第一章　简介

第二章 Collaborative Multi Agent Reinforcement Learning

在协作的多智能体系统（multiagent system）中，每个 agent 决策的目标是选择出对整体系统最优的 action

第三章 Coordination Graphs and Variable Elimination

在协作式的多智能体系统中，每个 agent 的动作选择会对其他 agent 产生潜在的影响，即系统中各个 agent 之间存在依赖关系，一个 agent 动作的选择会取决于其他 agent 的决定。比如：TODO ： an example。所以保证各个 agent 每个时刻选择的动作都是针对整个系统的最优决策，对提高系统的整体收益具有重要的意义。通常这种问题被定义为协调问题(Coordination Problem)。本章节，我们首先回顾由 Guestrin et al. (2002a)提出的问题，计算对由n个 agents 组成的协作式多智能体系统整体最优的动作组合。系统中每个 agent i 从各自的动作集合 $\boldsymbol{A}_i$ 中选择一个 action a_i 整体组成一个动作向量（联合动作） $\boldsymbol{a} = (a_1, a_2 \dots a_n)$，进一步系统得到环境提供的一个收益 $u(\boldsymbol{a})$。协调问题的目标是选择一个动作向量 $\boldsymbol{a}^*$ 以最大化系统的整体收益 $R(\boldsymbol{a})$，即 $\boldsymbol{a}^* = argmax_a u(\boldsymbol{a})$。

针对这个问题，可以遍历所有可能的动作向量，并且选择可以最大化 $u(\boldsymbol{a})$ 的动作向量。但是，很快发现这个思路是不现实的，因为问题的解空间 $|\boldsymbol{A}_1 \times \boldsymbol{A}_2 \times \dots \times \boldsymbol{A}_n|$ 的规模，随着系统中 agent 的数量 n 成指数增长。幸运的是，现实的很多问题中，每个 agent 的决策只依赖于与其非常相关的一少部分。

由 Guestrin et al., 2002a 提出的协调图(coordination graphs ,CGs)架构是解决此类策略相互依赖问题的一种方式。此架构假设对一个 agent i，其动作的选择只依赖与与其相关的 agent $j \in \Gamma(i)$ 集合。系统整体的收益 $R(\boldsymbol{a})$ 由系统中每个 agent i 的收益 $r(i)$ 之和组成，即

$$R(\boldsymbol{a}) = \sum_{i=1}^{n} r(\boldsymbol{a_i}) \tag{1}$$

每个 agent i 的收益 $r(i)$ 取决于与其密切相关（有依赖关系）的所有 agent 的动作选择，$\boldsymbol{a_i} \subseteq \boldsymbol{a}$， $\boldsymbol{a_i} = \boldsymbol{A_i} \times (\times_{j \in \Gamma(i)} \boldsymbol{A}_j)$，这种相互依赖关系可以通过无向图 $G = (V, E)$ 表示，其中每个节点 $i \in V$ 表示 agent，每条边 $(i, j) \in E$ 表示相关的 agents i, j 需要协调各自动作的选择，$j \in \Gamma(i)$ 并且 $i \in \Gamma(j)$。于是整个系统的协调问题，被拆分为一定数量的局部协调问题，并且减小了问题的规模。

第四章 Payoff Propagation and Max-Plus Algorithm

第五章 Coordination Set Selection

第六章 实验 Accelerating Norm Emergency

在这一章，针对四五章讨论的方法，用实验进行验证。实验基于单状态(single state)的多 agent、多 action 协调 game

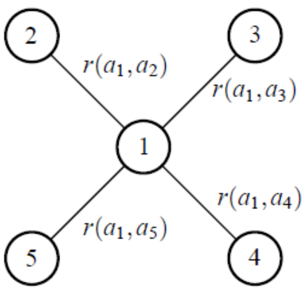
6.1 基于单状态的协调问题

定义，实验环境是由 n 个 agent 组成的协调系统，每个 agent 独立决策，并且通过对环境的探测与学习，选择对整体最优的动作，以最大化系统整体的收益。系统中每个 agent i 根据自己的策略，选择出动作 action a_i，随机地与邻居进行交互。随即，当动作执行后，一轮游戏结束，并且每个 agent i 各自收到一个回报 r_i。每个 agent i 的目标是选择出各自最优的动作 a_i^* 以最大化系统整体收益 $R(\boldsymbol{a}^*) = \sum_{i=1}^{n} r(a_i^*)$。

