Министерство образования и науки РФ

ФГБОУ ВО "Государственный академический университет гуманитарных наук"

Факультет: Экономический

Магистерская программа: Междисциплинарный анализ социально-экономических процессов

Кафедра: Математической экономики

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На тему: разработка модели формирования графа кооперации фирм-олигополистов в среде с неопределенным спросом

Студент: Белявский Е.В.

_____

Научный руководитель:
д.ф.-м.н., профессор, Леонидов А.В.

_____

Рецензент:
к.ф.- м.н., доцент, Пильник Н.П.

_____

Содержание

Введение **3**

1 Глава I **5**

1.1 Краткое вступление 5

1.2 Описание модели 5

2 Глава II **15**

2.1 Анализ модели 15

2.2 Пример 1 15

2.3 Пример 2 25

Заключение **29**

Список литературы **30**

Приложение **31**

Введение

Цель данной работы заключается в исследовании вопроса динамического формирования сети взаимодействий между фирмами, где каждая фирма имеет цель максимизировать свою собственную функцию полезности за счет принятия или отвержения котракта сотрудничества с другими фирмами, посредством построения математической модели и её анализа. Что именно мы подразумеваем под понятием сотрудничесва между фирмами будет показано в дальнейшем в данной работе в рамках описания построенной модели.

Отметим, что в работе рассматривается функция полезности, состоящая из двух частей: получаемой доходности и неприязнь к риску. Именно желание фирм оптимизировать соотношение между риском и доходностью заставляет их формировать альянсы.

Следует уточнить, что в данной работе проводится иследование только стилизованной модели. Рассмотрение "реальных" примеров по данной модели предполагается как возможное направление в будущих исследованиях.

Также, из названия данной работы "Разработка модели формирования графа кооперацийфирм-олигополистов в среде с неопределенным спросом" мы можем увидеть, что речь идет о фирмах-олигополистах, чья прибыль зависит от различных её источников. Однако, в данной работе мы не будем анализировать конкретные виды источников прибыли, а будем считать саму величину прибыли компании случайной величиной. Более подробно об этом будет написано в "Описание модели".

Можно заметить, что в столь произвольной формулировке цели нашей работы, у нас появляется множество степеней свободы в разработке модели, некоторые из которых будут сформулированны в форме задач, составляющие интерес для дальнейшего исследования.

Итак, для исследования нашей работы предварительно рассматриваются следующие вопросы и ставятся следующие задачи:

- ознакомление с некоторыми уже существующими моделями, акцент в которых сделан на формирования связей взаимодействия между фирмами;

- изучение вопроса возможных правил взаимодействий между фирмами (рассмотрение функций полезности для фирм) и разработка новых правил для нашей модели;

- описание самой процедуры принятия решений о сотрудничестве между фирмами в рамках модели;

- разработка и написание модели;

- разработка и построение алгоритмов для решения задач в рамках модели;

- анализ результатов, полученных в ходе численного решения задач.

Данная работа состоит из двух глав. В первой главе произведен краткий обзор моделей со схожей тематикой, с целью выделить основное направление развития нашей модели, а также подчеркнуть возможные её вариации. Выполнено описание модели и приведены некоторые суждения "философского" характера по поводу того, как фирмы могут "договариваться" между собой. Вторая глава включает в себя анализ построенной нами модели, посредством рассмотрения примеров, а также визуализацую результатов их симуляции. Также, в работу включено приложение, содержащее код программы, написанной в среде программирования RStudio.

1 Глава I

1.1 Краткое вступление

В качестве вступления к нашей работе приведем пример работ, в которых исследовались модели со схожей тематикой.
Так, например в статье (Brown, Chiang, 2000) речь идет об изучении взаимоосвязи между несистематическим риском на рынке продуктов и образовании коалиций среди фирм-олигополистов. Их модель основана на некооперативной игре, которая была предложена Селтеном (1981) и Рубинштейном (1982), где игроки делают последовательные предложения для формирования коалиции. Их решение присоединения к коалиции опирается на игру в расширенной форме: предложения и встречные предложения.
А в книге (Jackson, 2008) в разделе, посвященном сетям сотрудничества среди фирм, речь идет о рынках труда и передачи информации и об обменных сетях, где и в одном и в другом случаях сделки происходят между связанными агентами. Они строят свою модель с целью описать эти явления.
Взяв за основу некоторые ключевые идеи из расмотренных выше моделей, мы попытаемся создать свою модель, описывающую динамику взаимодействия между фирмами.

1.2 Описание модели

Итак, приступим к непосредственному описанию модели и попутно будем вводить некоторые обозначения и определения, которые нам понадобятся. Пусть $N = \{1, ...,n\}$ — множество вершин (фирм). A — симметричная матрица[1] смежности данных вершин размерности $n \times n$, состоящая из $\{0, 1\}$. Т.е. если $A_{ij} = 1$, то i-ая вершина (фирма) взаимодействует с j-ой, иначе (при $A_{ij} = 0$) — не взаимодейсвует. Отметим, что $A_{ii} = 0$. За A_i обозначим вектор смежности i-ой вершины.

[1]Случай, когда матрица смежности несимметричная, также представляет большой интерес для отдельного исследования, но и таит в себе определенные трудности с её интерпретацией. Так, например, это может означать, что фирма $\mathcal{A}$ сотрудничает с фирмой $\mathcal{B}$, с целью снизить свои риски, а фирма $\mathcal{B}$ берет с фирмы $\mathcal{A}$ определенную плату, не понижая при этом свои риски.

