ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
Мэдээлэл, Холбооны Технологийн Сургууль

F.CS213 Биоалгоритм

Graphs and Biological Networks

Граф ба Биологийн сүлжээ

Лекц 13

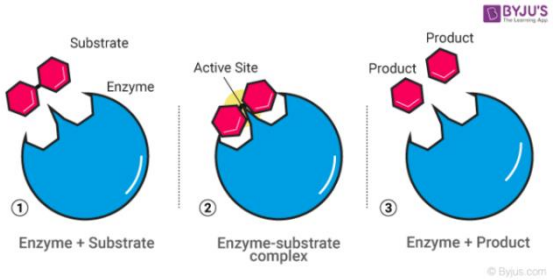
Багш ***Ганбаатарын ГАНБАТ.*** *Өрөө* ***#304,*** *ganbatg@must.edu.mn,* ***99999467***

Лекцийн агуулга

- Биологийн сүлжээ
- Сүлжээний граф дүрслэл
  - Метаболик сүлжээний Пайтон класс
  - Бодит организмын Метаболик сүжлээ
- Сүлжээний тополог шинжилгээ
  - Зэргийн тархалт
  - Богино замын шинжилгээ
  - Кластрын коэффицент
  - Хабууд (Hubs) ба төвлөрөл (Centrality)-ийн хэмжүүр
- Бодисын солилцооны боломжийг үнэлэх

Биологийн сүлжээ (Biological networks)

- Эс зэрэг төрөл бүрийн биологийн системийг дүрслэх, симуляц хийхэд ашигладаг
  - уураг кодлодог генүүд,
  - урвалд оролцож буй метаболитууд (metabolites),
  - ген хөрвүүлэлтийг зохицуулдаг эсвэл нэгдлүүд рүү холбогддог уургууд
- *Биологийн сүлжээ* - граф
  - *Зангилаа:* биологийн объект (ген, уураг, химийн нэгдлүүд)
  - *Ирмэг:* зангилаануудын хоорондын харилцан үйлчлэл, хамаарал



Системийн биологиийн хувьд: Эсийн байгууламж болон фенотип орчлын глобал шинжилгээ

Бодисын солилцооны сүлжээ (Metabolic network)

- Урвал (тэдгээрийн кодлож буй энзимууд)
- Метаболит (Бодисын солилцооны урвалд оролцож буй нэгдлүүд)
- Хоол тэжээлийн химийн хувиралуудын олонлог (enzyme-catalyzed) харуулдаг.

Зохицуулалтын сүлжээ (Regulatory networks)

- Ген
- Уураг
- ген хувиргалтын түвшинг удирдах боломжийг эсэд бий болгодог процессийг харуулдаг

Дохио солилцооны сүлжээ (Signaling network)

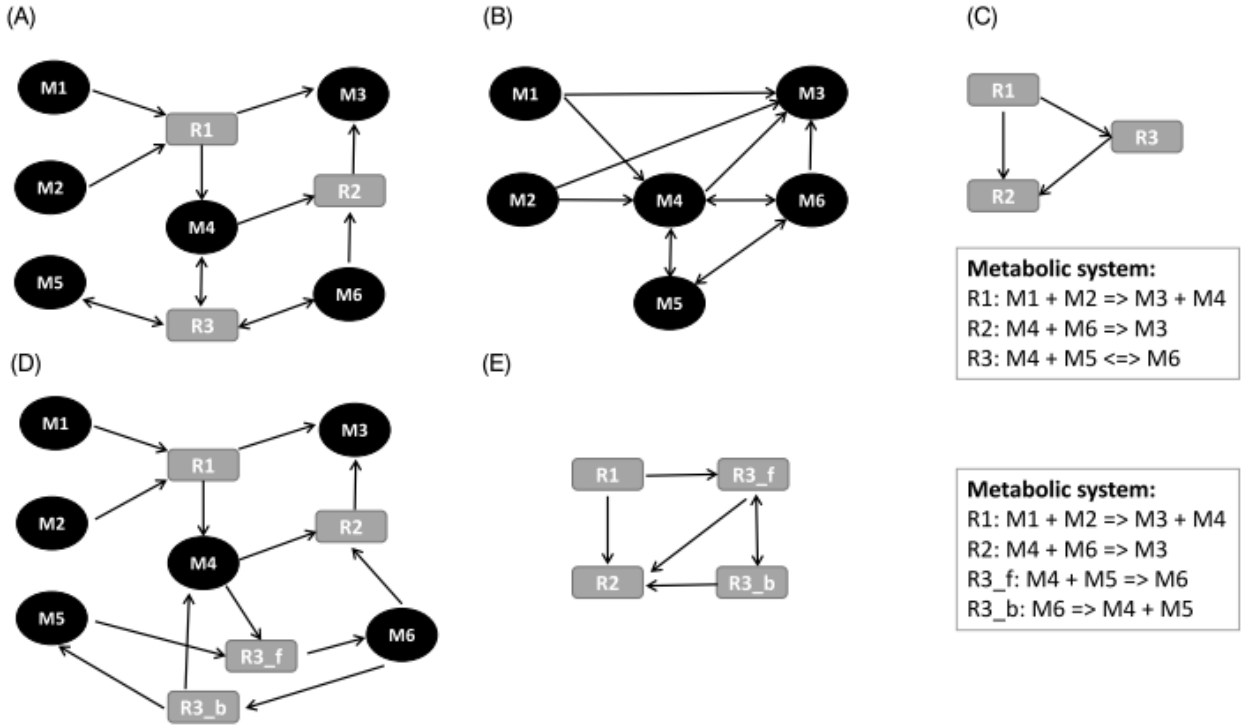
- Эсийн дотоод зохицуулалтын болон бодисын солилцооны системд гаднаас дохио шилжүүлэхэд оролцдог уургуудын шатлалыг харуулдаг