每个 agent i 在每一轮收到的回报 r_i 取决于与其交互的邻居 agent j。依赖关系可以通过无向图 $G = (V, E)$ 进行表示，其中每一条边 $(i, j) \in E$ 对应于相邻节点 agent i, j选择各自动作 a_i, a_j 后的收益 $r(a_i, a_j)$，如图 xxx 所示。收益函数 $r(a_i, a_j)$ 由系统提前设定(但每个 agent 不能直接获取其准确信息，需要通过学习对收益函数建模)。例如，对每个 agent i，$a_i \in \boldsymbol{A}_i$，$\boldsymbol{A}_i = \langle a_1, a_2, \ldots a_k, \rangle$，回报函数 $r(a_i, a_j)$ 定义如下：

$$
\begin{array}{cc} & a_j \\ r(a_i, a_j) = a_i & \begin{bmatrix} 1 & \cdots & -1 \\ \vdots & \ddots & \vdots \\ -1 & \cdots & 1 \end{bmatrix} \end{array}
$$

如果 agent i, j 同时选择在对角线上的动作组合 a_i, a_j，其中 $i = j$,则双方各自收到 reward +1，否则协调失败，收到 reward -1，如图 xxx 所示。



action agent i \ action agent j	1	2	3	4
1	1	-1	-1	-1
2	-1	1	-1	-1
3	-1	-1	1	-1
4	-1	-1	-1	1

(b) Example $r(a_i, a_j)$ function.

6.2 实验定义

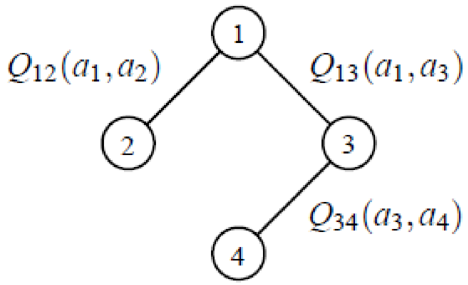
由于实验环境中，每个 agent 不能直接获取系统预设的回报函数（或 reward table），因此需要通过学习不断与环境进行交互、探测，进而对自己动作集合 A 中的每个 action 的优劣进行评估。这里使用 Q-learning 来对相邻 agent 的学习行为进行建模。

针对实验，做出以下定义：

- n 是系统中 agents 的数量。

- 每个 agent 只有一个状态。
- $\boldsymbol{A}_i = \langle a_1, a_2, \ldots a_k, \rangle$ 是 agent i 的动作空间，即 agent i 有 k 个可选动作。$\boldsymbol{A} = \boldsymbol{A}_1 \times \ldots \times \boldsymbol{A}_n$ 是系统 agents 的联合状态空间。其中 $\boldsymbol{a} = \langle a_1, a_2, \ldots a_n, \rangle$，$\boldsymbol{a} \in \boldsymbol{A}$，表示当前所有 agents 的动作选择。
- $r(a_i, a_j)$ 是系统预设的 reward table，a_i, a_j是环境中相邻 agents i,j 选择的 action，当 $a_i = a_j$，$r(a_i, a_j) = 1$否则，$r(a_i, a_j) = -1$。系统整体收益$R(\boldsymbol{a}) = \sum_{i=1}^{n} r(a_i)$。
- 假定每个 agent i 可以观察到与其交互的 agent j 的 action 选择，并且可以统计最近时间段内，对手选择各个 action 的频率。
- $Q(i,j)$ 用来记录相邻 agent i,j 之间的学习经验，以对 agent 每个 action 的优劣进行评估。
- π_i 是 agent i 选择 action 的策略，$\pi_i \rightarrow a_i$。$\boldsymbol{\pi} = argmax_{\boldsymbol{a} \in \boldsymbol{A}} \boldsymbol{Q}(\boldsymbol{a})$，是系统的整体策略（global policy）。

