Санкт-Петербургский государственный университет
Кафедра математической теории игр и статистических решений

Курносых Злата Александровна

Выпускная квалификационная работа бакалавра

Эволюционные игры на сетях

Направление 010400.62
Прикладная математика, фундаментальная информатика и программирование

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Губар Е. А.

Санкт-Петербург
2017

Содержание

Введение

Окружающий мир усложняется с каждым днем и нуждается в качественных моделях, описывающих динамические процессы, протекающие в нем. В случае взаимодействия большого количества действующих лиц (далее будем называть их агентами), результаты, полученные с использованием моделей классической теории игр, не всегда дают видение полной картины. В отличие от классической теории, эволюционная игра описывает адаптацию агентов: при выборе стратегии агенты ориентируются не только на матрицу выигрышей, действуя автономно, но также склонны перенимать поведение более успешных агентов. В работах [11] и [12] описаны различные виды динамик, являющиеся «правилами подражания» агентов внутри популяции. Данный подход к описанию изменения сложных систем, путем рассмотрения поведения его элементов, был назван *эволюционной теорией игр.*

Эволюционные игры — это молодой раздел теории игр. Зарождение данной дисциплины связано с работой [9], в которой были даны основные определения и рассмотрены примеры ее применения в биологии. Позднее были введены базовые положения эволюционной модели:

- число агентов велико,
- отдельно взятый агент оказывает незначительное влияние на других агентов популяции,
- выбор взаимодействующих агентов случаен: функция выигрыша каждого агента зависит от поведения других агентов популяции и распределения их выборов стратегий,
- количество возможных чистых стратегий конечно: все агенты популяции делятся на группы в зависимости от их стратегии поведения (чистой либо смешанной).

Дальнейшие исследования показали возможности данного аппарата в решении более обширного круга задач: процесс смены тех или иных общественных предпочтений, распространение информации, приспособление биологических организмов или экономических агентов к различным условиям и т. д.

Однако при построении подобных динамических моделей возникает ряд сложностей. Во-первых, большое число агентов не позволяет рассматривать поведение всей популяции путем перебора возможных вариантов развития, а аппроксимационные методы работают только при введении дополнительных предположений, которые зачастую искажают настоящую картину. Во-вторых, структура популяции редко бывает однородной, т. е. взаимодействие «всех со всеми» в реальном мире почти невозможно. Однако в настоящее время приобретают популярность эволюционные модели, учитывающие структуру популяции, которая может быть представлена в виде сети. В нашем случае сеть представляет способ задания связи между агентами популяции, ее структура оказывает влияние на процесс адаптации агентов к окружающей среде, так как в отличие от неструктурированных моделей, любые взаимодействия агентов могут происходить только при наличии связи между ними.

В данной работе представлено общее описание эволюционного процесса с учетом различных сетевых структур, разработан новый алгоритм, сочетающий в себе как точный подход к исследованию поведения агентов, так и его локальную аппроксимацию, приведена специальная процедура на языке Python, позволяющая проследить процесс эволюции, и результаты численных экспериментов.

Постановка задачи

Определим для начала классическую постановку эволюционной игры [11], введем основные понятия и обозначения и далее перейдем к определению эволюционной игры на сети [10].

Классическая постановка

Рассмотрим отображение $F : \Delta \to R^n$, являющееся *эволюционной игрой*, где $S = \{1, \ldots, n\}$ — набор чистых стратегий, N — количество агентов в популяции, Δ — множество возможных состояний популяции. Состояние $\delta \in \Delta$ есть вектор $\delta = (\delta_1, \ldots, \delta_s)$, в котором каждая компонента δ_i есть доля агентов, использующих i-ю стратегию. Значением игры $F(\delta)$ является вектор выигрышей $\pi = (\pi_1, \ldots, \pi_n)$, где π_i — выигрыш при использовании чистой i-й стратегии.

При изменении стратегии каждый из агентов может использовать специальную процедуру, которая описывает *когда* сменить стратегию и *какую* стратегию выбрать. Будем называть ее протоколом принятия решений.

Определение. *Протоколом принятия решений* ρ называется отображение $\rho : R^n \times \Delta \to R_+^{n \times n}$. Элемент $\rho_{ij}(\pi, \delta)$ называется коэффициентом условного переключения.

Если $\sum_{j \in S} \rho_{ij}(\pi, \delta) = R$, то, нормируя коэффициенты условного переключения, получим ρ_{ij}/R — вероятность смены агентом i-й стратегии на j-ю.

Базовое уравнение динамики эволюционного процесса (mean dynamics):

$$\dot{\delta}_i = \sum_{j \in S} \delta_j \rho_{ji}(F(\delta), \delta) - \delta_i \sum_{j \in S} \rho_{ij}(F(\delta), \delta), \tag{1}$$

где первое слагаемое — доля агентов перешедших на стратегию i с других всевозможных стратегий, второе слагаемое — доля агентов, изменяющая свою текущую стратегию i на любую другую. Изменяя способ задания протокола принятия решений, можно получить различные виды динамик.

Наиболее известной в эволюционной теории игр является репликативная динамика (replicator dynamics). Уравнение репликативной динамики

получается, когда протокол принятия решения задается следующим образом:

$$\rho_{ij}(\pi, \delta) = \delta_j[\pi_j - \pi_i]_+.$$

Подставив это выражение в базовое уравнение динамики, получим:

$$\dot{\delta}_i = \delta_i \sum_{j \in S} \delta_j(\pi_i - \pi_j), \tag{2}$$

при рассмотрении взаимодействия агентов на основе *симметричной биматричной игры* , с матрицей выигрышей M, получим:

$$\sum_{j \in S} \delta_j \pi_i = [M\delta]_i, \sum_{j \in S} \delta_j \pi_j = \delta M \delta,$$

тогда перепишем уравнение репликативной динамики в матричном виде:

$$\dot{\delta}_i = \delta_i([M\delta]_i - \delta M \delta), \tag{3}$$

где $[M\delta]_i$ — i-й элемент вектора. В данном выражении это слагаемое есть выигрыш i-й чистой стратегии против смешанной стратегии популяции δ.

Определение [2]. *Симметричной* называется биматричная игра, если множества чистых стратегий игроков совпадают, т. е. $S_i = S_j$ и выполняется условие $M_i = M_j^T$, где M_i и M_j — матрицы выигрышей i-го и j-го игроков соответственно, $\forall i, j \in \mathcal{V}$.

Эволюционная игра с учетом графа взаимодействий

Пусть $\mathcal{G}$=$<\mathcal{V}, \mathcal{E}>$ — неориентированный граф, где $\mathcal{V} = \{1, \ldots, n\}$ — множество игроков (агентов популяции) и $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ — набор ребер, каждое из которых представляет собой симметричную игру между двумя соседними агентами.

Предполагаем, что в каждый момент времени агент выбирает одну чистую стратегию против всех оппонентов. Обозначим *состояние игры* с помощью вектора $x(t) = [x_1(t), \ldots, x_n(t)]^T$, где $x_i(t) \in S_i$ — стратегия i-го игрока в момент времени t.