Сүлжээний граф дүрслэл

Гурван урвал бүхий энгийн бодисын солилцооны системд зориулсан Бодисын солилцооны сүлжээ:
(A) метаболит-урвал сүлжээ; (B) метаболит-метаболит сүлжээ; (C) урвал-урвал сүлжээ;

(D) ба (E) нь (A) ба (C) хувьд харих урвалыг хоёр чиглэлд ялгаж харуулсан.

- => нь энгийн урвал
- <=> нь харих урвал



- $G = (V, E)$ нь *хоёр талт (bipartite)* граф
  - $V_1 \cup V_2 = V$ ба $V_1 \cap V_2 = \emptyset$
  - E нь зөвхөн V_1 элементийг V_2 элементтэй харгалзуулсан хосууд,
- Ижил олонлог дахь зангилаануудыг хооронд нь холбосон ирмэг байхгүй.
- Бодисын солилцооны сүлжээ (жнь: метаболит-урвал)
  - V_1 нь метаболитын олонлог,
  - V_2 нь урвалын олонлог

СҮЛЖЭЭНИЙ ГРАФ ДҮРСЛЭЛ

MetabolicNetwork класс

- 13-р бүлгийн **MyGraph** классыг хөгжүүлнэ.
  - Сүлжээний төрөл: "метаболит-урвал", "метаболит-метаболит", "урвал-урвал",
  - Зангилааны төрөлд зангилаануудыг харгалзуулах атрибут
  - Харих урвалыг 2 чиглэлд задалсан эсэх Булийн хувьсагч
- Хоёр талт графын ирмэг, зангилаануудыг нэмдэх функц

```
from MyGraph import MyGraph

class MetabolicNetwork (MyGraph):

    def __init__(self, network_type = "metabolite-reaction",
   split_rev = False):
        MyGraph.__init__(self, {})
        self.net_type = network_type
        self.node_types = {}
        if network_type == "metabolite-reaction":
            self.node_types["metabolite"] = []
            self.node_types["reaction"] = []
        self.split_rev =  split_rev
    def add_vertex_type(self, v, nodetype):
        self.add_vertex(v)
        self.node_types[nodetype].append(v)
```

```
def test1():
    m = MetabolicNetwork("metabolite-reaction")
    m.add_vertex_type("R1","reaction")
    m.add_vertex_type("R2","reaction")
    m.add_vertex_type("R3","reaction")
    m.add_vertex_type("M1","metabolite")
    m.add_vertex_type("M2","metabolite")
    m.add_vertex_type("M3","metabolite")
    m.add_vertex_type("M4","metabolite")
    m.add_vertex_type("M5","metabolite")
    m.add_vertex_type("M6","metabolite")
    m.add_edge("M1","R1")
    m.add_edge("M2","R1")
    m.add_edge("R1","M3")
    m.add_edge("R1","M4")
    m.add_edge("M4","R2")
    m.add_edge("M6","R2")
    m.add_edge("R2","M3")
    m.add_edge("M4","R3")
    m.add_edge("M5","R3")
    m.add_edge("R3","M6")
    m.add_edge("R3","M4")
    m.add_edge("R3","M5")
    m.add_edge("M6","R3")
    m.print_graph()
    print("Reactions: ", m.get_nodes_type("react
```

