Cradle-to-Gate Life Cycle Assessment for Geopolymer Concrete

The purpose of this LCA is to

1- identify the most critical process (that has principal contribution in more impact categories) with less effort, less time and more accuracy, as compared to traditional stacked bar charts

2- Coalesce midpoint, endpoint LCA's, and process contributions in a visually appealing way to summarize LCA results (also usefull for posters and graphical abstracts of scientific publications)

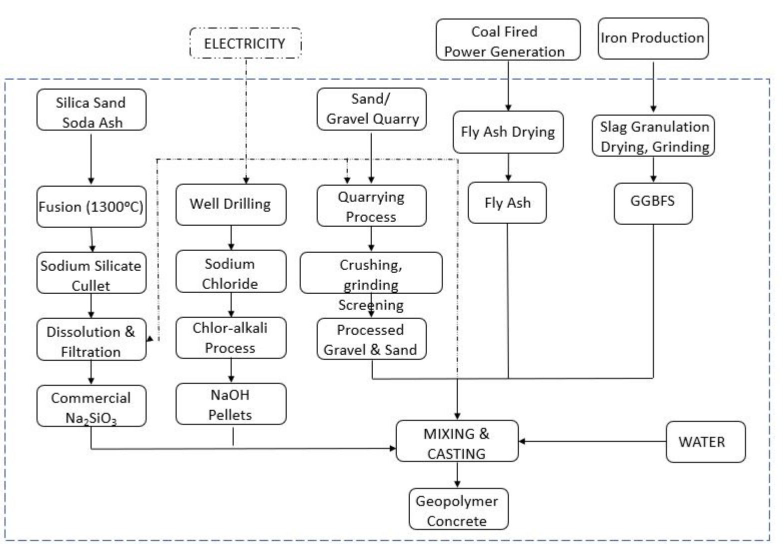
In [1]:
```
import pandas as pd
import uuid
import math
from datetime import datetime
from matplotlib import pyplot as plt
import matplotlib.mlab as mlab
from matplotlib import rcParams
import matplotlib.patches as mpatches
import seaborn as sns
import plotly.express as px
import plotly.io as pio
pio.renderers.default = "plotly_mimetype+notebook"
import plotly.graph_objects as go
import pandas as pd
%matplotlib inline
```

Goal and Scope

The goal of this LCA is to identify the most critical process contributing to the environmental impacts of geopolymer concrete. The functional unit is 1m3 of concrete. The cradle-to-gate system boundary is selected and is presented in figure below.

In [2]:
```
from PIL import Image
img = Image.open('Sys Boundary.jpg')
img
```

Out[2]:



Life Cycle Inventory Analysis

In [3]:
```
geopolymer_data = pd.read_csv('geopolymer.csv')
print("The testing matrix has been taken from experimental work by us in the lab")
geopolymer_data
```

```
The testing matrix has been taken from experimental work by us in the lab
```

Out[3]:

	in/out	flow	amount	units
0	in	Fly ash	480	Kg
1	in	GGBFS	120	Kg
2	in	Sand	600	Kg
3	in	Gravel	900	Kg
4	in	NaOH	80	Kg
5	in	Sodium Silicate	160	Kg
6	in	Mixing	2	kWh
7	out	Geopolymer	1	m3

In [4]:
```
pip install -U olca-ipc
```

```
Requirement already satisfied: olca-ipc in c:\users\hp\anaconda3\lib\site-packages (0.0.12)
Requirement already satisfied: requests in c:\users\hp\anaconda3\lib\site-packages (from olca-ipc) (2.27.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\hp\anaconda3\lib\site-packages (from requests->olca-ipc) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\anaconda3\lib\site-packages (from requests->olca-ipc) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\hp\anaconda3\lib\site-packages (from requests->olca-ipc) (1.26.9)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hp\anaconda3\lib\site-packages (from requests->olca-ipc) (3.3)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
cd C:\Users\HP\Downloads\olca-ipc.py-master
```

```
C:\Users\HP\Downloads\olca-ipc.py-master
```

In [6]:
```
import olca

params = {'mathtext.default': 'regular' }

client = olca.Client(8080)
client
```

Out[6]:
```
<olca.ipc.Client at 0x255e1b543d0>
```

In [7]:
```
dt_object = datetime.fromtimestamp(datetime.timestamp(datetime.now()))
```

In [8]:
```
#creating flows
volume = client.find(olca.FlowProperty, 'Volume')

geopolymer = olca.product_flow_of('geopolymer', volume)
geopolymer
geopolymer.description = '1 m3 Geopolymer concrete added via olca-ipc on %s.' % (dt_object)
client.insert(geopolymer)
```

