Modeling Workgroup Meeting
Quarterly Review

Optimization update: Development of an Optimization API

Kalyanmoy Deb, Pouyan Nejadhashemi, Gregorio Toscano, Vahid Rafiei, and Hoda Razavi.

OCT 2022

MICHIGAN STATE UNIVERSITY

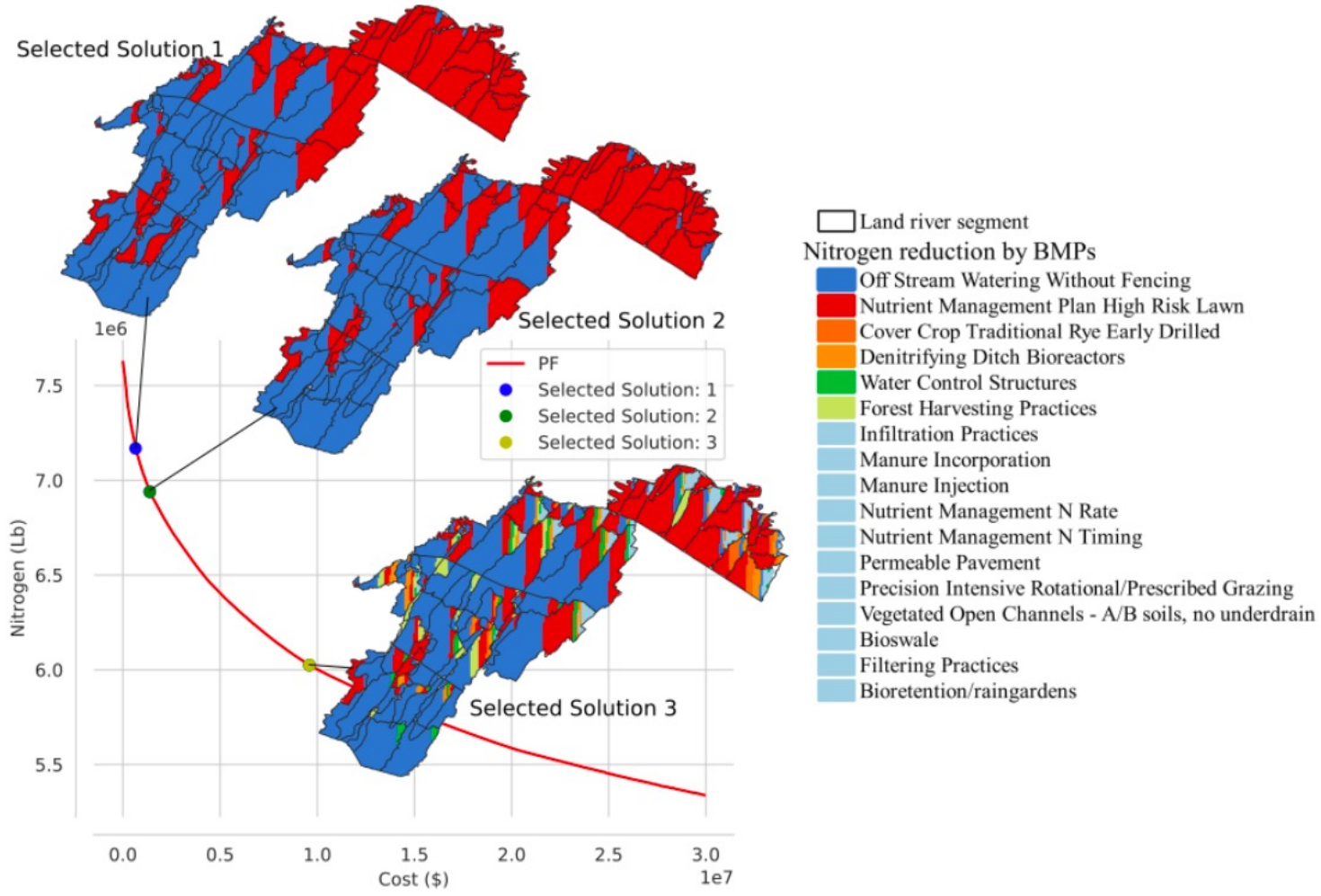
Agenda

- Objective 2: Development of Efficient Multi-objective Optimization Procedures
- Current Accomplishments
- Development of a RESTful API
- Next Steps

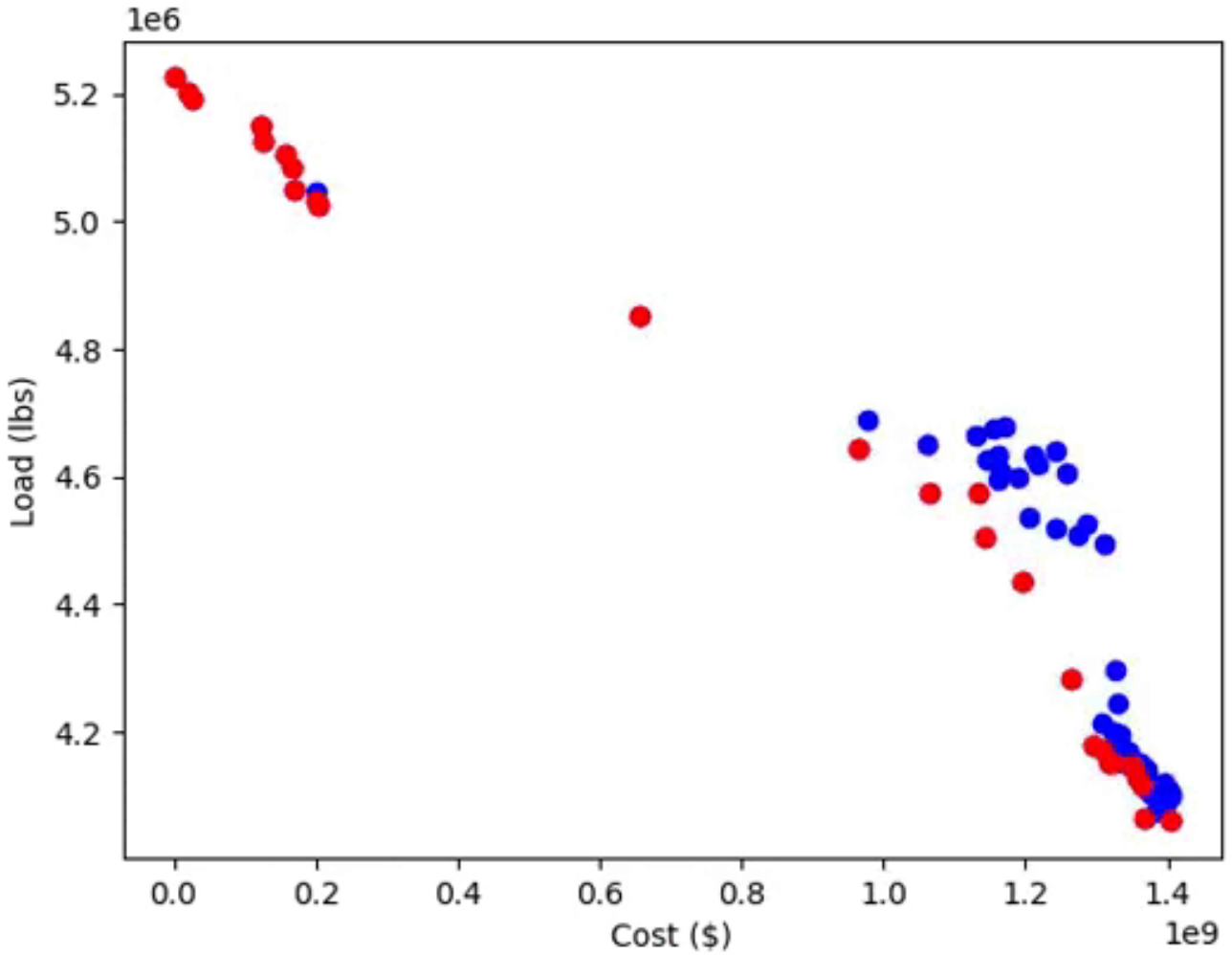
Benefits of multi-objective optimization

BMP Allocation

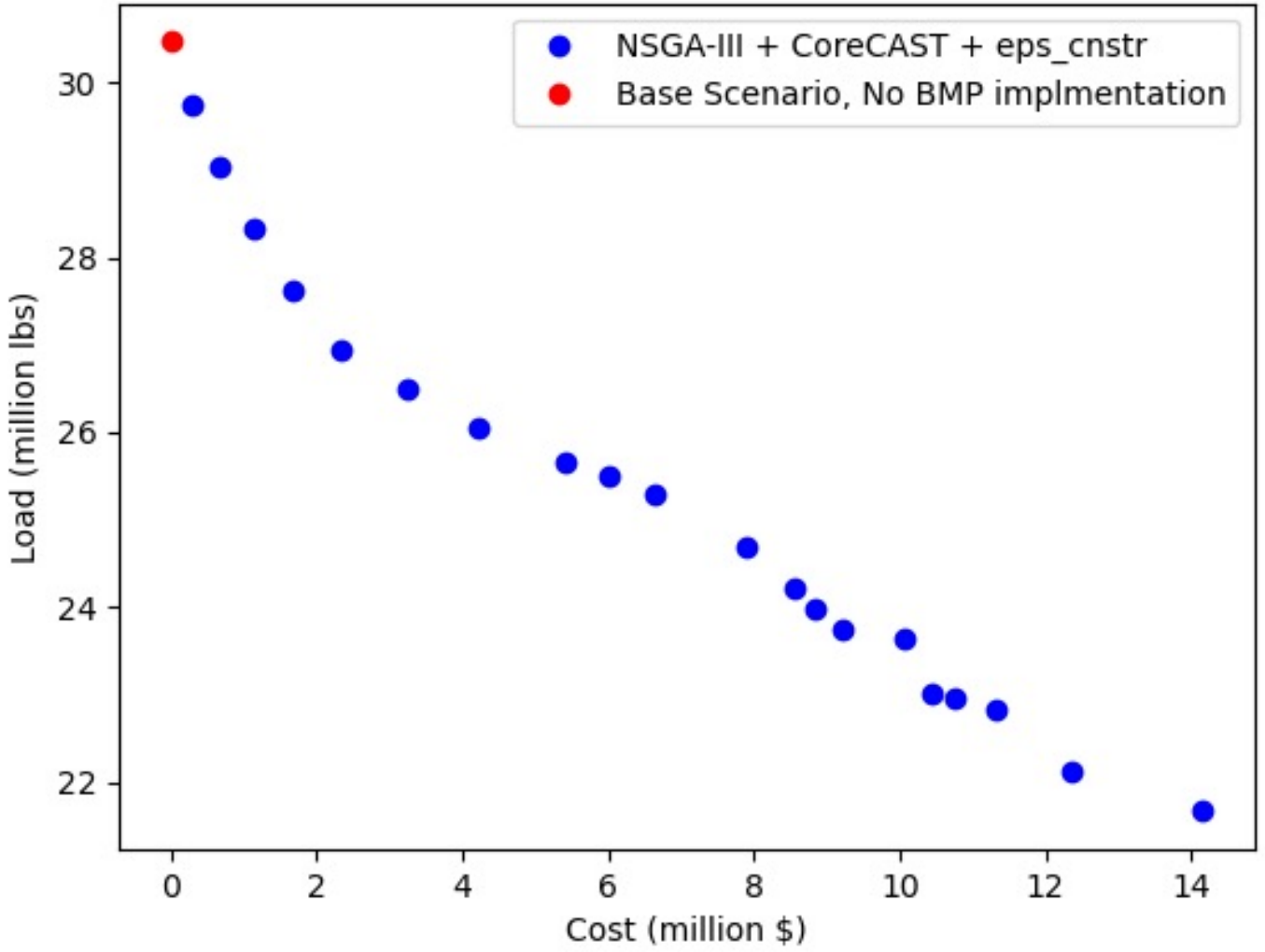
- Two objectives: Cost and Load
- Each point represents a solution with BMPs implementation required to achieve the referred Cost and Load.
- There is a compromise between the Cost and the Loads, we can achieve better load reductions but it will cost more.