Сүлжээг файлаас унших

- MetabolicNetwork классын load_from_file функцийг ашиглана.
  - Файлаас мэдээллийг ачаалж, классын сүлжээний төрлөөр сүлжээг үүсгэдэг
  - Харих урвалыг хэрхэн зохицуулахыг зааж өгсөн флаг хувьсагчийг хэрэглэнэ.
  - Эхлээд "метаболит-урвал" үүссэнэ;
  - Өөр сүлжээ хэрэгтэй бол convert_metabolite_net, convert_reaction_graph функцийг ашиглана.
- load_from_file функц
  - тусгаарлагчийг ашиглан оролтын *str* тэмдэгт мөр дээр ажиллана, (regular expression-тэй ижил)
  - str-г тусгаарлагчийн илрэлүүдээр хувааж *дэд тэмдэгт мөрүүд (tokens)*-ийн жагсаалт болгоно.

```
R1: M1 + M2 => M3 + M4
R2: M4 + M6 => M3
R3: M4 + M5 <=> M6
```

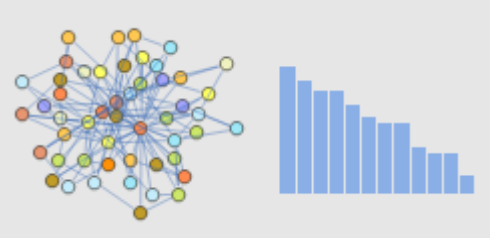
Жишээ: *Escherichia coli* бактери

- Өмнөх ойлголтуудыг илүү бодит хувилбараар харуулахын тулд мэдэгдэж буй загвар организмын бодисын солилцооны сүлжээг байгуулъя.
  - Судлаачдын үүсгэн байгуулсан хамгийн түгээмэл Бодисын солилцооны сүлжээнд суурилна.
- Рид (Reed) нар. *iJR904* бодисын солилцооны загвар.
  - Энэ загварын доторх урвалуудыг "ecoli.txt" (өмнө үзсэн форматтай) файлаас авна (номын вэбсайт).
  - 931 урвал, 761 метаболит
  - "метаболит-урвал" сүлжээнд 5000 гаруй ирмэг,
  - "урвал-урвал" сүлжээнд 130,000 гаруй ирмэг.
- Энэхүү илүү том, илүү бодит сүлжээг топологийн шинжилгээний зарим үр дүнг харуулахын тулд ашигладаг.

```
def test3():
    print("metabolite-reaction network:")
    ec_mrn = MetabolicNetwork("metabolite-reaction")
    ec_mrn.load_from_file("ecoli.txt")
    print(ec_mrn.size())

    print("metabolite-metabolite network:")
    ec_mmn = MetabolicNetwork("metabolite-metabolite")
    ec_mmn.load_from_file("ecoli.txt")
    print(ec_mmn.size())

    print("reaction-reaction network:")
    ec_rrn = MetabolicNetwork("reaction-reaction")
    ec_rrn.load_from_file("ecoli.txt")
    print(ec_rrn.size())

test3()
```

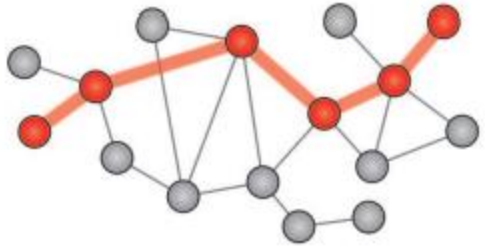
Сүлжээний топологийн шинжилгээ

- Графын бүтцэд уялдуулан тооцоолдог
- Сүлжээний биологийн ойлголтод хүргэж болзошгүй шинж чанарыг шалгахад ашиглаж боломжтой хэмжүүрүүд.



Зэргийн тархалт

(Degree Distribution)

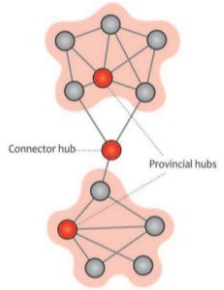


Богино замын шинжилгээ

(Shortest Path Analysis)

