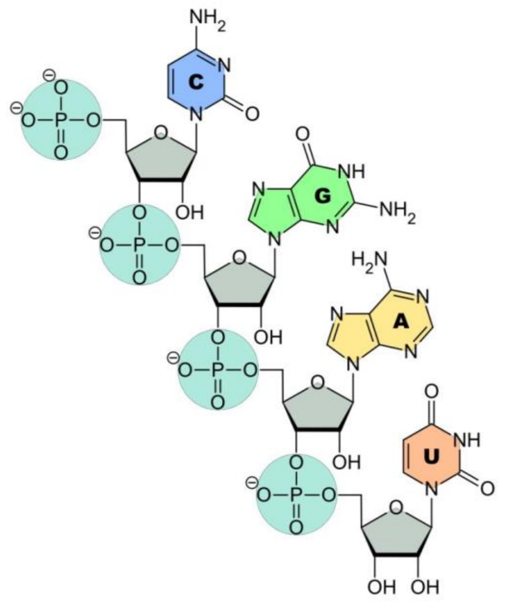
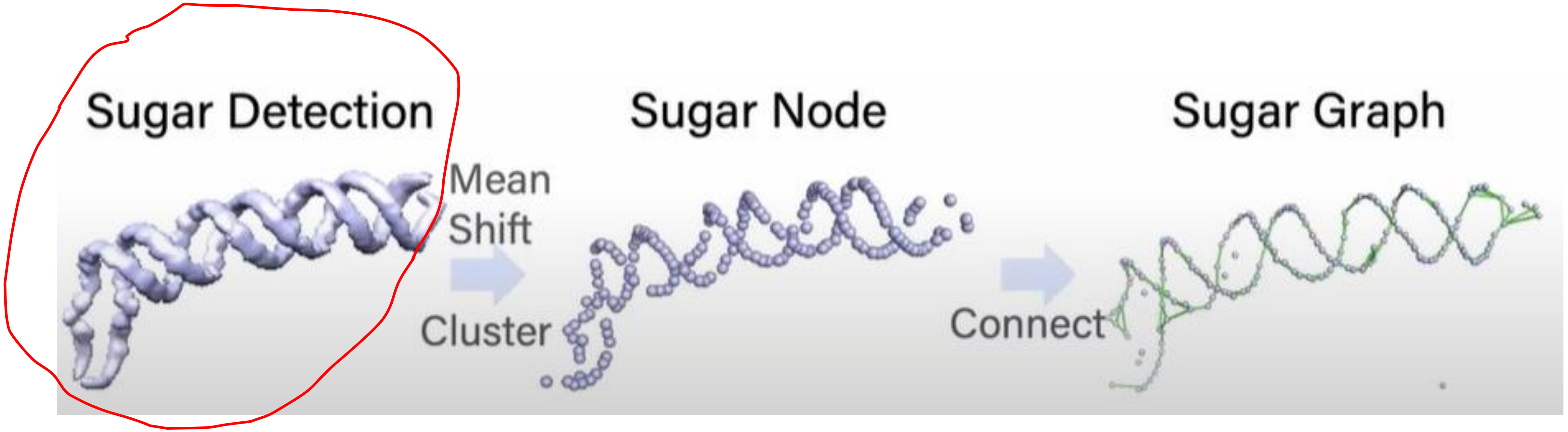
How CryoREAD Traces Backbones