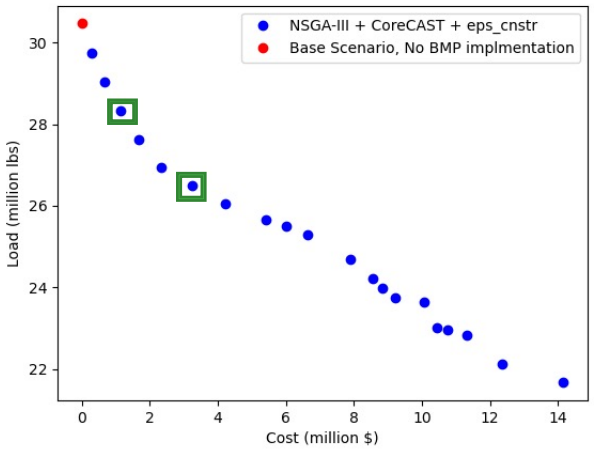
NSGA—III Convergence plot

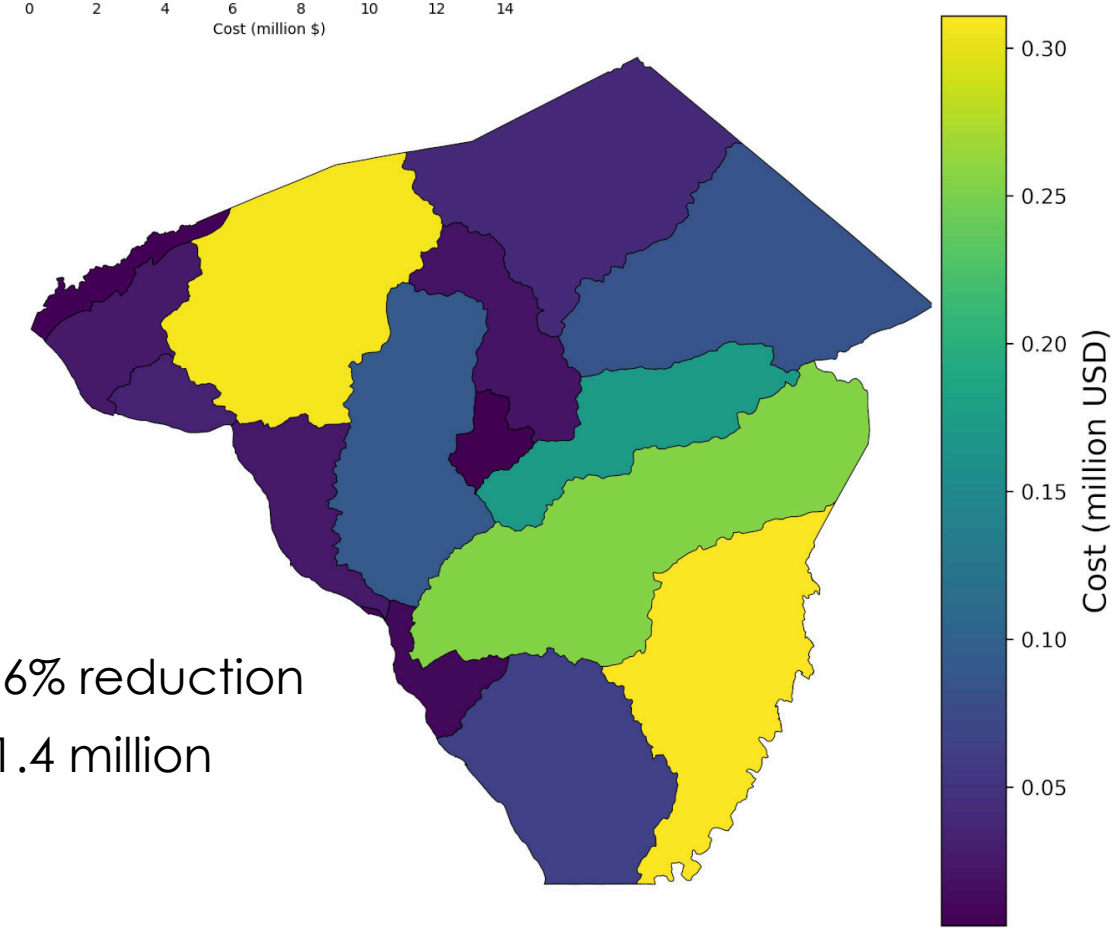


Lancaster, PA



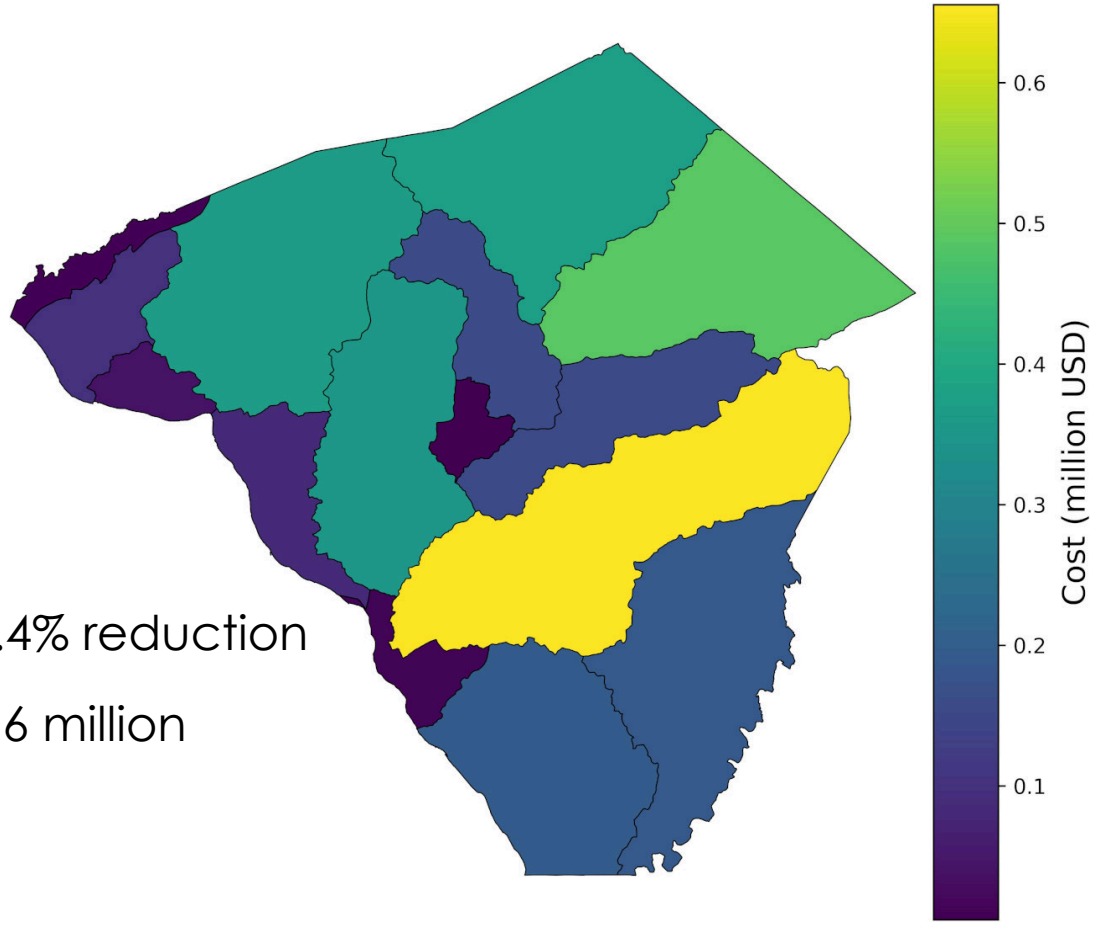
Cost





3.6% reduction

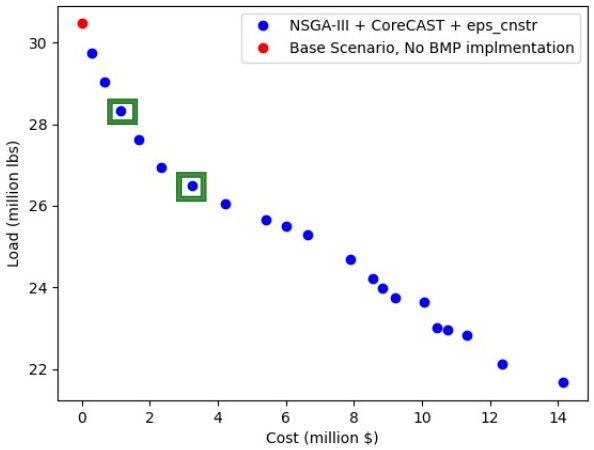
$1.4 million

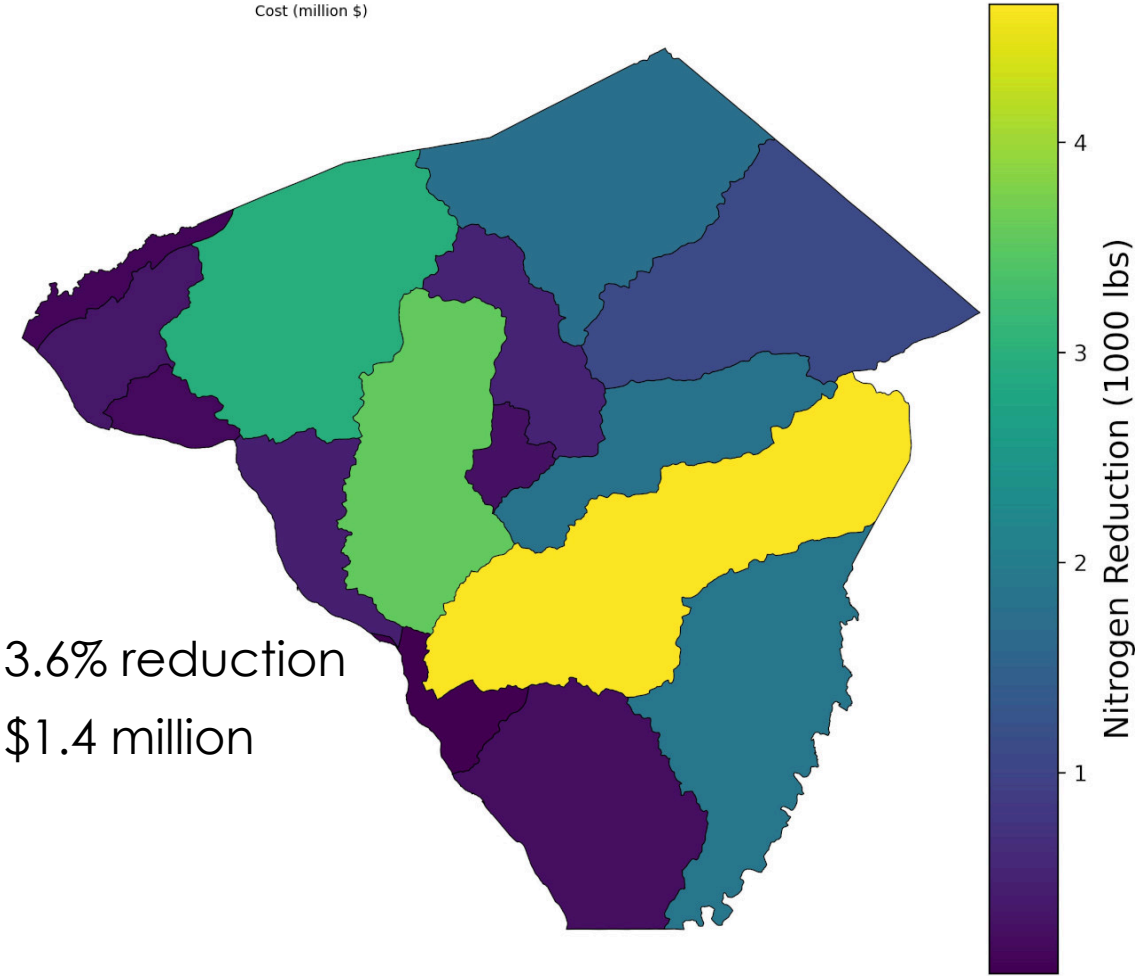


10.4% reduction

$2.6 million

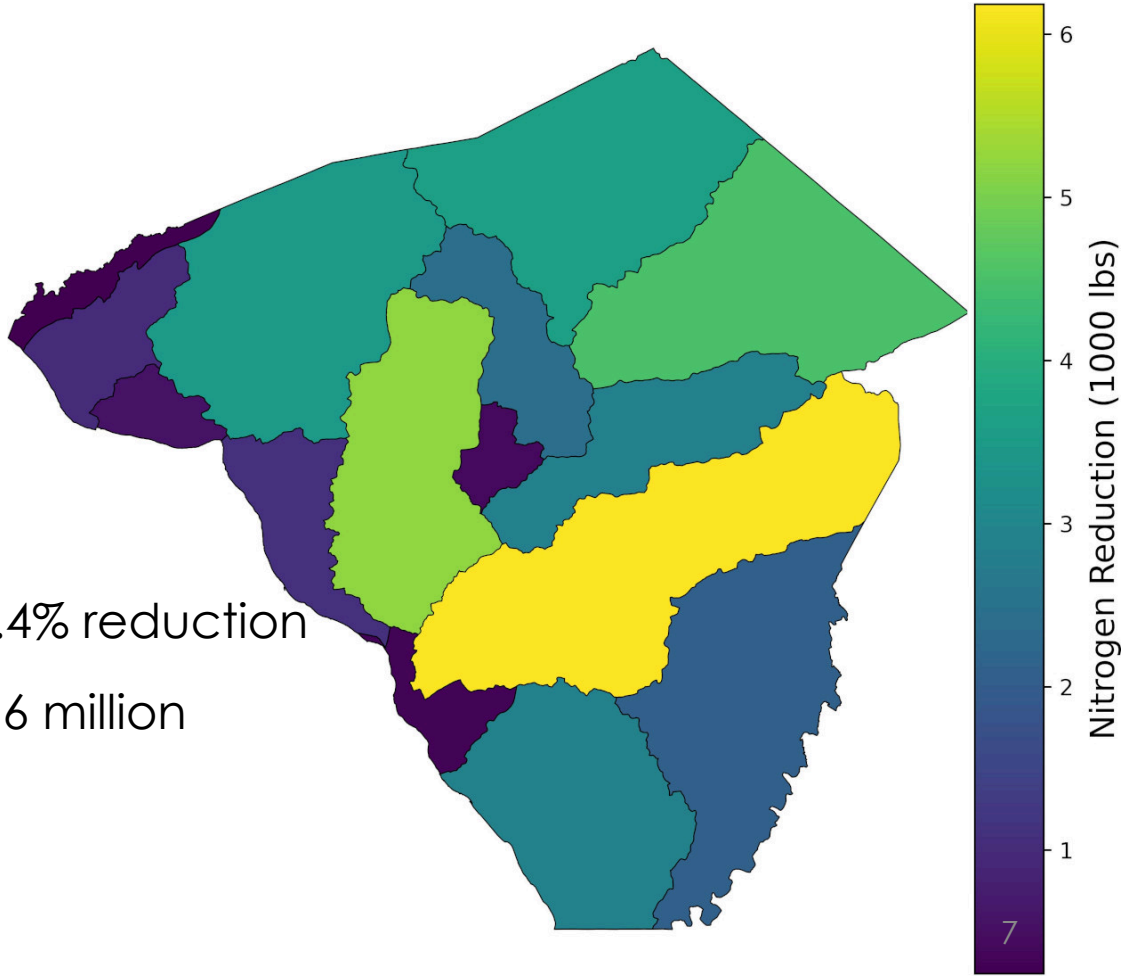
Nitrogen Reduction





3.6% reduction

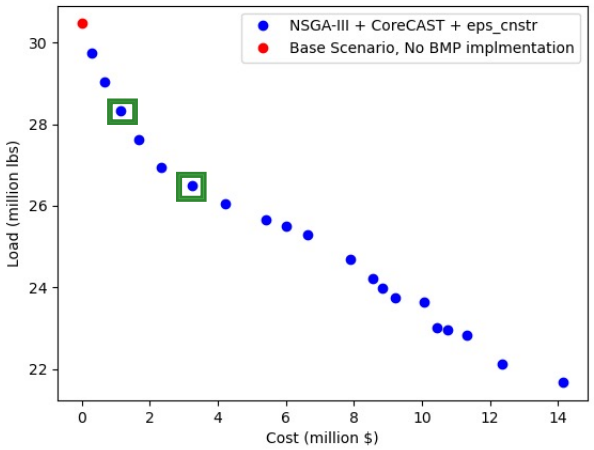
$1.4 million

