Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича»

Птицына Л.К.

Системы представления и приобретения знаний

Конспект лекций

Направление 09.04.02 – Информационные системы и технологии

Направленность (профиль) – Коммуникационные технологии

Санкт-Петербург

2016

Введение

На современном этапе развития общества высокая значимость знаний признаётся, прежде всего, в контексте социально-экономического развития стран. В стратегии социально-экономического развития России до 2020 года определяются пути формирования инновационной экономики. Инновационная экономика позиционируется как экономика, которая соответствует объективным требованиям внедрения достижений науки в производство, повышения качества современного образования, технологической и информационной революций.

При инновационной экономике наука становится неотъемлемой частью промышленного производства и непосредственной производительной силой. Инновации рассматриваются как новые комбинации производственных и интеллектуальных ресурсов, обеспечивающие создание новых технологий, производств, товаров и услуг. Благодаря созданию новых технологий, производств и продуктов формируются новые рынки, наблюдается их развитие. Интеллектуальная собственность становится объектом разнообразных сделок и отношений. Для обеспечения устойчивой конкурентоспособности привлекаются новые знания.

В качестве ключевой опоры для инновационной экономики выбираются наука, система образования, интеллект и навыки работающих, нацеленные на создание, распространение и целевое использование интеллектуальных ресурсов. Инновационная экономика отличается возрастающей ролью знаний, их созданием и распространением, реализацией и приспособлениями к определённым нуждам, к процессам производства новых продуктов и оказания услуг.

В условиях инновационной экономики появляется новый вид предпринимательства – интеллектуальное предпринимательство. По определению предпринимательство – это процесс культурных инноваций. При интеллектуальном предпринимательстве акцентируется внимание на социально значимых аспектах деятельности человека, риски принимаются на себя, изыскиваются возможности, открывается и создаётся знание, порождаются инновации, организуется сотрудничество и решаются проблемы в любых сферах социальной реальности: корпоративной, некоммерческой, государственной и образовательной. Отличительной особенностью интеллектуального предпринимательства является эффективное использование существующих и новых знаний.

По мере развития инновационной экономики знание становится рыночным продуктом и начинается новый этап развития – экономика знаний, при котором знания играют главную роль, а их производство является источником роста. Знание рассматривается как некая самостоятельная сущность, с которой можно производить самые разнообразные операции: передавать от одного лица к другому, хранить, обладать, продавать, покупать, уничтожать и другие операции.

Становление и развитие экономики знаний ассоциируется с пятым технологическим укладом, который определяется как уклад информационных и коммуникационных технологий.

С целью повышения эффективности организации и её конкурентоспособности осуществляется систематическое формирование, обновление и применение знаний, определяемое как управление знаниями. В основу управления знаниями

закладывается использование людей, искусственного интеллекта, процессов и технологий, позволяющее организации оптимизировать обмен знаниями, сохранить и приумножить знания.

Во многих секторах современного глобального рынка для создания конкурентного преимущества требуется наличие у фирмы динамических способностей. Динамические способности фирмы – это её потенциал распознавания новых возможностей и реконфигурации знаний как активов, компетенций и комплементарных активов и технологий в целях достижения устойчивого конкурентного преимущества.

В соответствии с вышеизложенным актуализируется изучение дисциплины «Системы представления и приобретения знаний».

1 ОПРЕДЕЛЕНИЯ ЗНАНИЯ В ОБЛАСТИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

1.1 ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ ЗНАНИЯ

Понятие знания характеризуется многообразием определений.

В философских исследованиях знания рассматриваются в контексте проблем разума. При этом на центральное место выдвигается связь между знаниями и действиями.

В профессиональной деятельности часто употребляются следующие определения:

знание – понимание, приобретаемое практическим опытом;

знание – состояние осведомлённости о чём-то или обладание информацией;

знание – диапазон осведомлённости или информированности;

знание – осознание, понимание и толкование определённой информации с учётом путей наилучшего её использования для достижения конкретных целей;

знание – эта система представлений о предметной области в виде сущностей, т.е. вещей, явлений, процессов, их проявлений в форме фактов и отношений или объектов и их связей;

знание – комбинация опыта, ценностей, контекстной информации, экспертных оценок, которая даёт возможность оценивать и инкорпорировать новый опыт и информацию;

знания – способность к инновациям, предрасположенность к самостоятельной творческой деятельности.

Задача из предметной области может представляться конкретными сущностями и их проявлениями в форме, определяемой выбранной моделью знаний.

Многими профессионалами разделяется мнение, что результаты решения задач, истинная, проверенная информация, обобщённая в виде систем взглядов, законов и теорий представляют собой знания.

Наиболее часто IT-специалисты ориентируются на приводимые ниже определения.

В области информатики знания определяются как вид информации, хранимой в базах знаний и отражающей знания специалистов в определённой предметной области.

В области ситуационного управления знания определяются как множество всех текущих ситуаций в объектах данного типа и способы перехода от одного описания объекта к другому. При этом различаются априорные (доопытные) и апостериорные (послеопытные) ситуации.

Специалистами в области информационных систем знания определяются как всевозможная информация об общих свойствах и закономерностях предметной области, которая хранится в системе вне зависимости от того, решается или не решается в некоторый момент времени задача и выраженная в терминах некоторой модели представления знаний.

Знания характеризуются внутренней интерпретируемостью, структурируемостью, связностью, семантической метрикой и активностью.

Достаточно часто пользуются образными определениями знания:

знания = факты + убеждения + правила,
знания = факты + гипотезы + правила.

В жизненном цикле экспертных систем пристальное внимание уделяется глубинным знаниям. При этом глубинные знания трактуются как модели функциональных и каузальных отношений.

На начальных этапах жизненных циклов информационных систем и технологий рассматриваются декларативные знания – информация, не содержащая в явном виде описания каких-либо процедур, которые необходимо выполнить.

Декларативные знания образуются множествами описаний состояний и условий переходов между ними, которые носят символьный характер и не содержат в явном виде описания исполняющих процедур.