Week 7 – 11/13/14

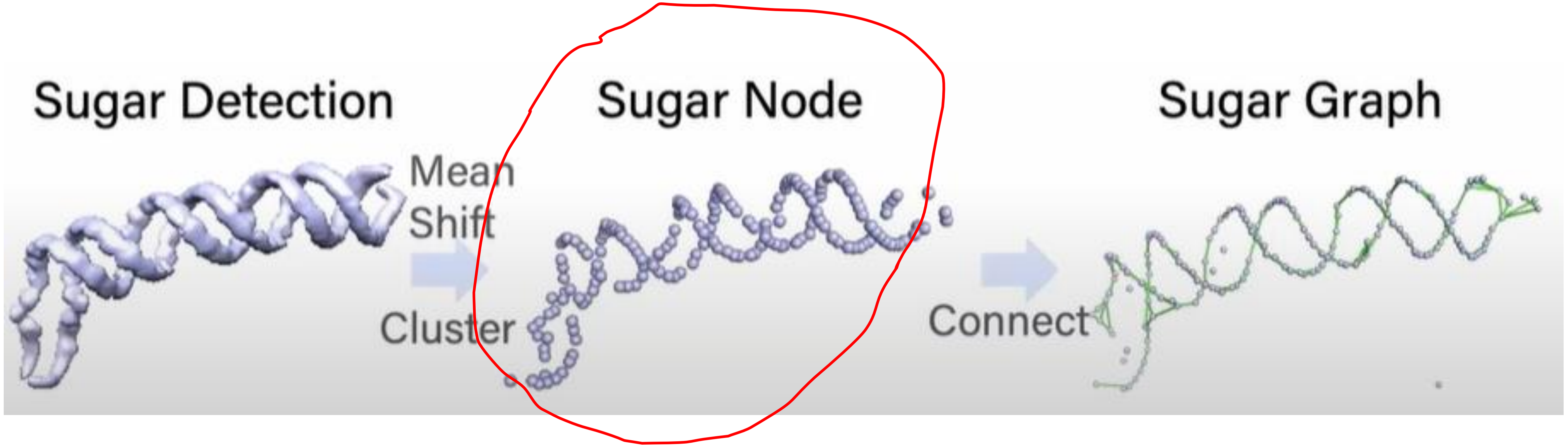


Backbone Tracing Workflow - Overview

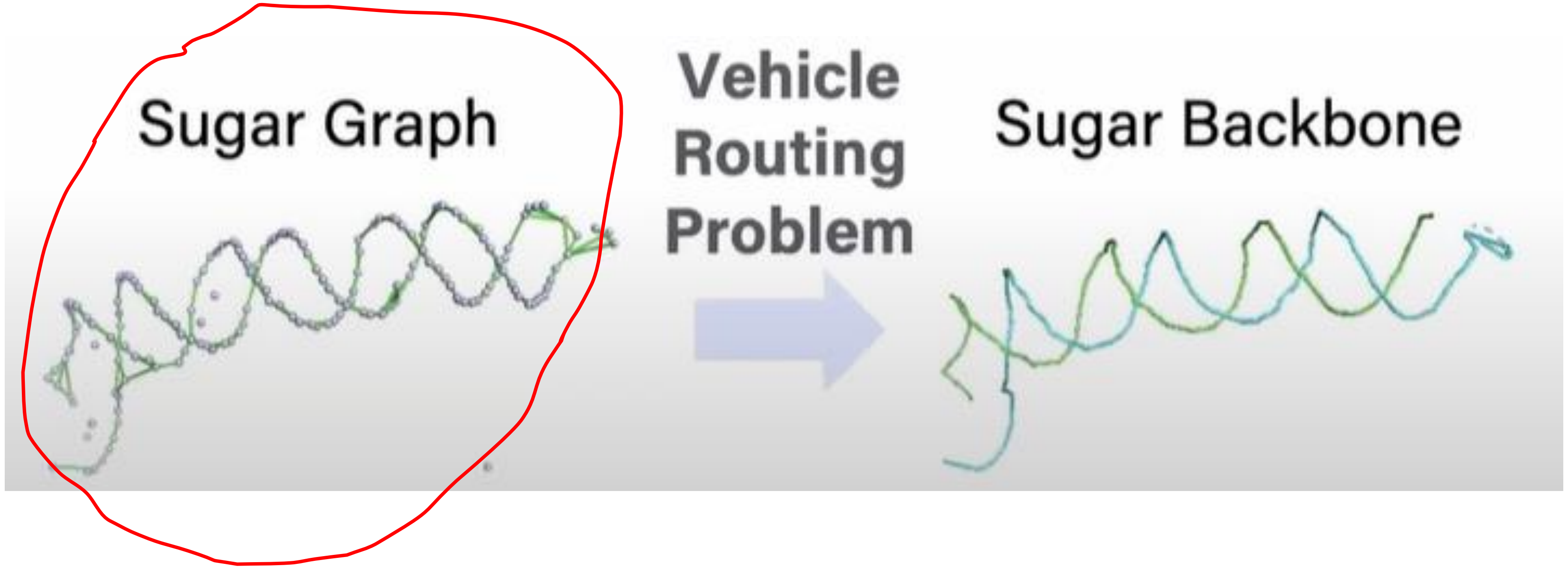
- 1. Sugar Detection → Sugar Nodes
- 2. Sugar Nodes → Sugar Graph
- 3. Sugar Graph → Sugar Backbone