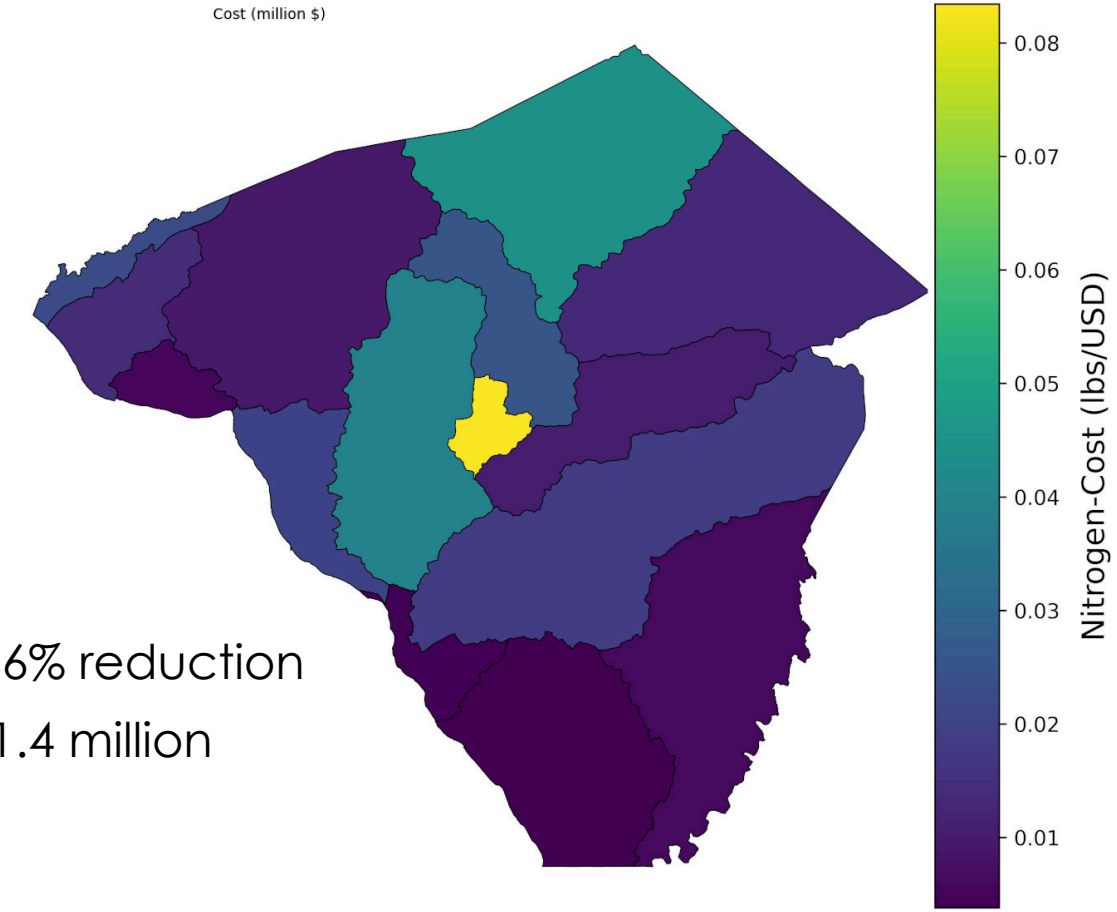


10.4% reduction

$2.6 million

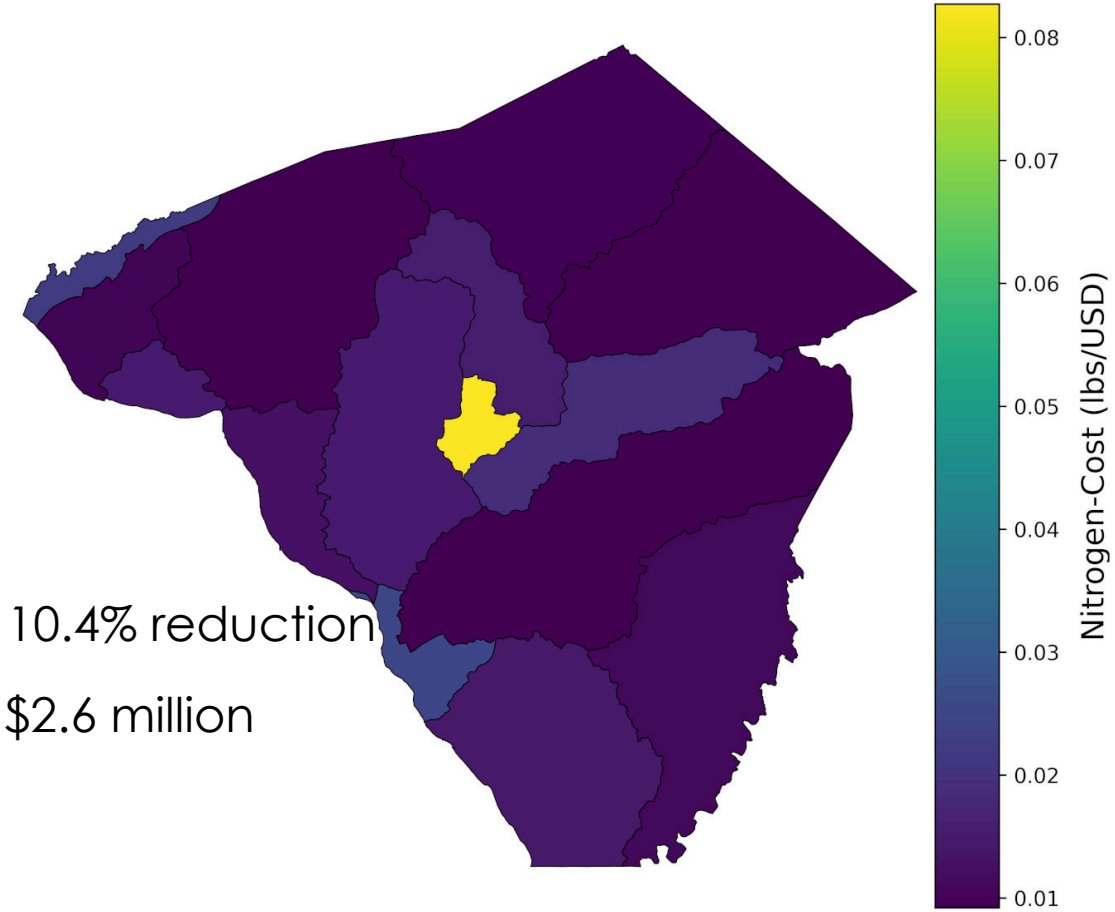
Nitrogen/Cost





3.6% reduction

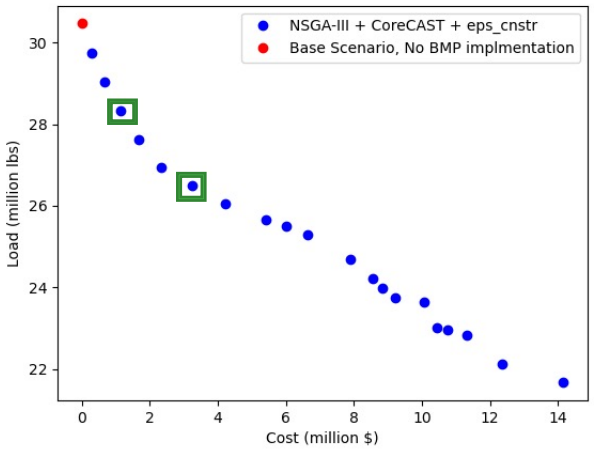
$1.4 million

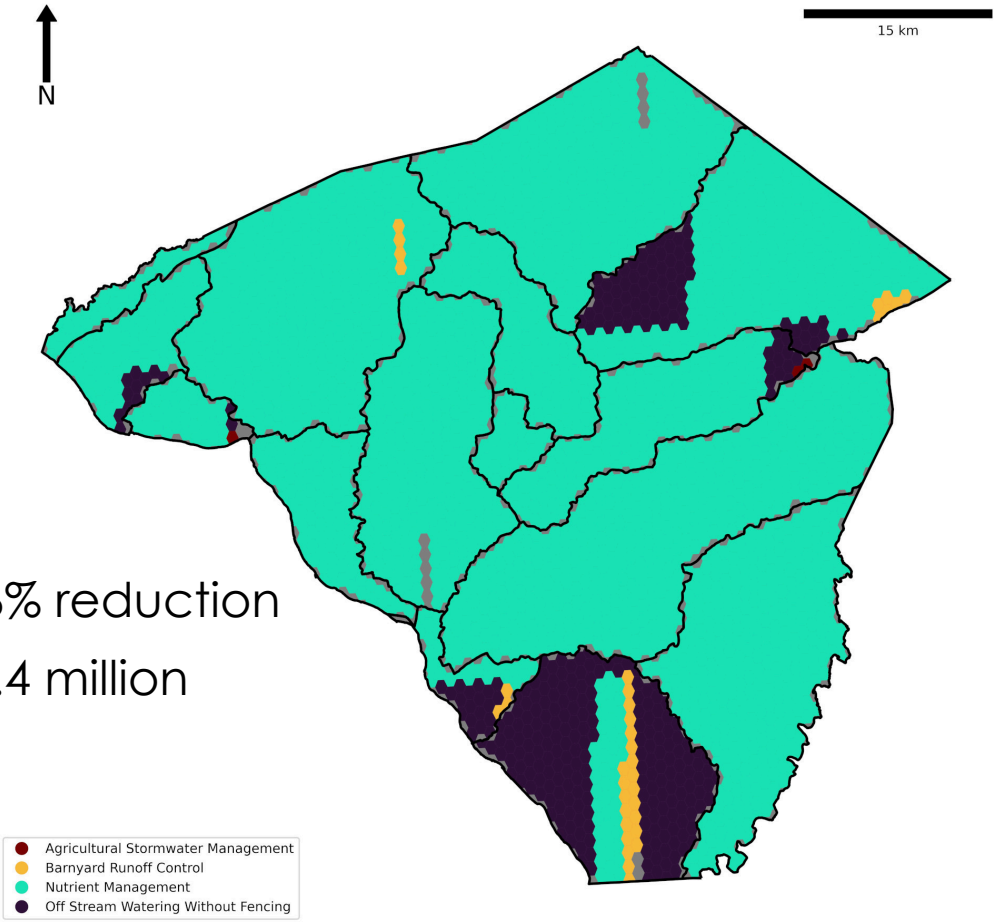


10.4% reduction

$2.6 million

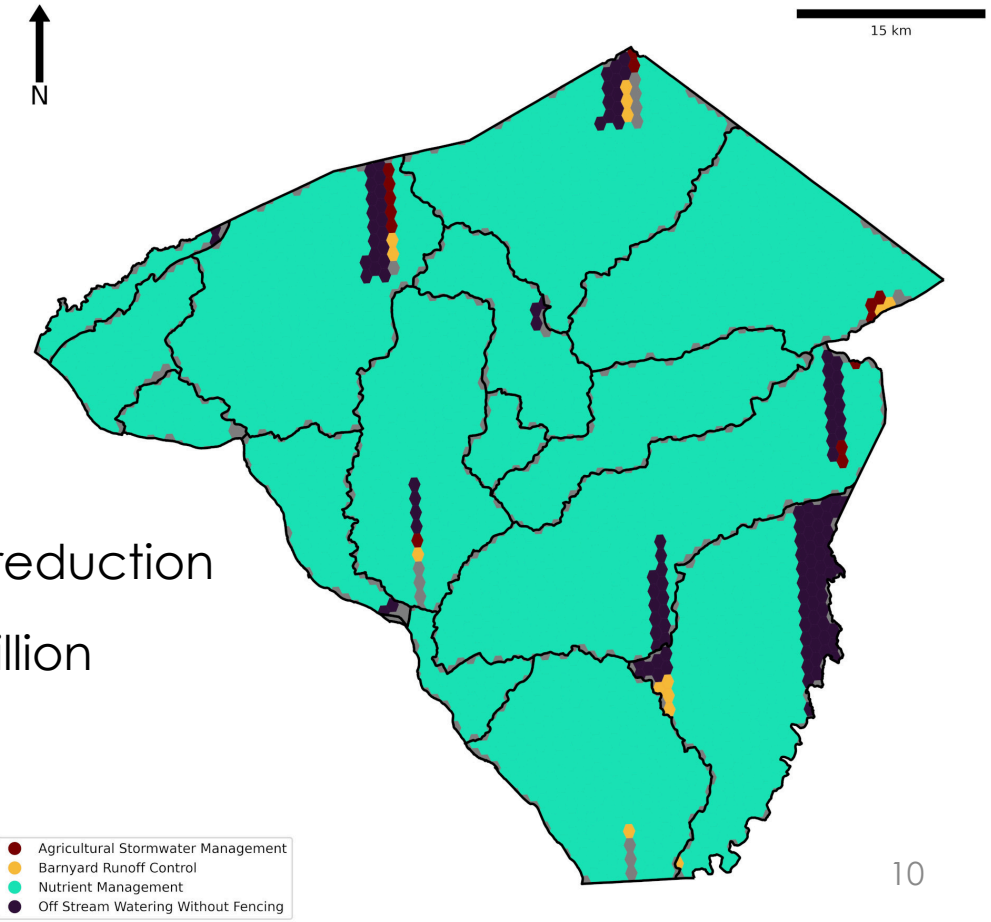
BMP Implement





3.6% reduction

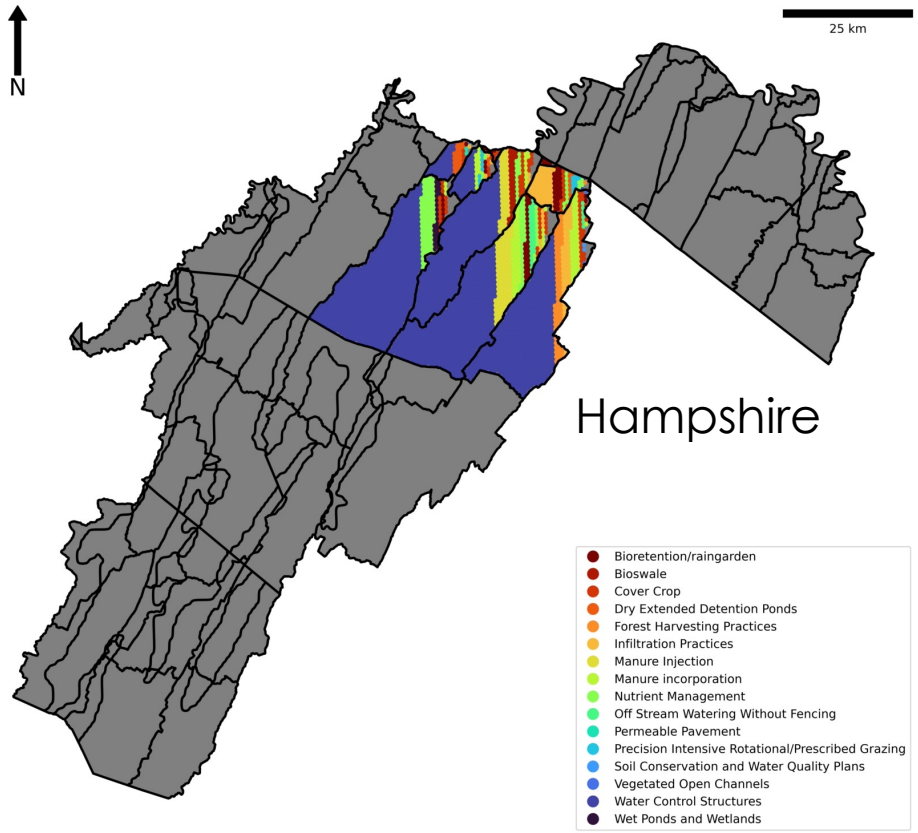
$1.4 million

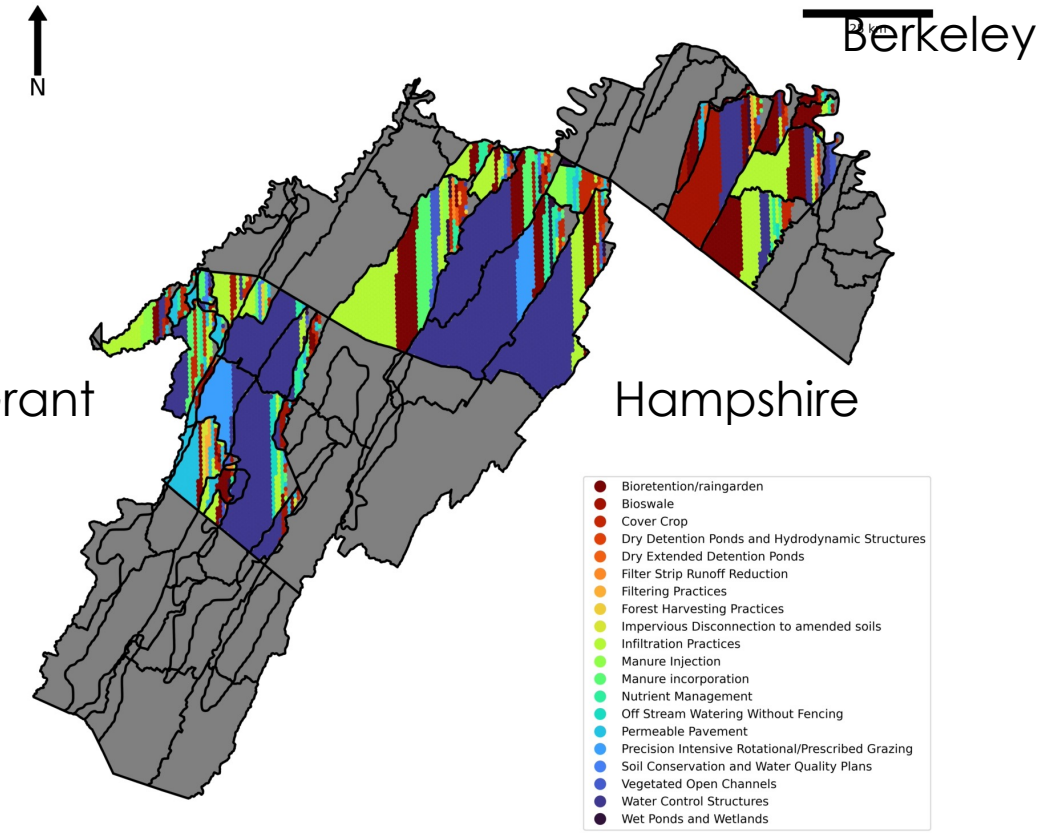