Альтернативой декларативным знаниям являются процедурные знания.

Процедурными знаниями охватываются исходные состояния и явные описания процедур, обрабатывающих исходные знания при необходимости получения состояния полного множества производных знаний. В описания процедур вводится семантика. Процедуры ориентируются на генерацию синтаксических знаний. Благодаря подобной функциональности появляется возможность отказаться от хранения всех состояний баз знаний, требуемых при решении задач предметной деятельности. В подобном случае экономится память при хранении знаний.

В экономике обращается внимание на неявные знания.

Неявные знания в принципе не формализуются и могут существовать лишь вместе с их обладателем – человеком или группой лиц. При этом рассматриваются два вида неявного знания:

- технические навыки, которые проявляются у мастеров своего дела и выступают, как правило, результатом многолетней практики;
- верования, идеалы, ценности и ментальные модели, которые используются в деятельности.

Ценность неявного знания считается не меньшей, чем ценность явного знания.

Наряду с вышеприведёнными определениями в области информационных технологий пользуются определениями типовых разновидностей знаний.

Каузальные знания – информация о причинно-следственных отношениях.

Непосредственные знания – знания, полученные в результате прямого опыта, прямого восприятия без накопленных ранее знаний и правил логического мышления.

Конструктивные знания – знания о структуре и взаимодействии различных объектов и (или) процессов.

Неточные знания – знания, отличающиеся неполнотой, приблизительностью или даже противоречивостью.

Семантические знания – знания о состоянии объектов и процессов и отношений между ними.

Эвристические знания – знания, основанные на эвристиках. Эвристика – наука, изучающая продуктивное творческое мышление.

1.2 ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

Представление знаний является одним из направлений искусственного интеллекта.

Искусственный интеллект это область науки, в которой определяются, систематизируются, решаются, анализируются, оптимизируются и автоматизируются интеллектуальные задачи.

В искусственном интеллекте выделяются следующие категории интеллектуальных задач:

1. Задачи воспроизведения способностей человека.
2. Задачи обеспечения рациональности, когда все действия, относящиеся к некоторой системе, выполняются правильно, в условиях обладания знаниями о том, что является правильным.
3. Задачи воссоздания окружающей среды для объектов, (и/или) субъектов и их взаимодействия.

Интеллектуализация ИИ обеспечивается посредством решения всех категорий задач.

В настоящее время различаются четыре категории систем искусственного интеллекта:

1. Системы с воспроизведением мыслительных процессов.
2. Системы с воспроизведением рациональных рассуждений.
3. Системы реализации функций, требующих интеллектуальности при их выполнении людьми.
4. Системы реализации рациональных действий.

В жизненном цикле систем искусственного интеллекта применяются следующие подходы к их созданию:

1. Подход, основанный на когнитивном моделировании.
2. Подход, основанный на использовании законов мышления.
3. Подход, основанный на использовании Теста Тьюринга.
4. Подход, основанный на использовании модельно-аналитического интеллекта.
5. Подход, основанный на использовании интеллектуальных агентов.

Тест Тьюринга касается функционального определения систем. Согласно Тесту Тьюринга в интеллектуальную систему должны включаться:

1. Средства обработки текстов на естественных языках.
2. Средства представления знаний.
3. Средства автоматического формирования логических выводов.
4. Средства машинного обучения.
5. Средства машинного зрения для восприятия объектов.
6. Средства робототехники для манипулирования объектами и перемещения в пространстве.

При любом подходе к созданию систем искусственного интеллекта представление знаний является неотъемлемым компонентом их жизненного цикла. В жизненном цикле знаний различается множество фаз. К типовым фазам жизненного цикла знаний относится хаос знаний, поле знаний, база знаний, анализ знаний, генерация знаний, управление знаниями.

В области информатики представление знаний определяется как совокупность методов и моделей отображения знаний в памяти компьютерных систем в такой форме, которая позволяет проводить их автоматическую обработку и оперировать ими для решения задач проблемной области.

Основная часть ранних работ в области представления знаний привязывалась к языку и подпитывалась исследованиями в области лингвистики, которые, в свою очередь, основывались на результатах философского анализа языка.

В последующих исследованиях представление знаний объединилось с проведением рассуждений, которые заключаются в том, что всегда полезно иметь формальное, явное представление о мире, а также о том, как действия влияют на мир и как приобрести способность манипулировать подобными представлениями с помощью дедуктивных процессов.

При создании систем, обеспечивающих понимание естественного языка, основной акцент делается на проблемах представления и формирования рассуждений с помощью знаний, требуемых для понимания.

В области искусственного интеллекта различается несколько разновидностей представления знаний.

Эвристически адекватное представление знаний – представление, заключающееся в допустимости лингвистического выражения последовательности рассуждений, приводящих к решению задачи.

Эпистомологически адекватное представление знаний – представление, обеспечивающее соответствие модели предметной области.

Эпистомологически неполное представление знаний – описание части фактов о внешнем мире, необходимых для решения определённого класса задач.

Интенсиональное представление знаний – представление знаний проблемной области.

Экстенсиональное представление знаний – представление знаний предметной области.

IT-специалисты пользуются каждым из приведённых определений в жизненном цикле информационных систем и технологий.

По характеру описания выделяются структурные и параметрические представления знаний.

При структурном представлении знаний характеризуются отношения фактов или объектов. Структура может изменяться эволюционно или динамически.

Параметрическое представление знаний характеризуется фиксированной структурой и изменяемыми параметрами в фактах или объектах.

Знания рассматриваются в непосредственной связи с процессами. К типовым процессам относятся извлечение, структурирование, формализация и анализ знаний.

Помимо процессов, представляющих типовые процессы обработки знаний, выделяется и множество других процессов, реализуемых в жизненном цикле систем, основанных на знаниях:

– отображение;
– тиражирование;
– сопровождение;
– защита;

– обучение;
– генерация;
– развитие;
– мониторинг;
– управление;
– интеграция;
– агрегирование;
– накопление;
– прогнозирование.