Каждая фирма характеризуется своей функцией полезности $u_k = u_k(\bar{E}_k, \sigma_k^E)$, где $\bar{E}_k$ - средний доход, σ_k^E - стандартное отклонение этого дохода. Мы будем рассматривать функции полезности вида:

$$u_k(\bar{E}_k, \sigma_k^E) = \bar{E}_k - \frac{\sigma_k^E}{\mathcal{R}_k},$$

где $\mathcal{R}_k$ - некоторый параметр, характеризующий степень неприязни к риску у вершины k.

Наша задача заключается в том, чтобы построить динамическую модель взаимодействия этих фирм, максимизирующих собственную функцию полезности. Для того, чтобы описать модель, мы сначала кратко и неформально опишем основные этапы работы модели, а затем более подробно опишем алгоритм каждого этапа.

Итак, т.к. модель динамическая, значит её выполнение можно разбить на временные шаги. Теперь каждый временной шаг модели разобьём на следующие этапы:

1. вершины графа получают экзогенную прибыль;
2. происходит этап перераспределния полученных прибылей между вершинами, в зависимости от вида матрицы смежности A на этом шаге, по заданному ниже правилу в пункте *Этап 2*;
3. затем, каждая вершина решает задачу максимизации собственной функции полезности и в результате получает набор "желаемых соседей";
4. после того, как у каждой вершины сформировался набор "желаемых соседей", строится новый граф, где ребро между вершинами создается только в том случае, когда они входят в набор "желаемых соседей" друг у друга;
5. происходит переход на следующий временной шаг к первому этапу;
6. осуществляется остановка выполнения временных шагов по некоторым критериям, описанным в пункте *Этап 6*.

Теперь перейдем к более подробному описанию каждого этапа.

Этап 1

Каждая вершина k в периоде времени l "извне" получает некоторую случайную прибыль π_k. Предположим, что эта прибыль распределена нормально $\pi_k \sim \mathcal{N}(a_k,\sigma_k)$. (На самом деле, можно использовать любое произвольное, не обязательно нормальное, распределение, где a_k — математическое ожидание, а σ_k — стандартное отклонение данной случайной величины, и это замечание заслуживает внимания для проведения дальнейшего исследования.) Если вершины несвязаны, то π_k - это и есть то значение прибыли, которое фирма получит в данном периоде, если же вершины связаны, то каждая фирма обязана поделиться частью своей прибыли со своими соседями (более формально алгоритм перераспределения прибылей будет описан в *Этап 2*).

Этап 2

На этом этапе определяется правило перераспределение прибылей между вершинами графа. Это один из самых ключевых и интересных моментов в построении нашей модели. Можно придумать множество различных правил взаимодействий между вершинами, основываясь на простых соображениях или придумывая достаточно ветьеватые схемы, от которых результат будет зависеть существенным образом. Здесь мы приведем несколько рассуждений и возможных путей развития данного этапа модели. Но стоит отметить, что в данной работе будет проведено иследование всего одного правила взаимодействия вершин, которое представляется наиболее естественным образом и поэтому является удобной отправной точкой для дальнейших исследований.

Начнем с того, что на некоторый период времени l вершины характеризуются своей матрицей смежности $A(l)$ и полученной в *Этап 1* прибылью π_k. Главной предпоссылкой к далее полученому правилу является то, что вершина должна делиться со своими соседями частью этой прибыли, а также должна получать некоторую часть прибыли от этих соседей. Пусть вершина k оставляет себе долю полученной прибыли равную $\alpha_k \in (0,1)$, а оставшуюся часть распределяет равномерно между своими соседями. Тогда

на выходе она получит прибыль π_k^{new} равную

$$\pi_k{}^{new}(l) = \alpha_k \pi_k + \sum_j A_{jk}(l) \frac{1-\alpha_j}{d_j(l)} \pi_j,$$

здесь $d_j(l) = \sum_i A_{ji}(l)$ — степень j-ой вершины (или иначе, количество её соседей). То же самое можно переписать в матричной форме:

$$\pi^{new}(l) = [\alpha + A(l)(E-\alpha)D^{-1}(l)]\pi,$$

здесь π — вектор прибылей, α — диагональная матрица, где $\alpha_{ii} = \alpha_i$, D — диагональная матрица, где $D_{ii} = d_i$.

Казалось бы, что после проделанной процедуры деления, вершины получат свою новую прибыль π_k^{new} и на этом "успокоятся". Как вариант, это можно оставить так и переходить к следующим этапам выполнения модели. Однако, как не трудно заметить, в данном случае полученная прибыль будет зависить лишь от соседей первого порядка. Нам бы хотелось получить прибыль вершины, в зависимости от "полного состояния дел" (от матрицы смежности в целом), т.е. так, чтобы перераспределенная прибыль зависела от соседей второго и более высокого порядка[2]. Этого мы можем добиться следующим образом: пусть каждая вершина проводит перераспределения своих прибылей по выше описанному закону, а затем снова повторяет то же самое, но в качестве первоначальных π_k использует полученные π_k^{new}, затем снова повторяет ту же процедуру и т.д. Более формально это будет выглядеть следующим образом:

$$\pi^{t+1}(l) = [\alpha + A(l)(E-\alpha)D^{-1}(l)]\pi^t(l),$$

где π^t — t-ое повторение процедуры, $\pi^0 := \pi$. Что является равносильным

$$\pi^t(l) = [\alpha + A(l)(E-\alpha)D^{-1}(l)]^t\pi.$$

[2]Соседями j-го порядка вершины k называются те вершины, которые имеют путь к вершине k длиной j.