10.4% reduction

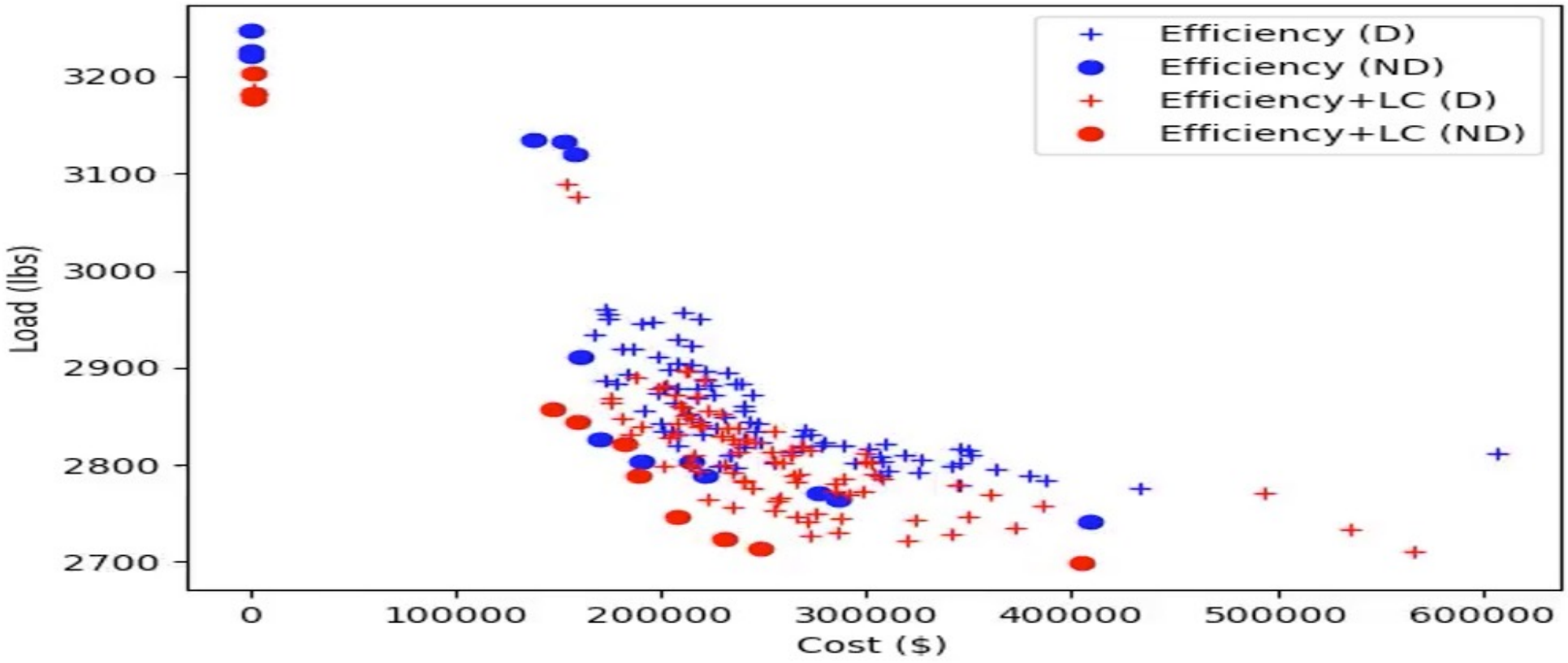
$2.6 million

From previous studies the optimization can benefit when optimizing more counties

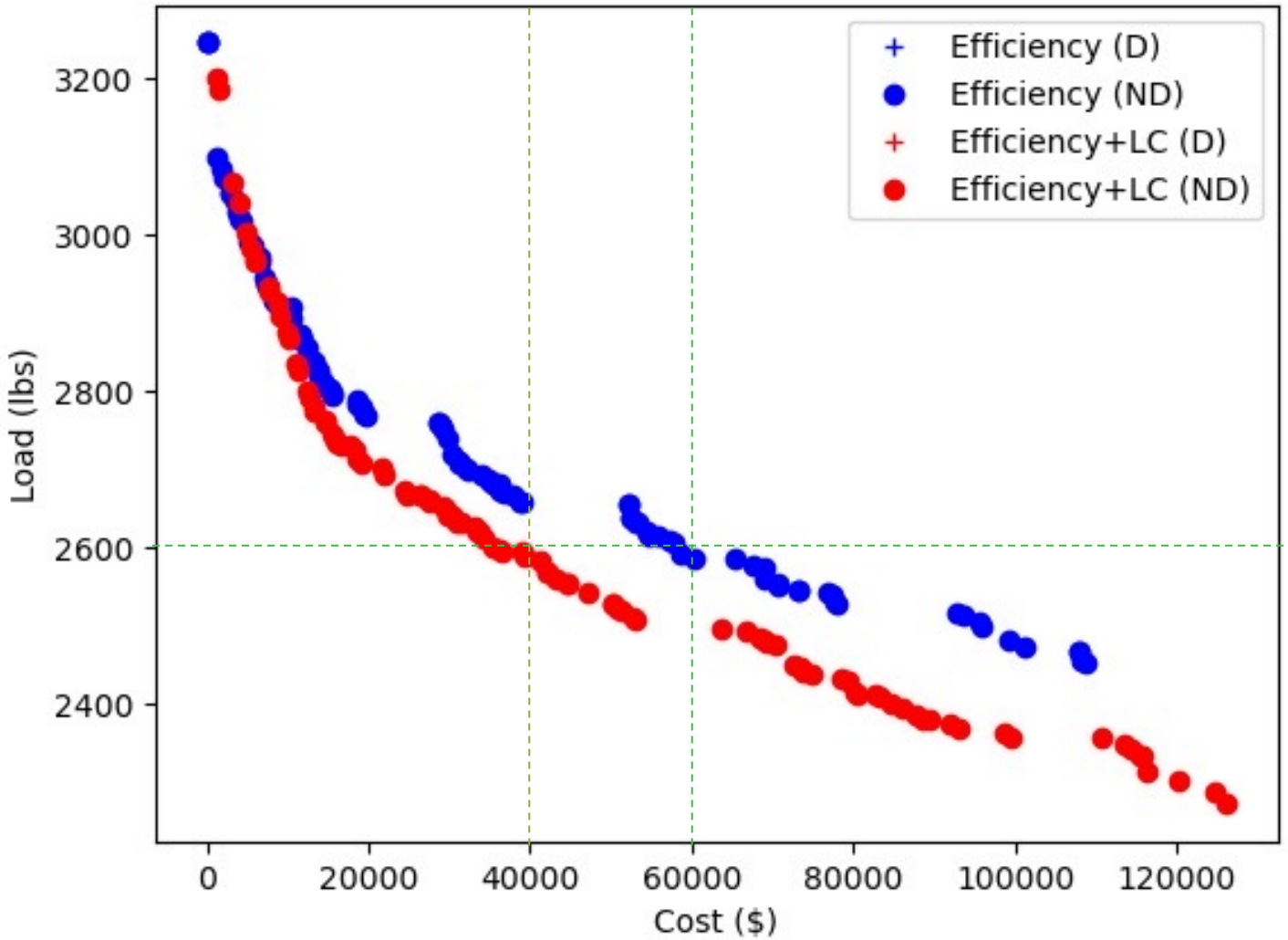




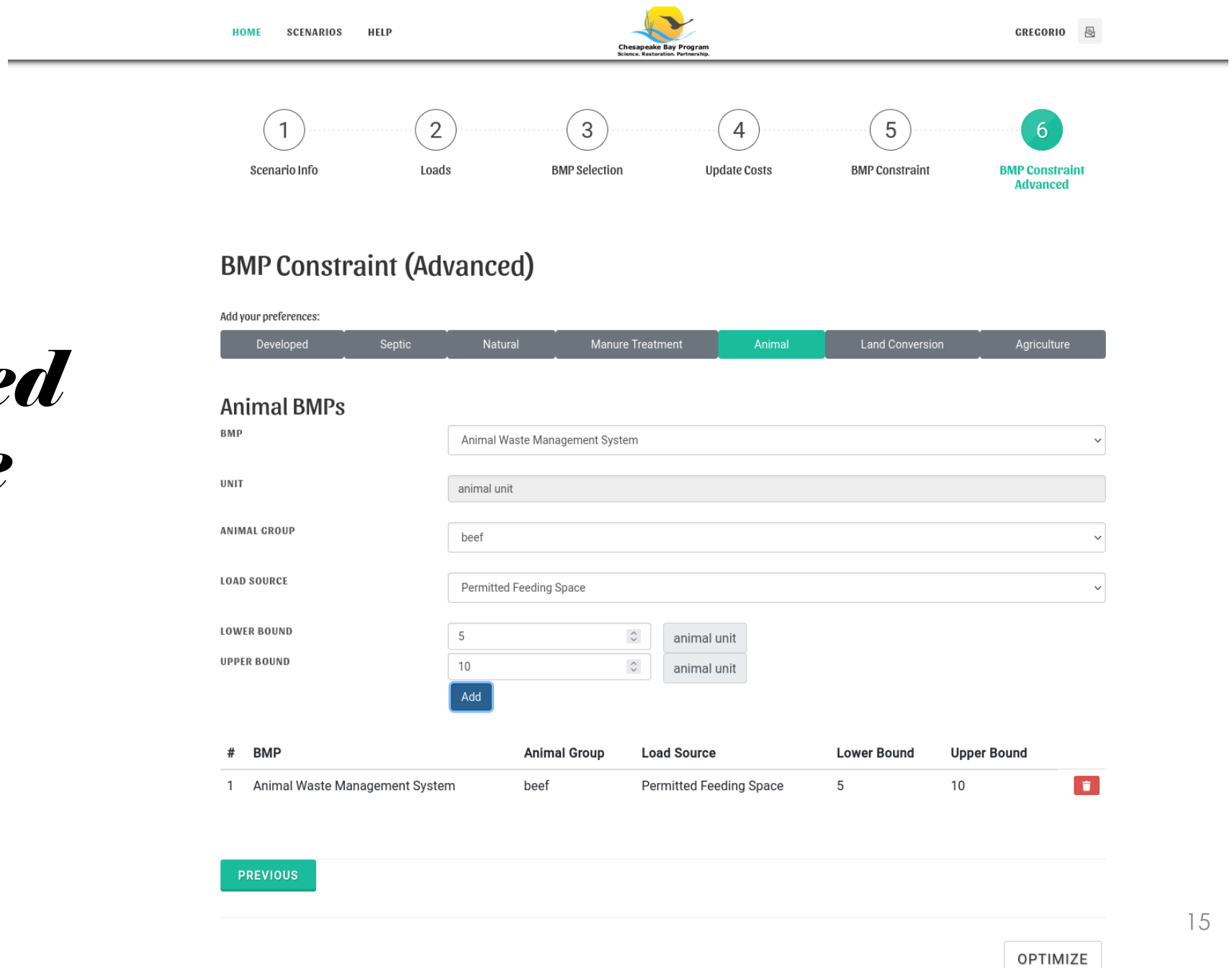
Comparison run: Efficiency vs Efficiency + Land Conversion (CoreCAST)



Comparison run on Tucker county: E vs E + LC (CoreCAST)



Developed interface



Web Interface

- We have an interface web interface that lets set up the users' preferences.