Каждый из приведённых процессов представляется как:

– система событий, определяемая объектом или явлением и наблюдаемая при некоторых условиях;

или

– течение событий, происходящих в результате выполняемых действий при определённых условиях в соответствии с намеченной целью.

2 ИСТОЧНИКИ И КЛАССИФИКАЦИЯ ЗНАНИЙ

2.1 ИСТОЧНИКИ ЗНАНИЙ

Множество источников знаний считается открытым и расширяемым. Среди всевозможных источников знаний выделяется метаисточник – вселенная. К менее общим по мощности потенциальных знаний относятся окружающий мир; объекты, представители и явления окружающего мира; деятельность и результаты деятельности социума и его представителей.

В узком понимании понятия к источникам знаний относятся предметы или устройства, от которых человек может получать информацию, связанную с решением задач.

К источникам знаний относятся сжатые и полные описания сущностей. Сжатые описания оформляются в виде утверждений, эвристик, концепций, стратегий. Полные описания могут быть вербальными, математическими и алгоритмическими.

Носителями знаний являются человек, человечество, живая и неживая природа.

Рукописные материалы, печатные материалы, средства массовой информации, компьютер, базы знаний, информационные ресурсы относятся к посредникам (медиаторам) между носителем и получателем знаний и могут в определённых случаях выполнять функции носителей знаний.

Согласно Федеральному закону «Об информации, информатизации и защите информации»:

информационные ресурсы – отдельные документы и отдельные массивы документов, документы и массивы документов в информационных системах (библиотеках, архивах, фондах, банках данных, других информационных системах).

С позиций системного подхода информационный ресурс это знания, подготовленные людьми для социального использования в обществе и зафиксированные на материальном носителе.

В философии описываются две составляющие познания: чувственные (ощущения, восприятия, представления) и рациональные (мышление в многообразных формах: понятиях, суждениях, умозаключениях, гипотезах, теориях).

2.2 КЛАССИФИКАЦИЯ ЗНАНИЙ

В контексте применения знаний в обучении акцентируется внимание на априорных знаниях о мире. В большинстве случаев априорные знания представляются в виде общих логических теорий.

Системы классификаций знаний относятся к открытым системам. При классификации знаний используются следующие принципы:

– научный принцип;
– принцип формирования (построения, конструирования);
– принцип дедуктивного вывода;
– принцип двузначности;
– принцип многозначности;
– принцип полезности;
– принцип действия;
– принцип интеграции;
– принцип коллективного доступа;
– принцип концептуализации;
– принцип полной согласованности;
– принцип качества.

3 ОСНОВНЫЕ ФАЗЫ ОБРАБОТКИ ЗНАНИЙ И ИХ РЕЗУЛЬТАТЫ

При разработке систем, основанных на знаниях, различаются три фазы домашинной обработки знаний:

– извлечение знаний из источников;
– структурирование разрозненных фрагментов знаний в единую модель;
– формализация знаний.

Традиционные фазы обработки знаний постоянны и не зависят от предметной области.

В результате извлечения знаний из источников образуется хаос, который характеризуется огромным количеством гетерогенных (разнохарактерных) и даже противоречивых фрагментов знаний. Извлечённые знания хаотического характера распределяются по различным средствам и субъектам.

При выполнении второй фазы домашинной обработки формируется поле знаний.

Поле знаний характеризуется слабо формализованным представлением. На этой фазе осуществляется процесс концептуализации.

Поле знаний рассматривается как условное неформальное описание основных понятий и взаимосвязей между понятиями предметной области, выявленных из систем знаний экспертов. Это образование представляется в виде одного из способов компрессии знаний.

Компрессия знаний осуществляется с помощью ментальных, концептуальных и семантических моделей.

Большинством психологов, занимающихся психологией познания, отмечается иерархичность понятийной структуры поля знаний. В связи с этим поле знаний стратифицируется, т.е. рассматривается на различных уровнях абстракции понятий. При подобной стратификации формируется «пирамида знаний», в которой каждый следующий уровень обеспечивает переход на новую ступень обобщения и углубления знаний предметной области. Достаточно часто уровни «пирамиды знаний» ассоциируются с глубиной профессионального опыта.

Для ряда задач, например, задач принятия решений, в поле знаний различаются концептуальные и функциональные модели.

Формирование концептуальных моделей основывается на выявлении понятийной структуры предметной области. В структуру функциональной модели включаются понятия предметной области и основные функциональные связи или отношения между понятиями. С помощью функциональных связей и отношений отражаются принимаемые стратегии решения поставленных задач.

Формализация поля знаний осуществляется посредством языков представления знаний. В результате формализации поля знаний образуется база знаний, описанная на выбранном специализированном языке. После формирования базы знаний может начинаться их машинная обработка, связанная со множеством процессов функциональной спецификации систем, основанных на знаниях. Среди элементов этого множества наиболее часто организуются процессы анализа и генерации знаний.

4 КОМПОНЕНТЫ ФОРМАЛИЗАЦИИ ЗНАНИЙ

Основой формализации знаний является модель. Знание рассматривается как некоторая самостоятельная сущность. Модель сущности представляет собой систему математических зависимостей, отражающих её свойства и закономерности жизненного цикла. Основное требование к модели заключается в её адекватности реальности.

Широкая палитра свойств и закономерностей в жизненном цикле знаний предопределяет многообразие моделей представления знаний. Любая модель является некоторой абстракцией (формализмом), соответствующей некоторой парадигме.

Формализм представляется универсальным математическим аппаратом для корректного формального описания и построения процедуры решения задачи. Различаются следующие виды формализмов: функциональный, логический, алгоритмический и объектный.

Описание каждой абстракции осуществляется с помощью математических объектов и зависимостей.

В реальных условиях обилия неопределённостей осуществляется постепенное формирование модели представления знаний. Подобный подход базируется на идее постепенной формализации проблемных ситуаций с неопределённостью посредством поочерёдного использования средств методов активации интуиции и опыта специалистов и методов формализованного представления систем.

