

СОГЛАСОВАНО

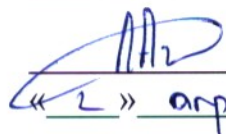
УТВЕРЖДАЮ

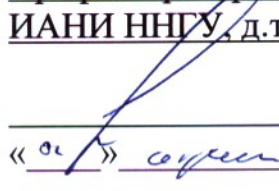
Сторона ЗАКАЗЧИКА

Сторона ИСПОЛНИТЕЛЯ

Профессор кафедры  
ИАНИ ННГУ, д.ф.-м.н.

Профессор кафедры  
ИАНИ ННГУ, д.т.н.

  
Л.Г. Афраймович  
« 2 » апреля 2019 г.

  
Н.В. Старостин  
« 02 » апреля 2019 г.

### Отчет

**«Обзор известных подходов к решению задачи планирования графика спортивных мероприятий»**

**Этап 1. Подготовка обзоров на существующие подходы к решению задачи**

**НИР «Разработка и реализация программного обеспечения планирования графика спортивных мероприятий»**

**(Шифр ПО «График»)**

Ответственный исполнитель

 А.А. Алабин

« 02 » апреля 2019 г.

## Оглавление

1 Обзор алгоритмов комбинаторной оптимизации .....	3
1.1 Классификация алгоритмов по типу решения .....	3
1.2 Классификация приближенных алгоритмов .....	3
1.2.1 Принцип принятия решения.....	5
1.2.2 Сложность структуры .....	5
1.2.3 Тип используемых пространств.....	7
1.2.4 Тип формируемой траектории .....	8
1.2.5 Влияние на ландшафт поиска .....	8
1.2.6 Использование памяти.....	9
1.2.7 Наличие адаптации/обучения .....	10
1.2.8 Наличие специальной модели задачи.....	10
2 Обзор методов построения календаря матчей.....	10

# **1 Обзор алгоритмов комбинаторной оптимизации**

## **1.1 Классификация алгоритмов по типу решения**

Алгоритмы комбинаторной оптимизации (КО) можно классифицировать по многим характеристикам. Одной из самых важных является тип получаемого решения. По типу решения можно выделить точные, приближенные и эвристические алгоритмы.

Точные алгоритмы за конечное время гарантированно возвращают оптимальное решение или делают вывод, что его не существует, если задача неразрешима. Эти алгоритмы можно разделить на два класса: общие и специальные методы. Общие методы могут быть применены к весьма широкому кругу задач. Наибольшее распространение получили метод полного перебора, метод ветвей и границ, метод ветвей и сечений, последовательный анализ и отсеивание вариантов, динамическое программирование.

Приближенные алгоритмы — это алгоритмы, гарантированно возвращающие вариант решения за конечное время (если оно существует), и для которых может быть получена оценка точности найденных вариантов решения. Оценки точности бывают двух типов: априорные и апостериорные. Априорная оценка точности — гарантированная оценка, задаваемая до начала решения задачи. Апостериорные оценки вычисляют непосредственно во время процесса решения или после его окончания, учитывая при этом и само полученное решение.

Под эвристическими алгоритмами («эвристиками») чаще всего понимают алгоритмы, для которых отсутствует либо неизвестна оценка точности. Это означает отсутствие гарантий того, что для конкретной решаемой задачи алгоритм не вернет как угодно «плохое» (в смысле целевой функции) решение или что он вернет решение вообще. В то же время эвристики корректны в том смысле, что не возвращают варианты решения, которые не являются допустимыми решениями задачи. Для многих практических задач такие алгоритмы показали свою эффективность, и, более того, нередко эвристики могут быть единственным способом получить «хорошее» решение за приемлемое время.

По сути, все вычислительные подходы к решению сложных задач КО могут быть описаны как поисковые процедуры, идея которых состоит в итерационном порождении/генерировании вариантов решений и их оценивании. Такой подход к описанию методов решения задач КО является общим, хотя и не отражает многих характеристик структуры.