Out[8]:
```
'ok'
```

In [9]:
```
geopolymer_data['flow'][0]
geopolymer
```

Out[9]:


```
{
  "@id": "63eb19b1-88ed-40c3-a4f4-8f65c81a28ab",
  "@type": "Flow",
  "description": "1 m3 Geopolymer concrete added via olca-ipc on 2023-01-08 01:11:56.159817.",
  "flowProperties": [
    {
      "@id": "",
      "@type": "FlowPropertyFactor",
      "conversionFactor": 1.0,
      "flowProperty": {
        "@id": "93a60a56-a3c8-22da-a746-0800200c9a66",
        "@type": "FlowProperty",
        "name": "Volume"
      },
      "referenceFlowProperty": true
    }
  ],
  "flowType": "PRODUCT_FLOW",
  "lastChange": "2023-01-07T20:11:56.608458Z",
  "name": "geopolymer",
  "version": "00.00.000"
}
```


In [10]:
```
mass = client.find(olca.FlowProperty, 'Mass')
x = [0,1,2,3,4,5]

for a in x:
    flow_name = geopolymer_data['flow'][a]
    print(flow_name)
    flow_name = olca.product_flow_of(geopolymer_data['flow'][a], volume)
    flow_name.description = 'flow for', geopolymer_data['flow'][a],'added via olca-ipc on %s.' % (dt_object)
    client.insert(flow_name)
flow_name
```

```
Fly ash
GGBFS
Sand
Gravel
NaOH
Sodium Silicate
```

Out[10]:


```
{
  "@id": "1123a426-5ddd-4a5c-b8fa-07e30ed02318",
  "@type": "Flow",
  "description": [
    "flow for",
    "Sodium Silicate",
    "added via olca-ipc on 2023-01-08 01:11:56.159817."
  ],
  "flowProperties": [
    {
      "@id": "",
      "@type": "FlowPropertyFactor",
      "conversionFactor": 1.0,
      "flowProperty": {
        "@id": "93a60a56-a3c8-22da-a746-0800200c9a66",
        "@type": "FlowProperty",
        "name": "Volume"
      },
      "referenceFlowProperty": true
    }
  ],
  "flowType": "PRODUCT_FLOW",
  "lastChange": "2023-01-07T20:11:57.163476Z",
  "name": "Sodium Silicate",
  "version": "00.00.000"
}
```


In [11]:
```
#Creating Processes
dt_object = datetime.fromtimestamp(datetime.timestamp(datetime.now()))
x=[0,1,2,3,4,5,6,7]

for a in x:
    process_name = geopolymer_data['flow'][a]
    print(process_name)
    process_name = olca.process_of(geopolymer_data['flow'][a])
    process_name.description = 'Added via olca-ipc on %s.' % (dt_object)
    client.insert(process_name)
process_name
```

```
Fly ash
GGBFS
Sand
Gravel
NaOH
Sodium Silicate
Mixing
Geopolymer
```

Out[11]:


```
{
  "@id": "e734854c-6d46-4506-9750-b3897e95279a",
  "@type": "Process",
  "description": "Added via olca-ipc on 2023-01-08 01:11:57.205133.",
  "lastChange": "2023-01-07T20:11:57.466676Z",
  "name": "Geopolymer",
  "processType": "UNIT_PROCESS",
  "version": "00.00.000"
}
```


In [12]:
```
#adding input/output flows in the geopolymer process

target_refs = []

for a in x:
    all_obj = client.get_descriptors(olca.Flow)
    cache = [obj for obj in all_obj if geopolymer_data['flow'][a] == obj.name]
    target_refs.append(cache)
```

In [13]:
```
process_descriptor = client.get_descriptors(olca.Process)
process_list = []
id_list = []

for process in process_descriptor:
    process_list.append(process.name)
    id_list.append(process.id)

process_df = pd.DataFrame(list(zip(process_list,
                          id_list)),
                          columns=['name', 'id'])
process_df
```

Out[13]:

	name	id
0	heat and power co-generation, biogas, gas engi...	[illegible]
1	heat and power co-generation, biogas, gas engi...	[illegible]
2	market for land tenure, arable land, measured ...	[illegible]
3	heat and power co-generation, natural gas, ...	[illegible]
4	electricity voltage transformation from medium...	[illegible]
...	...	...
19579	Gravel	[illegible]
19580	NaOH	[illegible]
19581	Sodium Silicate	[illegible]
19582	Mixing	[illegible]
19583	Geopolymer	e734854c-6d46-4506-9750-b3897e95279a

