Cloud Optimization Solver API

By Eric Burger, 2014

The Cloud Optimization Solver is a web application capable of solving simple Linear Programs and Mixed-Integer Linear Programs. To use the Cloud Optimization Solver, you must interact with the app according to the RESTful Web API below.

Quick Notes:

- To use the app, you must:
  - Set the base url as http://127.0.0.1:5000
  - Set the api_key to ce186 in the query string
- To use the LP solver, set the endpoint to /v1/solve/lp
- To use the Mixed Integer LP solver, set the endpoint to /v1/solve/milp
- An integer or binary variable can be declared using the index of the variable as described below.
- The boundary constraints of a variable (e.g. x is greater than 0 and less than 42) COULD be declared as 2 inequality constraints. However, to make things a little easier, you can set the upper and lower bounds of a variable using the "bounds" list.

Quick Definitions:

- Method: The HTTP Verb/Method (GET or POST) that the app accepts.
- Base URL: The URL address at which a web service is being hosted. This is the persistent portion of a website address (e.g. google.com, berkeley.edu, etc.).
- Endpoint: The address or connection point of a web service.
- API Key: Key or password that permits access to a web service.
- Query String: The portion of a URL address that begins with ? and is followed by key-value pairs in the form: key1=value1&key2=value2&key3=value3
- Body: Content being sent from one computer to another through HTTP communication.
- Content-Type: A description of the content contained in the Body (e.g. HTML, JSON, text, csv).

Linear Program Format:

The Cloud Optimization Solver API requires that Linear Programs are communicated in the matrix format:

$$\text{Minimize: } f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + r$$

$$\text{s.t.}$$

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

$$\boldsymbol{E}\boldsymbol{x} = \boldsymbol{d}$$

$$w \leq x_i \leq y \quad for\ 0 \leq i \leq n-1$$

Where:

x and **c** are vectors with n elements
r is a constant
A is an m by n matrix
b is a vector with m elements
E is a p by n matrix
d is a vector with p elements
w is a lower bound for each value in **x**
y is an upper bound for each value in **x**

Transforming a linear program from the standard format to the matrix format is fairly simple. The example below shows an LP written in both the standard form and the matrix form.

LP Standard Format Example:

$$\text{Minimize: } f(x_0, x_1, x_2) = r + \sum_{i=0}^{n-1} c_i x_i = c_0 x_0 + c_1 x_1 + c_2 x_2 + r$$

$$\begin{aligned} &\text{s.t.} \\ &A_{00}x_0 + A_{01}x_1 + A_{02}x_2 \le b_0 \\ &A_{10}x_0 + A_{11}x_1 + A_{12}x_2 \le b_1 \\ \\ &E_{00}x_0 + E_{01}x_1 + E_{02}x_2 = d_0 \\ &E_{10}x_0 + E_{11}x_1 + E_{12}x_2 = d_1 \end{aligned}$$

$$w \le x_0, x_1, x_2 \le y$$

LP Matrix Format Example:

$$\text{Minimize: } f(x_0, x_1, x_2) = \begin{bmatrix} c_0 & c_1 & c_2 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + r$$

$$\text{s.t.}$$

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

$$\begin{bmatrix} E_{00} & E_{01} & E_{02} \\ E_{10} & E_{11} & E_{12} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \end{bmatrix}$$

$$w \le x_0, x_1, x_2 \le y$$

Linear Program Solver Requests:

LP problems can be sent to the API using the HTTP POST method. To access the API, you must set a valid api_key in the query string (use "ce186"). By default, the API will return results as a JSON string. The contents of the LP are sent to the API as a JSON string contained in the body of the HTTP request. This fact must be noted in the header of the request. The JSON string contains 6 key-value pairs which define the parameters (costs, constraints, equalities, etc) of the LP problem. Because the app is still in an alpha development phase (before beta), the app does not check for incomplete or malformed problem statements. Therefore, all 6 keys must have a value, even if that value is a 0 or an empty list []. Additionally, lists are interpreted as vectors. Therefore, when expressing matrices, each of the inner lists is a row of the matrix. Finally, by default, the solver assumes that variables are free (have bounds of -∞ and ∞). These bounds can be changed by editing the "bounds" list. Each expression in the "bounds" list is another list specifying an index, upper bound, and lower bound.

```
"bounds": [[index, lower, upper], [index, lower, upper]]
```

For example, if "bounds" contains the list [1,0,20], then the variable x_1 will have a lower bound of 0 and an upper bound of 20. To set the lower bound of x_0 to negative infinity or the upper bound to infinity, use the string "None".

```
"bounds": [[0, "None", "None"], [1, 0, 20]]
```

You can also change the default lower and upper bounds by replacing the index with the string "Default". For example, if "bounds" is set to [["Default", 0, 20], [1,0, "None"], then the variable x_1 will have a lower bound of 0 and an upper bound of infinity and every other variable will have a lower bound of 0 and an upper bound of 20.

```
"bounds": [["Default", 0, 20], [1, 0, "None"]]
```

The API returns a JSON string containing 4 key-value pairs. The "code" key contains an integer indicating the success of the solver. The "message" key contains a short string message. The "objective" key contains the value of the cost function and the "x" key contains the variable values that generated the optimal solution for the given problem.