Выигрыши игроков обозначим:

$$y_i(t) = \frac{1}{|N_i|} \sum_{j \in N_i} M_{x_i(t), x_j(t)}, \tag{4}$$

где $N_i := \{j \in \mathcal{V} : i, j \in \mathcal{E}\}$ — множество оппонентов i-го игрока, а $M_{x_i(t),x_j(t)}$ — выигрыш i-го игрока при взаимодействии с j-м, $|N_i|$ — количество соседей i-го игрока. Вектор выигрышей агентов $y(t) = [y_1(t), \ldots, y_n(t)]^T$. Далее можно в общем виде записать *правило изменения стратегий* i-го игрока как функцию от элементов состояния популяции (стратегий) и выигрышей агентов:

$$x_i(t+1) = f(\{x_j(t), y_j(t) : j \in N_i \cup \{i\}\}). \tag{5}$$

Одним из известных правил изменения состояния популяции является механизм *пропорциональной имитации*: каждый игрок выбирает соседа случайно, и если он в предыдущий момент времени достигает лучшего результата, используя другую стратегию, то агент может поменять свою стратегию с вероятностью, пропорциональной разности их выигрышей

$$p(x_i(t+1) = x_j(t)) = \left[\frac{\lambda}{|N_i|}(y_j(t) - y_i(t))\right]_0^1, \tag{6}$$

где $j \in N_i$ (j — случайно выбранный сосед i-го агента), $\lambda > 0$ — произвольный коэффициент пропорциональности, варьируя данный коэффициент можно получить различные виды динамик, удовлетворяющие конкретным примерам, квадратные скобки $[Z]_0^1$ означают $\max(0, \min(1, Z))$. Заметим, что приведенная выше вероятность — аналог *протокола принятия решений* из классической популяционной игры.

Построение модели

Модель эволюционной игры, представленная в данной выпускной квалификационной работе, основывается на построении протокола принятия решений для *каждого агента* на каждом этапе эволюции популяции. В отличие от описанной вероятности (6), используемый протокол принятия решений является вероятностью того, что i-ый агент примет стратегию s. Данный подход аналогичен протоколу для репликативной динамики, однако из-за неоднородности структуры популяции мы не можем оперировать только долями популяции, относящимися к той или иной стратегии. Зададим вероятность смены стратегии для *каждого* агента, однако, исходя из предположения, что встречи агента с оппонентами из множества его

возможных соседей равновероятны [4], опишем окружающую его «популяцию соседей» как в классической популяционной игре. Таким образом, комбинируя описанные выше представления протокола принятия решений, введем значение вероятности того, что i-й агент сменит свою стратегию на стратегию s следующим образом:

$$p(x_i(t) = s) = \delta_s^{N_i}[y_{N_i}^s(t) - y_i(t)]_0^1, \tag{7}$$

где $\delta_s^{N_i}$ — доля соседей i-го агента, использующих стратегию s, $y_{N_i}^s$ — средний выигрыш соседей i-го агента, использующих стратегию s, y_i — средний выигрыш i-го агента против всех его возможных оппонентов, квадратные скобки $[Z]_0^1$, как и в выражении (6), означают $\max(0, \min(1, Z))$. Также следует учесть, что если $1 - \sum_{k \in S} p_{ik} > p_{i\hat{s}}$, для любых $\hat{s} \in S$, то агент не меняет свою стратегию на данном этапе процесса.

Новая модель фактически исключает случайную составляющую процесса, учитывая вероятность встречи агента с оппонентами, использующими одну из стратегий, и разность их средних выигрышей.

Глава 1. Репликативная динамика

1.1 Общий вид уравнения

В начале предыдущего раздела (см. Постановка задачи) были даны основные определения эволюционной теории игр и описана одна из наиболее известных динамик эволюционного процесса, а именно репликативная динамика. Данное уравнение дает «хорошую» аппроксимацию в случае выполнения ряда предположений:

1. Число агентов в популяции велико.
2. Популяция хорошо смешана, т. е. взаимодействия между всеми агентами равновероятны.
3. Агенты используют пропорциональную имитацию: агент меняет свою стратегию на стратегию оппонента с вероятностью, пропорциональной разности их выигрышей.

В случае возможности описания попарного столкновения агентов симметричной матричной игрой, уравнение репликативной динамики выглядит следующим образом:

$$\dot{\delta}_i = \delta_i([M\delta]_i - \delta M\delta), \quad i \in S, \tag{8}$$

где S — множество чистых стратегий, A — матрица выигрышей в симметричной игре, $\delta = [\delta_1, \ldots, \delta_s]$ — вектор долевого соотношения агентов на множестве стратегий: δ_i — доля агентов, использующих стратегию i, $\sum_{i=1}^{s} \delta_i = 1$. Данное уравнение иллюстрирует зависимость изменения доли агентов популяции, использующих чистую стратегию от разности выигрыша данной чистой стратегии против смешанной стратегии популяции и выигрыша популяции против самой себя.

1.2 Модели репликативной динамики на примере классических биматричных игр

Рассмотрим некоторые классические биматричные симметричные игры, построим на их основе модели репликативной динамики и с помощью

процедуры в среде Maple (см. Приложение 1) проследим эволюционное развитие популяций, с помощью описания фазового портрета уравнения динамики. Представленные в данном параграфе результаты широко известны, однако, они необходимы для последующего сравнения с результатами, полученными экспериментально.

Пример 1. Составим и исследуем уравнение репликативной динамики популяции, попарное взаимодействие агентов которой подобно симметричной игре «Дилемма заключенного» [6]. Общий вид матрицы выигрышей M:

$$\begin{array}{c} \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cc} C & D \end{array} \\ \begin{pmatrix} (a,a) & (b,c) \\ (c,b) & (d,d) \end{pmatrix} \end{array},$$

где $c > a > d > b$ и $2a > c + b$. В нашем случае использовались коэффициенты: $a = 4, b = -1, c = 5, d = 0$.

Данную игру можно интерпретировать так: агенты либо сотрудничают (cooperate) ради общего блага (C), либо заботятся только о собственной выгоде (don't cooperate) (D). В итоге для двух агентов возможно четыре исхода:

- оба сотрудничают (C, C) и получают выигрыш по 4 единицы;
- оба предают (D, D) и получают выигрыш 0;
- первый предает, второй сотрудничает (D, C) — выигрыши 5 и -1 соответственно;
- первый сотрудничает, второй предает (C, D) — выигрыши -1 и 5.

На изображении (см. рис. 1) можем наблюдать, что из любого состояния популяция придет в точку $(0, 1)$, следовательно, все агенты выберут стратегию D.

Пример 2. В данной популяции попарное взаимодействие агентов характеризуется игрой «Ястреб-Голубь»[9].

Широко известна классическая интерпретация данной игры. Есть два вида стратегий: ястреб (Howk) и голубь (Dove). Между агентами популяции происходит борьба за ресурсы. В случае столкновения двух ястребов (H, H), ресурс делится пополам, но они также наносят друг другу увечья

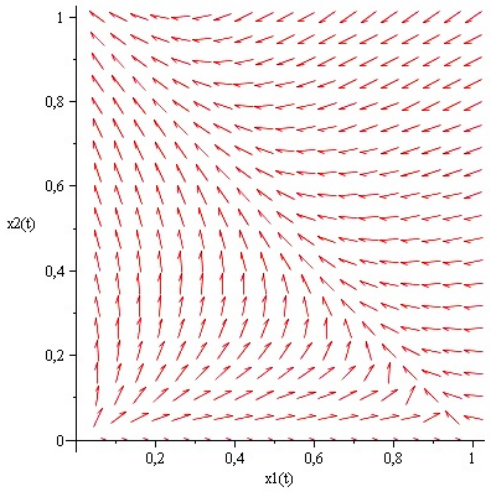


Рис. 1: «Дилемма заключенного». Фазовый портрет уравнения репликативной динамики.

в ходе борьбы. Когда сталкиваются два голубя (D, D), они просто делят ресурс пополам. При столкновении ястреба и голубя (H, D) или (D, H), ястреб забирает себе весь ресурс, а голубь отступает ни с чем. Множество стратегий $S = \{H, D\}$. Матрица взаимодействий M:

$$\begin{array}{c} \\ H \\ D \end{array} \begin{array}{c} \begin{array}{cc} H & \qquad D \end{array} \\ \begin{pmatrix} (1/2(V-C), 1/2(V-C)) & (V, 0) \\ (0, V) & (1/2V, 1/2V) \end{pmatrix} \end{array},$$

где V – вознаграждение за получение ресурса, C – ранение при столкновении двух ястребов. В модели используются значения: $V = 4, C = 6$.