Теперь остется вопрос, до каких пор нам продолжать эту процедуру? (Какое π^t нас устроит?) Одним из вариантов является условие остановки: $\pi^{t+1} = \pi^t$ — или иначе, получение равновесного вектора π^t. Этого мы можем добиться, устремив $t \to \infty$, что приведет нас к вектору, который сходный по смоему смуслу с понятием Eigenvector Centrality (Newman, 2010). Т.е. в результате данного этапа перераспределения прибылей между вершинами будет получен равновесный вектор прибылей $\pi^{eq} = \lim_{t\to\infty} \pi^t$.
Также, ниже приведем несколько замечаний относительно описанной выше процедуры.

Замечание 1 *Предварительно, перед начлом действий, связанных с перераспределением прибылей, разобьем наш граф на компоненты связности. Затем будем применять описанную выше процедуру для каждой компоненты связности отдельно в тех же матричных обозначениях, что и в исходной задаче (здесь мы пользуемся условием, что несвязные компоненты - независимы друг от друга).*

Замечание 2 *Показанный выше способ получения вектора перераспределенных прибылей с учетом соседей "всех" порядков является далеко не единственным. Так, можно, например, модифицировать наше правило следующим образом:*

$$\bar{\pi}^{new}(l) = A^{eq}(l)\pi,$$

где $A^{eq}(l) = \lim_{t\to\infty} \frac{1}{\lambda_1^t} A^{t+1}(l)$. *Здесь* λ_1 — *максимальное собственное значение матрицы* $A(l)$. *Тем самым, мы получили новые перераспределенные прибыли и всего за одну такую итерацию учли связи с соседями "всех порядков". Однако, введение такого правила является намного менее интуитивным, и в дальнейшей работе оно рассмотренно не будет (хотя все же представляет некоторый интерес для изучения).*

Также в описании этого этапа мы произведем несколько примечаний, касающихся матрицы $W(l) = \alpha + A(l)(E - \alpha)D^{-1}(l)$ размерности $r \times r$ (в данном случае r — это размерность рассматриваемой компоненты связности), которые нам пригодятся в дальнейшем.

Примечание 1 *Пусть* $\beta(l) = \lim_{t\to\infty} W^t(l)$. *Тогда*

$$\beta(l) = \underbrace{(\nu_1, \ldots, \nu_1)}_{r},$$

где ν_1 *— собственный вектор, отвечающий максимальному собственному значению матрицы* $W(l)$, *которое равно единице.*

Примечание 2 *Если в качестве единичной матрицы* α *использовать одну константу* $\tilde{\alpha}$, *т.е.* $\alpha_{ii} = \tilde{\alpha}$, $\forall i \in N$, *то получим матрицу* $W(l) = E\tilde{\alpha} + (1-\tilde{\alpha})A(l)D^{-1}(l)$. *Особенность этой матрицы состоит в том, что её собственный* ν_1 *имеет следующий вид:*

$$\nu_{\mathbf{1}} = \begin{pmatrix} \frac{d_1}{d} \\ \frac{d_2}{d} \\ \vdots \end{pmatrix},$$

где $d = \sum_i d_i$, $\forall\tilde{\alpha} \in (0{,}1)$. *Таким образом, перераспределение прибылей будет происходить лишь по степеням вершин и не будет зависеть от выбора параметра* $\tilde{\alpha}$. *Полученный вектор прибылей соответствует Degree Centrality (Newman, 2010).*

Итак, перейдем к следующему этапу модели.

Этап 3

В итоге, после выполения *Этап 2*, каждая вершина может определить полученную итоговую прибыль в зависимости от вида матрицы $A(l)$. Но каждая вершина решает для себя задачу не максимизации собственной прибыли, а максимизации собственной функции полезности

$$u_k(\bar{E}_k, \sigma_k^E) \to \max_{A_k(l)}.$$

На этапе рассмотрения вариантов функции полезности снова возникает множество открытых вопросов, связанных с её выбором. Однако, на этом мы останавливаться долго не будем, приведем лишь выбранную нами функцию полезности и несколько рассуждений - какие ещё можно рассмотреть

для дальнейшейго исследования.

Итак, будем рассматривать функцию полезности, которую мы вводили в самом начале описания модели. Напомним её вид: $u_k(\bar{E}_k, \sigma_k^E) = \bar{E}_k - \frac{\sigma_k^E}{\mathcal{R}_k}$ и сделаем несколько пояснений.

В качестве среднего дохода, мы будем использовать комбинацию из математических ожиданий наших случайных величин π_k

$$\bar{E}_k = \sum_{j \in N_k} \beta_{jk} a_k,$$

а в качестве стандартного отклонения от среднего дохода возьмем

$$(\sigma_k^E)^2 = \tilde{\sigma}_k^2 = \sum_{j \in N_k} \beta_{jk}^2 \sigma_j^2,$$

где $N_k = \{i \in N |$ i-ая вершина связна с k-ой$\}$, β_{ij} — элемент матрицы $\beta(l)$ из *Примечание 1*. Отметим, что введенная выше формула верна, если мы предполагаем, что ковариационная матрица σ случайных величин π_k является диагональной (т.е. эти случайные величины независимы). Таким образом, оптимизационная задача для каждой вершины будет выглядеть следующим образом:

$$u_k(\bar{E}_k, \sigma_k^E) = \bar{E}_k - \frac{\tilde{\sigma}_k}{\mathcal{R}_k} \to \max_{A_k(l)}.$$

Можно заметить, что в данной функции полезности учитывается взаимодействие между вершинами, однако никаких издержек на создание или разрушение ребра не взимается. Этот факт можно также учесть и, тем самым, изменив вид введенной выше функции полезности, получить задачу, представляющую интерес для дальнейших исследований.