19584 rows × 2 columns

In [14]:
```
#creating product systems
product_system = client.create_product_system(process_df['id'][process_df.last_valid_index()],
                                               default_providers='prefer',
                                               preferred_type='UNIT_PROCESS')
```

In [15]:
```
psID = product_system.id
psID
```

Out[15]:
```
[illegible]
```

Life Cycle Impact Assessments

The LCIA can be performed by clicking "quick calculations" button in OpenLCA, available on the product system that has been created with [illegible]

In [16]:
```
cd
```

```
C:\Users\HP
```

In [17]:
```
LCIA_results= pd.read_csv('LCIA.csv')
LCIA_results
```

Out[17]:

	Impact Category	Unit	Value	Ecosystem	Human Health	Resources
0	Climate Change	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1	Ozone Depletion	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2	Particulate Matter Formation	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3	Acidification	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
4	Eutrophication	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

In [18]:
```
process_contributions= pd.read_csv('process_contribution.csv')
process_contributions
```

Out[18]:

	Flow	GHG	Ozone Depletion	Particulate Matter	Acidification	Eutrophication
0	Gravel	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1	Sand	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2	Fly Ash	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3	GGBFS	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
4	SS	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
5	SH	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Interpretation/Visualization

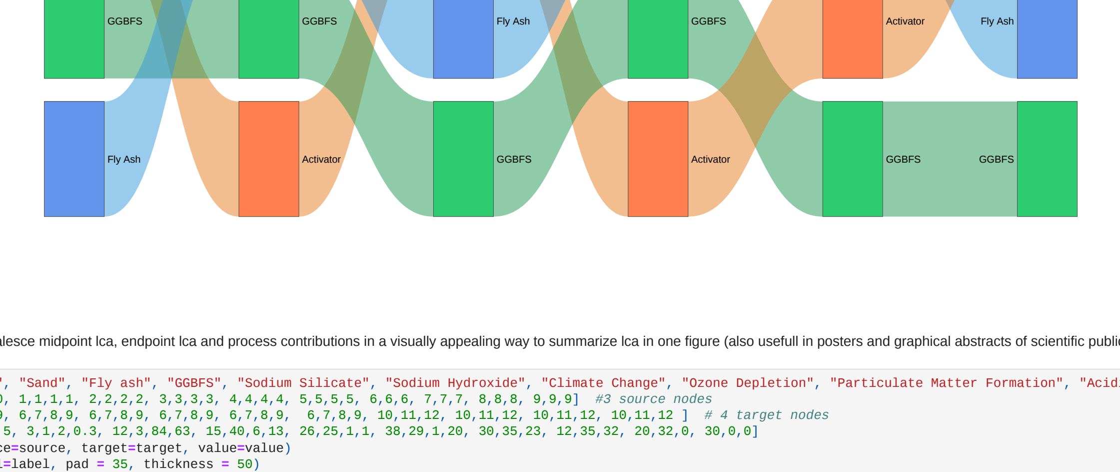
Visualization 1:

This visualization indicates the most critical process (that has principal contribution in more impact categories)

The traditional stacked bar chats used in LCA visualization are intricate to analyze the most critical product and more often than not, the conclusions drawn are not accurate. For example, previously it was concluded that activators are the most critical process in impacts of geopolymer concrete, whereas this visualization indicates that both activators and fly ash are principal contributors in 3 impact categories, and therefore both should be deemed as critical processes

In [19]:
```
label = ["Activator", "GGBFS", "Fly Ash", "Activator", "GGBFS", "Fly Ash", "Activator", "Fly Ash", "GGBFS", "Activator", "GGBFS", "Activator", [illegible]
source = [illegible]
target = [illegible]
link = [illegible]
node = [illegible]
data= go.Sankey(link=link, node=node)
fig = go.Figure(data)
fig.update_layout(
    hovermode = 'x',
    font=dict(size = 12, color = 'black')
)
fig.add_annotation(
    x= 0.1,
    y=1.065,
    text = "Climate Change",
    font=dict(size = 12, color = 'black')
)
fig.add_annotation(
    x= 0.25,
    y=1.065,
    text = "Ionizing Radiation",
    font=dict(size = 12, color = 'black')
)
fig.add_annotation(
    x= 0.4,
    y=1.065,
    text = "Ozone Depletion",
    font=dict(size = 12, color = 'black')
)
fig.add_annotation(
    x= 0.55,
    y=1.065,
    text = "Particulate Matter",
    font=dict(size = 12, color = 'black')
)
fig.add_annotation(
    x= 0.7,
    y=1.065,
    text = "Acidification",
    font=dict(size = 12, color = 'black')
)
fig.add_annotation(
    x= 0.85,
    y=1.065,
    text = "Eutrophication",
    font=dict(size = 12, color = 'black')
)

fig.show()
```