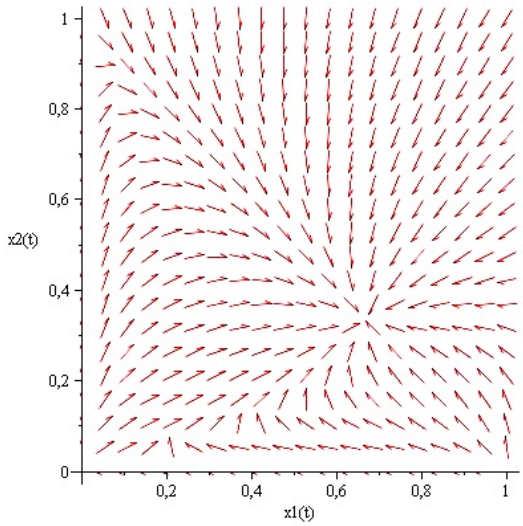


Рис. 2: «Ястреб-Голубь». Фазовый портрет уравнения репликативной динамики

На изображении (см. рис. 2) видим, что в этой игре устойчивое состояние равновесия в смешанных стратегиях: $\overline{\delta} = [2/3, 1/3]$, где компоненты

2/3 — доля популяции ястребов, 1/3 — доля голубей.

Пример 3. Агенты рассматриваемой популяции попарно взаимодействуют подобно игре «Охота на оленя» (Stag Hunt).
Впервые данная игра была описана в работе [7] французским философом Жан-Жаком Руссо:

> *«Если охотились на оленя, то каждый понимал, что для этого он обязан оставаться на своем посту; но если вблизи кого-либо из охотников пробегал заяц, то приходилось сомневаться, что этот охотник без зазрения совести пуститься за ним вдогонку и, настигнув добычу, весьма мало будет сокрушаться о том, что таким образом лишил добычи своих товарищей.»*

Множество стратегий: $S = \{s, r\}$. Матрица взаимодействий M:

$$\begin{array}{c} \\ s \\ r \end{array} \begin{array}{c} \begin{array}{cc} s & r \end{array} \\ \begin{pmatrix} (a,a) & (b,c) \\ (c,b) & (d,d) \end{pmatrix} \end{array},$$

где $a > c >= d > b$. В построенной модели использовались коэффициенты: $a = 3, b = 0, c = 1, d = 1$.

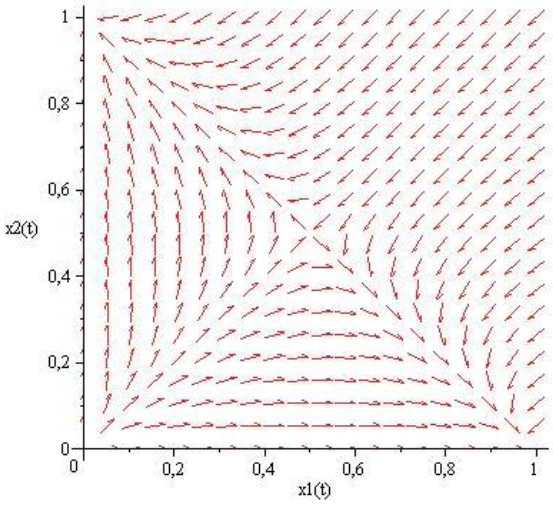


Рис. 3: «Охота на оленя». Фазовый портрет уравнения репликативной динамики

В данной игре популяций (см. рис. 3) есть 2 устойчивых равновесия: ситуации $\overline{\delta_1} = [1, 0]$ и $\overline{\delta_2} = [0, 1]$, т. е. в случае, когда все выберут действовать сообща и охотиться на оленя, либо все побегут за зайцем. Также есть

одно не устойчивое равновесие: $\overline{\delta_3} = [1/2, 1/2]$ — когда половина охотится на оленя, а половина — бегут за зайцем, однако малейшее отклонение от этой стратегии приведет к «чистому» равновесию.

Глава 2. Эволюционная игра с учетом структуры популяции

В данной главе будут представлены результаты экспериментов по построению модели эволюционной игры с заданной схемой возможных взаимодействий. Для вычислений был использован ряд процедур, написанных на языке программирования Python [5]. В ходе экспериментов использовались различные классические биматричные игры, описание которых представлено выше, различные сетевые структуры, а также варьировалось число агентов в популяции и начальное состояние системы (начальное распределение стратегий).

2.1 Виды структур популяций

При построении модели использовались несколько основных видов расположения и взаимосвязи агентов популяции:

1. *Кольцо* — у каждого агента есть 2 соседа, каждый агент может приходиться соседом только двум игрокам. Агенты расположены последовательно и образуют замкнутую структуру.

2. *Решетка* — у каждого агента возможно 2, 3 или 4 соседа, в зависимости от их расположения. Структура ячеистая.

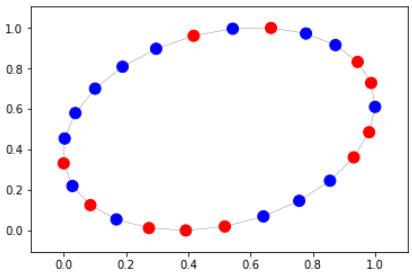
3. *Случайная сеть* — генерация случайного графа. Распределение ребер на графе равномерное: с вероятностью 20 % есть ребро между двумя вершинами.

2.2 Результаты экспериментов

В данном параграфе будут представлены наиболее интересные результаты экспериментов, их описание и частичный сравнительный анализ. Код процедуры, используемой для проведения экспериментов, представлен в Приложении (см. Приложение 2).

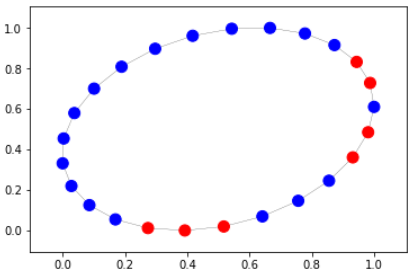
Пример 1.1 Рассмотрим популяцию из 25-ти агентов, характер взаимодействия которых, описывается классической игрой «Дилемма заключенного». Рассмотрим процесс эволюции на графе типа *кольцо*. На изоб-

ражениях красные узлы — агенты, использующие стратегию C, синие — агенты, использующие стратегию D



а)

Начальное состояние. Стратегия C : 11 агентов, D : 14 агентов.



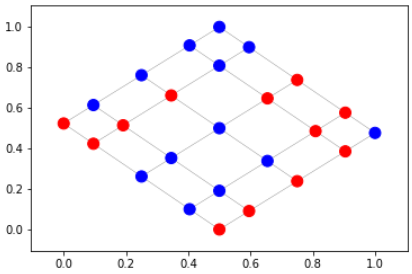
:б)

Стационарное состояние. Стратегия C : 7 агентов, D : 18 агентов.

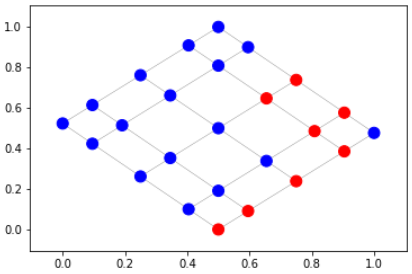
Рис. 4.