Теперь перейдем непосредственно к описанию процесса выбора вершиной "желаемых соседей", т.е. к решению указанной выше оптимизационной задачи.

Задача состоит в следующем: максимизировать функцию полезности для каждой вершины, в зависимости от матрицы $A(l)$, где вершина вольна менять своих собственных соседей, в предположении, что окружающие её соседи сами менять ничего не будут. (Данное предположение также является

одним из ключевых моментов для нашей модели. Интерес для дальнейших исследований представляет изменение "бездействия" фирм на реализацию ими различного рода стратегий поведения.)

Итак, каждая k-я вершина может изменять свой вектор $A_k(l)$ в матрице $A(l)$ и только его (ну, и т.к. мы предполагаем, что матрица $A(l)$ - симметричная, то вместе с изменением своего вектора, она соответственно меняет и симметричные к нему элементы), тем самым находя для себя оптимальный набор соседей $\overline{A_k}(l)$. Однако в данной процедуре сама матрица $A(l)$ остаётся неизменной.

Как именно происходит пересчет функции полезности вершины k? Для этого снова воспользуемся результатами уже рассмотренного выше этапа модели *stage* 2. Итак, k-я вершина выбирает свой вектор $\overline{A_k}(l)$. В данном случае он не является оптимальным, а про процедуру его выбора чуть ниже. Затем происходит этап перераспределения прибылей с измененной матрицей $A(l)$, в которой вместо вектора $A_k(l)$ используется вектор $\overline{A_k}(l)$, и также изменены все симметричные к этому вектору элементы матрицы. Затем, определяется измененная соответсвующим образом матрица $\beta(l)$ и уже с учетом этого вычисляется значение функции полезности $u_k(A(l),\overline{A_k}(l))$.

Следует отметить, что в данном случае мы имеем дело с задачей целочисленного программирования. Как решать такую задачу? Можно пойти несколькими путями.

Итак, путь первый. Проводим аналитический анализ этой максимизационной задачи, если это возможно (что, в принципе, большая удача) и делаем выводы относительно получаемых вершинами предпочтительных векторов $\overline{A_k}(l)$. Однако, на практике это сделать очень сложно.

Путь второй. Решаем задачу полным перебором. Т.е. перебираем все возможные варианты относительно вида вектора $\overline{A_k}(l)$ и выбираем тот, который является решением нашей оптимизационной задачи. Однако, в этом случае необходимо перебрать 2^n вариантов, что на практике является, мягко говоря, весьма проблематичной задачей, особенно при большом n.

Путь третий. Использовать численные алгоритмы решения данной задачи. Именно этим путем мы и пойдем. А в качестве одного из таких методов выберем метод отжига[3], описание которого мы опустим.

[3]С самим алгоритмом этого метода можно ознакомиться здесь (Лопатин, 2005), а со всеобъемлющей его теорией здесь (Винклер, 2002)

В результате выполения *Этап 3* (решая задачу с помощью метода отжига), мы получим набор предпочтительных векторов $\overline{A_k}(l)$ для каждой вершины k.

Этап 4

Итак, теперь у нас имеется набор векторов $\overline{A_k}(l)$. Построим матрицу

$$T = (\overline{A_1}(l), \ldots, \overline{A_n}(l))$$

и составим с её помощью новую симметричную матрицу:

$$\bar{T} = T * T^T,$$

где T^T— транспонированная матрица T.
По сути, это означает, что если обе вершины готовы друг с другом "сотрудничать" , то они создают между собой связь (ребро). А если хоть одна из них этого не хочет, то связь не создаётся.
Теперь рассмотрим следующий этап.

Этап 5

Мы переходим на новый $(l+1)$-ый временной шаг, где в качестве новой матрицы используем матрицу, полученную на предыдущем этапе, т.е. $A(l+1) := \bar{T}$.

Этап 6

Наша основная задача - рассмотреть динамику получаемых матриц по временным шагам l. Здесь возможно несколько вариантов развития событий.
Вариант первый. Динамика сходится к некоторому равновесному состоянию. Т.е. если $A(l) = A(l-1)$, то симуляция временных шагов прекращается, а полученное состояние, характеризующееся $A(l)$, в связи с вышеперечисленными предположениями, будет являться равновесием Нэша (Osborne, 2004).
Вариант второй. Динамика имеет некоторый повторяющийся цикл длиной $m \leq l$ (динамическое равновесие). Т.е. при $A(l) = A(l-m)$ симуляция временных шагов прекращается.

Вариант третий. За разумное количесво временных шагов l_{max} мы не наблюдаем ни первый, ни второй случаи. Т.е. при $l = l_{max}$ симуляция временных шагов прекращается.

На этом описание модели окончено. Перейдем к рассмотрению неоторых примеров и к частичному анализу модели, основываясь на результатах, полученных в этих примерах.

2 Глава II

2.1 Анализ модели

Как говорилось ранее, исследование модели будет производиться посредством рассмотрения некоторых примеров, решение которых будет производиться с помощью написанной нами программы в среде программирования RStudio, код которой будет приведен ниже в *приложении*.