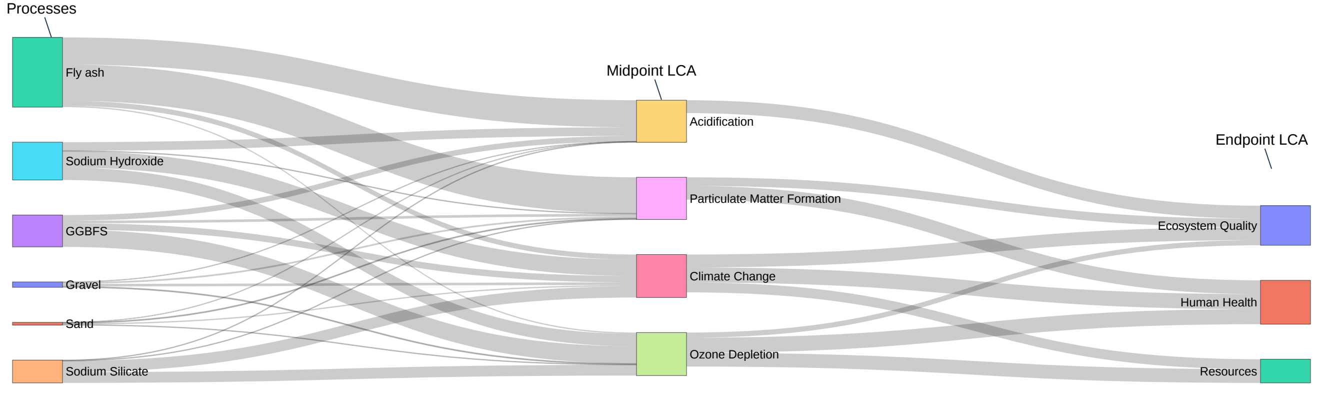


Visualization 2:

This visualization coalesce midpoint lca, endpoint lca and process contributions in a visually appealing way to summarize lca in one figure (also usefull in posters and graphical abstracts of scientific publications)

In [20]:
```
label = ["Gravel", "Sand", "Fly ash", "GGBFS", "Sodium Silicate", "Sodium Hydroxide", "Climate Change", "Ozone Depletion", "Particulate Matter Formation", "Acidification", "Ecosystem Qualit
source = [0,0,0,0, 1,1,1,1, 2,2,2,2, 3,3,3,3, 4,4,4,4, 5,5,5,5, 6,6,6, 7,7,7, 8,8,8, 9,9,9]  #3 source nodes
target = [6,7,8,9, 6,7,8,9, 6,7,8,9, 6,7,8,9, 6,7,8,9,  6,7,8,9, 10,11,12, 10,11,12, 10,11,12, 10,11,12 ]  # 4 target nodes
value = [6,2,4,0.5, 3,1,2,0.3, 12,3,84,63, 15,40,6,13, 26,25,1,1, 38,29,1,20, 30,35,23, 12,35,32, 20,32,0, 30,0,0]
link = dict(source=source, target=target, value=value)
node = dict(label=label, pad = 35, thickness = 50)
data= go.Sankey(link=link, node=node)
```

In [21]:
```
fig = go.Figure(data)
fig.update_layout(
    hovermode = 'x',
    font=dict(size = 12, color = 'black')
)
fig.add_annotation(
    x= 0.03,
    y=1,
    text = "Processes",
    font=dict(size = 15, color = 'black')
)
fig.add_annotation(
    x= 0.5,
    y=0.82,
    text = "Midpoint LCA",
    font=dict(size = 15, color = 'black')
)
fig.add_annotation(
    x= 0.97,
    y=0.62,
    text = "Endpoint LCA",
    font=dict(size = 15, color = 'black')
)
fig.show()
```



Acknowledgments

I would like to thank Mr. Julian Rickert for his amazing tutorials in integrating open LCA and Jupyter Notebooks

I am gratefull to open LCA for providing open source software license

I thank ecoinvent for providing academic license of ecoinvent database