- We placed our efforts in producing an easy to use and self-content platform
- Users can review the optimization results directly on the platform
- The MSU team has new members that are analyzing the results.
- However, web interfaces are usually time-consuming for massive experimentation and validation

Solved problems

- Call CoreCAST
- Communication errors
- Synchronization errors
- We have executed CoreCAST thousands of times, geographically distant, many possible problems in these circumstances have appeared.
- Optimization working
- Plotting

HTTP

User

GET
POST
PUT
DELETE

Request

Response

Representation of a resource

A Server

REpresentational State Transfer API (Roy Fielding)

CBPO Infrastructure

GET
POST
PUT
DELETE

Practitioner

Request

REST API endpoint URL + API method + parameters

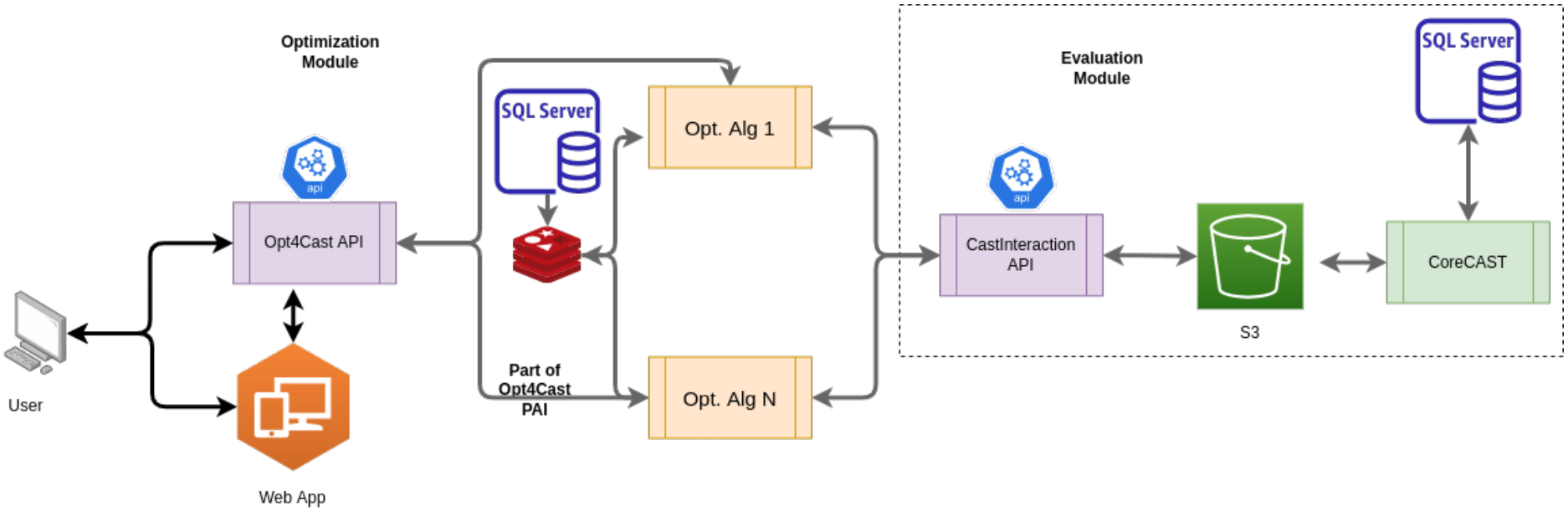
Response

Representation of a resource (JSON)