2.2 Пример 1

Зададим начальные параметры модели.
Пусть $n = 10$, матрица α задаётся одной константой $\tilde{\alpha}$, а как мы отметили в *Примечание 2*, что от выбора этой константы ничего не зависит, поэтому неважно какое значение она принимает, и пусть для определённости $\tilde{\alpha} = \frac{1}{2}$.
Также зададим параметры, определяющие неприязнь к риску, для каждой вершины, равные константе $\mathcal{R}_k = \mathcal{R}$, $\forall k \in N$.
Дальше сгенерируем для каждой вершины параметры распределения её прибыли, т.е. набор (a_k, σ_k). Отметим, что, вообще говоря, наша модель зависит от того, как именно мы будем выбирать эти параметры распределения. Однако, в рассматриваемых примерах, мы не будем исследовать вопрос зависимости поведения модели от этих параметров. Поэтому зададим их следующим (произвольным) образом: $a_k \sim \mathcal{N}(1,\, 0.2)$, $\sigma_k \sim |\mathcal{N}(0,\, 0.2)|$ (т.е. σ_k принимает лишь положительные значения).
Сделаем замечание, что для каждой конкретно взятой модели, где решение происходит с помощью численного метода, следует также тщательно настраивать параметры внутри этого метода, но, как правило, это, достаточно, непростая задача, требующая проведения огромного количества экспиреметов (для некоторых отдельных примеров в модели настройка параметров может производиться втечении нескольких месяцев), но т.к. нам важен общий характер поведения динамики в модели (здесь мы допускаем возможность получения точек, "близких" к глобальному максимуму в задаче оптимизации функции полезности для вершины), то будем делать поправки

на анализ результатов, учитывая все вышеперечисленные замечания.
Теперь осталось лишь определить начальное состояние нашей системы, т.е. задать начальную матрицу смежности. Вообще говоря, выбор начальной точки (начального состояния графа) может оказаться параметром, влияющим на результат. Однако, допускаетя предположение, что в рамках нашей модели результат не зависит от выбора оного (по крайней мере, в рамках рассматриваемых примеров), однако, т.к. этот факт не является доказанным (доказательство этого факта включено в дальнейшее исследование), то во всех симуляциях будем использовать одну и ту же начальную точку.
Итак приступим к симуляции нашей модели, задав начальную матрицу смежности так, как проиллюстрировано на *Рис. 1.*

Симуляция 1. 1

Приведем иллюстрацию матриц смежности на разных шагах выполнения нашей симуляции, при заданном параметре неприязни к риску $\mathcal{R} = 1$.

step 0

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	0	1	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	1	0
1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	1	0

Рис. 1: Матрица смежности, симуляция 1.1, шаг 0

step 1

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	0

Рис. 2: Матрица смежности, симуляция 1.1, шаг 1

step 2

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	1	1	1	1	1	0	0
1	0	1	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	0	0
1	1	1	0	1	1	1	1	0	0
1	1	1	1	0	1	1	1	0	0
1	1	1	1	1	0	1	1	0	0
1	1	1	1	1	1	0	1	0	0
1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Рис. 3: Матрица смежности, симуляция 1.1, шаг 2

step 3

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	0	0	1	1	1	0	0
1	0	1	0	0	1	1	1	0	0
1	1	0	0	0	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	1	0	0
1	1	1	0	0	1	0	1	0	0
1	1	1	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Рис. 4: Матрица смежности, симуляция 1.1, шаг 3

step 4

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	0	1	1	0	0	0	0	1
0	0	1	1	1	0	0	1	0	1
0	1	0	1	1	0	0	1	0	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	1
0	0	0	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1
0	0	0	1	1	0	0	0	0	1
1	1	1	1	1	1	1	1	1	0

Рис. 5: Матрица смежности, симуляция 1.1, шаг 4

step 5

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	0

Рис. 6: Матрица смежности, симуляция 1.1, шаг 5

Как видно из результатов, полученных при *Симуляции 1.1* нашего примера, мы получили динамическое равновесие, при котором граф взаимодействий разбивается на полные компоненты связности во всех шагах, за исключением *шага 4*.

Симуляция 1. 2

Рассмотрим этот же пример, но с очень большим заданным параметром неприязни к риску так, чтобы вторая часть функции полезности вершины "не играла" значения для решения задачи её оптимизации, а значит, и в динамике формирования графов. Так, например, зададим $\mathcal{R} = 1000$. И также проиллюстрируем матрицы смежности в этом случае.

step 0

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	0	1	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	1	0
1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	1	0

Рис. 7: Матрица смежности, симуляция 1.2, шаг 0

step 1

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	0

Рис. 8: Матрица смежности, симуляция 1.2, шаг 1

step 2

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0
0	1	0	1	1	1	1	1	0	0
0	1	1	0	1	1	1	1	0	0
0	1	1	1	0	1	1	1	0	0
0	1	1	1	1	0	1	1	0	0
0	1	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Рис. 9: Матрица смежности, симуляция 1.2, шаг 2

step 3

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
1	1	1	1	1	1	1	1	0	1
1	0	0	0	0	0	0	0	1	0

Рис. 10: Матрица смежности, симуляция 1.2, шаг 3

step 4

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	0

Рис. 11: Матрица смежности, симуляция 1.2, шаг 4

Стоит отметить матрицу, полученную на *шаге 3* данной симуляции, т.к. данная конфигурация графа практически представляет из себя звезду. Конфигурации, полученные на других шагах так же, как и в предыдущей симуляции, имеют вид связных полных компонент.