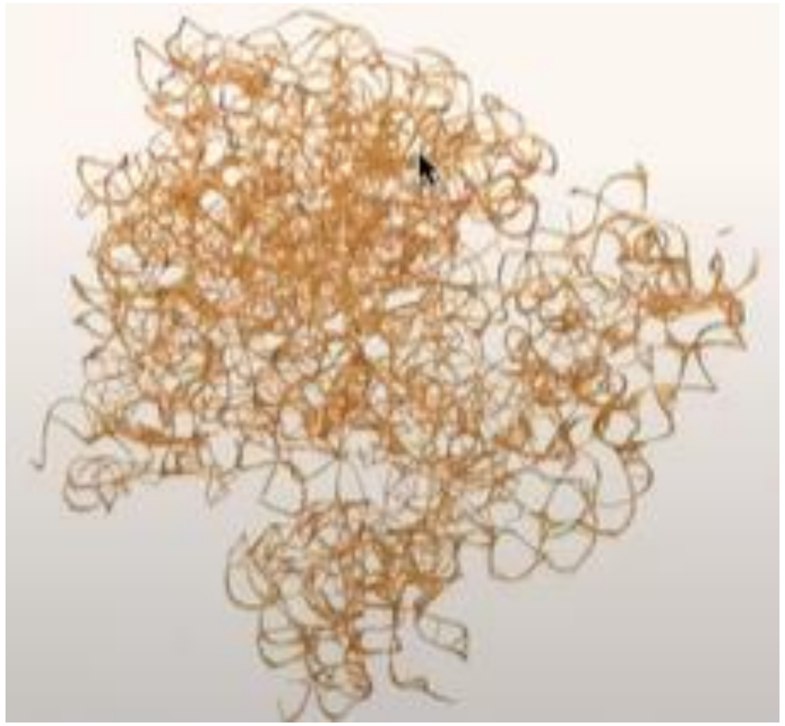
Our starting point
(we have this)