На изображении (см. рис. 4) можем видеть как одиночные агенты с красной стратегией (C) меняют ее на синюю (D), однако скопление агентов со стратегией C устойчиво к изменениям, благодаря тому, что их соседнее расположение обеспечивает этим агентам больший выигрыш, чем у игроков со стратегией (D).

Пример 1.2 Теперь рассмотрим поведение популяции 25-ти агентов со структурой *решетка*. В данной популяции (см. рис. 5) есть некое подобие



а) Начальное состояние. Стратегия C : 12 агентов, D : 13 агентов.



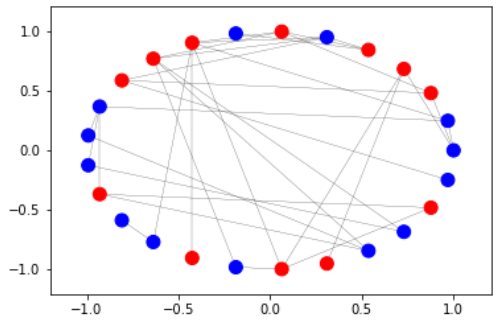
б) Стационарное состояние. Стратегия C : 8 агентов, D : 17 агентов.

Рис. 5.

иерархии по количеству возможных соседей, благодаря этому достижение стационарного состояния популяцией происходит за большее число шагов, чем в предыдущем случае. В данном примере это произошло за пять эволюционных периодов.

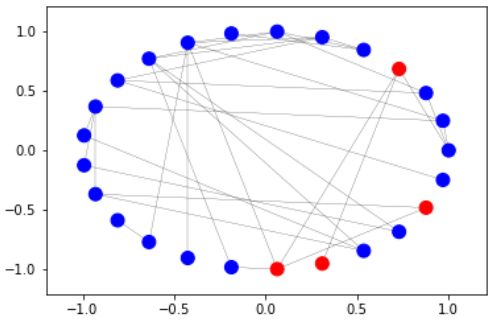
На изображениях выше можем наблюдать, что в стационарном состоянии образовалась группа агентов со стратегией C, которой при данном расположении выгодно использовать эту стратегию и нет необходимости ее менять. Однако в углу решетки остался один агент со стратегией D, которому также не выгодно ее менять, несмотря на то, что все его соседи «красные». Если взглянуть на матрицу выигрышей (см. п. 1.2), то можно понять, что такое соседство наиболее выгодно для «синего» игрока, именно при взаимодействии стратегии D против стратегии C выигрыш агента D максимален.

Пример 1.3 Рассмотрим последний пример для популяции, характер взаимодействия агентов которой основан на игре «Дилемма заключенного». Размер популяции также 25 агентов, граф возможных взаимодействий — *случайный* (см. п. 2.1).



а)

Начальное состояние. Стратегия C : 12 агентов, D : 13 агентов.



б)

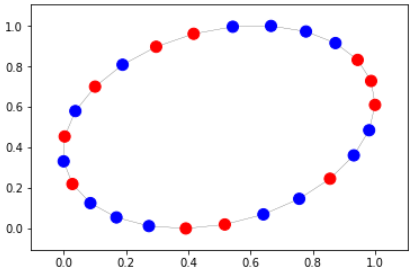
Стационарное состояние. Стратегия C : 4 агентов, D : 21 агент.

Рис. 6.

Здесь можем наблюдать (см. рис. 6), что в стационарном состоянии осталось только 4 агента со стратегией C. Это произошло из-за того, что на случайном графе больше ребер, а значит обширнее возможные взаимодействия. С увеличением числа связей между агентами, популяция становится все более однородной, в итоге превращаясь в характер взаимодействия «всех со всеми» (см. п. 1.1), данная структура более предсказуема и для нее есть хорошие аппроксимационные уравнения, в отличие от рассмотренных в данной главе примеров.

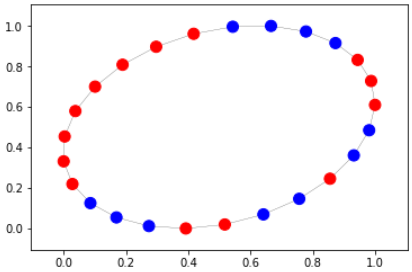
Пример 2.1 Рассмотрим поведение популяции, состоящей из 25-ти агентов, с матрицей взаимодействия классической биматричной игры

«Ястреб-Голубь». На изображениях красные узлы — стратегия ястреба (H), синие — голубя (D). В первом случае популяция задана графом *кольцо*.



а)

Начальное состояние. Стратегия H : 11 агентов, D : 14 агентов.



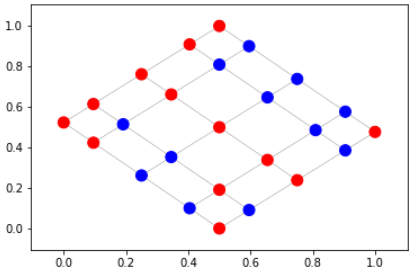
б)

Стационарное состояние. Стратегия H : 14 агентов, D : 11 агентов.

Рис. 7.

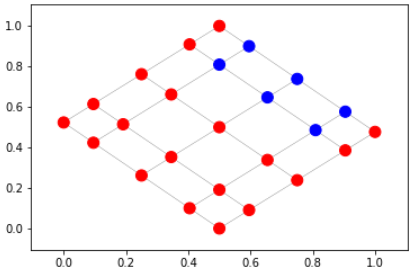
На изображениях (см. рис. 7) видно, что, так как при взаимодействии ястреба с голубем ястреб получает свой максимально возможный выигрыш, а голубь — минимально возможный, то соседство голубя с двумя ястребами вынуждает его сменить стратегию на более выгодную. Однако ястреб, окруженный двумя голубями, не меняет свою стратегию на противоположную, таким образом, в стационарном состоянии преобладает доля агентов-ястребов.

Пример 2.2 Рассмотрим развитие популяции с таким же характером взаимодействия на сети типа *решетка*.



а)

Начальное состояние. Стратегия H : 13 агентов, D : 12 агентов.



б)

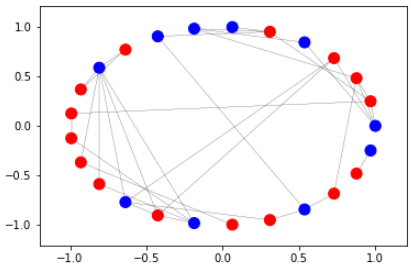
Стационарное состояние. Стратегия H : 19 агентов, D : 6 агентов.

Рис. 8.

Изображения (см. рис. 8) иллюстрируют распространение страте-

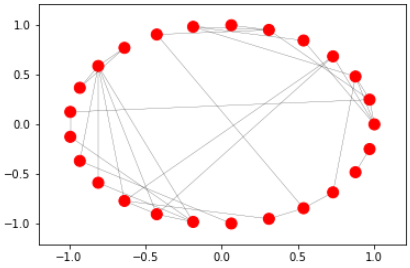
гии ястребов, однако скопление голубей способно обеспечить друг другу больший выигрыш, чем скопление ястребов: это обусловлено тем, что ястребы при соперничестве несут дополнительные потери во время борьбы за ресурс, в отличие от голубей, которые просто делят его пополам (см. п. 1.2, Пример 2). Таким образом, в стационарном состоянии присутствуют как агенты-ястребы так и агенты-голуби и их соотношение близко к равновесию в случае однородной структуры популяции.

Пример 2.3 Также исследуем поведение популяции «Ястреб-Голубь» на *случайном* графе.



а)

Начальное состояние. Стратегия C : 15 агентов, D : 10 агентов.