Ранние версии рассматриваемого подхода строились на базе концепции структурно-лингвистического моделирования. В последующем этот подход стал развиваться как самостоятельное направление.

Постепенная формализация обеспечивается некоторой методикой, в которой сочетаются относительно малоформальные методы, понятные для человека, и формальные, знаковые представления, позволяющие привлекать достижения математических теорий, информационных технологий и инфокоммуникационных систем. Ключевой особенностью постепенной формализации является развитие представлений. В указанном подходе предусматривается многоразовый выбор методов моделирования по мере расширения, углубления и обновления знаний.

Типовые этапы методики постепенной формализации заключаются в следующем:

1. Разрабатывается или выбирается знаковая система.
2. Выбирается подход к моделированию и определяются правила преобразования, применяемые при формировании и анализе модели.
3. Фиксируются элементы модели и связи между ними в соответствии с выбранной знаковой системой.
4. Получают новые, ранее неизвестные компоненты, взаимоотношения, зависимости и структуры.
5. Включаются полученные новые результаты в первоначальное описание, процедура преобразования повторяется до тех пор, пока не будет найдено адекватное представление знаний.

При выборе знаковых систем учитываются возможные вариации в их средствах.

На первом этапе могут использоваться: естественный язык, средства продукций, средства лингвистики, средства представления стереотипных ситуаций, средства теории множеств, средства теории вероятностей, средства теории графов, средства теории автоматов, средства теории логик, средства теории нечётких множеств, средства теории регрессионного анализа, средства теории нейронных сетей, средства теории эволюции и другие средства известных теорий.

По типу знаковых систем классифицируется система моделей представления знаний. Каждому из возможных типов знаковых систем ставится в соответствие определённый класс моделей представления знаний.

Внутри каждого типа знаковой системы выделяется ряд специфических признаков, позволяющих объединять модели по определённым особенностям выбранных средств, характерных для предусматриваемых условий проблемной области. В результате образуется система классификаций моделей представления знаний.

Достаточно часто модель представления знаний формируется с применением нескольких типов знаковых систем.

На втором этапе могут применяться правила структуризации или декомпозиции, правила композиции, правила поиска мер близости на пространстве состояний элементов.

Адекватность модели доказывается последовательно по мере её формирования путём оценки правильности отражения в каждой последующей модели компонентов и связей, необходимых для достижения поставленных целей.

При возникновении затруднений в процессе формирования модели используется переключение образного и формального мышления.

Модели знаний разделяются на реляционные, объектные и ассоциативные. В каждой разновидности имеется ряд форм, определяющих её конкретные реализации.

Примерами реляционных моделей считаются предикатная, продукционная и лингвистическая модели; примерами объектных – семантическая, фреймовая модели; примерами ассоциативных – логическая, нейросетевая модели.

5 ОПРЕДЕЛЕНИЯ МОДЕЛЕЙ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Известные модели представления знаний могут разделяться на три группы:

1) Реляционные модели знаний;
2) Объектные модели знаний;
3) Ассоциативные модели знаний.

В каждой группе существует несколько различных форм, отражающих присущие им особенности.

При реляционной модели знаний система описывается следующим кортежем:

$$\mathbf{R} = < \mathbf{T}, \mathbf{P}, \mathbf{A}, \mathbf{F} >,$$

где **R** – система отношений;
T – множество базовых элементов;
P – множество синтаксических правил, позволяющих строить из элементов множества **T** синтаксически правильные выражения;
A – множество априорно истинных выражений (аксиом);
F – семантические правила вывода, позволяющие расширить множество **A** за счёт других выражений.

В основе реляционной модели лежит теория логики.

Наиболее часто используются предикатная, продукционная и лингвистическая формы реляционной модели знаний.

Объектная модель знаний задаётся кортежем:

$$\mathbf{N} = < \mathbf{C}, \mathbf{O}, \mathbf{S}, \mathbf{I} >,$$

где **N** – сеть объектов, связанных разными отношениями;
C – множество классов объектов, связанных отношениями классов;
O – множество объектов, связанных отношениями объектов;
S – структура классов и объектов, определяющая конкретные связи между ними;
I – правила преобразования объектов и вывода на сети объектов.

В основе объектной модели может лежать теория семантических сетей или теория фреймовых сетей. Общим для этих подходов является понятие объекта знаний как основного действующего элемента сетевой системы знаний.

Ассоциативная модель знаний определяется в виде кортежа:

$$\mathbf{A} = < \mathbf{U}, \mathbf{C}, \mathbf{L}, \mathbf{I} >,$$

где **A** – ассоциативная сеть представления знаний;
U – множество узловых элементов ассоциативной сети;
C – множество коннекций элементов;
L – множество правил построения сети и определения параметров коннекций;
I – правила ассоциативного вывода (процедуры процессирования знаний).

Разные проявления ассоциативной модели вылились в ассоциативную логическую форму, нейронную форму и универсальную форму.

Универсальная форма ассоциативной модели знаний использует базис узловых элементов сети, настраиваемый из условий наилучшего отображения системы знаний.

6 МЕНТАЛЬНЫЕ МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

К ментальным моделям, как правило, относятся интеллект-карты, концептуальные графы и таблицы решений. Основным видом формализации являются графы.

Структуризация знаний начинается с формирования отображения радиантного мышления человека. Графическое выражение процесса радиантного мышления человека представляется в виде интеллект-карты. Интеллект-картами описываются связи и древовидные структуры произвольных фрагментов знаний.

Интеллект-карты характеризуются следующими признаками:

- объект изучения выделяется в центральном образе;
- основные темы, связанные с объектом изучения, расходятся от центрального образа в виде ветвей, которые не поясняются или поясняются ключевыми словами или образами;
- вторичные идеи также отражаются посредством ветвления;
- ветви формируют связанную структуру.

В настоящее время существует представительное множество программных приложений для формирования интеллект-карт (майнд мэппинг, например, Mind Manager).

