Previous Next

4.3.5 Problem Set: Cell Tower Study

Bookmark this page

Now we are ready to optimize the positions of multiple cell towers. In the file `cellopt.py`, we have provided the function `calc_objfun`, which calculates the p-norm approximation of the objective function J_{minmax}, given a set of base locations in `basesv` and user locations in `pdict`.

In `plot_optimize`, we call `gd.gradient_descent` just like in `gd_tests.py`, except we pass in `calc_objfun` instead of `J0` or `J1`. We then visualize the gradient descent history alongside the user locations for reference. On top of all this, `optimize_bases123` runs three scenarios of `plot_optimize` with varying number and initial locations of base stations.

In the `__main__` block of `cellopt.py`, we have supplied two configurations of user locations. For the first two parts below, you will use the configuration where users are located along roads at a T-intersection. Then in the last part below, you will switch over to the locations of MIT's undergraduate dorms, and use the settings found previously to optimize cell tower locations for campus.

1. In the case where we optimize the location of only one base station, we also want to plot contours of the objective function in the x-y plane. (Why don't we plot contours for two and three base stations?)

   Complete the implementation of `plot_objfun_onebase` in `cellopt.py`, according to its docstring and the following steps:

   - Find the boundaries of the plotting region, which are defined to be the bounding box of the user locations, with an additional 10% margin.
   - Evaluate the objective function (call `calc_objfun`) throughout the region, and plot its contours.
   - Find the location of the objective function's minimum, and plot it with a red asterisk (`marker='*'`) using the `plot` function.
   - Specify the minimum objective and its (x, y) location in the title. Use scientific notation with two digits beyond the decimal.
   - Finally, return the `ContourSet`, so the autograder can inspect it.
   - **Note:** Some of the code you wrote for `run_test1` will be useful to copy over. However, you need not worry about the plot's aspect ratio, axes labels, or grid, because those are already handled by the `plot_users` helper function.

Discussions

All posts sorted by recent activity

Exact minimum location? "Find the locat
m_powers
2

- **Note:** For a NumPy approach to finding the minimum's location, consider using `np.argmin` and `np.unravel_index`.
- **Note:** The format of the title should exactly match that shown in Figure 4.13.

To test, run `cellopt.py` using the `users_T` locations. With the default settings of `alpha=0.1` and `nStop=10` in `optimize_bases123`, you should see the plots in Figures 4.13 through 4.15. And the printed output should be:

```
Optimization with 1 base station:
   Jmin   = -4.48e-01
   Base 0 = (-2.12e-01,  0.00e+00)

Optimization with 2 base stations:
   Jmin   = -5.07e-01
   Base 0 = (-6.10e-01, -8.35e-02)
   Base 1 = ( 1.20e-01,  4.76e-01)

Optimization with 3 base stations:
   Jmin   = -5.02e-01
   Base 0 = (-6.11e-01, -1.15e-01)
   Base 1 = ( 3.28e-02,  9.75e-03)
   Base 2 = ( 2.31e-02,  5.31e-01)
```