б)

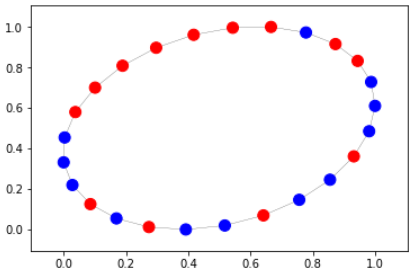
Стационарное состояние. Стратегия C : 25 агентов.

Рис. 9.

Из начального состояния популяции (см. рис. 9. а)), в которой преобладает число ястребов, в течение трех эволюционных периодов вся популяция приходит к стратегии ястребов (см. рис. 9. б)).

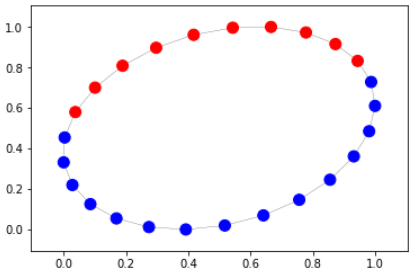
Пример 3.1 В последней серии экспериментов будет описано развитие популяции с характером взаимодействия «Охота на оленя» также для трех различных графов возможного взаимодействия. На изображениях красные узлы — стратегия «ожидания оленя» (s), синие — «отвлечься на кролика» (r). Первый пример на сети типа *кольцо*.

На изображении (см. рис. 10) видим, что агенты со стратегией «ждать оленя», окруженные агентами с противоположной стратегией, меняют ее, так как при взаимодействии с «отвлекшимися на кролика» получают меньший выигрыш, однако скопление «ожидающих оленя» при взаимодействии друг с другом обеспечивают себе максимальный выигрыш, заставляя изменить свою стратегию одиночного среди них агента со стратегией «отвлечься». Таким образом, популяция стремится к однородности в окрестности



а)

Начальное состояние. Стратегия C : 13 агентов, D : 12 агентов.



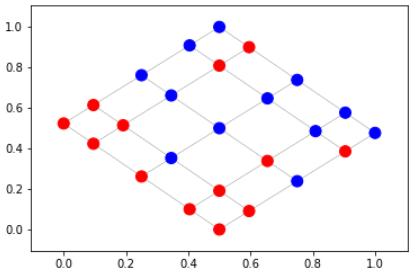
б)

Стационарное состояние. Стратегия C : 10 агентов, D : 15 агентов.

Рис. 10.

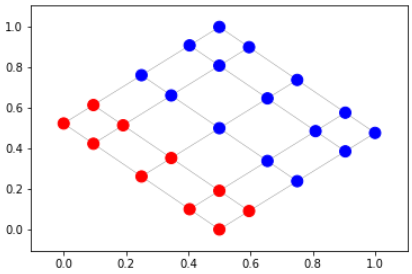
скопления той или иной стратегии и на изображениях выше мы видим разделение агентов на две группы по расположению.

Пример 3.2 Рассмотрим данную популяцию на сети *решетка*.



а)

Начальное состояние. Стратегия C : 13 агентов, D : 12 агентов.



б)

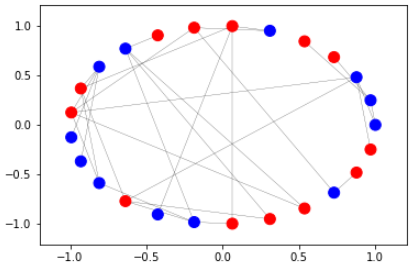
Стационарное состояние. Стратегия C : 10 агентов, D : 15 агентов.

Рис. 11.

В данном случае расположения соседей можем также наблюдать (см. рис. 11) что в стационарном состоянии образовалось четкое разделение агентов на две группы. В начальном состоянии скопления агентов с той или иной стратегией влияли на обособленных агентов с противоположной стратегией, превратив популяцию в два обособленных лагеря. Таким образом, начальное расположение агентов повлияло на численность и расположение агентов в стационарном состоянии.

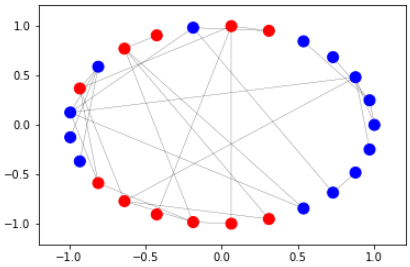
Пример 3.3 Популяция «Охота на оленя» на случайном графе.

Для случайного расположения (см. рис. 12) можно сказать, что так как большое скопление красных агентов влияет на одиночных синих и, на-



а)

Начальное состояние. Стратегия C : 12 агентов, D : 13 агентов.



б)

Стационарное состояние. Стратегия C : 11 агентов, D : 14 агентов.

Рис. 12.

оборот, в течение эволюционного процесса численное соотношение агентов зачастую колеблется (не наблюдается строгого роста численности агентов с одной стратегией и убывания с другой). Таким образом, связи между агентами оказывают очень значимое влияние. Во время проведения экспериментов именно для этого характера взаимодействия иногда было невозможно достижение популяцией стационарного состояния, из-за зацикливания изменений.

Глава 3. Задача налогообложения

Также в рамках экспериментов была модифицирована модель одной практической задачи. Идея подобного процесса описана в работах [1, 8].

3.1 Постановка задачи

Пусть имеется популяция из n агентов-налогоплательщиков, каждый имеет доход i_k, где $k = \overline{1,n}$. По окончании налогового периода, каждый налогоплательщик декларирует доход r_k, $r_k \leq i_k$ и выплачивает по налог, равный ψr_k в соответствии с текущей налоговой ставкой ψ. У налогоплательщика имеется две чистых стратегии $S = \{P, E\}$: платить налоги $r_k = i_k$ (pay) или уклониться от выплаты $r_k = 0$ (evade).
Из предположения, что агент может декларировать часть своих доходов $r_k < i_k$, что аналогично вероятностному распределению на множестве чистых стратегий, введем *смешанное расширение игры* [6]. Здесь смешанная стратегия — долевое соотношение декларированного дохода к не декларированному.

$$\xi_k = \left[\frac{r_k}{i_k}, 1 - \frac{r_k}{i_k}\right]$$

В начале игры каждый из агентов, исходя из своих предпочтений, выбирает ту или иную стратегию. Далее по популяции проходит «волна проверок». Факт проверки — случайное, независимое внешнее воздействие на популяцию с заданной вероятностью. В результате проверки, неплательщики выплачивают сам налог и штраф: $(\psi + \pi)(i_k - r_k)$ в соответствии со штрафной ставкой π. В модели, как и в [1] штраф пропорционален сокрытому доходу. Таким образом, для каждого агента возможно 4 ситуации в чистых стратегиях:

- Агент заплатил, и его проверили (check): ситуация (PC)
- Агент не заплатил, его проверили и, как следствие, оштрафовали: ситуация (EC)
- Агент заплатил, и его не проверили (do not check): ситуация (PD)
- Агент не заплатил, и его не проверили: ситуация (ED)

После проверок начинается взаимодействие между налогоплательщиками, представляющее собой *имитационную динамику* [13].

Общий вид матрицы симметричной игры двух агентов:

$$\begin{array}{c} \\ PC \\ EC \\ PD \\ ED \end{array} \begin{array}{c} \begin{array}{cccc} PC & EC & PD & ED \end{array} \\ \begin{pmatrix} \text{-}\psi i_k, -\psi i_k & \text{-}\psi i_k, -(\psi+\pi) i_k & \text{-}\psi i_k, -\psi i_k & \text{-}\psi i_k, 0 \\ \text{-}(\psi+\pi) i_k, -\psi i_k & \text{-}(\psi+\pi) i_k, -(\psi+\pi) i_k & \text{-}(\psi+\pi) i_k, -\psi i_k & \text{-}(\psi+\pi) i_k, 0 \\ \text{-}\psi i_k, -\psi i_k & \text{-}\psi i_k, -(\psi+\pi) i_k & \text{-}\psi i_k, -\psi i_k & \text{-}\psi i_k, 0 \\ 0, \text{-}\psi i_k & 0, \text{-}(\psi+\pi) i_k & 0, \text{-}\psi i_k & 0, 0 \end{pmatrix} \end{array}$$

Выигрыш агента равен его затратам с учетом наличия/отсутствия проверки. Таким образом, данная величина зависит как от лично выбранной игроками стратегии, так и от внешнего воздействия (была ли проверка).