Кластр коэффицент

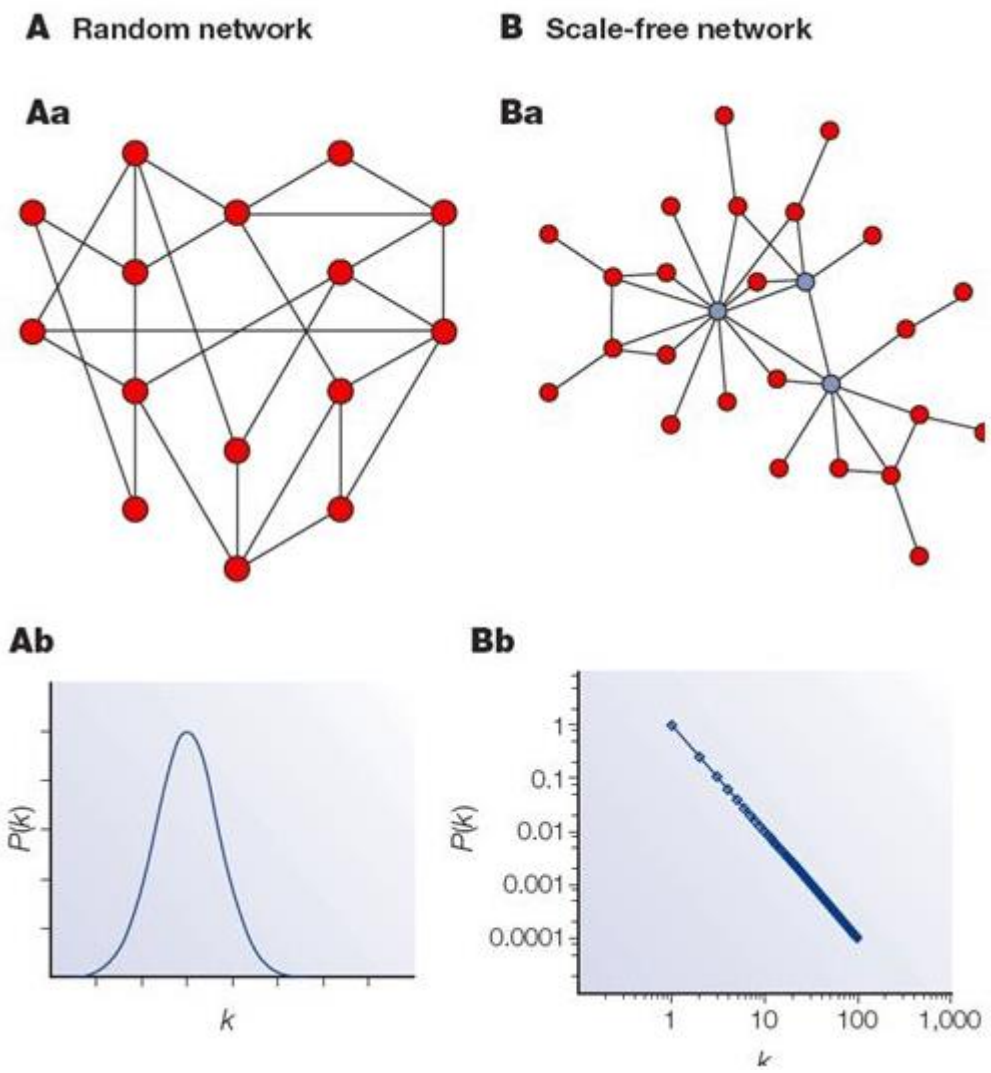
(Clustering Coefficients)



Хабууд ба төвлөрлийн хэмжүүр

(Hubs and Centrality Measures)

Зэргийн тархалт



- *Дундаж зэрэг (mean degree)* k
  - Графын бүх зангилаа дээрх зэргийн дундаж утга.
  - Чиглэлт графын хувьд энэ утгыг оролтын, гаралтын зэрэг (in-degrees, out-degrees) тооцоолж болно.
  - Сүлжээний холболтын бүх түвшинд үнэлж болно.
- Зангилааны зэргийн тархалт
  - Граф дахь давтамжаар зангилаа нь k зэрэгтэй байх $P(k)$ магадлалыг тооцоолно.
  - *Санамсаргүй байдлаар үүсгэгдсэн (Рандом) сүлжээ*
    - k дундажтай $P(k)$ нь Нормал тархалттай
  - Рандом эсвэл бусад сүлжээтэй төстэй бүтэцтэй эсэхийг шалгахад чухал параметер.
- Түгээмэл төрөл нь *scale-free сүлжээ*
  - $P(k)$ нь ихэвчлэн чадлын хууль (power law)-тай ойролцоо байдаг, жнь $P(k) \simeq k^{-\gamma}; 2 < \gamma < 3$.
  - Ихэнх зангилаанууд бага зэрэгтэй байдаг бол их зэрэгтэй цөөн хэдэн зангилаа байдаг.
  - $P(k)$ нь логарифм хэмжээстэй, шугаман хэлбэртэй.