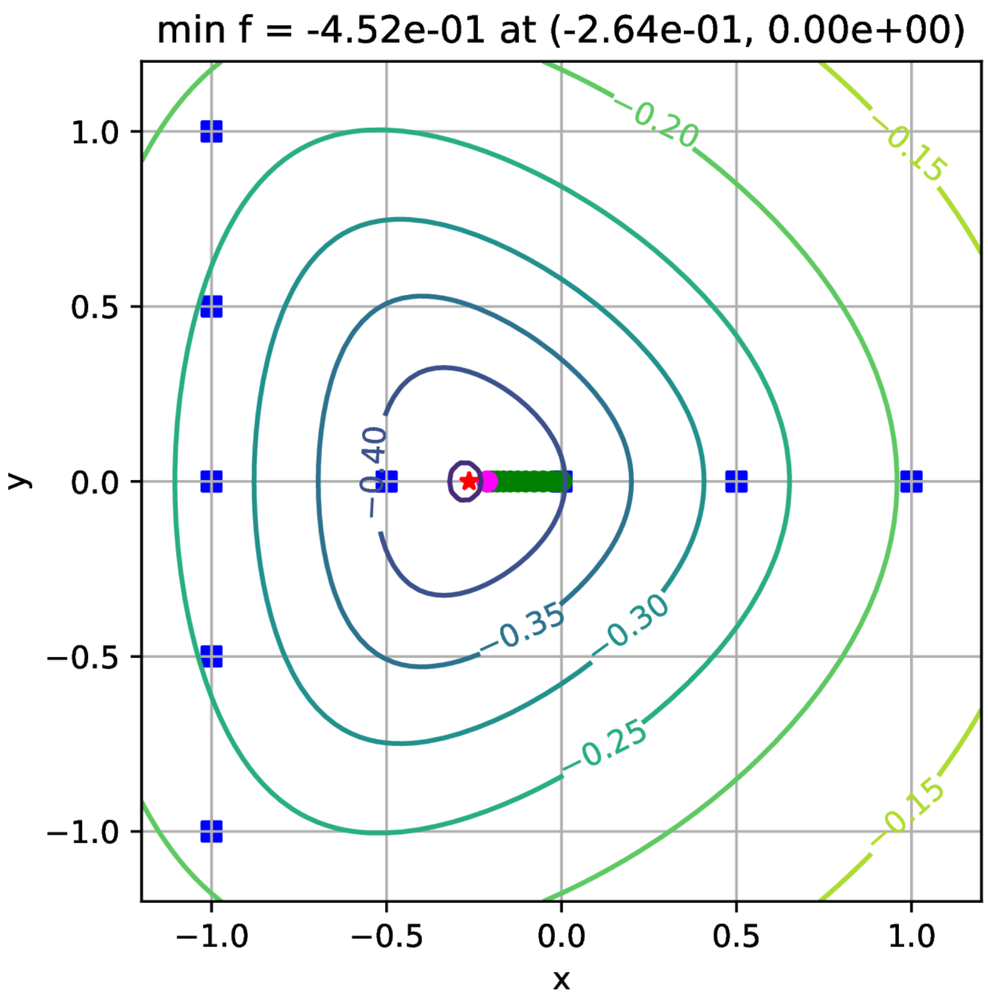
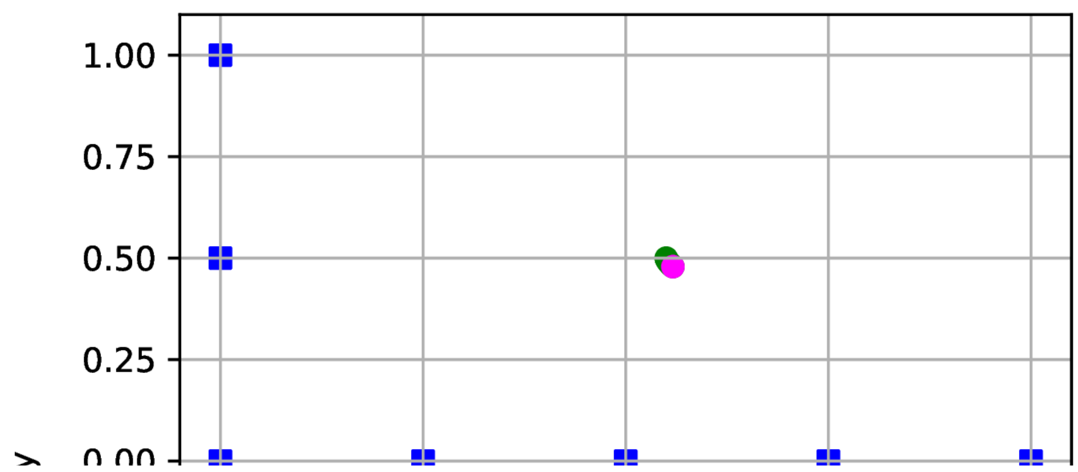
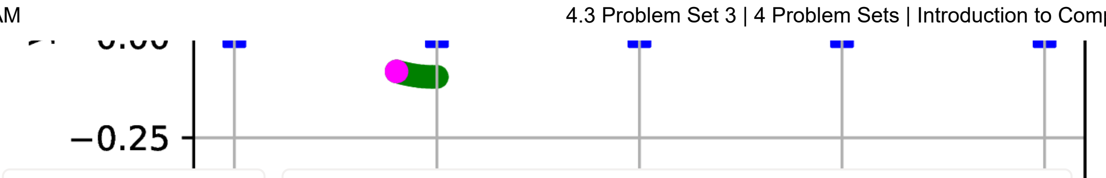


Figure 4.13: Iteration history for 1 base station during gradient descent minimization in the T-intersection configuration.





check if you found the right values.

- Keeping `alpha` fixed, increase `nStop` in multiples of 25 (i.e. 25, 50, 75, ...) until the reported `Jmin` no longer changes (at the precision they are printed at) in all three scenarios (1 base, 2 bases, 3 bases). You may still see minor changes in the positions of the base stations, which is okay if the gradient of J is very shallow at those locations. Suppose you found that `nStop` of 75 and 100 produced the same values of `Jmin` (though `nStop` of 50 is different). Then, the value of `nStop` you should return in `my_nStop` is 75.

- Using this larger value of `nStop`, increase `alpha` in increments of 0.1 (i.e. 0.2, 0.3, ...). If `alpha` is increased too much, the gradient descent algorithm will go unstable or oscillate. Based upon this, determine the largest `alpha` that produces, in all three cases, non-oscillatory convergence to the minimum.

- Finally, with this new value of `alpha`, are you able to use fewer iterations (i.e. smaller `nStop`) and still converge to the minimum objective? Specifically, with this new `alpha`, find the minimum value of `nStop` in multiples of 25 (i.e. 25, 50, 75, ...) such that the reported `Jmin` values do not change (at the precision they are printed at).

3. We are finally ready to improve cell phone reception at MIT's undergrad residences. In the `__main__` block, comment out the