3.1 Моделирование процесса

Построим матрицу взаимодействий. Примем $i_k = 34600$, для любого $k = \overline{1, n}$ — средний заработок агентов [1], налоговая ставка $\psi = 0.13$ и штрафная ставка $\pi = 0.05$. Средняя вероятность проверки равна 20%.

Во всех примерах будут представлены графы, характеризующие план возможных взаимодействий. Цвет узлов характеризует стратегию соответствующего агента: красный — агент выплачивает весь налог, синий — агент не платит ничего, переходные цвета характеризуют смешанное поведение. В первой колонке будут представлены изображения состояния всей популяции, во второй — результат проверки: цветом стратегии подсвечены только проверенные агенты, непроверенные же окрашены зеленым. Также рассчитаем выигрыш системы для каждого этапа эволюционного процесса. Выигрышем системы является сумма выигрышей всех агентов, вычисляемая для каждого налогового периода t:

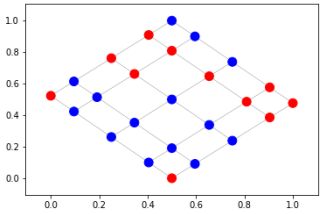
$$W(t) = \sum_{i=1}^{n} x_i(t) M x_i(t), \qquad t = 0, 1, \ldots, T,$$

где T — момент достижения популяцией стационарного положения. Несмотря на то, что внешнее воздействие затрагивает случайных агентов, следовательно мы не можем с точностью судить о стационарном состоянии (если только не приходим к одной стратегии у всех агентов), в качестве критерия остановки также выберем условие неизменности состояния популяции на

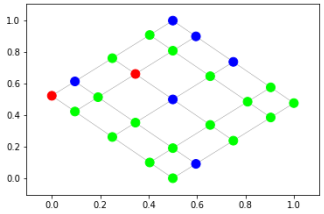
следующем шаге. Процедура для моделирования данного эволюционного процесса представлена в Приложении (см. Приложение 3).

Пример 1. Рассмотрим поведение модели в случае 25-ти агентов на сети типа решетка с начальным состоянием: 11 агентов выплачивают весь налог, 14 — не платят ничего. Всего понадобилось 26 периодов пересмотра стратегий для достижения системой стационарного положения. На рисунках представлены только некоторые этапы.

Рис. 13: Начальное состояние.

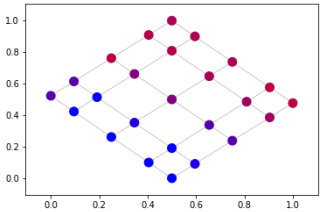


а) 11 агентов платят, 14 не платят. Доход системы без проверки $W(0) = -49764$

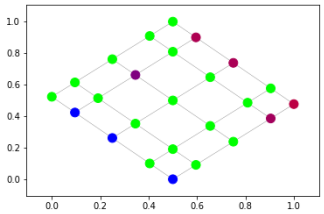


б) Проверка. Проверено 8 агентов. Доход системы при условии проверки $\hat{W}(0) = -87348$

Рис. 14: Четвертый налоговый период.

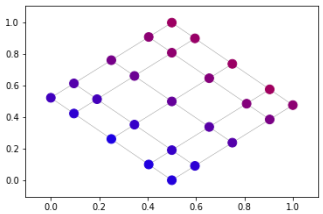


а) Доход системы без проверки $W(4) = -45784.6$

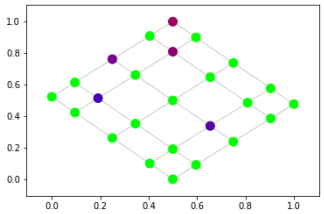


б) Проверка. Проверено 8 агентов. Доход системы при условии проверки $\hat{W}(4) = -75712.6$

Рис. 15: Двенадцатый налоговый период.

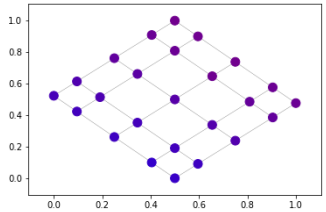


а) Доход системы без проверки $W(12) = -43378.4$

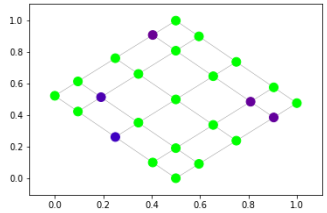


б) Проверка. Проверено 5 агентов. Доход системы при условии проверки $\hat{W}(12) = -60581.7$

Рис. 16: Двадцать шестой налоговый период.



а) Доход системы без проверки $W(26) = -38615$



б) Проверка. Проверено 5 агентов. Доход системы при условии проверки $\hat{W}(26) = -59018.4$

В течение изменения популяции относительно выбираемой агентами стратегии можем наблюдать (см. рис. 13-16), что расходы системы, т. е. общая сумма выплачиваемого налога агентами, уменьшается. В этом и есть смысл «эволюции» или обучения популяции: с каждым шагом агенты стараются уменьшить свои расходы.

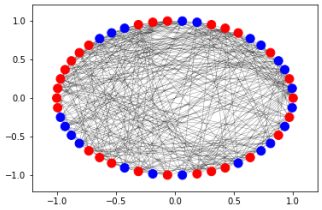
Таким образом, в данной ситуации с помощью проверок (в представленном количестве) налоговая может только взимать недоимки с попавшихся неплательщиков и слабо влияет на отношение самих агентов к выплате налогов.

Пример 2. Рассмотрим поведение системы 50-ти агентов со случайным расположением (схема взаимодействий задана случайным графом).

В данном примере (см. рис. 17-18) стационарное состояние достигнуто за 30 налоговых периодов. В стационарном состоянии популяция более согласована в выборе стратегии, в отличие от предыдущего примера, состояние популяции описывает вектор смешанной стратегии $\hat{\delta} = [0.26, 0.74]$, т. е. агенты считают наиболее выгодным выплачивать только четверть требуемого налога.

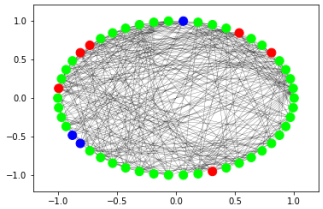
Полученные результаты иллюстрируют влияние числа связей между агентами системы на ее адаптацию: если оно достаточно велико, агенты более солидарны в выборе оптимального решения.

Рис. 17: Начальное состояние.



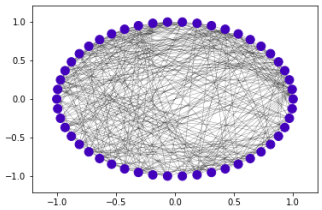
а) Доход системы без проверки

$W(0) = -126672$



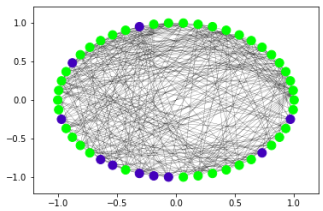
б) Проверка. Проверено 5 агентов. Доход системы при условии проверки $\hat{W}(0) = -145464$

Рис. 18: Стационарное состояние.