Start with this to trace a graph



Run VRP solver on this



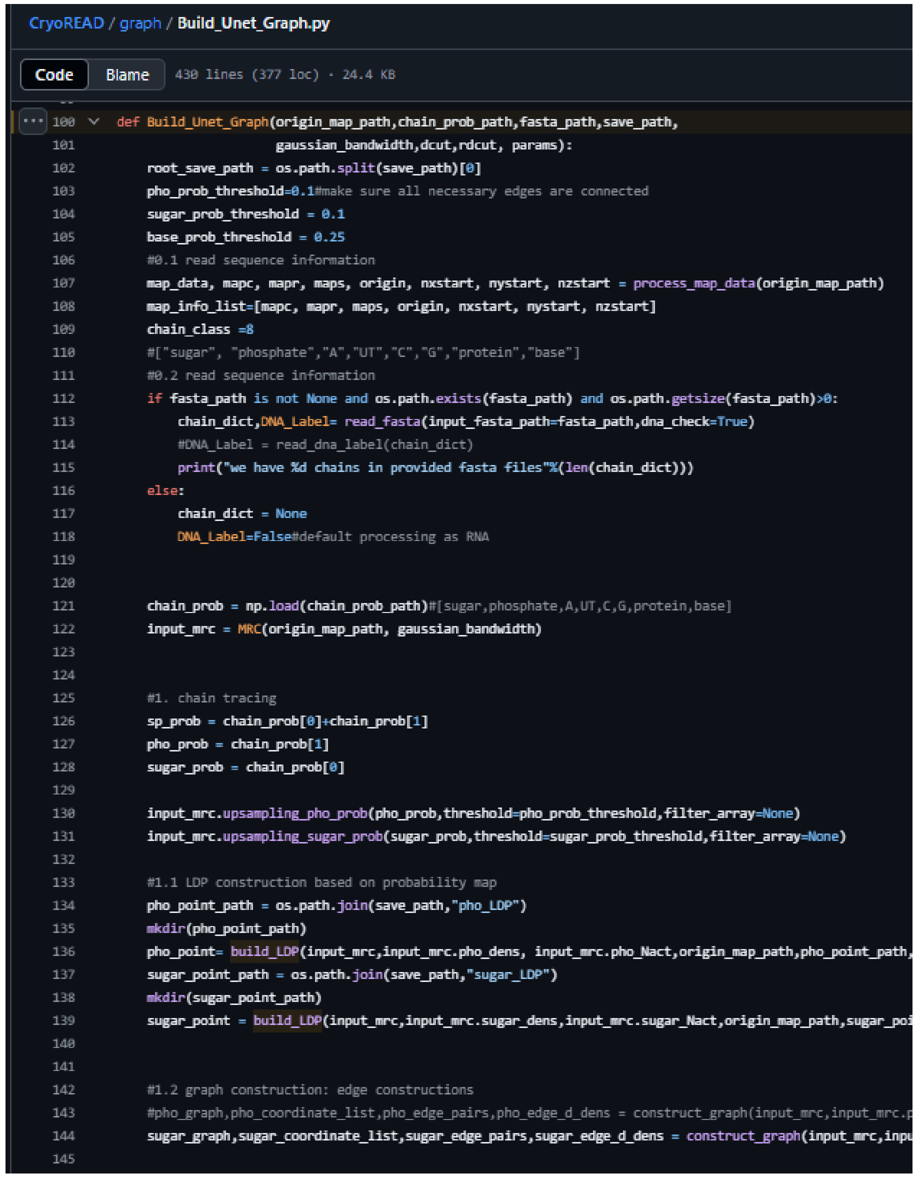
CryoREAD’s Code

https://github.com/kiharalab/CryoREAD

Overview

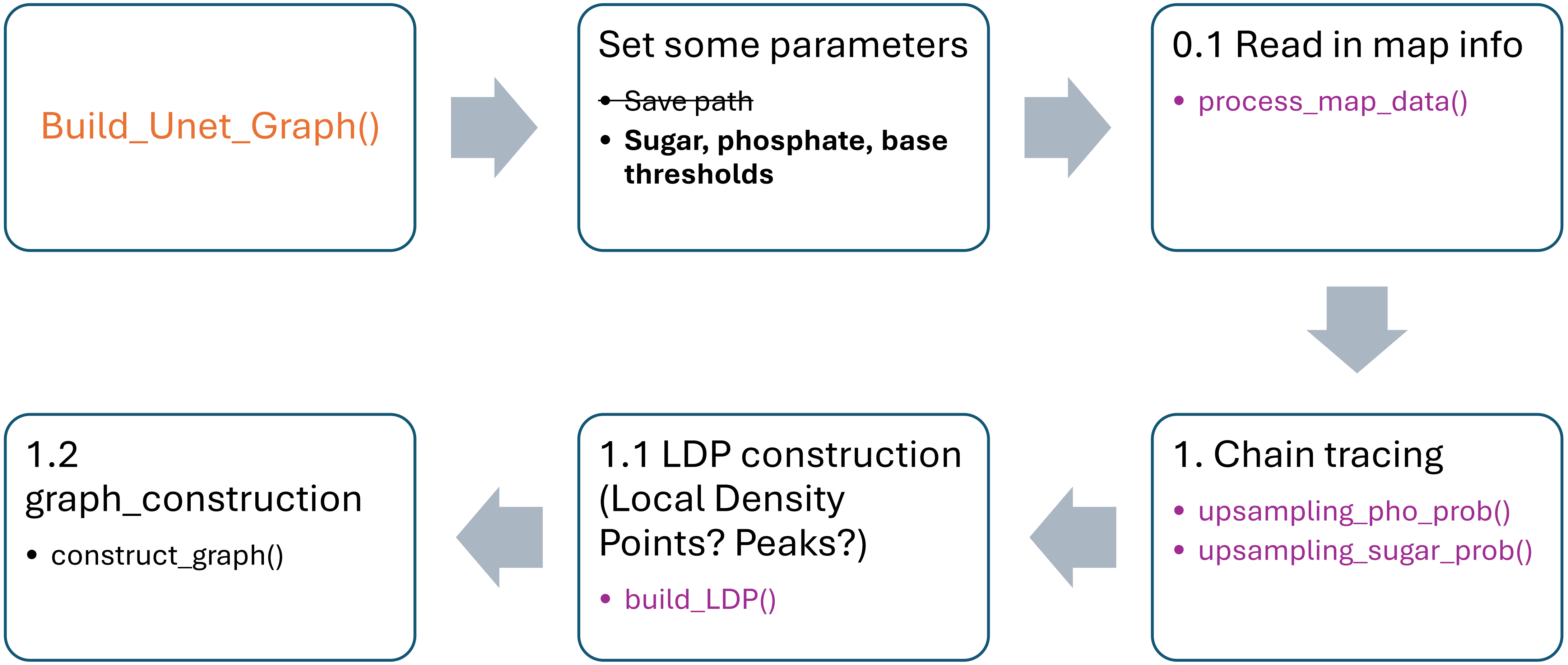
- 1. main.py > line 116 calls Build_Unet_Graph() function
- 2. Build_Unet_Graph() is defined in graph/Build_Unet_Graph.py at line 100.
  - *Starting point/hub for backbone tracing process.*
- 3. Build_Unet_Graph() calls a number of functions
  a. process_map_data()
  b. upsampling_pho_prob(), upsampling_sugar_prob()
  c. build_LDP()
  d. construct_graph()