于是，对于此问题，各 agent 之间的依赖关系，可以通过协调图 $G = (V, E)$ 表示，其中每个节点 $i \in V$ 表示每个 agent，每条边 $(i,j) \in E$ 表示相关的 agents i,j 的局部 Q 函数 $Q(i,j)$，如下图 xxx 所示。



我们的目标是，找到一个策略$\boldsymbol{\pi} = argmax_{\boldsymbol{a} \in \boldsymbol{A}} \boldsymbol{Q}(\boldsymbol{a})$，以最大化系统的整体收益。对于一个包含多状态的 MDP 问题，可以简单的对整体使用 single Q-learning：

$$\mathbf{Q}(\mathbf{s}_t, \boldsymbol{a}^t) = (1-\alpha)\boldsymbol{Q}(\mathbf{s}_t, \boldsymbol{a}^t) + \alpha[r^t + \gamma \boldsymbol{max}_{\boldsymbol{a}^{t+1}} \boldsymbol{Q}(\mathbf{s}_{t+1}, \boldsymbol{a}^{t+1})] \qquad (1)$$

但是，由于系统整体的策略空间随 agents 的数量 n，并且往往无法观察到其他 agent 的所有信息，因此进一步把整体的 **Q** 函数拆分成各个 agent Q 函数的线性组合，即：

$$\mathbf{Q}(\mathbf{s}_t, \boldsymbol{a}^t) = \sum_{(i,j) \in E} Q_{ij}(s_{i,j}^t, a_i, a_j) \qquad (2)$$

于是，等式(1) 可以被重新表示为：

$$\sum_{(i,j) \in E} Q_{ij}(s_{i,j}^t, a_i, a_j) = (1-\alpha) \sum_{(i,j) \in E} Q_{ij}(s_{i,j}^t, a_i, a_j) + \alpha\left[{r_{ij}}^t + \gamma \boldsymbol{max}_{\boldsymbol{a}^{t+1}} \boldsymbol{Q}(\mathbf{s}_{t+1}, \boldsymbol{a}^{t+1})\right] (3)$$

上式中，因为$\boldsymbol{max}_{\boldsymbol{a}^{t+1}} \boldsymbol{Q}(\mathbf{s}_{t+1}, \boldsymbol{a}^{t+1})$ 取决于对整体最优的联合 action $\mathbf{a}^*$，因此不能直接拆分为各个 agent 局部最优 Q 值之和。但是我们可以通过 VE 或 Max-Plus 等方式，通过使每个 agent i 选择出对整体最优的 action a_i^* 来计算出对整体最优的联合 action $\mathbf{a}^*$。其中 $\boldsymbol{max}_{\boldsymbol{a}^{t+1}} \boldsymbol{Q}(\mathbf{s}_{t+1}, \boldsymbol{a}^{t+1}) = \boldsymbol{Q}(\mathbf{s}_{t+1}, \boldsymbol{a}^*) = \sum_{(i,j) \in E} Q_i(s_{i,j}^{t+1}, a_i^*, a_j^*)$。于是对于每一个 agent 对，

有：

$$Q_{ij}(s_{i,j}^{t},a_i,a_j) = (1-\alpha)Q_{ij}(s_{i,j}^{t},a_i,a_j) + \ \alpha r(a_i,a_j) + \ \gamma Q_{ij}(s_{i,j}^{t+1},a_i^*,a_j^*) \qquad (4)$$