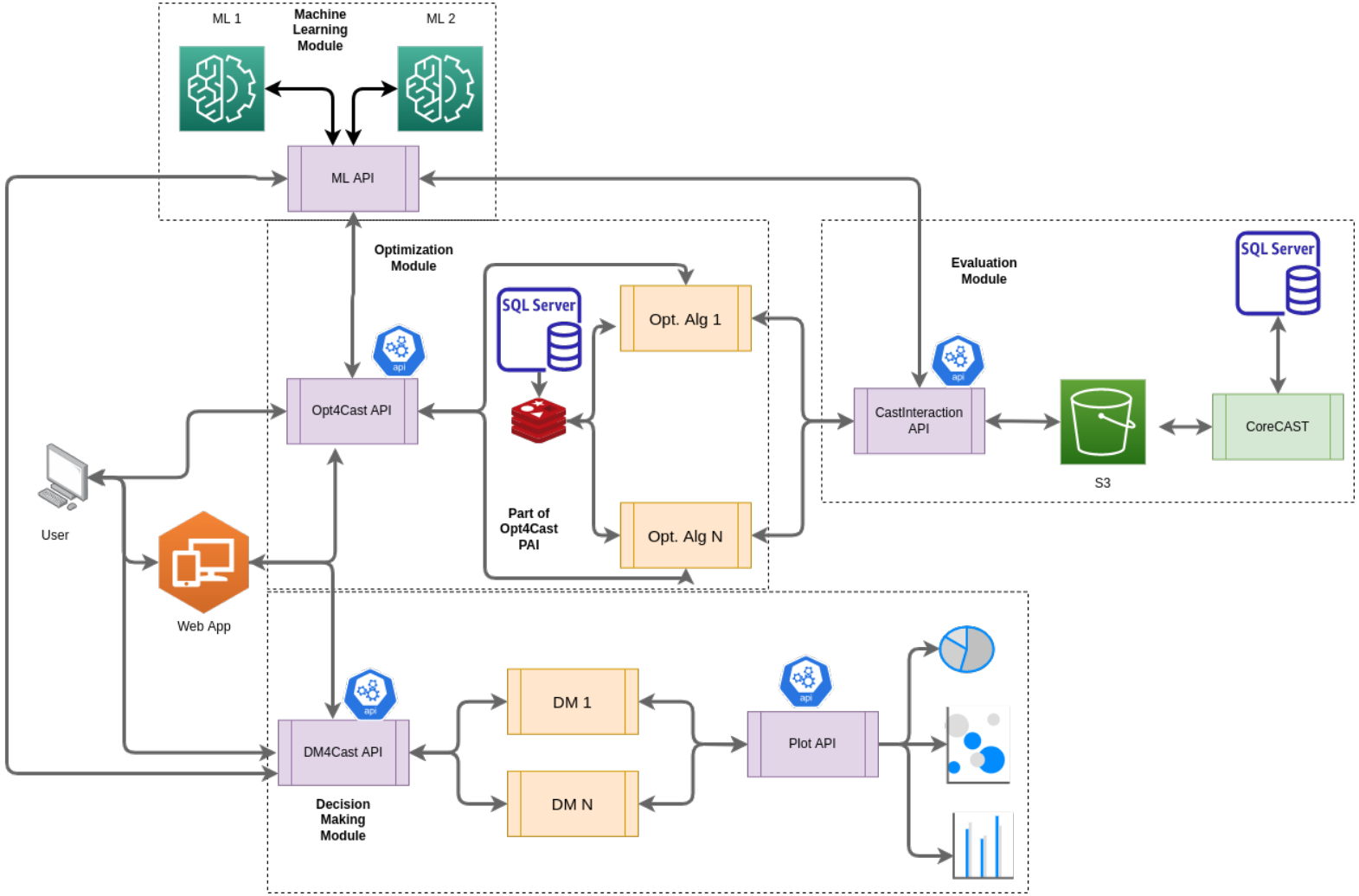
Optimization API (API4Opt)

- 3 different layers: base scenarios, scenario evaluation, optimization execution
  - i.e., users can set up a base scenario, evaluate the reduction of certain BMPs, or trigger a specific algorithm. Users can also download results.
- HTTP request let users use their preferred language to call our API and to manage the results.
- Users can access the results and data any where through the web.

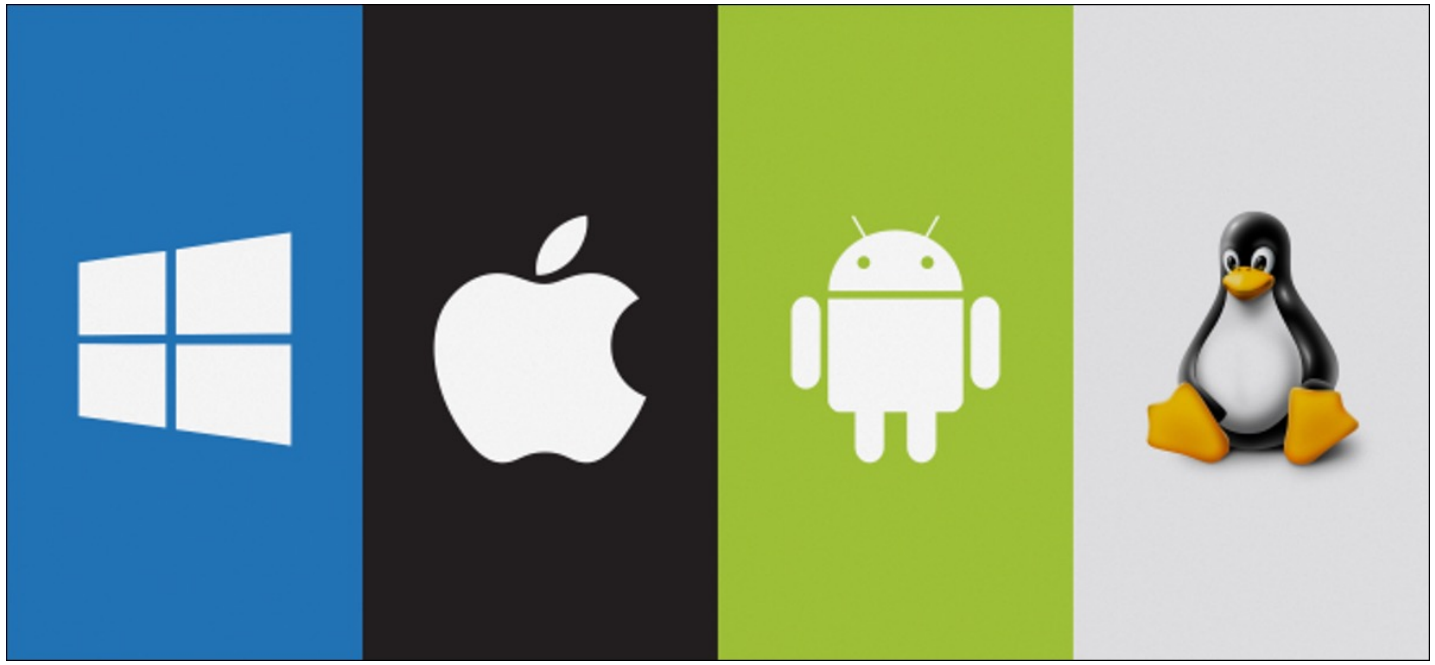
API4Opt = Opt4Cast + CASTInteraction APIs



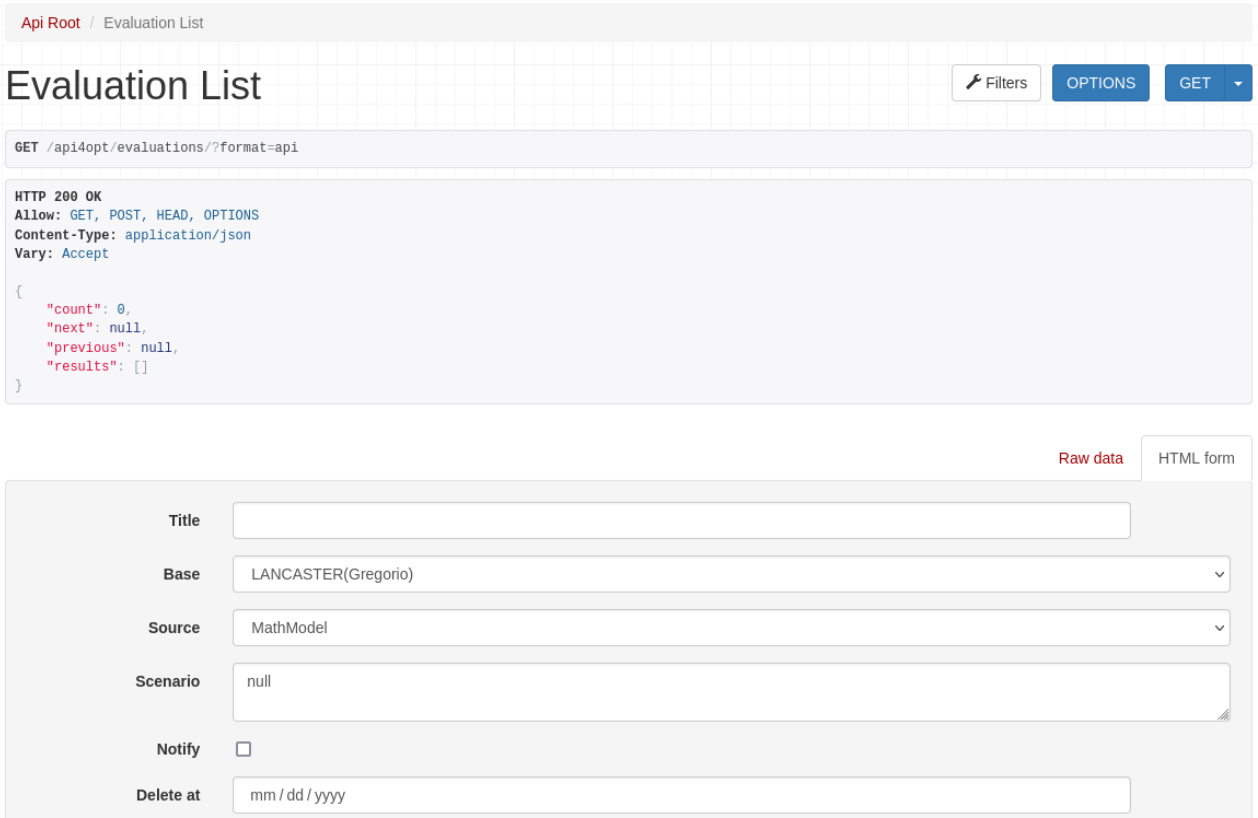
API4Opt = Opt4Cast + CastInteraction APIs

