 0.90, but i was concerned that this could lead to new (and poor) subregions with very few points, and that the algorithm would waste time choosing to explore those bad subregions (so i didn’t actually do it)
  + i have a feeling i should probably compute *log* volumes, and choose smaller *log* volumes, rather than true volumes (for numerical reasons)
  + how long did the splits take?
    - 6 min?
  + how many of the top 10 EI points did the chosen splits tend to have?
    - looks like usually 10/10, sometimes 8/10 or 9/10
* compare thursday’s results with regular dice results
  + regular dice results are in 5jul24meeting
  + thursday’s results are in 19jul25meeting
* one speedup would be to explore any promising regions in the promising regions list in parallel, i.e., as soon as a new promising region is added
  + this will presumably affect temporal nature of the plots
  + i don’t know much about this
* my thoughts about next steps: we have not really “broken” regular BO yet, so I don’t think these tests are ideal for showing superiority of our method
  + i think there are two main further tests we could do:
    - we could run with the same parameters except total subsequent obs = 500, and see how the two methods do
    - we could set n\_max = 200, since regular BO seems to do reasonably well here, and maybe total subsequent obs 1,000–5,000, and just see what happens
    - maybe do 20+ runs of these tests?
    - eventually, possibly higher-dimension tests (e.g., 20)
    - note that for these tests i should almost certainly request 4 GB memory
  + other ideas: maybe try more EI values (e.g., 10,000?); tol param seems reasonable; what about changing top 10 EI points?; maybe change the percentile\_vec?

*Post-meeting notes:*

* to do: log volume, and job submit script for BO and then plotting after (latter is relatively unimportant)
  + plotting job might need 4 GB memory
  + i should modify my script save\_dir to be more informative (it should probably contain some information about parameters of my method; and maybe it doesn’t need to say anything about num\_runs)
* at the end of split(), we could go straight into explore\_region(), exploring the “better” new subregion, i.e. the one containing more of the top n EI points
  + could you do this by calling explore\_region() at the end of explore\_region(), inputting the better of the two new subregions?
    - i think this might work, but i would need to figure out how I’d eventually exit explore\_region(), and if I do multiple splits, how do I eventually return all the splits and information I’ve gathered?
* relatively minor changes:
  + every time calling max\_EI, we want to use a plugin for the global y\_min, i.e. smallest y observed so far
  + also use plugin when evaluating EI at lhs points during split\_and\_fit()
  + when deciding which region to explore (roughly line 280 in bo\_partition.R), we could just choose to explore the region with highest a\_max (i.e. forget about y\_min)
    - the a\_max should have been computed with the global y\_min as a plugin
    - maybe add a print statement printing out all the region a\_maxes
  + maybe re-run my code with just these couple of changes
    - maybe evaluate EI at 1,000 points and focus on the top 10 EI points
* in a 2-dim rastrigin example, plot the initial points, the points EI chooses, and the points at which we evaluate EI
  + can we also plot the local minima of rastrigin?
    - need to differentiate rastrigin and figure out where its local minima/maxima are (maybe ask for help on stackexchange if necessary)
    - could do this on a separate plot if necessary, or if the local minima are “regular”, could I adjust the axes so that the intersection of axes represents a local minimum?
  + the point is that we want to get a better feel for a region is being explored in explore\_region()
* Another idea: if every region's value is just its a\_max (with the global y\_min as plugin), could I make a “region\_value\_vec” and feed it to explore\_region() so that instead of setting tol= 0.1, my tol is the next highest a\_max value? So you explore the region with the highest a\_max until its a\_max drops below the second-highest a\_max?
  + You don't need the whole vector, just set the tol to be the second-highest value, right?
* maybe i also need to email mairead about being a TA for welch’s course