## **1.2 Классификация приближенных алгоритмов**

Приближенными алгоритмами часто называют все «неточные» алгоритмы — как алгоритмы с оценкой точности, так и эвристические алгоритмы. Ввиду сложности многих

практически важных задач КО применимость точных алгоритмов ограничена, и маловероятно, что удастся разработать эффективные точные методы, применяемые к реальным задачам. Кроме того, схема точных алгоритмов часто не позволяет решать с их помощью некоторые типы задач КО, такие как динамические задачи или задачи с неопределенностями. Поэтому именно приближенные алгоритмы находят все более широкое применение на практике.

Классификация и примеры приближенных алгоритмов представлены на Рисунках 1, 2.



Рисунок 1. Приближенные алгоритмы КО

Первые три класса имеют длительную историю применения и достаточно полно описаны в литературе.

Эволюционные методы являются широким и разнообразным классом алгоритмов, использующих принципы биологической эволюции для решения оптимизационных задач.

Роевой интеллект в контексте КО представлен классом алгоритмов, которые являются децентрализованными системами простых агентов, локально взаимодействующих со средой и между собой. Несмотря на отсутствие централизованной структуры управления агентами, локальное взаимодействие между ними приводит к проявлению глобального поведения всей системы в целом.

В методах сканирования пространства, принадлежащих к числу популяционных алгоритмов, на каждой итерации происходит формирование направления поиска в пространстве решений на основе нескольких имеющихся вариантов.

Предлагается классифицировать приближенные методы КО по таким характеристикам: принцип принятия решений; сложность структуры; тип используемых пространств; тип формируемой траектории; влияние на ландшафт поиска; использование памяти; наличие адаптации/обучения; наличие специальной модели задачи.

#### **1.2.1 Принцип принятия решения**

В приближенных методах, прежде всего, различают два принципа принятия решений по наличию рандомизации — детерминированный и стохастический. Примером детерминированного метода является простой локальный поиск — алгоритм, который, начиная с некоторого начального решения, итерационно пытается его заменить лучшим (в смысле целевой функции) из окрестности. Один из наиболее известных стохастических алгоритмов — алгоритм имитационного отжига. Этот подход является развитием локального поиска: разрешается принятие в качестве текущего решения не только лучшего из окрестности, но и худшего по значению целевой функции решения. Выбор решения определяется вероятностным механизмом на основе значений целевой функции.

#### **1.2.2 Сложность структуры**

По сложности структуры различают простые алгоритмы, гибридные алгоритмы, метаэвристики, гибридные метаэвристики и гиперэвристики. Классификация по структуре весьма важна, но, поскольку, несмотря на широкое употребление понятий эвристики, метаэвристики и гиперэвристики, отсутствуют их общепринятые формальные определения, затруднительно четко разграничить алгоритмы согласно этой характеристике. Термин «метаэвристика» введен в работе, и упрощенно метаэвристические методы можно понимать как общую технику или подход, который используется для управления встроенными проблемнозависимыми процедурами (методами) с целью повысить их эффективность или робастность. Такие процедуры часто

являются отдельными алгоритмами решения той же задачи, что и метод в целом. Метаэвристические методы описываются как общие подходы, которые конкретизацией проблемнозависимых компонентов могут быть адаптированы к конкретной задаче или подклассу задач.