A more in depth look at Build_Unet_Graph()


CryoREAD / graph / Build_Unet_Graph.py

Code Blame 430 lines (377 loc) · 24.4 KB

```python
def Build_Unet_Graph(origin_map_path,chain_prob_path,fasta_path,save_path,
                     gaussian_bandwidth,dcut,rdcut, params):
    root_save_path = os.path.split(save_path)[0]
    pho_prob_threshold=0.1#make sure all necessary edges are connected
    sugar_prob_threshold = 0.1
    base_prob_threshold = 0.25
    #0.1 read sequence information
    map_data, mapc, mapr, maps, origin, nxstart, nystart, nzstart = process_map_data(origin_map_path)
    map_info_list=[mapc, mapr, maps, origin, nxstart, nystart, nzstart]
    chain_class =8
    #["sugar", "phosphate","A","UT","C","G","protein","base"]
    #0.2 read sequence information
    if fasta_path is not None and os.path.exists(fasta_path) and os.path.getsize(fasta_path)>0:
        chain_dict,DNA_Label= read_fasta(input_fasta_path=fasta_path,dna_check=True)
        #DNA_Label = read_dna_label(chain_dict)
        print("we have %d chains in provided fasta files"%(len(chain_dict)))
    else:
        chain_dict = None
        DNA_Label=False#default processing as RNA


    chain_prob = np.load(chain_prob_path)#[sugar,phosphate,A,UT,C,G,protein,base]
    input_mrc = MRC(origin_map_path, gaussian_bandwidth)


    #1. chain tracing
    sp_prob = chain_prob[0]+chain_prob[1]
    pho_prob = chain_prob[1]
    sugar_prob = chain_prob[0]

    input_mrc.upsampling_pho_prob(pho_prob,threshold=pho_prob_threshold,filter_array=None)
    input_mrc.upsampling_sugar_prob(sugar_prob,threshold=sugar_prob_threshold,filter_array=None)

    #1.1 LDP construction based on probability map
    pho_point_path = os.path.join(save_path,"pho_LDP")
    mkdir(pho_point_path)
    pho_point= build_LDP(input_mrc,input_mrc.pho_dens, input_mrc.pho_Nact,origin_map_path,pho_point_path,
    sugar_point_path = os.path.join(save_path,"sugar_LDP")
    mkdir(sugar_point_path)
    sugar_point = build_LDP(input_mrc,input_mrc.sugar_dens,input_mrc.sugar_Nact,origin_map_path,sugar_poi


    #1.2 graph construction: edge constructions
    #pho_graph,pho_coordinate_list,pho_edge_pairs,pho_edge_d_dens = construct_graph(input_mrc,input_mrc.p
    sugar_graph,sugar_coordinate_list,sugar_edge_pairs,sugar_edge_d_dens = construct_graph(input_mrc,inpu

```



Build_Unet_Graph()

Set some parameters

- ~~Save path~~
- **Sugar, phosphate, base thresholds**

0.1 Read in map info

- process_map_data()