# Fri. 26 July 2024

*Pre-meeting notes below (listing what I wanted to talk about):*

* i emailed mairead asking if she could give me some work in stat 404; she did it
* explain the main four variations i have tried in the last week (v2, 3, 4, 5)
  + v2: use global y min as plugin whenever maximizing or evaluating EI
  + v3: like v2, but only take sample if the EI value is at least tol, and at least as high as the second-highest EI value (for another region) (i.e. region switching)
  + v4: like v3, but only consider rejecting a region after it has n\_max obs
  + v5: like v4, but allow new subregions to share observations within point\_sharing\_tol of split border with other new subregion
    - the region boundaries don’t overlap (i.e. the space is partitioned), but a region may “contain” points that are not in the region
    - this can lead to erroneous/nonsensical percentile calculations for split points, because these calculations may involve points that are not actually in the region.
    - I am trying to resolve this by, when splitting on dimension d, only considering the points “in” the region that are within the region’s bounds in dimension d
* show desmos 2dim w/contour
  + this was done with the “minor changes”
  + compare with the plots in OneDrive/bo\_partition/plots/bo\_runs/rastr\_same\_init
    - I think I was using EGO in these plots, not Dice
  + these plots seem to show that 2dim Rastr is basically a solved problem with standard BO
  + my gut reaction when looking at our method in desmos is that we’re splitting too soon
    - meaning: n\_max too low
    - because it seems like we end up wasting a lot of time exploring sub-optimal subregions
    - (but splitting too early could be possibly mitigated if you switch regions when the a\_max drops below the next-highest a\_max, right?)
* show desmos 2dim switch w/contour
  + seems to be exactly the same as previous one through first 20 observations
  + i still think we are splitting too soon
  + it seems like the algorithm is trying to investigate the same region from both sides (i.e. the border of two adjacent regions)
    - this seems dumb and sub-optimal
* show the crash i had in seed 8 w/ switching (load the .rda file)
  + my solution was to put in an if statement checking if one of the new regions would have zero points, and go to the next possible split if so
  + i am operating under the assumption that when a region is split, the points in the region should be partitioned into the two new sub-regions, and that which inequality is strict and which is not is essentially arbitrary
    - but should points on the border (or within epsilon of the border) be common to both new subregions? In other words, should we not actually partition the points, necessarily?
      * maybe one way to do this would be to allow regions to overlap by some epsilon, like maybe the min\_split\_width
      * this would bring us close to n\_max and thus make us more likely to split a region, which i think is maybe another reason that we need to increase n\_max
    - i wonder if allowing subregions to share points at/near their common border would reduce the algorithm’s propensity to try to explore a border from both sides. **i guess i need to implement this and make another desmos (probably with the seed 1 points) and see how it does.**
* show the 10 run results
  + v2 and v3 results seem to show a lot of early quitting because a\_max < tol
    - i.e., we quit early because all regions have been rejected
    - maybe we should only reject if a\_max < tol AND we have n\_max observations? i.e., we make the “reject or split” decision only once we have n\_max observations in a region?
      * this could work well in tandem with region switching, because regions with low a\_max will be de-prioritized, i.e. they will be explored later
  + v4 and v5 seem better, maybe v5 possibly better than v4 (not sure)
* differences between v4 and v5:
  + it looks like point-sharing as in v5 can lead to significant differences in EI optimization. e.g. in seed 10, we see v4 and v5 perform identically up to the first split, and then choose the same split, but v5 reg1 contains one point shared with reg2, whereas in v4, the points are partitioned as normal. you can then see that the EI optimization for reg1 in v4 and v5 proceed very differently, presumably only because the v5 reg1 has one additional point that it doesn’t have in v4
    - note that the reg2 optimization proceeds exactly the same in both (seemingly), because no points were shared, i.e. v4 and v5 reg2 are identical. (this is because i use a non-strict inequality when partitioning the points for reg2, in my prep\_subregions() code)
* other thoughts:
  + if the optimized EI value for one of the new subregions created by a split is really low (like < 0.01), we could consider rejecting that new subregion right away, so that the split only results in one new subregion that we wish to explore further
  + should i maybe try considering the top 20 or 30 EI values? i am concerned that if the top 10 EI values are very concentrated around one point, then it makes the splits very short-term focused like they’re optimizing the split with respect to the next point that we want to sample. i wonder if upping from 10 to 25 (say) could make the splits a little more conservative or long-term focused
  + i have been trying to trying to get benchmark results from EGO.nsteps() from the Dice packages, but i keep running into chol.default(R) leading minor issues
    - i have been trying progressively bigger nuggets, from 1e-09 all the way to 1e-01, but i get similar errors every time
    - perhaps i will have to move ot higher-dimensional tests (e.g. 3, 4, 5) to compare my methods with vanilla BO as in DiceOptim
  + i guess possibly a general principle at play here is that we want to partition to gain speed from having fewer observations in a region, but we want to i) avoid committing to a bad sub-region (hence the switching), and use information from outside the region as long the cost of doing so is zero or minimal (hence the global min EI plugin, and the point sharing)
  + sign test (or its possible generalization, the friedman test), might be useful to compare the performance of different optimization methods over 10 or 100 runs (same starting points for each run)?