Концептуальные карты формируются по результатам анализа предметной области с позиций некоторой концепции. Как правило, основные положения выбираемой концепции формулируются вербальным образом. Концептуальной картой описывается система понятий и поименованных отношений между ними, согласованных с выбранной концепцией.

Типовые этапы построения концептуальной карты заключаются в следующем:

- выбор концепции решения задачи предметной области;
- определение контекста посредством формирования конкретного фокусирующего вопроса, определяющего главную задачу;
- выделение концептов – базовых понятий данной предметной области;
- построение связей между концептами, определение отношений и взаимодействий базовых понятий;
- упорядочивание графа – уточнение, удаление лишних связей, снятие противоречий.

Ниже приводится матричная форма описания концептуальной модели взаимодействия двух открытых систем при условии, что первая открытая система является источником информации, а вторая приёмником информации.

Матрица I_1 описывает интерфейсы источника, матрица I_2 – интерфейсы приёмника, матрица c_v – виртуальные связи между системами, а матрица c_p – физические связи между системами.

Матрица I_1

$$\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Матрица I_2

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}.$$

Матрица c_v

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матрица c_p

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

В случае, когда первая система становится приёмником, а вторая система – источником, то матричная форма описания взаимодействия двух открытых систем примет иной вид.

Матрица I_1

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}.$$

Матрица I_2

$$\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Матрица c_v

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

Матрица c_p

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

7 ИНСТРУМЕНТАЛЬНЫЕ СИСТЕМЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Инструментальные системы представления знаний обеспечивают формирование среды для построения соответствующих моделей. Ключевые особенности формирования среды раскрываются на примере приложения Cmap Tools, признанного IT-специалистами в качестве лидера в области представления знаний. Cmap Tools приложение предназначается для создания и изучения концептуальных карт. Этим определяется расширяемость его сферы использования различными организациями и частными лицами.

Основным назначением приложения CmapTools является стимулирование сотрудничества и обмена мнениями при создании моделей знаний, базирующихся на концепт-картах. Работать с программной очень удобно: каждый новый бокс генерируется простым щелчком мыши в любом месте поля, любой бокс можно произвольно перетаскивать по рабочему полю.

Программной средой CmapTools обеспечивается пользователям возможность выявлять глубинную структуру знаний.

Далее представляются действия по инсталляции приложения.

Приложение находится в глобальной сети Internet Cmap Tools и осуществляется переход по первой ссылке (рис. 7.1).

Поиск в сети Internet Cmap Tools

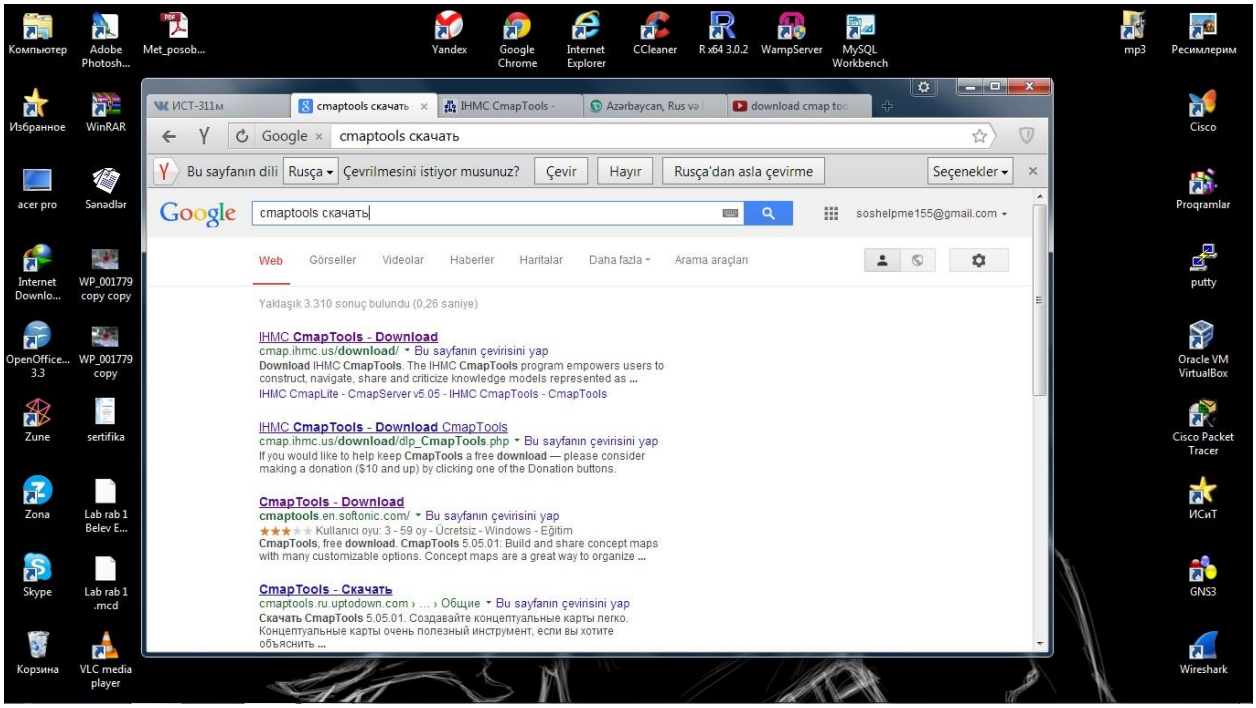


Рис 7.1

Затем необходимо познакомиться с программой и зарегистрироваться на сайте. Последовательно указывается email, имя, фамилия, страна, принадлежность к определенной организации/компании, также вводится название данной организации

и ее веб-сайт (рис. 7.2). Затем требуется указать источник знаний о программном средстве Cmap Tools и выбрать из списка используемую платформу. После окончания ввода данных нажимается кнопка «Submit», и выполняется переход к странице выбора средств инструментальной среды.