1. Chain tracing

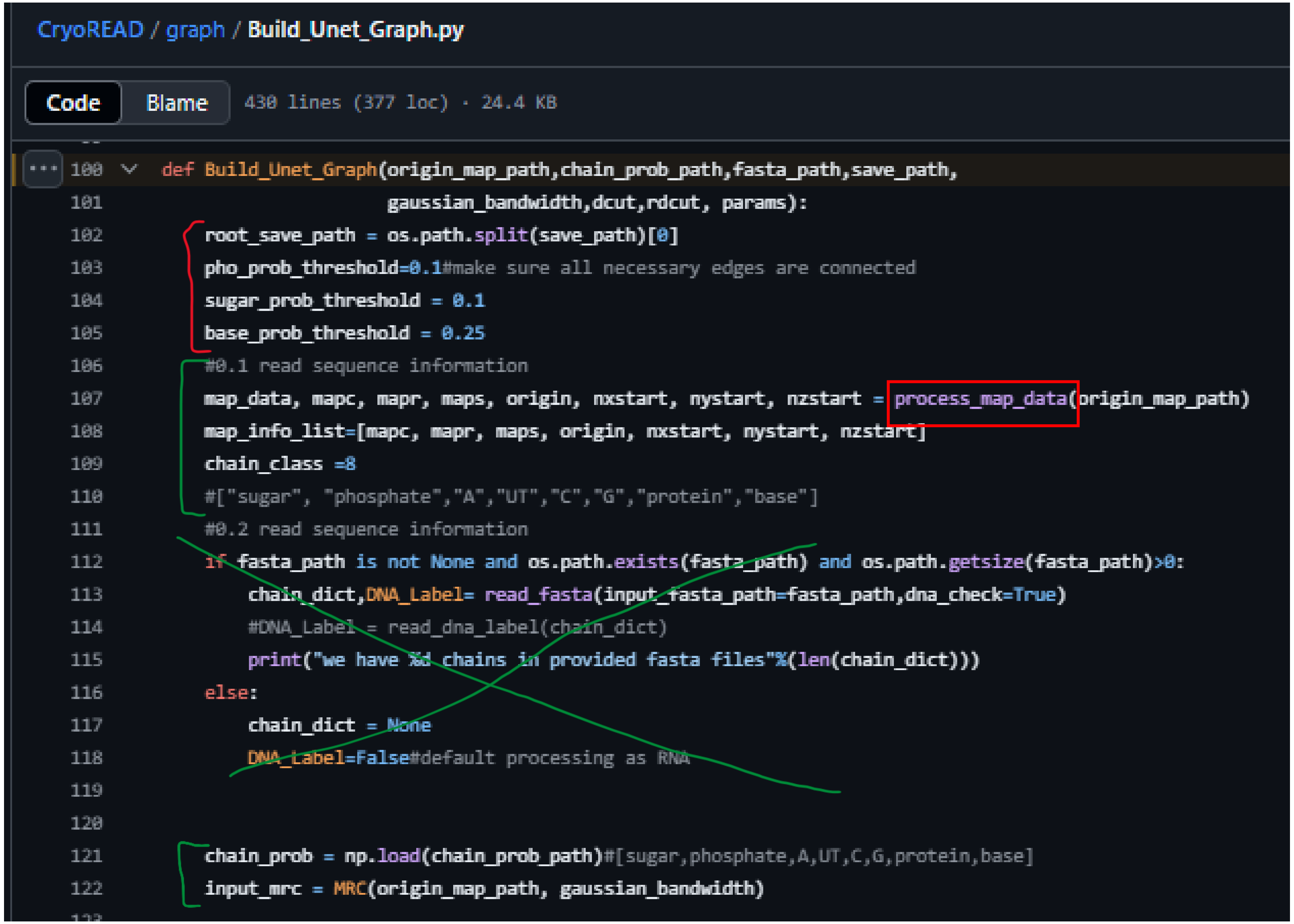
- upsampling_pho_prob()
- upsampling_sugar_prob()

1.1 LDP construction (Local Density Points? Peaks?)

- build_LDP()