ТОПОЛОГИЙН ШИНЖИЛГЭЭ

Богино замын шинжилгээ

- 2 зангилааны хоорондох хамгийн богино замын уртад суурилан сүлжээний глобал хэмжигдэхүүнүүдийг тооцоол.
- Хамгийн энгийн нь *дундаж зай (mean distance) L*
  - Бүх хос зангилааны хоорондох хамгийн богино замуудын уртын дундаж
- Хэрэв граф холбоост биш байхад заримдаа зай нь хязгааргүй (зангилаанууд холбогдоогүй тул).
  - Энэ тохиолдолд эдгээр хосуудыг орхидог
  - Гэхдээ, хэдэн замын тохиолдолд боломжтойг мэдэх нь бас чухал байж болно.
- Энэ хэмжигдэхүүн нь зарим сүлжээний онцлог шинж чанарыг үнэлэхэд тусална. Жнь:
  - Ижил хэмжээний *Рандом сулжээ*-тэй харьцуулахад *L* нь харьцангуй бага байдаг *small-world сулжээ*.

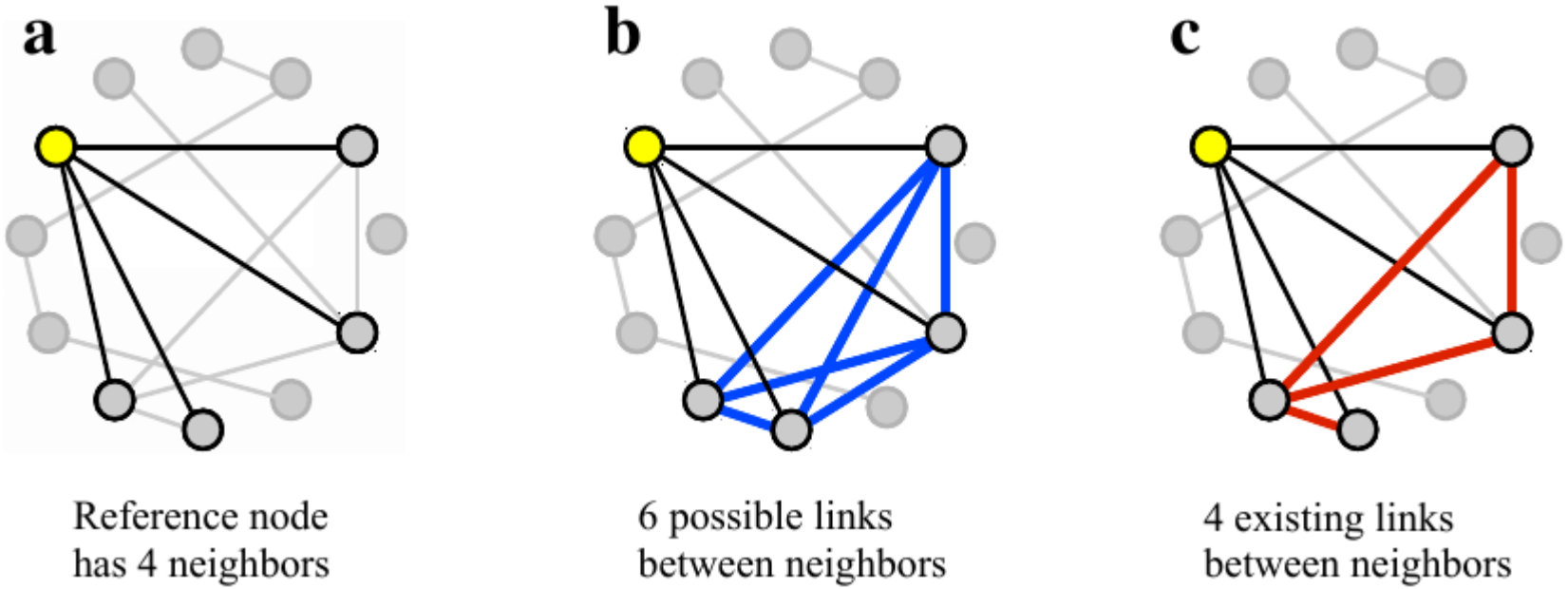
```
class MyGraph:

    (...)

    def mean_distances(self):
        tot = 0
        num_reachable = 0
        for k in self.graph.keys():
            distsk = self.reachable_with_dist(k)
            for _, dist in distsk:
                tot += dist
            num_reachable += len(distsk)
        meandist = float(tot) / num_reachable
        n = len(self.get_nodes())
        return meandist, float(num_reachable)/((n-1)*n)
```

```
def test2():
    (...)

    print(mmn.mean_distances())
    print(mrn.mean_distances())

test2()
```