Симуляция 1. 3

Проведем еще одну симуляцию данного примера, с параметром риска, "близким" к нулю, т.е. задаим его $\mathcal{R} = 10^{-3}$, тем самым, "подавив" первую часть функции ползности вершин. Начальную матрицу смежности оставим в том же виде, что и в придыдущей симуляции и проиллюстрируем матрицы смежности в этом случае, начиная с первого шага.

step 1

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0

Рис. 12: Матрица смежности, симуляция 1.3, шаг 1

step 2

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Рис. 13: Матрица смежности, симуляция 1.3, шаг 2

step 3

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Рис. 14: Матрица смежности, симуляция 1.3, шаг 3

Как мы видим, результатом этой симмуляции является то, что динамика графа сходится к равновесию Нэша, где взаимодействуют между собой лишь две вершины, имеющие наименьшие значения стандартного отклонения. Отметим, что, в результате решения задачи оптимизации, для $k \neq s$-ой вершины в списе "желаемых соседей" будет находиться лишь вершина $S = s_1 \in N | \ \sigma_{s_1} = \min_i \sigma_i, i \in N$ (т.е. имеющая минимальное значение стандартного отклонения из множества вершин N), а у самой s_1 в этот список будет включена вершина $(s_2 \in N \setminus S | \ \sigma_{s_2} = \min_i \sigma_i, i \in N \setminus S)$ (т.е. имеющая минимальное значение стандартного отклонения среди множества вершин $N \setminus \ S$). При уловии $|S| = 1$ (т.е. мощность множества S равно 1) мы будем получать результат, сходный с тем, что мы получили в результате *Симуляции 1.3*. Если $|S| > 1$, то пар соединенных между собой вершин может быть больше одной в равновесном состоянии.

Таким образом, динамика графа в этом примере зависит от параметра риска $\mathcal{R}$. Где в предельных случаях по этому параметру, мы получаем либо динамическое равновесие либо равновесие Нэша. Поэтому в рамках *примера 1* можно сделать следующее предположение.

Предположение 1 *Существуют некоторые критические значения параметра $\mathcal{R}_{TopCr}$ и $\mathcal{R}_{LowCr}$ выше и ниже которых динамика графа сходиться либо к динамическому равновесию либо к равновесию Нэша.*

2.3 Пример 2

Этот пример будет отличаться от предыдущего лишь выбором матрицы α. Конечно, динамика графа будет зависеть от вида матрицы α, но нам важно лишь то, что она не константная, а значит матрица $\beta(l)$ из *Заметка 1* принимает более общий вид. Поэтому заполним нашу диагональную матрицу α случайным образом, т.е. $\alpha_{ii} \sim \mathcal{U}(0,\ 1)$[4].

Симуляция 2. 1

Рассмотрим симуляцию данного примера с параметром риска $\mathcal{R} = 1$.

step 0

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	0	1	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	1	0
1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	1	0

Рис. 15: Матрица смежности, симуляция 2.1, шаг 0

[4]Равномерное распределение на отрезке [0, 1].

step 1

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	0

Рис. 16: Матрица смежности, симуляция 2.1, шаг 1

step 2

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	1	1	1	1	1	0	0
1	0	1	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	0	0
1	1	1	0	1	1	1	1	0	0
1	1	1	1	0	1	1	1	0	0
1	1	1	1	1	0	1	1	0	0
1	1	1	1	1	1	0	1	0	0
1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Рис. 17: Матрица смежности, симуляция 2.1, шаг 2

step 3

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1	0	0
0	1	0	1	0	1	1	1	0	0
0	1	0	1	1	0	1	1	0	0
0	1	0	1	1	1	0	1	0	0
0	1	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Рис. 18: Матрица смежности, симуляция 2.1, шаг 3

step 4

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1	0

Рис. 19: Матрица смежности, симуляция 2.1, шаг 4

step 5

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	0

Рис. 20: Матрица смежности, симуляция 2.1, шаг 5

Для анализа этого примера ограничимся лишь одной его симуляцией. По результатам полученных матриц смежности, можно сделать вывод, что динамика графа в данном примере не сильно отличается от динамики графа в предыдущем примере, где параметром риска задан таким же образом. Однако, можно обратить внимание на *step 4* в этой симулиции и в *симуляции 1.1*, где имеется существенное различие.

Понятно, что для полного исследования модели рассмотрение лишь выше приведенных примеров недостаточно (так, хотя бы, стоит рассмотреть примеры при $n > 10$), однако, некоторые основные моменты динамики нашей модели были наблюдаемы в наших примерах. Таким образом, данная задача представляет интерес для дальнейшего исследования.

Заключение

В соответствии с поставленной целью было выполнено:

- ознакомление с некоторыми уже существующими моделями с похожей тематикой;
- изучен вопрос о возможных правилах взаимодействий между фирмами в рамках модели;
- описана процедура принятия решений о сотрудничестве между фирмами;
- разработана и написана модель;
- разработан и построен алгоритм для нашей модели;
- произведен анализ результатов, полученных в ходе чиссленного исследования примеров модели;
- реализован программный код.

Отметим, что в данной работе были также поставлены некоторые открытые вопросы и задачи, представляющие интерес для дальнейшего исследования.

Список литературы

Wilson R. J. Introduction to Graph Theory // Oliver and Boyd, Edinburgh. – 1972.

Feller W. An introduction to Probability Theory and its applications // John Wiley & Sons, University of Reading, New York. – 1970.

Newman M.E.J. Networks. An Introduction // University of Michigan and Santa Fe Institute, Oxford Univercity Press. – 2010.

Кострикин А.И. Введение в алгебру. Часть II. Линейная алгебра // Учебник для вузов. - М.:Физико-математическая литература. – 2000.

Roketskiy N. Competition and Networks of Collaboration // New York University. – January, 2012.

Brown M., Chiang S.-H. Unsystematic risk and coalition formation in product markets // International Journal of Industrial Organization – 2000.

Jackson M.O. Social and Economic Networks // Princeton University Press. – February 2008.