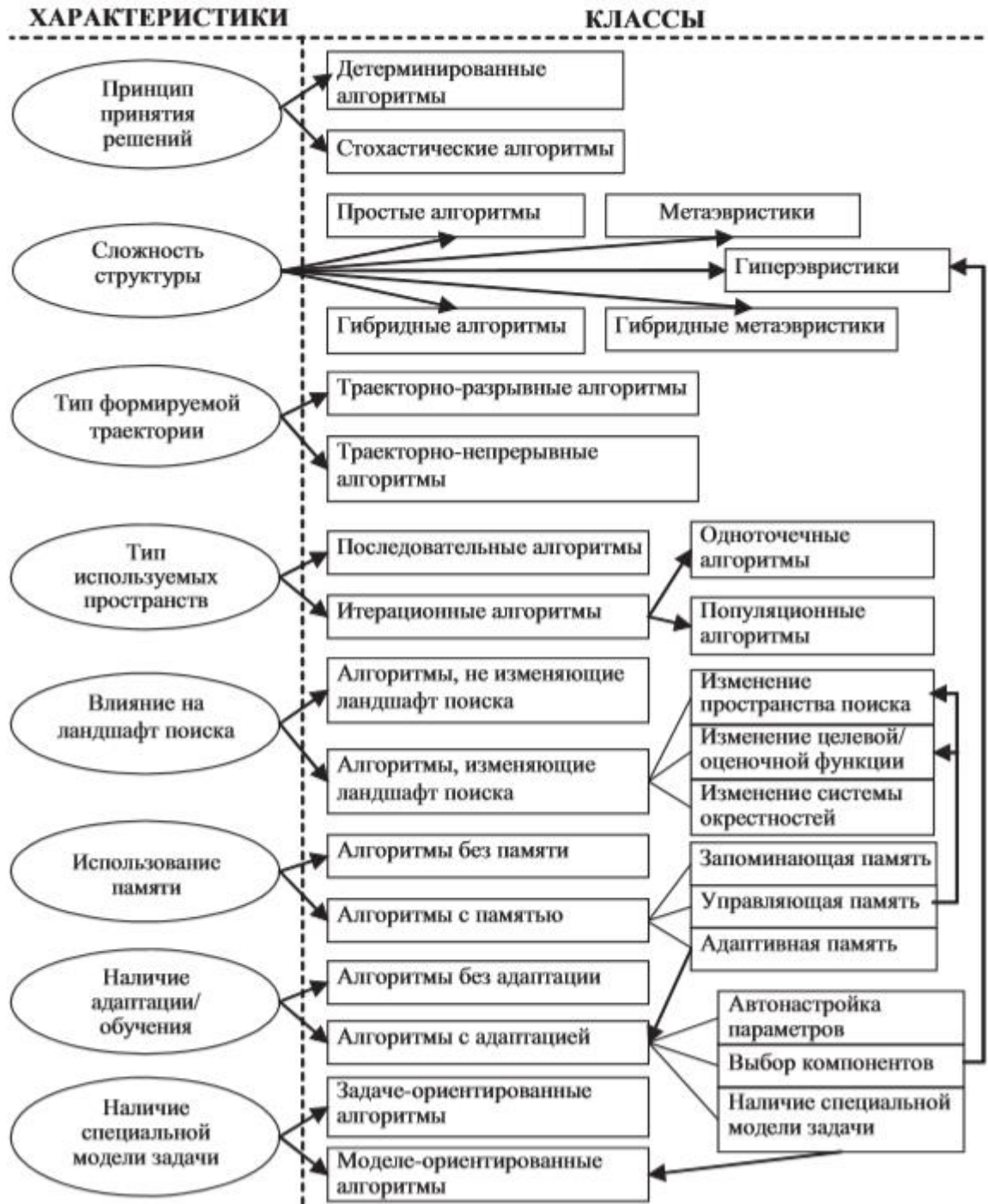


Рисунок 2. Критерии классификации приближённых алгоритмов

Под гибридными алгоритмами понимают такое объединение простых алгоритмов, которое еще не порождает метаэвристичный алгоритм. Например, это может быть простое

последовательное или параллельное выполнение нескольких алгоритмов. На практике простые (как и гибридные) алгоритмы используются все реже, поскольку развитие метаэвристических подходов позволяет получать лучшие результаты, а темпы роста производительности вычислительных систем дают возможность применять все более сложные алгоритмы к задачам повышенной размерности. Исключение составляют случаи, когда удастся разработать удачный специализированный алгоритм для отдельной задачи или узкого класса задач (эффективно учесть при разработке алгоритма особенности конкретной задачи), а также алгоритмы для задач повышенной размерности или задач с трудоемким вычислением целевой функции.

Гибридными метаэвристиками называют приближенные алгоритмы КО, которые комбинируют в себе компоненты из двух и более отдельных метаэвристик, точных методов, специализированных алгоритмов и т.д. К гибридным метаэвристикам также отнесем многоагентные системы, в которых агенты представляют отдельные алгоритмы (например, метаэвристики, точные или специализированные алгоритмы), взаимодействующие между собой. По аналогии с последней возможно классифицировать гибридные алгоритмы по степени гибридизации, порядку выполнения и стратегии управления.