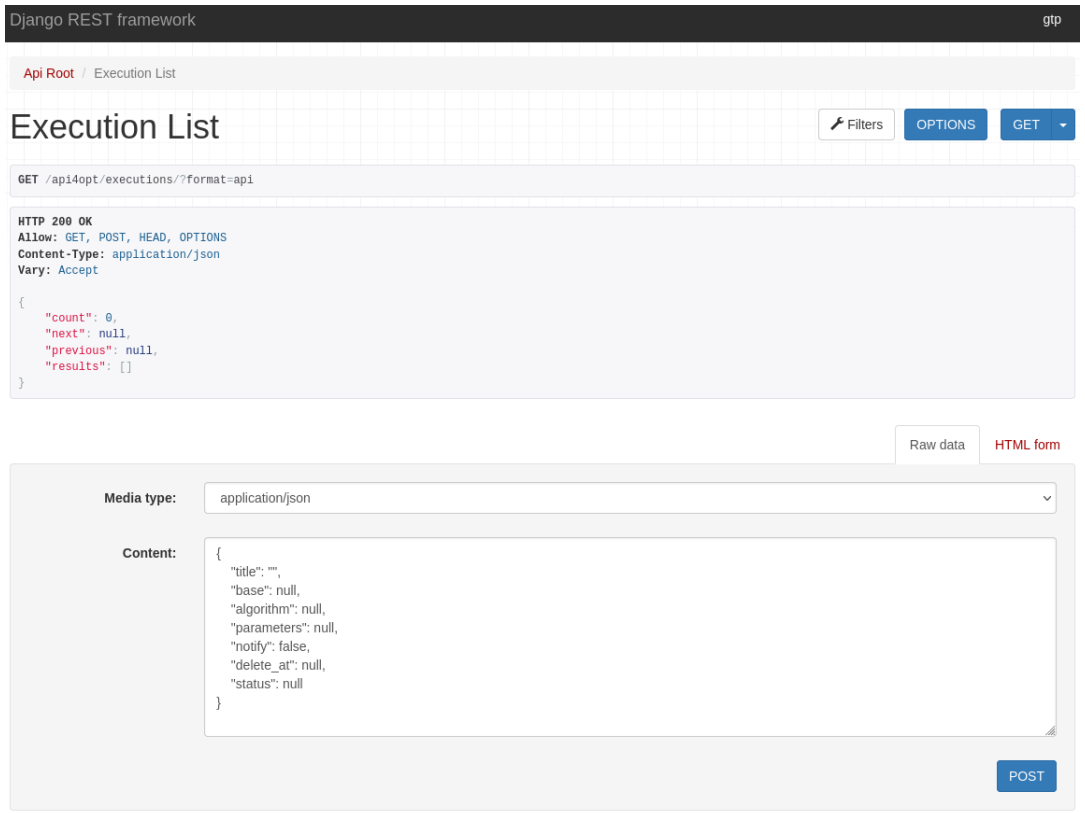


Independence



RESTful API (Web interface)





RESTful API (Code using http requests)

```
import time

data = {"title": "LANCASTER(Gregorio)",
  "atm_dep_data_set_id": 0,
  "back_out_scenario_id": 6611,
  "base_condition_id": 256,
  "base_load_id": 6,
  "cost_profile_id": 4,
  "climate_change_data_set_id": 59,
  "historical_crop_need_scenario_id": 6608,
  "point_source_data_set_id": 36,
  "scenario_type_id": 2,
  "soil_p_data_set_id": 31,
  "source_data_revision_id": 8,
  "counties": [332],
}

url = url_base+'bases/'
response = requests.post(url=url, json=data, headers=head)
if response.status_code == 200:
    cid = json.loads(response.text)['id']
    print('ID: {}'.format(cid))
```

RESTful API (Code using http requests)

Send a scenario to evaluate and wait for its execution

Using CoreCAST

To use CoreCAST to retrieve the information, please set "source" to "C".

By default, source is set to "C" (CoreCAST)

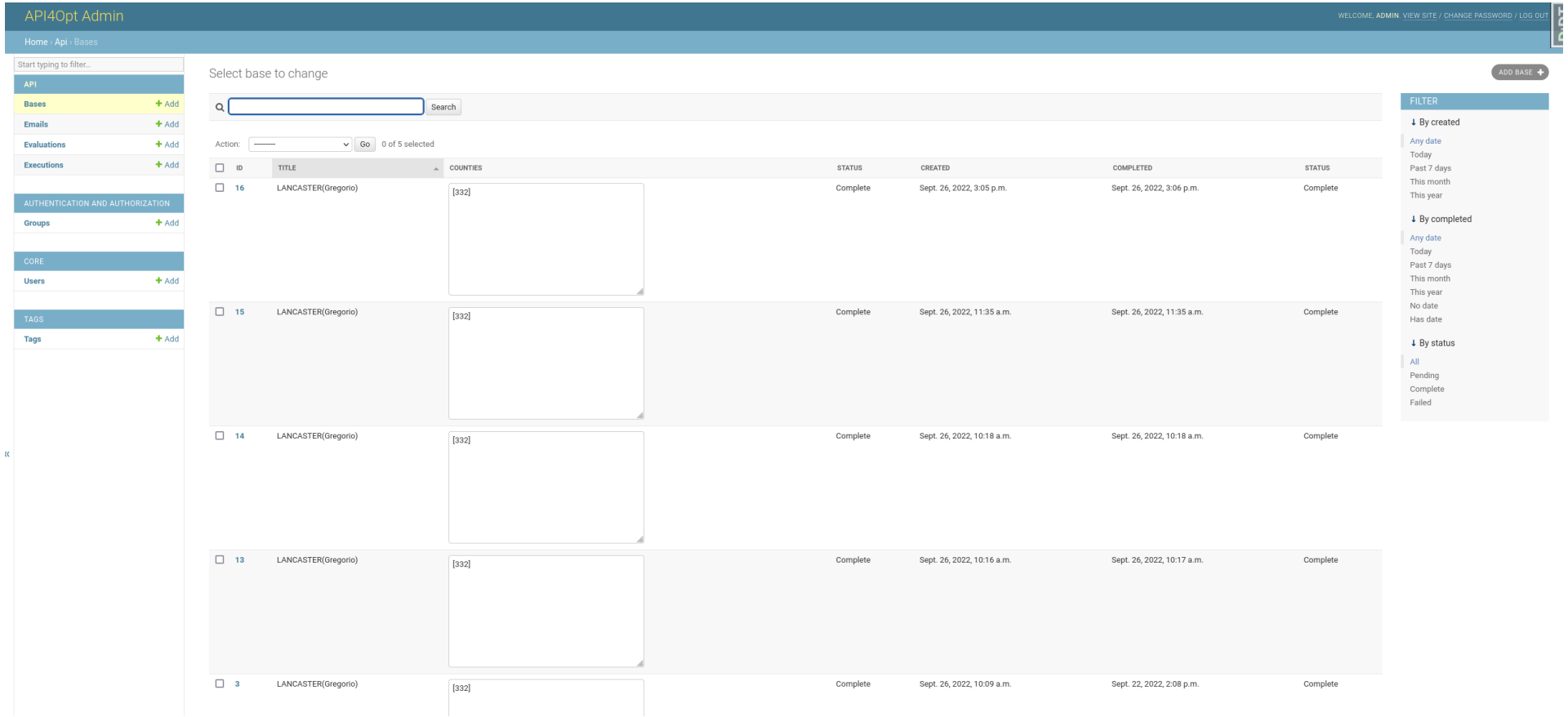
```
import time

base = 3
scenario_lst = [[998, 9, 3, 137, 1.0], [999, 9, 3, 137, 1.0], [999, 9, 6, 137, 1.0], [998, 9, 6, 137, 1.0]]
scenario = json.dumps(scenario_lst)
print (scenario)
data = {
    "source": "C",
    "title" : "Using Corecast",
    "base" : base,
    "scenario" : scenario,
    "notify" : False
}
# Land River Segment, Agency, Load Source, BMP, percentage [0-1]

url = url_base+'evaluations/'
response = requests.post(url=url, json=data, headers=head)


if response.status_code == 200:
    cid = json.loads(response.text)['id']
    print('ID: {}'.format(cid))
```

RESTful API (Admin interface) Base Scenarios



Conclusions and Future Work

- We have developed a RESTful API that provides an additional interface to developers and testers of our optimization approaches.
- The implemented RESTful API provides a mechanism to manage base scenarios, evaluate scenarios and execute optimization algorithms.
- Such a tool will help us reduce the time to evaluate and analyze our optimization algorithms.

- Add this work to the web interface (future work).
- Incorporation of remaining BMPs (future work).
- Improve the decision-making process (future work).

Thank you