Процесс регистрации на сайте cmap.ihmc.su

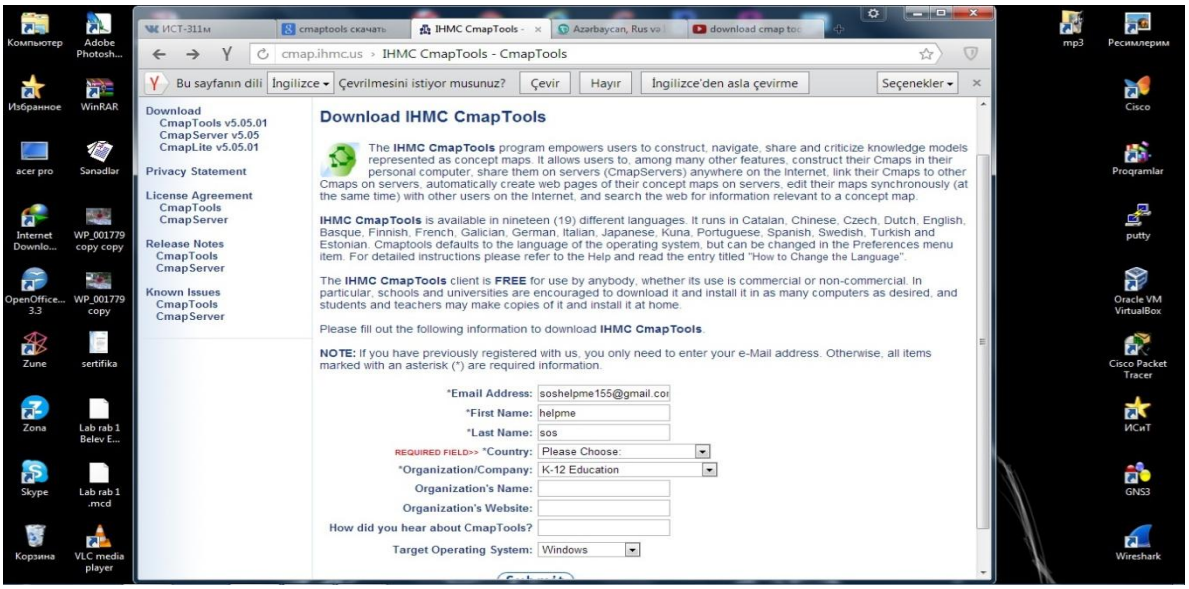


Рис. 7.2

Затем предоставляется выбор между четырьмя платформами: Windows, Mac OS X, Linux и Solaris. Выбирается платформа Windows и начинается скачивание программы на персональное средство информации (рис. 7.3, рис. 7.4).