对于单状态的协调问题，下一个状态的 Q 函数没有定义，因此在本实验中，每个 agent i 在每一轮中，选择自己的 action 时，直接考虑选择对当前系统整体最优的 action a_i^*，并且以一定的探索率 ε 随机对动作空间中的 action 进行探索。

6.3 Coordination action selection

如上节所示，实验中各 agents 的协调图 CG（Coordination Graph）如图 xxx 所示。于是，每个 agent i (CG 中的节点)，向它的邻居 agent $j \in \Gamma(i)$发送的消息定义为：

$$\mu_{ij}(a_j) = max_{a_i}\left\{Q_{ij}(a_i,a_j) + \sum_{k\in\Gamma(i)\backslash j}\mu_{ki}(a_i)\right\} + c_{ij}$$

其中 $\Gamma(i)\backslash j$ 表示 agent i 除了 j 以外的所有邻居，参数 c_{ij} 是为了标准化消息数值的取值范围，防止某些 agents 组成的网络中，存在环状结构，进而导致由 i 发送出去的消息，一定时间后，又发送到 i，进而导致消息值的无限增大。这个消息 μ_{ij} 是对给定一个目标 agent j 的动作 a_j，agent i 所能实现的最大收益值的近似。通过最大化与目标 agent j 之间的平均回报 $Q_{ij}(a_i,a_j)$ 以及 agent i 的所有邻居（j 以外的）向其发送的消息数值总和来计算当前消息 μ_{ij}。每个 agent 不断向邻居发送消息直到消息的值不再变化，或者到达指定的发送轮数。当网络中所有消息值都达到稳定时，每个消息中都包含了网络中所有边（i,j）上的收益，所以最大化当前消息值即最大化了系统的整体收益 **Q**,因此对每个 agent i，即找到了能最大化整体收益的 action a_i^*。

$$a_i^* = \ argmax_{a_i}\sum_{j\in\Gamma(i)}\mu_{ij}(a_j)$$

整个算法计算过程如 Algorithm XXX 所示，其中 $c_{ij} = \frac{1}{|\Gamma(i)|}\sum_{k\in\Gamma(i)}\mu_{ik}(a_k)$。在很多情况下，随着消息值的抖动，各个 agent 的最优 action a_i^* 也在不断变化，因此进一步拓展，只有当 agent 收到的收益 $g_i(a_i')$ 提高时，才对其最优 action a_i^* 进行更新。

Algorithm 1 runDCOP(centralized max-plus algorithm for CG(V,E))

```
initialize μ_ij = μ_ji = 0 for (i,j) ∈ E, m=−∞, fixed_point=false
while fixed_point = false and deadline to send action has not yet arrived do
    // run one iteration
    fixed_point = true
    for every agent i do
        for all neighbors j = Γ(i) do
        send j messages μ_ij(a_j) = max_{a_i}{Q_ij(a_i,a_j) + Σ_{k∈Γ(i)\j} μ_ki(a_i)} + c_ij
        if μ_ij(a_j) differs from previous message by a small threshold then
            fixed_point = false
        determine g_i(a_i) = Σ_{j∈Γ(i)} μ_ji(a_i) and a_i' = argmax_{a_i} g_i(a_i)
        if use anytime extension then
            if g_i(a_i') > m then
            a_i* = a_i' and m = g_i(a_i')
        else
```

$a_i^* = a_i'$
set best action for agent i = a_i^*
end for
end for

6.4 Coordination Set Selection

Algorithm runDCOP 中，消息的数量与系统的 CG（Coordination Graph）中，边的条数成正比。对于一个足够大网络结构来说，各个 agent 的相互依赖关系比较复杂，图中每个节点的度数可能比较大，因而消息发送的次数频繁。但是在现实环境中，每个 agent 通信的资源数量往往是有限的，并且通信的代价往往比较昂贵，因此我们设计了一种动态调整，选择出当前时刻，对各个 agent 最有益的最小协调子集（Coordination Set，$CS \subset \Gamma(i)$），以减少在 CG 中相互依赖的边的数量，进而减少每个 agent 发送 message 的数量，进而降低通信的代价。