Гиперэвристика — метод, автоматизирующий процесс выбора, комбинирования, генерирования или адаптации ряда простых алгоритмов (эвристик) или их компонентов с целью эффективного решения задачи. Упрощенно под гиперэвристикой можно понимать (мета)эвристики для выбора/конструирования (мета)эвристик. Одна из целей создания гиперэвристик — разработка оптимизационных систем, способных решать классы разных задач, а не только отдельные задачи.

### **1.2.3 Тип используемых пространств**

По такому критерию, прежде всего, выделим класс последовательных, или конструктивных алгоритмов. Их суть состоит в последовательном построении решения из отдельных частей по определенным правилам. Простейшим примером последовательного алгоритма является эвристика, называемая «жадной», реализация которой для задачи коммивояжера известна как «иди в ближний (город)».

Другой важный класс алгоритмов — итерационные. Такие алгоритмы работают в пространстве решений исходной задачи. Итерационные методы разделяются по числу вариантов решений, которыми алгоритм одновременно оперирует (генерирует и оценивает), на два класса: в случае одноэлементного множества алгоритмы называют одноточечными, а в случае, когда множество текущих вариантов решений содержит больше одного элемента, — популяционными. Одноточечные алгоритмы часто называют траекторными, поскольку процесс их работы можно

представить в виде непрерывной траектории (пути) на графе соседства (понятие траекторной непрерывности рассматривается ниже). Алгоритмы стохастического локального поиска, такие как, например, имитационный отжиг, управляемый локальный поиск и GRASP, являются одноточечными. Название «популяционные алгоритмы» введено по аналогии с терминологией, используемой в эволюционных методах. Эволюционные методы, такие как генетические алгоритмы, были одними из первых, оперировавших множеством вариантов решений одновременно. К числу ранних популяционных алгоритмов решения задач КО относятся и фронтальные алгоритмы.

Конструктивные методы обычно являются наиболее быстрыми методами решения задач КО, но, в общем случае, проигрывают итерационным методам в «качестве» решений. На практике конструктивные алгоритмы чаще всего используются либо для задач повышенной размерности или задач с затратным вычислением целевой функции, либо для генерирования начальных решений для других методов.

#### **1.2.4 Тип формируемой траектории**

Важным отличием между разными итерационными методами является то, формируют ли они одну траекторию поиска, что отвечает «непрерывным» переходам на графе соседства, или осуществляют большие «прыжки» на этом графе.

Имитационный отжиг и табу-поиск — типичные примеры траекторно-непрерывных методов. Алгоритмы, осуществляющие более сложные переходы, но которые представимы в виде более простых шагов, также можно интерпретировать как траекторно-непрерывные. К таким алгоритмам относятся, например, алгоритмы метода поиска переменной глубины, в частности алгоритм Лина–Кернигана, или алгоритмы, которые базируются на «выбрасывании цепей» (ejection chains).

Большинство популяционных алгоритмов являются траекторно-разрывными относительно любой одной фиксированной системы окрестностей. Среди исключений — фронтальный алгоритм. В модификации этого алгоритма, которая базируется на локальном поиске, на каждой итерации осуществляются параллельные локальные улучшающие шаги относительно некоторой наперед заданной системы окрестностей.

#### **1.2.5 Влияние на ландшафт поиска**

В отдельный класс выделяют методы КО, которые изменяют ландшафт поиска в процессе своей работы. Возможны следующие типы изменений: пространства поиска, целевой/оценочной функции и системы окрестностей. Рассмотрим каждый из них в отдельности.

Изменение пространства поиска. Под изменением пространства поиска понимается



включение/исключение элементов в/из него. В табу-поиске запрещено возвращение к недавно рассмотренным вариантам решения, что соответствует их временному исключению из пространства поиска. Большинство методов КО не изменяют пространство поиска.

Изменение целевой/оценочной функции. Некоторые алгоритмы модифицируют оценивание отдельных состояний процесса поиска во время выполнения алгоритма. Одним из примеров является метод «выламывания» (breakout), базовой идеей которого является введение штрафов на включение отдельных компонентов в решение, изменяющих значение целевой функции. В качестве развития этого подхода был предложен управляемый локальный поиск.