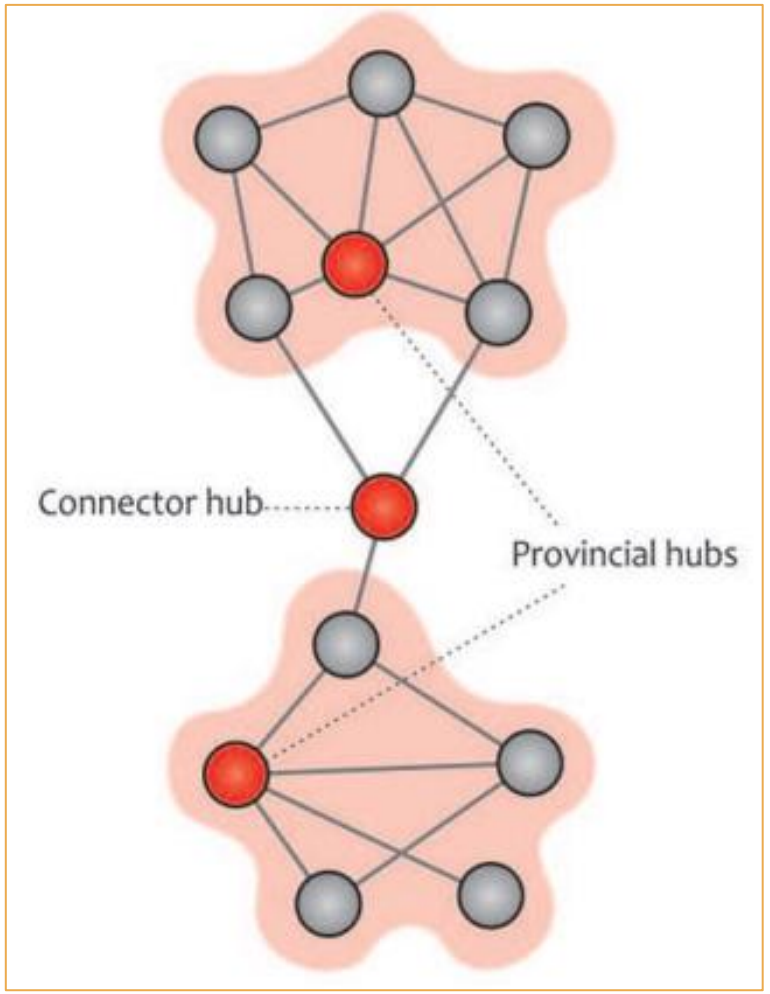
```
def test3():
    (...)

    print("Mean distance (M-R): ", ec_mrn.mean_distances())
    print("Mean distance (M-M): ", ec_mmn.mean_distances())

test3()
```

Кластрын коэффицент

- Илүү хүчтэй холбогдсон зангилаануудын локал бүлгүүд эсвэл кластруудыг үүсгэж буй зангилаануудын тодролтыг граф дээр хэмжих асуудал тулгардаг.
  - Зангилааны ***кластерын коэффициент (clustering coefficient)*** нь хөршүүдийн хоорондох ирмэгүүдийн тоо, мөн тэнд байж болох ирмэгүүдийн тооны харьцаагаар тодорхойлогддог
  - Кластрын коэффициент өндөртэй зангилаа нь өөр хоорондоо нягт холбогдсон хөршүүдтэй байна.
- Сүлжээн дэх бүх зангилааны хувьд ***дундаж кластерын коэффициент (mean clustering coefficient)*** C-ийг тухайн сүлжээний глобал хэмжигдэхүүн байхаар тодорхойлж болно.
  - $C(k)$ нь k зэрэгтэй бүх зангилааны кластерын коэффициентүүдийн дундаж байна.

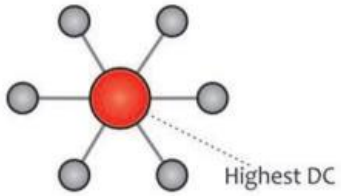


Хабууд ба төвлөрлийн хэмжүүр

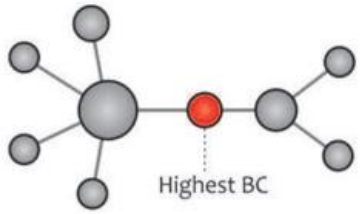


- Хамгийн чухал зангилаануудыг илрүүлэх нь Сүлжээний шинжилгээний чухал даалгавар байдаг.
  - Биологийн сүлжээнд эдгээр зангилаа нь системийн үйл ажиллагааг ойлгоход маш чухал
  - Сүлжээний тогтвортой, тэсвэртэй байдал зэрэг онцлогуудтай холбоотой.
- Хабууд (холбогч эсвэл муж) нь төвлөрөл ихтэй зангилаа
  - Янз бүрийн хэмжүүр ашиглан тодорхойлж болно.