定义一：在稳定状态的 Coordination Set（CS）中，对任意 agent i, 其邻居 agent j，将无条件的配合 agent i 的行为选择 action，以最大化其局部的整体最大收益。对于初始网络中 agent i 的邻居 $k \in \Gamma(i)$ and $k \notin CS$（$\Gamma(i)$ 是网络初始化时，agent i 的所有邻居组成的集合），agent i 能够根据对 agent k 行为的观察，统计出当前其选择各个 action 的概率，进一步可计算其对 i 收益的平均影响。

定义二：当选定 Coordination Set＝C 时，agent i 的预期最大收益（the potential expected utility） $\mathrm{PV}(a_i, C)$：

$$\mathrm{PV}(a_i, C) = \sum_{j \in C} max_{a_j} Q_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i),\ k \notin C} \sum_{a_k} P_k(a_k) Q_{ik}(a_i, a_k)$$

其中，$P_k(a_k)$，$k \in \Gamma(i)$ and $k \notin C$，是 agent i 对 agent k 最近一段时间选择各个 action 的可能性的统计概率。

定义三：不与 NC 协调而造成的预期损失（the potential loss in lack of coordination）

$$\mathrm{PLILOC}_i(\mathrm{NC}) = max_{a_i, i \in \Gamma(i)} \mathrm{PV}(a_i, \Gamma(i)) - max_{a_k, k \in \Gamma(i) \setminus NC} \mathrm{PV}(a_k, \Gamma(i) \setminus \mathrm{NC})$$

其中，$\mathrm{PLILOC}_i(\emptyset) = 0$。

整个算法过程描述如下：其中δ 代表系统允许的最大损失率，此处设置为 0.001

Algorithm 2 computeCoordinationSet(*i*)

initialize $maxLoss = \delta * \max\{|max_{a_i} \mathrm{PV}(a_i, \boldsymbol{\Gamma}(\boldsymbol{i}))|, \mathbf{PLILOC}_i(\boldsymbol{\Gamma}(\boldsymbol{i}))\}$
find $C \subset \boldsymbol{\Gamma}(\boldsymbol{i})$, such that
(1) $\mathbf{PLILOC}_i(\boldsymbol{\Gamma}(\boldsymbol{i}) \setminus C) \le maxLoss$
(2) $\mathbf{PLILOC}_i(\boldsymbol{\Gamma}(\boldsymbol{i}) \setminus D) > maxLoss$, for all $D \subset \Gamma(i)$ and $|D| < |C|$
(3) $\mathbf{PLILOC}_i(\boldsymbol{\Gamma}(i) \setminus C) \le \mathbf{PLILOC}_i(\boldsymbol{\Gamma}(\boldsymbol{i}) \setminus D)$ for all $D \subset \Gamma(i)$ and $|D| = |C|$
return C

6.5 Coordinated Learning process

Algorithm 3 The coordinated learning process

initialize learning rate $\alpha = 1$,explore rate $\varepsilon = 1$, loss rate $\delta = 0.001$

while not converge **do**

  runDCOP() to select the best action a_i^* for each agent i

  for every agent *i* **do**

    random select a neighbor *j* to interact

    each agent *i,j* select the its' action a_i, a_j(each select the best action a_i^*, a_j^*

      with some explore rate ε)

    each agent observed the reward $r(a_i, a_j)$,and observed each other's action (for

      record $P_j(a_j)$ *and* $P_i(a_i)$)