*Post-meeting notes:*

* make desmos contour plots for v5, for 2 or 3 of the 10 runs where v5 does well or poorly (e.g. seed 1, and maybe another one or two where v5 does well)
* try to get predictive mean and standard deviation (from the km() model) for a region containing the optimum (i.e. the origin) when that region is rejected
  + i.e. when the optimized EI value is computed to be < 0.1 (and the region was rejected), was this because of large predictive mean and/or small predictive variance?
    - there were 6 runs (out of 10) where the origin/optimum was not found, so what happened to the subregion containing the origin in those cases? I could try to gather predictive mean/sd information in each case where the subregion containing the origin was rejected
  + I guess take the km() model at the time of region rejection, make a grid of values in the region, and feed the km() model to predict()
    - and/or a contour plot of those values?
    - see the predict (DiceKriging) R documentation
* we could also re-run v5 for the same 10 runs, with the same sets of 20 initial points, but n\_max = 50 and total subsequent obs = 200, and see how v5 performs then
  + does gathering more information lead to different choices made by my v5 method? i guess particularly for the same set of initial starting points
    - maybe just re-do all 10 seed values/runs
    - and do v4 as well
  + could even consider doing n\_max = 100, total subsequent obs = 200, maybe
* if i set a large n\_max for some problem, and i find that the initial region gets rejected, i could wrap the region reject if statement in an if statement itself, like “if the total number of observations taken is > the total budget / 2”, meaning we won’t even consider rejecting until we’ve used up at least half our total budget
* since we don’t have another meeting for two weeks, maybe i should do a comparison of v5 (and maybe v4) with regular DIce EGO on a 5- or even 10-dim problem?

# Fri. 9 August 2024

*Pre-meeting notes below (listing what I wanted to talk about):*

* two main things to talk about:
  + first, the 50/200, 100/200 and 120/200 results (the plots)
  + second, i have tried to bring information that will hopefully illuminate selected cases where the algorithm performs poorly (i.e. rejecting a region containing the origin without finding < 0.05)
    - main content is desmos plots and km() mean and sd plots (discussed below)
* show v5 25/100 seed 1 on desmos
  + x
  + then show the km() plots
    - x
* show results of v4 and v5 n\_max = 50, subseq\_obs = 200 runs
  + seems that allowing bigger n\_max makes a significant positive difference
    - (i mentioned last meeting i suspected we were splitting too early)
    - v5 performs slightly better than v4 (v5 solves seed 8)
      * more (weak) evidence that v5 is slightly better than v4
  + biggest failing is runs 5 and 10, where entire region is rejected after 30 additional obs (50 total)
    - show v5 50/200 seed 5 on desmos
      * x
      * then show the km() plots
        + x
* show results of v4 and v5 n\_max = 100, subseq\_obs = 200 runs
  + v4 and v5 appear to perform identically (presumably because no splits were made)
  + again it appears that we have gained performance due to larger n\_max
    - notice that seed\_8 was figured out after approx. 55 observations (much faster than in 50/200 run)
    - interestingly, 50/200 did worse on seed 9 than 100/200
      * might be just a fluke
      * show v5 50/200 seed 9 and v5 100/200 seed 9 on desmos
* show results of v4 and v5 n\_max = 120, subseq\_obs = 200 runs, with mandatory split
  + x
* thoughts:
  + if the user thinks their function might be very difficult (require lots of exploration), maybe we could let the EI exponent be a parameter (higher meaning more biased towards exploration)
    - i say this because maybe EI would benefit from being a bit more biased towards exploration, at least in the examples where it performs poorly
    - but how you would tune this, and whether that would mess up the cases where it performs well, is unclear to me
    - so i’m not sure this is a good idea

*Post-meeting notes:*