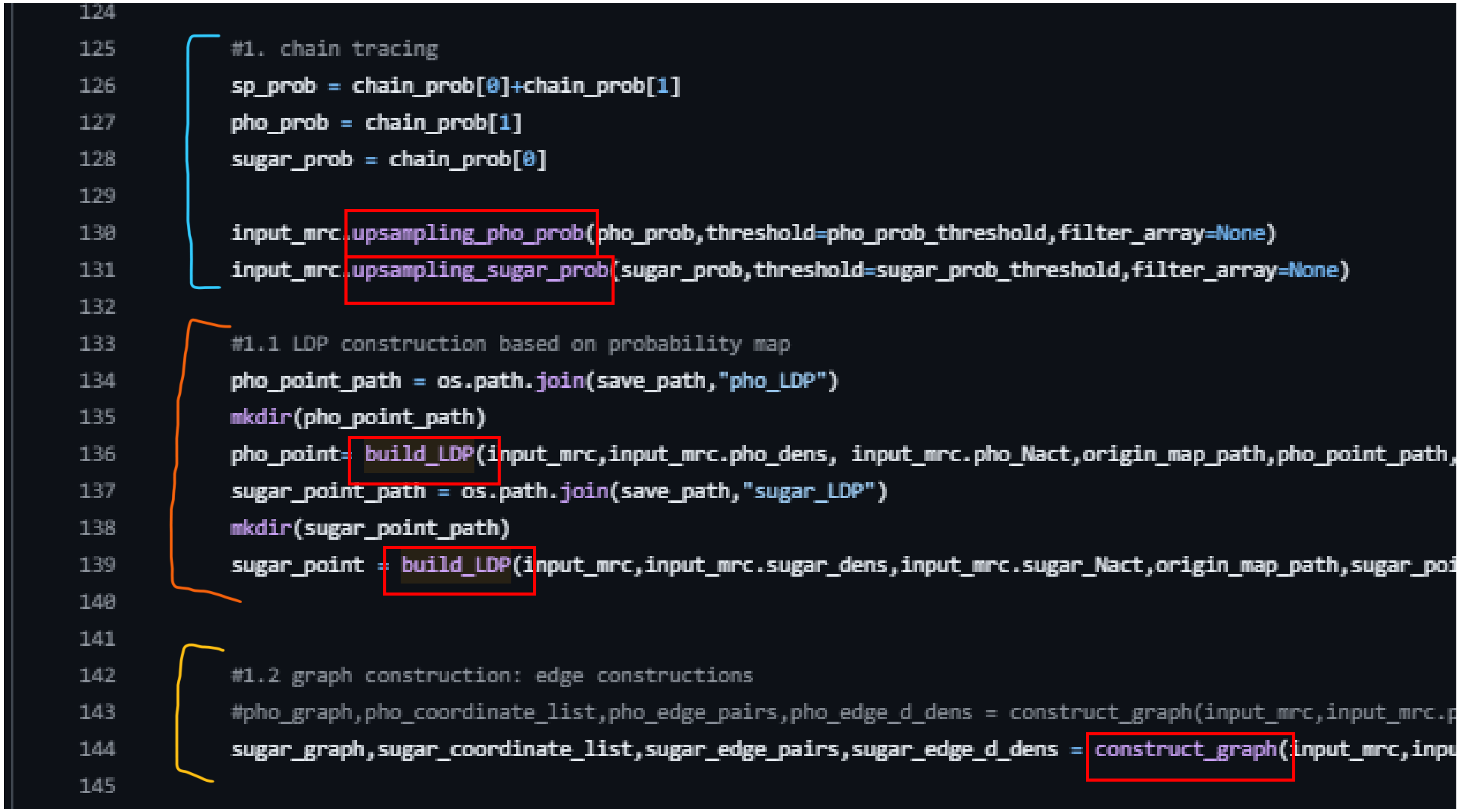
1.2 graph_construction

- construct_graph()