а) Доход системы без проверки

$W(30) = -59455.2$



б) Проверка. Проверено 5 агентов. Доход системы при условии проверки $\hat{W}(30) = -105641.4$

Заключение

В данной работе с помощью компьютерного моделирования эволюционного процесса был проведен ряд экспериментов для различных начальных данных, таких как: матрица выигрышей, начальное состояние популяции, граф возможных взаимодействий, количество агентов в исследуемой системе. Было выявлено заметное влияние вида графа взаимодействия агентов на характер дальнейших изменений в популяции (стационарное состояние популяции и необходимое число шагов для его достижения). В процессе работы была поставлена задача, имеющая практическое значение, для решения которой возможно использование описанного аппарата, также реализована процедура (на основе ранее созданных), для моделирования именно этого процесса, учитывая его специфику.

Данная тема имеет большой потенциал как в исследовании и модификации самого механизма эволюционных игр, так и в возможностях его практического применения. Благодаря достаточно обширному спектру регулируемых параметров модели возможен более точный анализ поведения любой крупной, замкнутой системы. На данный момент активно развивается идея построения алгоритмов оптимального управления, а именно задача достижения популяцией заданного конечного состояния[10].

Список литературы

[1] Буре В. М., Кумачева С. Ш. Теоретико-игровая модель налоговых проверок с использованием статистической информации о налогоплательщиках // Вестн. Санкт-Петерб. ун-та. 2010. Серия 10. 1–2. С. 16–24.

[2] Колесин И. Д., Губар Е. А., Житкова Е. М. Стратегии управления в медико-социальных системах: учеб. пособие. СПб.: Изд-во С.-Петерб. ун-та, 2014. 128 с.

[3] Колокольцов В. Н., Малафеев О. А. Теория игр для всех (моделирование процессов конкуренции и кооперации) СПб. 2010. 350 с.

[4] Курносых З. А., Губар Е. А. Моделирование эволюционной игры с учетом сетевой структуры. // Процессы управления и устойчивость. 2017. В печати.

[5] Лутц М. Программирование на Python, 4-е издание. Пер. с англ. СПб.:Символ-Плюс, 2011. – 992 с.,

[6] Петросян Л. А., Зенкевич Н. А., Шевкопляс Е. В. Теория игр: учебник. СПб.: БХВ-Петербург, 2012. 432 с.

[7] Руссо Ж. Ж. Рассуждение о происхождении и основаниях неравенства между людьми // Трактаты / Пер. с франц. А. Хаютина. М.: Наука, 1969. — С. 75.

[8] Kumacheva S., Gubar S., Zhitkova E., Kurnosykh Z., Skovorodina T. Evolutionary behaviour of taxpayers in the model of information dissemination // Constructive Nonsmooth Analysis and Related Topics. 2017.

[9] Maynard S. J., Price G. R. The logic of animal conflict. Nature Vol. 246, London, 1973. P.15-18.

[10] Riehl J. R., Cao M. Control of Stochastic Evolutionary Games on Networks. // 5th IFAC Workshop on Distributed Estimation and Control in Networked Systems, Philadelphia, PA, United States, 10-11 September. 2015.

[11] Sandholm W. H. Population Games and Evolutionary Dynamics, 2011. 442 p.

[12] Tembine H. Population games with networking applications. Université d'Avignon. 2009.

[13] Weibull J. W. Evolutionary Game Theory. MIT Press, Cambridge, 1995. 265 p.

Приложение

Приложение 1. Процедура в программном пакете Maple для решения и исследования уравнений репликативной динамики на устойчивость

```
> A := Matrix(2, 2, {(1, 1) = 3, (1, 2) = 1, (2, 1) = 5, (2, 2) = 0});
> X := Vector(2, {(1) = x1(t), (2) = x2(t)});
> P := A.X;
> with(MTM);
> Pvv := transpose(X).P;
> unwith(MTM);
  sys := {x1(0) = 0.1, x2(0) = 0.9,
diff(X(1), t) = simplify(X(1)*(P(1)-Pvv)),
  diff(X(2), t) = simplify(X(2)*(P(2)-Pvv))};
> F := dsolve(sys, [x1(t), x2(t)], numeric);
> plots[odeplot](F, [t, x1(t)], 0 .. 20);
> plots[odeplot](F, [t, x2(t)], 0 .. 20);
> plots[odeplot](F, [x1(t), x2(t)], 0 .. 100);
> with(DEtools);
> s1=0.5; s2 := 1-s1;
> DEplot({diff(X(1), t) = simplify(X(1)*(P(1)-Pvv)),
 diff(X(2), t) = simplify(X(2)*(P(2)-Pvv))},[x1(t), x2(t)],
 t = 0 .. 1, x1 = 0 .. 1, x2 = 0 .. 1, [[x1(0) = s1, x2(0) = s2]],
 scene = [x1(t), x2(t)], linecolor = COLOR(HUE, 0.5))
> restart;
```

Приложение 2. Набор функций на языке программирования Python для моделирования эволюционного процесса с учетом структуры популяции

```
## матрицы выигрышей
Dilemm = np.array([[4, -1],
                   [5, 0]])
YastrGol = np.array([[-1, 4],
                     [0, 2]])
SnowDrift = np.array([[3, 1],
                      [5, 0]])
StagHunt = np.array([[2, 0],
                     [1, 1]])
n=25 ## количество агентов
eps=0.01
def Norm(Array):
```

```
    Norma=0
    Sum=0
    for i in range(len(Array)):
        Sum+=(Array[i])**2
    Norma=mat.sqrt(Sum)
    return Norma
###
def formstrat(n): ## случайное задание стратегий агентов
    strat=dict.fromkeys(i for i in range(n))
    k=0;
    for i in range(n):
        k=random.randrange(0, 2, 1)
        if k==0:
            strat[i]=np.array([0,1])
        else:
            strat[i]=np.array([1,0])
    return strat
     ###
def sravnchist(graf, A, strat, n):## реализация протокола
    a=np.zeros(2)                  ## принятия решений
    winigr=np.zeros(n) ## средние выигррыши агентов
    winopon=dict.fromkeys(i for i in range(n)) ## средние выигрыши соседей
    ro=dict.fromkeys(i for i in range(n)) ## протокол принятия решений
    stratnew=dict.fromkeys(i for i in range(n)) ## стратегии в конце сравнения
    oponstr=dict.fromkeys(i for i in range(n)) ## смешанная стратегия популяци
    for igr in range(n):
        winopon[igr]=a
        ro[igr]=a
        oponstr[igr]=np.zeros(2)
        znam=0
        for agst in range(n):
            if graf[igr][agst]==1:
                oponstr[igr]+=strat[agst]
                znam+=1
        oponstr[igr]=oponstr[igr]/znam

        temp1=np.zeros(2)
        for j in range(len(A)):
            for l in range(len(A)):
                temp1[j]+=strat[igr][l]*A[l][j]
```

```
        for s in range(2):
            winigr[igr]+=temp1[s]*oponstr[igr][s]
    for igr in range(n):
        znam=[0.001,0.001]
        for agst in range(n):
            if graf[igr][agst]==1:
                if strat[agst][0]>strat[agst][1]:
                    winopon[igr][0]+=winigr[agst]
                    znam[0]+=1
                else:
                    winopon[igr][1]+=winigr[agst]
                    znam[1]+=1
        winopon[igr][0]=winopon[igr][0]/znam[0]
        winopon[igr][1]=winopon[igr][1]/znam[1]
        for k in range(len(A)):
            Z=oponstr[igr][k]*(winopon[igr][k]-winigr[igr])
            ro[igr][k]=min(1,max(0,Z))

        if ro[igr][0]>ro[igr][1]:
            stratnew[igr]=np.array([1, 0])
        if ro[igr][0]<ro[igr][1]:
            stratnew[igr]=np.array([0, 1])
        else:
            stratnew[igr]=strat[igr]
    return stratnew
                ###
def progon_epoh(graf, A, G, strat, n, eps): ## эволюционный
    k=0                                     ## процесс
    print(sum(strat.values()))
    while k>-1:
        nx.draw_networkx(G,pos=nx.shell_layout(G), node_size=100,
node_color = [(strat[i][0],0,strat[i][1]) for i in G.nodes()],
width=0.2,edge_color='k',vmin=0.0,vmax=2.0, with_labels=False)
        plt.show()
        k+=1
        razn=0
        strat1=sravnchist(graf, A, strat, n)
        print(strat1)
        for i in range(n):
            razn+=Norm(strat[i]-strat1[i]) ##пока суммарное изменение
```

```
        if razn < eps:                      ##стратегий не станет ниже eps
            print("Stacionarnoe sostoyanie dostignuto", razn, k)
            break
        strat=strat1
        print(sum(strat1.values()))
        plt.show()
    return(strat1)
```