Изменение системы окрестностей. Многие из приближенных алгоритмов КО используют в процессе поиска одну систему окрестностей. Это, например, такие алгоритмы, как имитационный отжиг, табу-поиск, G-алгоритмы. В повторяемом локальном поиске используется, по крайней мере, две системы окрестностей: вначале осуществляется локальный поиск до достижения локального минимума, после этого — возмущение найденного локального минимума. Фактически, возмущение может рассматриваться как шаг в другой системе окрестностей, отличной от той, по отношению к которой был найден локальный минимум. Эти соображения, в частности, развиты независимо в методах поиска в пульсирующих окрестностях и поиска в изменяемых окрестностях — в этих подходах используемые окрестности систематически изменяются в пределах заведомо заданного их перечня.

#### **1.2.6 Использование памяти**

Очень важной характеристикой, по которой различают приближенные алгоритмы КО, является использование памяти. Память — это набор специальных переменных, в которых отображается опыт (информация), накопленный в процессе поиска (работы алгоритма). В методах КО имеется несколько функций памяти. Во-первых, память может использоваться для хранения информации, которая будет использована на выходе алгоритма (например, наилучшее найденное решение). Во-вторых, память может использоваться для управления процессом поиска. Этот тип памяти может быть кратковременным или долговременным.

Кратковременная память подразумевает запоминание определенного количества последних сгенерированных решений, частей решений и принятых решений. Запоминание в табу-поиске последних сгенерированных решений является примером кратковременной памяти. Долговременная память — набор специальных дополнительных переменных, в которых запоминается информация обо всем осуществленном процессе поиска. Величины штрафов в алгоритме управляемого локального поиска являются примером долговременной памяти. Функция управления поиском тесно связана с влиянием на ландшафт поиска: алгоритмы с этой

функцией памяти изменяют ландшафт поиска.

### **1.2.7 Наличие адаптации/обучения**

Среди приближенных методов КО с памятью выделяют такие, которые обладают свойством адаптироваться/обучаться (изменять значения своих параметров в процессе работы на основе состояния памяти). Адаптация включает динамическую настройку значений параметров, автоматический выбор подчиненных процедур и наличие модели задачи.

Реактивный/реагирующий поиск предоставляет механизм для включения в алгоритм настройки варьируемых параметров: значения параметров настраиваются в автоматизированном цикле обратной связи соответственно «качеству» найденных решений, истории поиска и другим характеристикам. В реактивном табу-поиске, одном из первых реактивных алгоритмов, период запрещения автоматически настраивается в процессе поиска.

Процесс обучения активно используется для автоматизированного выбора компонентов, осуществляемого в алгоритмах, которые принадлежат к описанному классу гиперэвристики.

Адаптация может достигаться путем использования специальной структуры памяти — модели задачи, о чем пойдет речь далее.

Конкретные техники адаптации развиваются и в рамках отдельных классов методов. Например, в классе меметических алгоритмов развивается направление адаптивных методов, в которых совершается динамический выбор типа алгоритма встроенного локального поиска.

### **1.2.8 Наличие специальной модели задачи**

По отсутствию или наличию в алгоритме специальной модели задачи можно выделить класс задаче-ориентированных методов и класс модели-ориентированных алгоритмов. Большинство методов КО являются задаче-ориентированными, так как они генерируют новые варианты решений только на основе текущего решения или множества текущих вариантов решений. Это такие методы, как генетические алгоритмы, повторяемый локальный поиск и др. В модели-ориентированных методах варианты решений генерируются с использованием параметризированной вероятностной модели, которая обновляется на основе ранее рассмотренных вариантов решений так, чтобы поиск концентрировался в областях с вариантами решений «высокого качества». Широко известными модели-ориентированными методами являются метод оптимизации муравьиными колониями, метод кросс-энтропии и метод вычисления оценок распределений.