    each agent update its' Q table

    agent *i* update its' learning rate α and explore rate α with some decay

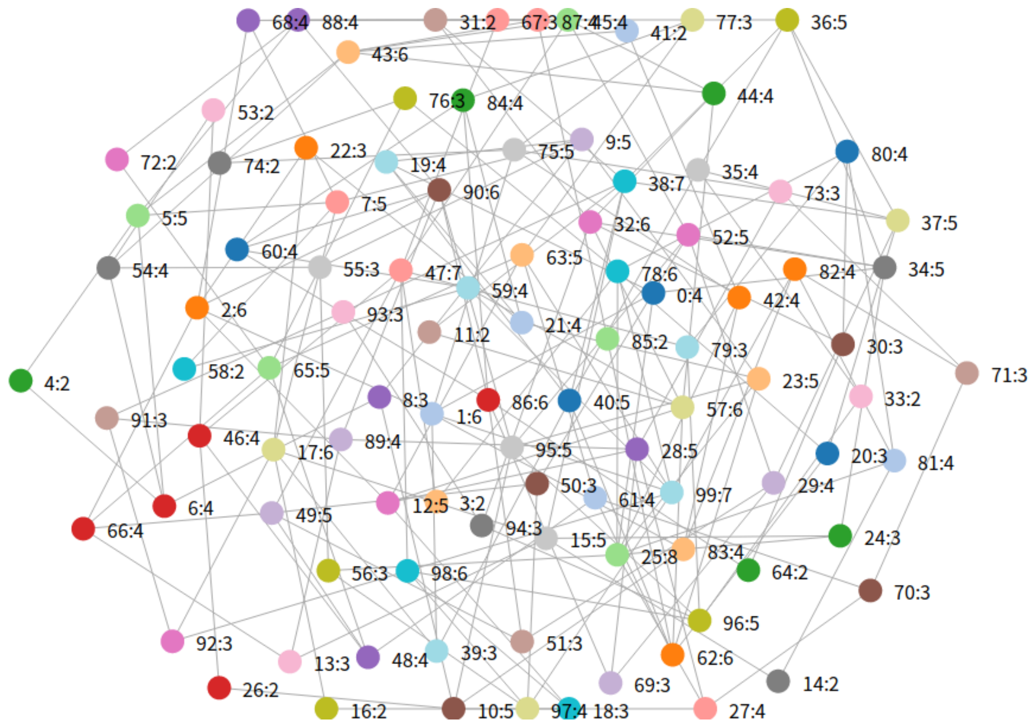
    computeCoordinationSet(*i*)