Приложение 3. Набор функций на языке программирования Python для моделирования эволюционного процесса популяции налогоплательщиков с генерацией случайных проверок

```
i_k=34800  #средний доход агентов
ksi=0.13   #налоговая ставка
pi=0.05    #штрафная ставка
#матрица выигрышей
nalogitrue=np.array([[-ksi*i_k, -ksi*i_k, -ksi*i_k, -ksi*i_k],
  [-(ksi+pi)*i_k, -(ksi+pi)*i_k,-(ksi+pi)*i_k, -(ksi+pi)*i_k],
  [-ksi*i_k, -ksi*i_k, -ksi*i_k, -ksi*i_k],
  [0, 0, 0, 0]])
n=25
eps=0.1
   ###
def Norm(Array): ## норма вектора
    Norma=0      ## для условия остановки
    Sum=0
    for i in range(len(Array)):
        Sum+=(Array[i])**2
    Norma=mat.sqrt(Sum)
    return Norma
   ###
def formstrat(n):          ## случайное задание стратегий
    strat=dict.fromkeys(i for i in range(n))
    k=0;
    for i in range(n):
        k=random.randrange(0, 2, 1)
        if k==0:
            strat[i]=np.array([1,0])
        else:
            strat[i]=np.array([0,1])
    return strat
```

```
###
def proverka(n, strat):    ## реализация проверки
    widestrat=dict.fromkeys(i for i in range(n))
    k=np.zeros(n);
    for i in range(n):
        widestrat[i]=np.zeros(len(A))
        if random.normalvariate(2, 9)>-5:
            k[i]=1
        if k[i]==0:
            widestrat[i][0]=strat[i][0]
            widestrat[i][1]=strat[i][1]
        else:
            widestrat[i][2]=strat[i][0]
            widestrat[i][3]=strat[i][1]
        #print(widestrat[i])
    return widestrat, k
###
def sravnchist(graf, A, strat, n): ## сравнение выигрышей
    winigr=np.zeros(n)             ## в чистых стратегиях
    winopon=np.zeros(n)
    stratnew=dict.fromkeys(i for i in range(n))
    oponstr=dict.fromkeys(i for i in range(n))
    for igr in range(n):
        oponstr[igr]=np.zeros(len(A))
        znam=0
        for agst in range(n):
            if graf[igr][agst]==1:
                oponstr[igr]+=strat[agst]
                znam+=1
        oponstr[igr]=oponstr[igr]/znam

        temp1=np.zeros(len(A))
        for j in range(len(A)):
            for l in range(len(A)):
                temp1[j]+=strat[igr][l]*A[l][j]
        for s in range(len(A)):
            winigr[igr]+=temp1[s]*oponstr[igr][s]
    for igr in range(n):
        znam=0
        for agst in range(n):
```

```
            if graf[igr][agst]==1:
                winopon[igr]+=winigr[agst]
                znam+=1
        winopon[igr]=winopon[igr]/znam
        if winigr[igr]>=winopon[igr]:
            stratnew[igr]=strat[igr]
        else:
            stratnew[igr]=oponstr[igr]
        M=max(stratnew[igr])
        if stratnew[igr][0]==stratnew[igr][1]==M or
stratnew[igr][0]==stratnew[igr][3]==M or
stratnew[igr][1]==stratnew[igr][2]==M:
            if strat[igr][0]>strat[igr][3] or strat[igr][2]>strat[igr][1]:
                stratnew[igr]=np.array([1,0])
            else:
                stratnew[igr]=np.array([0,1])
        elif M==stratnew[igr][0] or M==stratnew[igr][2]:
            stratnew[igr]=np.array([1, 0])
        elif M==stratnew[igr][1] or M==stratnew[igr][3]:
            stratnew[igr]=np.array([0, 1])
    print(winigr,winopon)
    return stratnew

def sravn(graf, A, strat, n): ## реализация протокола
    winigr=np.zeros(n)        ## пересмотра решений
    winopon=np.zeros(n)
    stratnew=dict.fromkeys(i for i in range(n))
    oponstr=dict.fromkeys(i for i in range(n))
    for igr in range(n):
        stratnew[igr]=np.zeros(2)
        oponstr[igr]=np.zeros(len(A))
        znam=0
        for agst in range(n):
            if graf[igr][agst]==1:
                oponstr[igr]+=strat[agst]
                znam+=1
        oponstr[igr]=oponstr[igr]/znam

        temp1=np.zeros(len(A))
        for j in range(len(A)):
```

```
            for l in range(len(A)):
                temp1[j]+=strat[igr][l]*A[l][j]
        for s in range(len(A)):
            winigr[igr]+=temp1[s]*oponstr[igr][s]

    for igr in range(n):
        znam=0
        for agst in range(n):
            if graf[igr][agst]==1:
                winopon[igr]+=winigr[agst]
                znam+=1
        winopon[igr]=winopon[igr]/znam
        if winigr[igr]>=winopon[igr]:
            stratnew[igr][0]=strat[igr][0]+strat[igr][2]
            stratnew[igr][1]=strat[igr][1]+strat[igr][3]
        else:
            stratnew[igr][0]=oponstr[igr][0]+oponstr[igr][2]
            stratnew[igr][1]=oponstr[igr][1]+oponstr[igr][3]
    print(winigr,winopon)
    return stratnew
       ###
def progon_epoh(graf, A, G, strat, n, eps): ## пошаговая эволюция
    k=0                                     ## популяции налогоплательщиков
    print(sum(strat.values()))
    while k>-1:
        nx.draw_networkx(G,pos=nx.spectral_layout(G), node_size=100,
node_color = [(strat[i][0],0,strat[i][1]) for i in G.nodes()],
width=0.2,edge_color='k',vmin=0.0,vmax=2.0, with_labels=False)
        plt.show()
        nach_nabor_str=strat.copy()
        [strat, pr]=proverka(n, strat)
        Dohod=sum(strat[i] for i in range(len(strat)))*A[:,1]
        print(sum(Dohod))
        nx.draw_networkx(G,pos=nx.spectral_layout(G), node_size=100,
node_color = [(strat[i][0],pr[i],strat[i][1]) for i in G.nodes()],
width=0.2,edge_color='k',vmin=0.0,vmax=2.0, with_labels=False)
        plt.show()
        k+=1
        razn=0
        strat1=sravn(graf, A, strat, n)
```

```
        print(strat1)
        for i in range(n):
            razn+=Norm(nach_nabor_str[i]-strat1[i]) ## критерий остановки
        if razn < eps:
            print("Stacionarnoe sostoyanie dostignuto", razn, k)
            break
        strat=strat1
        print(sum(strat1.values()))
        print(sum(pr))
        plt.show()
    return(strat1)
```