Лопатин А.С. Метод отжига // Санкт-Петербургский государственный университет. – 2005.

Винклер Г. Анализ изображений, случайные поля и динамические методы Монте-Карло // Перевод с английского С.М. Пригарина // Новосибирск: Филиал “Гео"Издательства СО РАН. – 2002.

Brandenburger A.M, Nalebuff J.N. Co-opetition 1. A revolutionary mindset that combines competition and cooperation 2. The game theory strategy that’s changing the game of business// Currency Doubleday. – 1996.

Osborne M. An Introduction to Game Theory // Oxford University Press, New York. – 2004.

Приложение

```
##CREATE MATRIX OF DEGREE
degree <- function(M){
numrow<-length(M[1,])
if(numrow>1){
Dmat<-matrix(c(rep(0,numrow)),nrow=numrow,ncol=numrow)
for(i in 1:numrow){
k<-0
for(j in 1:numrow){
k=k+M[i,j]
}
Dmat[i,i]<-1/k
}
return(Dmat)
}
else return(matrix(c(rep(1,numrow)),nrow=numrow,ncol=numrow))
}
##

## CREATE EDMAT
edmatrix <- function(n){
Dmat<-matrix(c(rep(0,n)),nrow=n,ncol=n)
for(i in 1:n){
Dmat[i,i]<-1
}
return(Dmat)
}
##

## USED MATRIX 1
u_mat1 <- function(MatX,alph=1/2){
demen<-length(MatX[1,])
D<-degree(MatX)
```

```
E<-edmatrix(demen)
MAT<-alph*E+(1-alph)*MatX%*%D
su<-0
for(i in 1:demen){
su<-su+1/D[i,i]
}
Dvec<-c(rep(0,demen))
for(i in 1:demen){
Dvec[i]<-(1/D[i,i])/su
}
B<-matrix(Dvec,nrow=demen,ncol=demen)
alo<-list(M1=MAT,Minf=B)
return(alo)
}
##

## USED MATRIX 2
u_mat2 <- function(MatX,ALFA){
demen<-length(MatX[1,])
D<-degree(MatX)
E<-edmatrix(demen)
MAT<-ALFA+MatX%*%(E-ALFA)%*%D
vec<-(eigen(MAT)$vectors[,1])/sum(eigen(MAT)$vectors[,1])
B<-matrix(vec,nrow=demen,ncol=demen)
alo<-list(M1=MAT,Minf=B)
return(alo)
}

## Function Centrality
centrality <- function(MatX,start_point){
CurMat<-u_mat1(MatX)$M1
if(length(CurMat)>1){
eigMatX <- eigen(CurMat)
m_e_v<-max_el_vector(eigMatX$values)
```

```
eigVec<-eigMatX$vectors[,m_e_v$num]
c1<-sum(start_point)/sum(eigVec)
return(c1*eigVec)
}
else{
return(start_point)
}
}
##
## Function Centrality2
centrality2 <- function(MatX,start_point,
num_used_matr=1,ALFA=MatX){
if (num_used_matr==1){
CurMat<-u_mat1(MatX)$M1
}
else {
CurMat<-u_mat2(MatX,ALFA)$M1
}
if(length(CurMat)>1){
eigMatX <- eigen(CurMat)
m_e_v<-max_el_vector(eigMatX$values)
eigVec<-eigMatX$vectors[,m_e_v$num]
c1<-sum(start_point)/sum(eigVec)
return(c1*eigVec)
}
else{
return(start_point)
}
}
##
## Function Centrality1
centrality1 <- function(MatX,start_point){
if(length(MatX)>1){
eigMatX <- eigen(MatX)
```

```
m_e_v<-max_el_vector(eigMatX$values)
eigVec<-eigMatX$vectors[,m_e_v$num]
c1<-sum(start_point)/sum(eigVec)
return(c1*eigVec)
}
else{
return(start_point)
}
}
##

## CONNECTED COMPONENT OF num's vertex
connected_component <- function(MatX,num){
flag <- TRUE
next_set_vertex<-MatX[,num]
next_set_vertex[num] <- 1
CurMat<-MatX
NN<-length(next_set_vertex)
d<-0
while(flag){
flag<-FALSE
CurMat<-CurMat%*%MatX
num_column<-CurMat[,num]
for(i in 1:NN){
if(next_set_vertex[i]==0){
if(num_column[i]>0){
next_set_vertex[i]<-1
flag<-TRUE
}
}
}
d<-d+1
}
vect_0<-c(rep(0,sum(next_set_vertex)))
```

```
k<-1
for(i in 1:NN){
if(next_set_vertex[i]>0){
vect_0[k]<-i
k<-k+1
}
}
comp<-list(vect=next_set_vertex,deep=d,num_full=vect_0)
return(comp)
}
##

## CREATE MATRIX
create_matrix <- function(Start_Mat,vectorr){
NN<-sum(vectorr)
N<-length(vectorr)
stroka<-0
colonka<-0
CurMat<-matrix(c(rep(0,NN)),nrow=NN,ncol=NN)
for(i in 1:N){
if(vectorr[i]>0){
stroka<-stroka+1
for(j in i:N){
if(vectorr[j]>0){
colonka<-colonka+1
CurMat[stroka,colonka]<-Start_Mat[i,j]
CurMat[colonka,stroka]<-Start_Mat[i,j]
}
}
colonka<-stroka
}
}
return(CurMat)
}
```

```