## **2 Обзор методов построения календаря матчей**

Многие спортивные лиги (например, футбол, хоккей, баскетбол) должны решать

проблемы с расписанием туров. Эти проблемы, как правило, содержат множество противоречивых ограничений и различные цели оптимизации, такие как минимизация пройденного расстояния, недоступность стадиона в определенные даты, минимальное количество дней между домашним матчем и соответствующим выездным матчем и т. д. Для решения этих проблем было предложено много подходов с различной степенью успеха: целочисленное линейное программирование, программирование с ограничениями, локальный поиск (имитация отжига, поиск табу), гибридный подход. Спортивное планирование было также исследовано с точки зрения окраски ребер графов. Сами алгоритмы не представляют интереса для поиска подходящего решения нашей задачи построения календаря матчей, т.к. она имеет совершенно иную структуру и критерий оптимальности. Полезны некоторые приемы оперирования с расписанием туров, т.к. расписание туров фактически является подзадачей задачи построения календаря матчей.

Одним из предложенных вариантов оперирования с расписанием туров является сведение задачи к реберной раскраске. Вершины соответствуют командам, а цвета ребер – турам (периодам) (Рисунок 3).

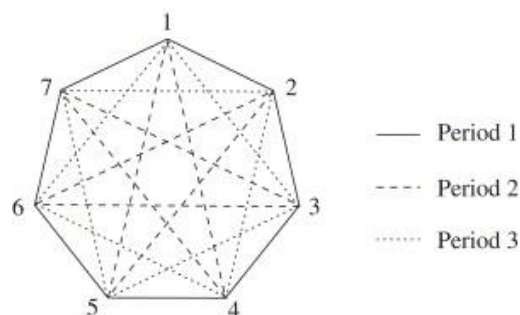


Рисунок 3. Реберная раскраска

Другим приемом может быть оперирование таблицей. Каждая  $i$ -ая строка относится к  $i$ -ой команде и хранит ее соперников для  $2(n-1)$  матчей, где  $n$  – число команд. Столбцы таблицы – туры чемпионата, в каждом из которых все команды играют ровно по одному разу и соперники матчей не могут повторяться до завершения полного круга из  $n-1$  матчей. На Рисунках 4-7 приведены варианты операций для локального поиска.

- 1) Обмен туров

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	@5	3	4	@4	@3	5	2	@6
2	5	1	4	@6	@3	3	6	4	@1	@5
3	@4	5	6	@1	2	@2	1	@6	@5	4
4	3	6	@2	@5	@1	1	5	@2	@6	@3
5	@2	@3	1	4	6	@6	@4	@1	3	2
6	@1	@4	@3	2	@5	5	@2	3	4	1

Рисунок 4. Обмен туров

2) Обмен команд

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	2	1	5	@2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	2	1	5	@2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@5	4	3	@2	@4	@3	2	5	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	2	5	@1	6	@5	1	@6	@2	4
4	3	6	@1	@2	@5	1	2	5	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@2	5	@3	2	@5	3	4	1

Рисунок 5. Обмен команд

3) Обмен отдельных туров

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	2	3	@5	@4	@3	5	4	@6
2	5	1	@1	@5	4	3	6	@4	@6	@3
3	@4	5	4	@1	6	@2	1	@6	@5	2
4	3	6	@3	@6	@2	1	5	2	@1	@5
5	@2	@3	6	2	1	@6	@4	@1	3	4
6	@1	@4	@5	4	@3	5	@2	3	2	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	4	2	3	@5	@4	@3	5	@2	@6
2	5	@6	@1	@5	4	3	6	@4	1	@3
3	@4	5	4	@1	6	@2	1	@6	@5	2
4	3	@1	@3	@6	@2	1	5	2	6	@5
5	@2	@3	6	2	1	@6	@4	@1	3	4
6	@1	2	@5	4	@3	5	@2	3	@4	1

Рисунок 6. Обмен отдельных туров



4) Обмен отдельных команд

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@6	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@1	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@6	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@1	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@1	@5	4	3	6	@4	@6	@3
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@3	@6	@2	1	5	2	@1	@5
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@1	@5	4	3	6	@4	@6	@3
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@3	@6	@2	1	5	2	@1	@5
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	2	3	@5	@4	@3	5	4	@6
2	5	1	@1	@5	4	3	6	@4	@6	@3
3	@4	5	4	@1	6	@2	1	@6	@5	2
4	3	6	@3	@6	@2	1	5	2	@1	@5
5	@2	@3	6	2	1	@6	@4	@1	3	4
6	@1	@4	@5	4	@3	5	@2	3	2	1

Рисунок 7. Обмен отдельных команд