Request:

Parameters

Method	POST
Base URL	http://127.0.0.1:5000
API Endpoint	/v1/solve/lp
Query String	{api_key = ce186 }

Headers

Content-type	application/json

Body

```
{
  "c": [c_0,c_1,c_2],
  "r": r,
  "A": [[A_00,A_01,A_02], [A_10,A_11,A_12]],
  "b": [b_0,b_1],
  "E": [[E_00,E_01,E_02], [E_10,E_11,E_12]],
  "d": [d_0,d_1,d_2,d_3],
  "bounds": [["Default", w, y]]
}
```

Response:

Parameters

Status Code	200 OK

Headers

No Headers

Body

```
{
  "code": 1,
  "message": "Optimal",
  "objective": f(x),
  "x": [x_0,x_1,x_2]
}
```

Code	Message
1	Optimal
0	Not Solved
-1	Infeasible

-2	Unbounded
-3	Undefined

Mixed-Integer Linear Program Solver Requests:

Integer Linear Programs (all variables, $\boldsymbol{x_i}$, are integers) can be expresses in the matrix form below.

$$\begin{aligned}
&\text{Minimize: } f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + r \\
&\text{s.t.} \\
&\boldsymbol{Ax} \leq \boldsymbol{b} \\
&\boldsymbol{Ex} = \boldsymbol{d} \\
&w \leq x_i \leq y \quad for\ 0 \leq i \leq n-1 \\
&x_i \in \mathbb{Z} \quad for\ 0 \leq i \leq n-1
\end{aligned}$$

In a Mixed-Integer Linear Program, only some of the variables are integers. In the example below, variables $\boldsymbol{x_0}$, $\boldsymbol{x_2}$, and $\boldsymbol{x_3}$ are integers but $\boldsymbol{x_1}$ is not. $\boldsymbol{x_0}$ and $\boldsymbol{x_1}$ have bounds of w and y, but $\boldsymbol{x_2}$ and $\boldsymbol{x_3}$ have bounds of 0 and infinity.

$$\text{Minimize: } f(x_0, x_1, x_2, x_3) = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} + r$$

$$\text{s.t.}$$

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

$$\begin{aligned}
&w \leq x_0, x_1 \leq y \\
&0 \leq x_2, x_3 \leq \infty \\
&x_0, x_2, x_3 \in \mathbb{Z}
\end{aligned}$$

The Cloud Optimization Solver API allows you to set variables as integers or binary integers by adding additional key-value pairs to the request body. The "integer" key contains a list of indexes representing integer variables. For example, to solve the MILP example above, you would set the "integer" key to [0, 2, 3]. By default, the solver assumes that integer variables are free (between -∞ and ∞). The "binary" key contains a list of indexes representing binary integer variables (0 or 1). The MILP example can be solved using the Cloud Optimization Solver by implementing the API below.

Request:

Parameters

Method	POST
Base URL	http://127.0.0.1:5000
API Endpoint	/v1/solve/milp
Query String	{api_key = ce186 }

Headers

Content-type application/json

Body

```
{
  "c": [c0,c1,c2,c3],
  "r": r,
  "A": [[A00,A01,A02,A03], [A10,A11,A12,A13]],
  "b": [b0,b1],
  "E": [],
  "d": [],
  "bounds": [["Default", w, y],[2, 0, "None"],[3, 0, "None"]]
  "integer": [0,2,3]
  "binary": []
}
```

Response:

(See Linear Program Solver Response above)

API Example Using Python and Requests Module:

Below we present an example of how to solve an LP problem using Python, the Requests module, and the Cloud Optimization Solver API.

Problem:
A furniture factory makes two products, wooden desks (x) and chairs (y). The productivity of the factor is limited by two machines: one to build the furniture and the other to paint it. Each desk requires 50 minutes to build and 24 minutes to paint. Each chair requires 30 minutes to build and 33 minutes to paint. Each week, a maximum of 40 hours can be spent building furniture and 35 hours can be spent painting. Finally, for every desk that the factory produces, it must also make at least one chair. If the profit is $400 per desk and $200 per chair, how many of each should the factory produce each week to maximize profits?

Standard Format:

$$\text{Minimize: } f(x,y) = -400x - 200y$$

$$\text{s.t.}$$

$$50x + 30y \le 40*60$$

$$24x + 33y \le 35*60$$

$$x - y \le 0$$

$$x, y \ge 0$$

Matrix Format:

$$\text{Minimize: } f(x,y) = \begin{bmatrix} -400 & -200 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\text{s.t.}$$

$$\begin{bmatrix} 50 & 30 \\ 24 & 33 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \le \begin{bmatrix} 2400 \\ 2100 \\ 0 \end{bmatrix}$$

$$x, y \ge 0$$

Python Code (CloudLPSolverExample.py):
The Python code below demonstrates how to solve the example problem above using the Cloud Optimization Solver API.

```
# Import requests and json modules
import requests
import json

def main():
    # Optimization Example:

    # Using an Application Programming Interface (API) for the
    # Cloud Optimization Solver App: Results returned in JSON format.
    # Set url address.
    base = 'http://127.0.0.1:5000'
    endpoint = '/v1/solve/lp'
    address = base + endpoint

    # Set query string (i.e. http://url.com/?key=value).
    query = {'api_key':'YOUR_API_KEY_HERE'}
    # Set header.
    header = {'Content-Type':'application/json'}

    # Formulate LP problem
    c = [-400,-200]
    A = [[50,30],[24,33],[1,-1]]
    b = [40*60,35*60,0]
    E = []
    d = []
    bounds = [["Default",0,"None"]]
    # Set body (also referred to as data or payload). Body is a JSON string.
    payload = {'c':c,'r':0,'A':A,'b':b,'E':E,'d':d,'bounds':bounds}
    body = json.dumps(payload)

    # Form and send request. Set timeout to 2 minutes. Receive response.
    response = requests.request('post', address, data=body, params=query, headers=header, timeout=12

    print response.url
    # Text is JSON string. Convert to Python dictionary/list
    #print response.text
    print json.loads( response.text )

main()
```