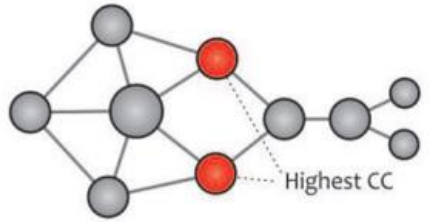
Degree centrality



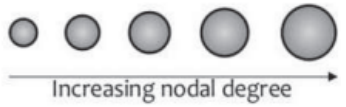
Betweenness centrality



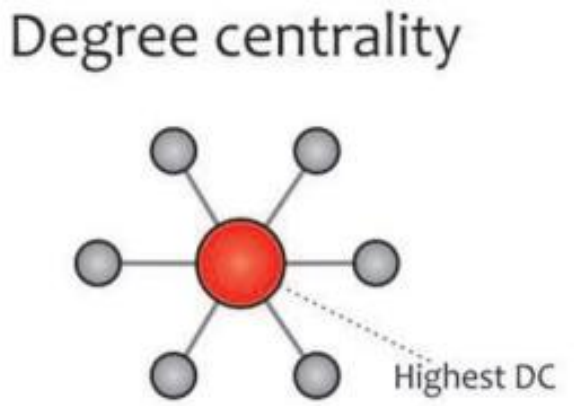
Closeness centrality



- Зангилааны хэмжээ нь зэрэгтэй нь шууд хамааралтай байдаг
  - Улаан зангилаанууд нь бага зэрэгтэй байсан ч чухал.



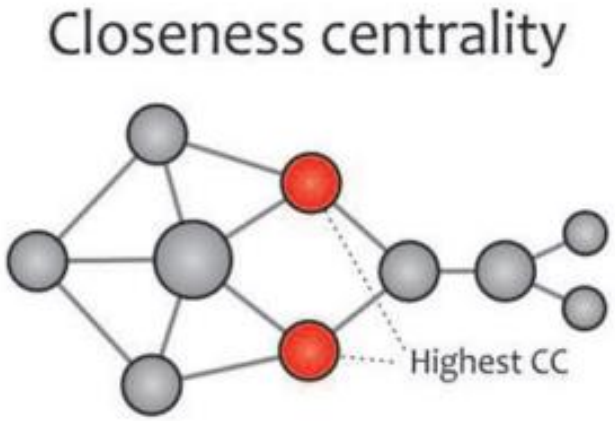
Зэргийн төвлөрөл




```
class MyGraph:

    (...)

    def highest_degrees(self, all_deg= None, deg_type = "inout", top=
    10):
        if all_deg is None:
            all_deg = self.all_degrees(deg_type)
        ord_deg = sorted(list(all_deg.items()), key=lambda x : x[1],
   reverse = True)
        return list(map(lambda x:x[0], ord_deg[:top]))
```

- Зангилааг ач холбогдлоор нь эрэмбэлэх хамгийн энгийн арга бол илүү олон хөрштэй байх
- Бодисын солилцооны сүлжээнд (жнь, "метаболит-метаболит") хэрэглэснээр олон урвалын гол хүчин зүйл болох highly-connected метаболитуудыг тодорхойлно.
- Заримдаа шинжилгээг хялбар болгохын тулд эдгээр метаболитуудыг устгах нь түгээмэл байдаг.

Closeness centrality



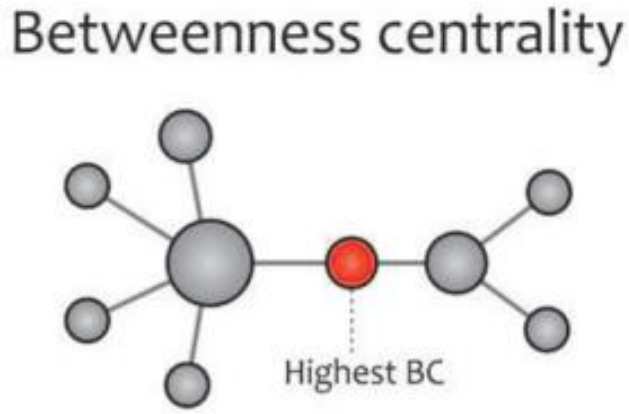
$$CC(v) = \frac{N-1}{\sum_{x \in V} d(v,x)}$$

$d(x,y)$ нь x ба y зангилааны хоорондох зай,
N нь граф дахь зангилааны тоо ($|V|$).

```
class MyGraph:

    (...)

    def closeness_centrality(self, node):
        dist = self.reachable_with_dist(node)
        if len(dist)==0: return 0.0
        s = 0.0
        for d in dist: s += d[1]
        return len(dist) / s

    def highest_closeness(self, top = 10):
        cc = {}
        for k in self.graph.keys():
            cc[k] = self.closeness_centrality(k)
        ord_cl = sorted(list(cc.items()), key=lambda x : x[1],
   reverse = True)
        return list(map(lambda x:x[0], ord_cl[:top]))
```

- Тухайн зангилаанаас бусад бүх зангилаа руу хэр хурдан нэвтэрч болохыг тоон үзүүлэлтээр илэрхийлдэг
  - Зангилаа хэдий чинээ төвлөрөлтэй байх тусам бусад бүх зангилаатай ойрхон байна
  - Холбоост графуудад (эсвэл бараг холбогдсон графуудад) тохиромжтой.
  - Хэрэв холболтгүй хос зангилаа байвал тооцооллоос хасна.
  - Эдгээр хосууд сүлжээнд олон байвал энэ нь үр дүнд ихээхэн нөлөөлнө.