Выбор между четырьмя платформами

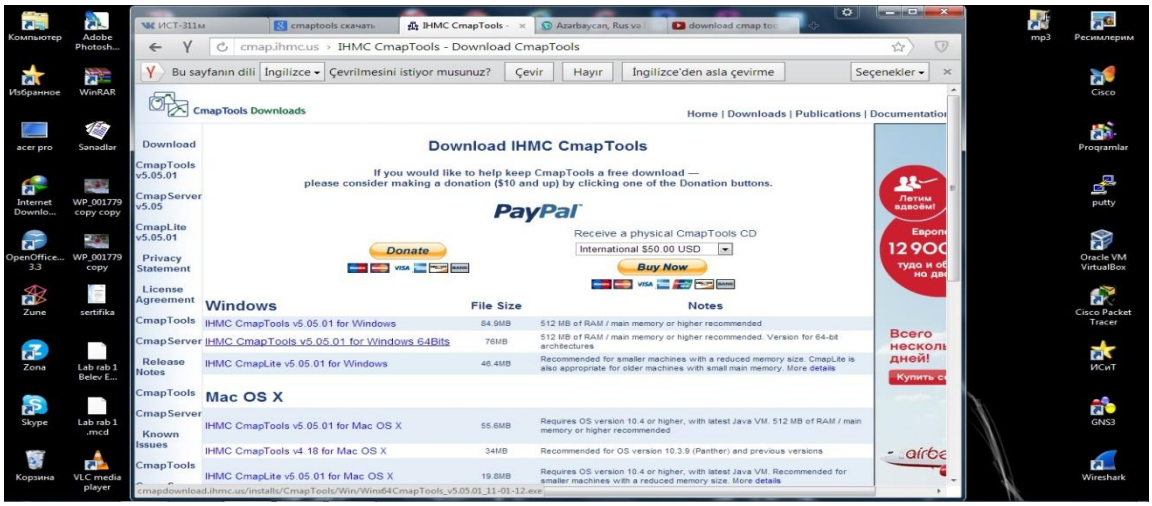


Рис. 7.3

Процесс скачивания программы

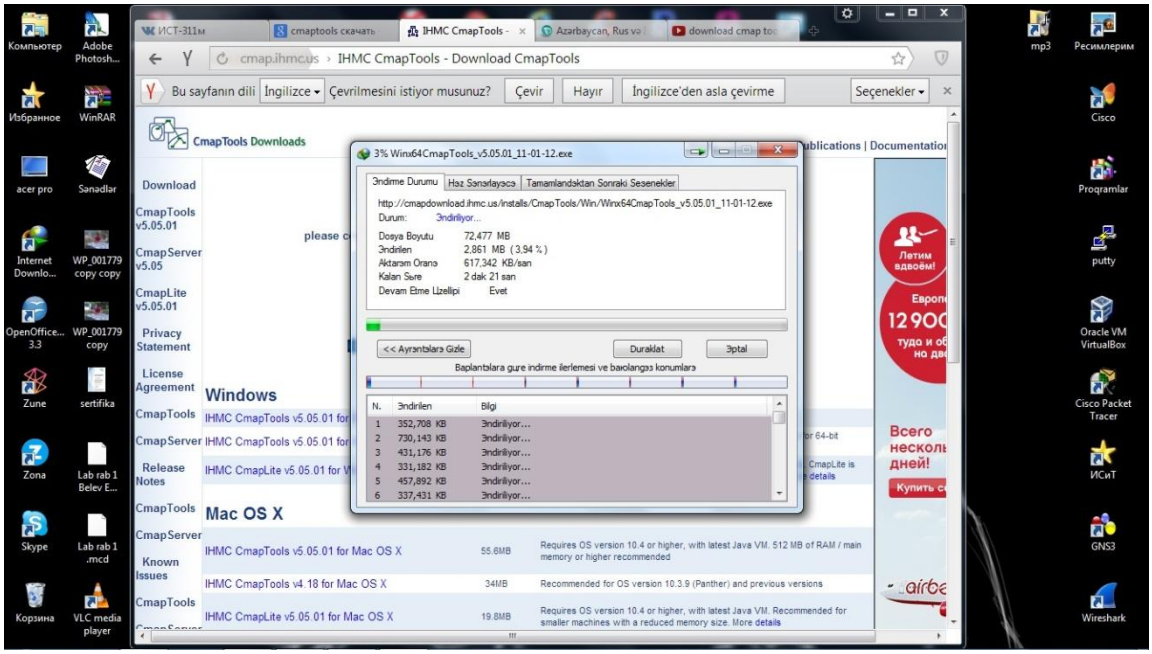


Рис. 7.4

После выполненных операций открывается окно «Введение» (рис. 7.5).

Приветствие программы

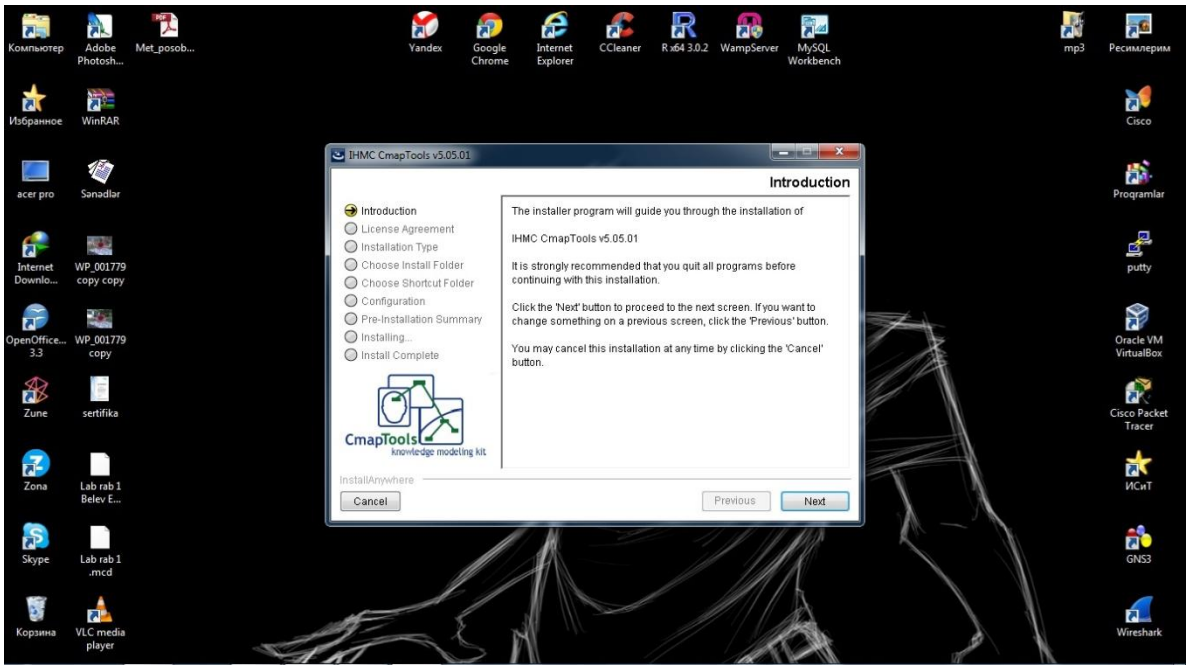


Рис. 7.5

После прочтения, и перехода на следующий этап инсталляции, открывается окно «Лицензионное соглашение» (рис. 7.6). Для продолжения инсталляции соглашаемся и нажимаем кнопку «next».

Лицензионное соглашение

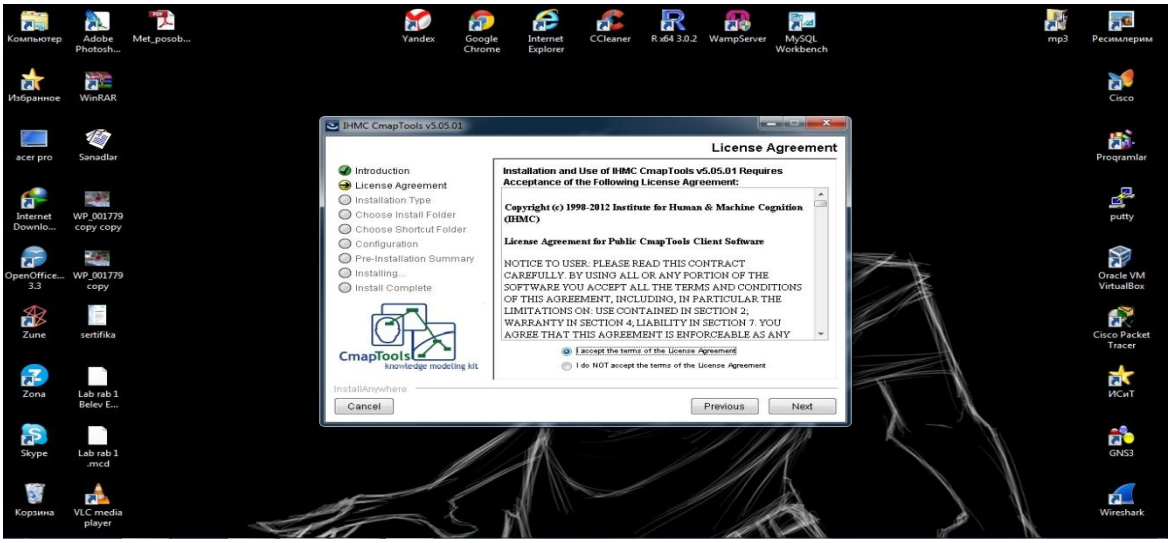


Рис. 7.6

Третье окно связано с типом инсталляции (рис. 7.7). Из предложенных двух типов выбираем наиболее подходящий первый тип, т.е типичная конфигурация (для пользователей с персональными компьютерами).