  end for

6.6 实验结果

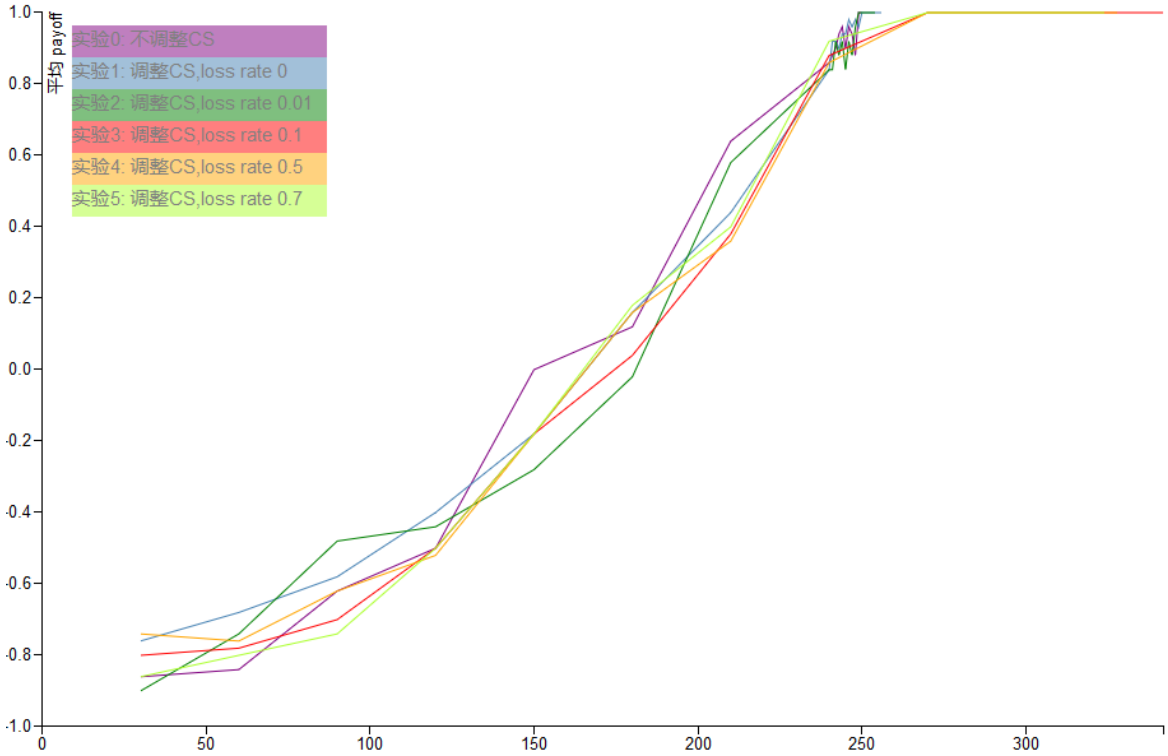
6.6.1 实验环境：100 agents,10 action

6.6.2 Small world 网络下



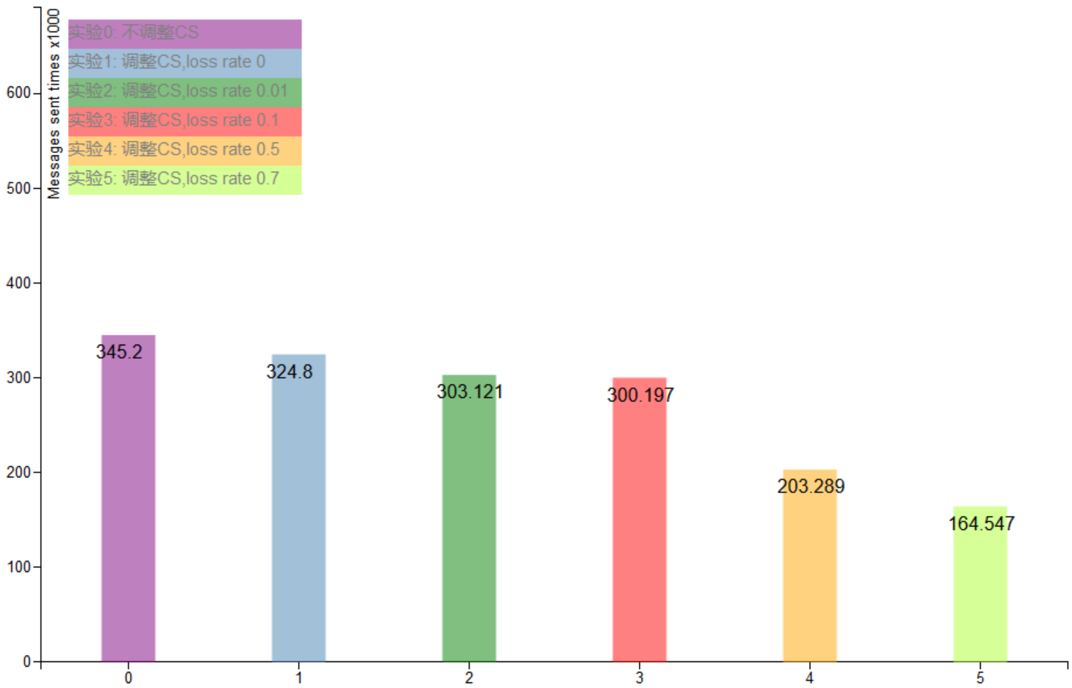
网络结构示意图 small-world

1）收敛情况及轮数对比图



收敛情况及轮数对比图

2）消息（Message）发送次数对比图



收敛时各实验 Message 发送次数

6.7