##

## Start profit of CONNECTED_COMPONENT
st_profits_comp <- function(vect,profits){
NN<-length(vect)
vect_0<-c(rep(0,NN))
for(i in 1:NN){
vect_0[i]<-profits[vect[i]]
}
return(vect_0)
}
##

## ZNACHENIE TSELEVOY FUNKTSII
value_of_func <- function(Cur_Mat,num_vertex,
st_profits,sigma,RISC,num_mat=1,ALFA=Cur_Mat){
N<-length(st_profits)
con_comp<-connected_component(Cur_Mat,num_vertex)
Con_Mat<-create_matrix(Cur_Mat,con_comp$vect)
st_prof<-st_profits_comp(con_comp$num_full,st_profits)
siigma<-st_profits_comp(con_comp$num_full,sigma)
CALFA<-create_matrix(ALFA,con_comp$vect)
if(num_mat==1){
cur_vec_energy<-centrality(Con_Mat,st_prof)
}
else{
cur_vec_energy<-centrality2(Con_Mat,st_prof,num_mat,CALFA)
}
num_count<-0
for(i in 1:N){
if(con_comp$vect[i]>0){
if (i<=num_vertex){num_count<-num_count+1}
}
}
```

```
if(num_mat==1){
B<-u_mat1(Con_Mat)$Minf ##Здесь для 1-го случая
}
else{
B<-u_mat2(Con_Mat,CALFA)$Minf ##Здесь для 2-го случая
}
sigm<-0
for(i in 1:length(siigma)){
sigm<-sigm+((B[num_count,i])^2)*((siigma[i])^2)
}
sigm<-(sigm^(1/2))
cur_vec_energy[num_count]<-cur_vec_energy[num_count]-(sigm/RISC)
return(-cur_vec_energy[num_count])
}
##

##RASPREDELENIE RAZNITSI ENERGIY

##

## Metropolis Method
metro_one_step <- function(CurMat,num_vertex,
st_profits,sigma,RISC,num_mat=1,ALFA=CurMat){
epsel <- -0.00001
x_cur<-CurMat[,num_vertex]
cur_energy <- value_of_func(CurMat,num_vertex,
st_profits,sigma,RISC,num_mat,ALFA)
x_min<-x_cur
Glob_min <- cur_energy
c=0.9
NNN<-length(st_profits)
Temp_vect_0<-c(rep(1,NNN))
Temp_vect<-Temp_vect_0
i_count<-c(rep(0,NNN))
```

```
for(k in 1:(20*NNN)){
print(k)
flag<-TRUE
while(flag){
NNextMat<-CurMat
ravn_vect<-runif(NNN,0,1)
z_vect<-sign(ravn_vect-0.5)*
Temp_vect*((1+1/Temp_vect)^abs((2*ravn_vect-1))-1)
x_next<-x_cur+z_vect
for(i in 1:NNN){
if(x_next[i]>0.5){
x_next[i]<-1
}
else {x_next[i]<-0}
NNextMat[i,num_vertex]<-x_next[i]
NNextMat[num_vertex,i]<-x_next[i]
}
x_next[num_vertex] <- 0
NNextMat[num_vertex,num_vertex]<-0
next_energy <- value_of_func(NNextMat,num_vertex,
st_profits,sigma,RISC,num_mat,ALFA)
if((next_energy-cur_energy)<0){
ah<-1
}
else{
ah<-exp((-next_energy+cur_energy)/mean(Temp_vect))
}
sluch_dem<-runif(1,0,1)
fla<-FALSE
if(sluch_dem<ah){
for(i in 1:NNN){
if(x_cur[i]!=x_next[i]){
fla<-TRUE
```

```
i_count[i]<-i_count[i]+1
Temp_vect[i]<-Temp_vect_0[i]*exp(-c*(i_count[i]^(1/NNN)))
}
}
if(fla){
cur_energy<-next_energy
x_cur<-x_next
flag<-FALSE
}
}
}
if(cur_energy-Glob_min<epsel){
Glob_min<-cur_energy
x_min<-x_cur
}
}

## Check of extreme cases
#1)
x_next<-x_cur
NNextMat<-CurMat
for(i in 1:NNN){
x_next[i]<-0
NNextMat[i,num_vertex]<-0
NNextMat[num_vertex,i]<-0
}
next_energy<-value_of_func(NNextMat,num_vertex,
st_profits,sigma,RISC,num_mat,ALFA)
if(next_energy-Glob_min<epsel){
Glob_min<-next_energy
x_min<-x_next
}
#2)
NNextMat<-CurMat
```

```
for(i in 1:NNN){
x_next[i]<-1
NNextMat[i,num_vertex]<-1
NNextMat[num_vertex,i]<-1
}
x_next[num_vertex] <- 0
NNextMat[num_vertex,num_vertex]<-0
next_energy<-value_of_func(NNextMat,num_vertex,
st_profits,sigma,RISC,num_mat,ALFA)
if(next_energy-Glob_min<epsel){
Glob_min<-next_energy
x_min<-x_next
}
####

minim<-list(val=Glob_min,vector=x_min)
return(minim)
}
##

## PROGRAMA 1 STEP
step_of_prog <- function(CurMat,st_profits,
sigma,RISC,num_mat=1,ALFA=CurMat){
NN<-length(st_profits)
Next_mat<-matrix(c(rep(0,NN)),nrow=NN,ncol=NN)
for(i in 1:NN){
vect<-metro_one_step(CurMat,i,st_profits,sigma,RISC)
Next_mat[,i]<-vect$vector
}
for(i in 1:(NN-1)){
for(j in (i+1):NN){
if(Next_mat[i,j]!=Next_mat[j,i]){
Next_mat[i,j]<-0
Next_mat[j,i]<-0
```

```
}
}
}
return(Next_mat)
}
##
```