CryoREAD / graph / Build_Unet_Graph.py

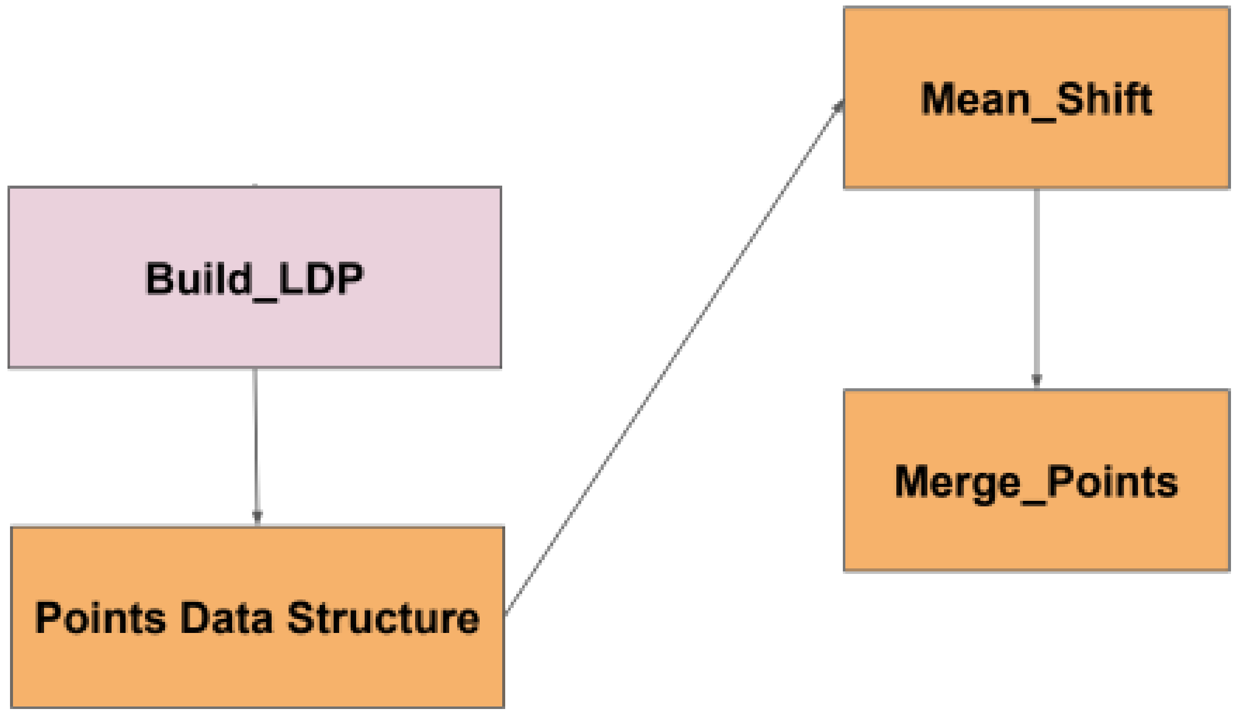
430 lines (377 loc) · 24.4 KB

```python
def Build_Unet_Graph(origin_map_path,chain_prob_path,fasta_path,save_path,
                     gaussian_bandwidth,dcut,rdcut, params):
    root_save_path = os.path.split(save_path)[0]
    pho_prob_threshold=0.1#make sure all necessary edges are connected
    sugar_prob_threshold = 0.1
    base_prob_threshold = 0.25
    #0.1 read sequence information
    map_data, mapc, mapr, maps, origin, nxstart, nystart, nzstart = process_map_data(origin_map_path)
    map_info_list=[mapc, mapr, maps, origin, nxstart, nystart, nzstart]
    chain_class =8
    #["sugar", "phosphate","A","UT","C","G","protein","base"]
    #0.2 read sequence information
    if fasta_path is not None and os.path.exists(fasta_path) and os.path.getsize(fasta_path)>0:
        chain_dict,DNA_Label= read_fasta(input_fasta_path=fasta_path,dna_check=True)
        #DNA_Label = read_dna_label(chain_dict)
        print("we have %d chains in provided fasta files"%(len(chain_dict)))
    else:
        chain_dict = None
        DNA_Label=False#default processing as RNA


    chain_prob = np.load(chain_prob_path)#[sugar,phosphate,A,UT,C,G,protein,base]
    input_mrc = MRC(origin_map_path, gaussian_bandwidth)
```



```
    #1. chain tracing
    sp_prob = chain_prob[0]+chain_prob[1]
    pho_prob = chain_prob[1]
    sugar_prob = chain_prob[0]

    input_mrc.upsampling_pho_prob(pho_prob,threshold=pho_prob_threshold,filter_array=None)
    input_mrc.upsampling_sugar_prob(sugar_prob,threshold=sugar_prob_threshold,filter_array=None)

    #1.1 LDP construction based on probability map
    pho_point_path = os.path.join(save_path,"pho_LDP")
    mkdir(pho_point_path)
    pho_point= build_LDP(input_mrc,input_mrc.pho_dens, input_mrc.pho_Nact,origin_map_path,pho_point_path,
    sugar_point_path = os.path.join(save_path,"sugar_LDP")
    mkdir(sugar_point_path)
    sugar_point = build_LDP(input_mrc,input_mrc.sugar_dens,input_mrc.sugar_Nact,origin_map_path,sugar_poi


    #1.2 graph construction: edge constructions
    #pho_graph,pho_coordinate_list,pho_edge_pairs,pho_edge_d_dens = construct_graph(input_mrc,input_mrc.p
    sugar_graph,sugar_coordinate_list,sugar_edge_pairs,sugar_edge_d_dens = construct_graph(input_mrc,inpu

```

1.1 LDP construction – build_LDP()



Tips

- Ctrl+click on function names to see their definitions, **makes it easier to trace thru code**
  - Works on github website and vscode (and probably other IDEs)