Betweenness centrality




```
class MyGraph:

    (...)

    def betweenness_centrality(self, node):
        total_sp = 0
        sps_with_node = 0
        for s in self.graph.keys():
            for t in self.graph.keys():
                if s != t and s != node and t != node:
                    sp = self.shortest_path(s, t)
                    if sp is not None:
                        total_sp += 1
                        if node in sp: sps_with_node += 1
        return sps_with_node / total_sp
```

- Сүлжээн дэх хос зангилааны хоорондох хамгийн богино зам дээр суурилна.
- Энэ хэмжигдэхүүн нь бүх хос зангилааны хоорондох зам нь тухайн зангилаагаар дамжих хамгийн богино замын эзлэх хувь байна.
  - 0-ээс 1 хүртэлх утгыг авна
- Ө.х: Тухайн зангилааг агуулсан сүлжээн дэх хамгийн богино замуудын харьцааг тооцоолох замаар салангид кластеруудын хооронд гүүр болох зангилааны үүргийг хэмждэг.

Бодисын солилцооны боломжийг үнэлэх

- Бодисын солилцооны сүлжээг хоёр талт график хэлбэрээр дүрслэх нь өмнө тайлбарласнаар хэмжигдэхүүнүүдэд суурилан бүтцийн шинжилгээг хийх боломжтой
- Мөн Бодисын солилцооны системийн үүсгэж болох метаболитуудын хувьд судлах боломжийг олгодог.
  - Оролтын үүрэгтэй метаболитуудаас хамааран урвалын зангилаа бүр идэвхтэй эсвэл идэвхгүй байж болно.
  - Шаардлагатай бүх оролцогч (substrate) байгаа тохиолдолд л эсийн дотор урвал явагдана.
  - "метаболит-урвал" сүлжээ тохиромжтой
  - Харих урвалууд нь урвалын чиглэл бүрт харгалзан хоёр болж хуваагддаг байна.
  - Урвал бүрийн оролцогч (ролт) ба бүтээгдэхүүн (гарц) -ыг тодорхой тэмдэглэдэг байх.
- Дараах функцүүдийг нэмж хэрэгжүүлнэ,
  - Тухайн эсэд идэвхтэй байх шаардлагатай метаболитуудын жагсаалтыг авч бүх идэвхтэй урвалыг ажиллуулах
  - Идэвхтэй урвалуудаар үйлдвэрлэж болох бүх метаболитуудыг тодорхойлох

БОДИСЫН СОЛИЛЦООНЫ БОЛОМЖИЙГ ҮНЭЛЭХ

Пайтон код

```
class MetabolicNetwork:

    (...)

    def active_reactions(self, active_metabolites):
        if self.net_type != "metabolite-reaction" or not self.
    split_rev:
            return None
        res = []
        for v in self.node_types['reaction']:
            preds = set(self.get_predecessors(v))
            if len(preds)>0 and preds.issubset(set(active_metabolites
    )):
                res.append(v)
        return res

    def produced_metabolites(self, active_reactions):
        res = []
        for r in active_reactions:
            sucs = self.get_successors(r)
            for s in sucs:
                if s not in res: res.append(s)
        return res
```

```
    def all_produced_metabolites(self, initial_metabolites):
        mets = initial_metabolites
        cont = True
        while cont:
            cont = False
            reacs = self.active_reactions(mets)
            new_mets = self.produced_metabolites(reacs)
            for nm in new_mets:
                if nm not in mets:
                    mets.append(nm)
                    cont = True
        return mets
```

```
def test4():
    mrsn = MetabolicNetwork("metabolite-reaction", True)
    mrsn.load_from_file("example-net.txt")
    mrsn.print_graph()

    print(mrsn.produced_metabolites(["R1"]))
    print(mrsn.active_reactions(["M1","M2"]))
    print(mrsn.all_produced_metabolites(["M1","M2"]))
    print(mrsn.all_produced_metabolites(["M1","M2","M6"]))

test4()
```

АНХААРАЛ ТАВЬСАНД БАЯРЛАЛАА