Тип инсталляции

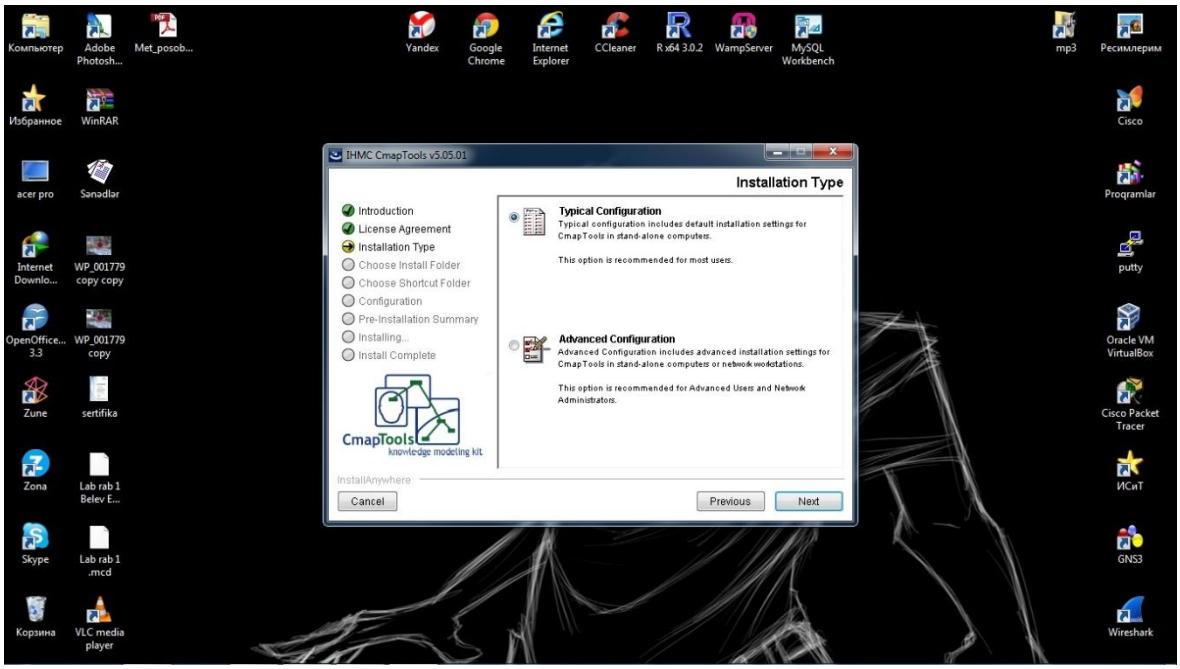


Рис. 7.7

Четвертое окно представляет процедуру выбора папки инсталляции (рис. 7.8). Оставляем папку по умолчанию.

Выбираем папку инсталляции

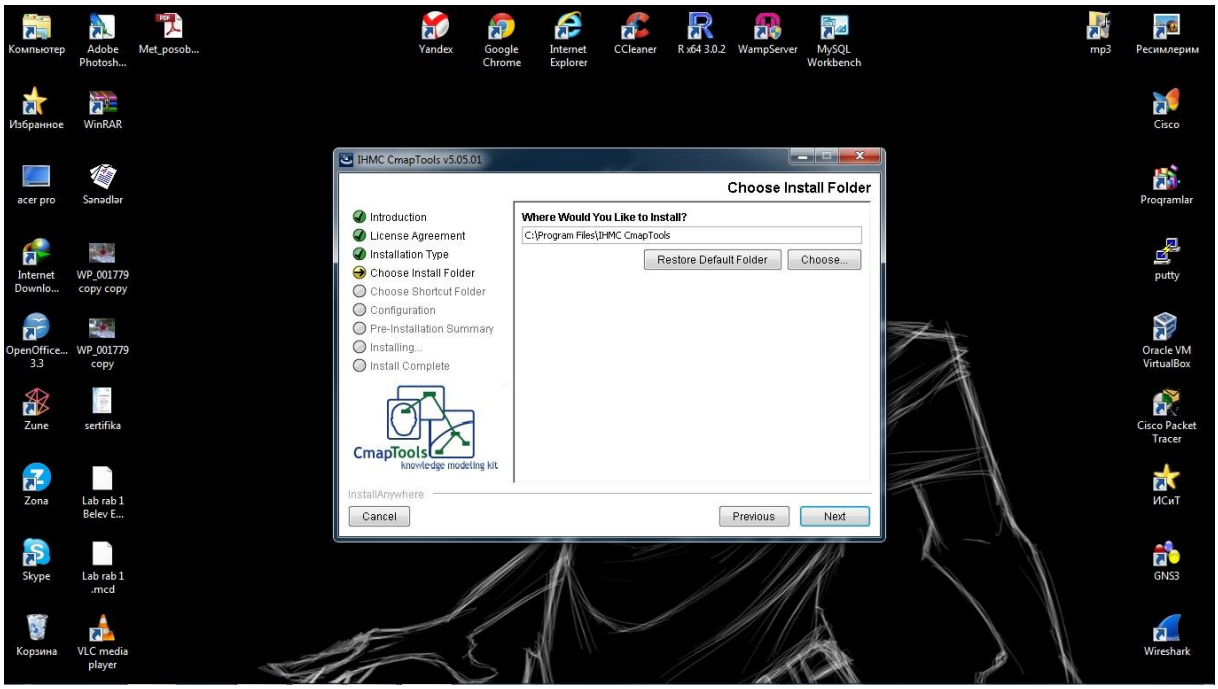


Рис. 7.8

В пятом окне предлагается выбрать ярлык или сокращенный путь инсталляции (рис.7.9).

Выбор сокращенного пути инсталляции

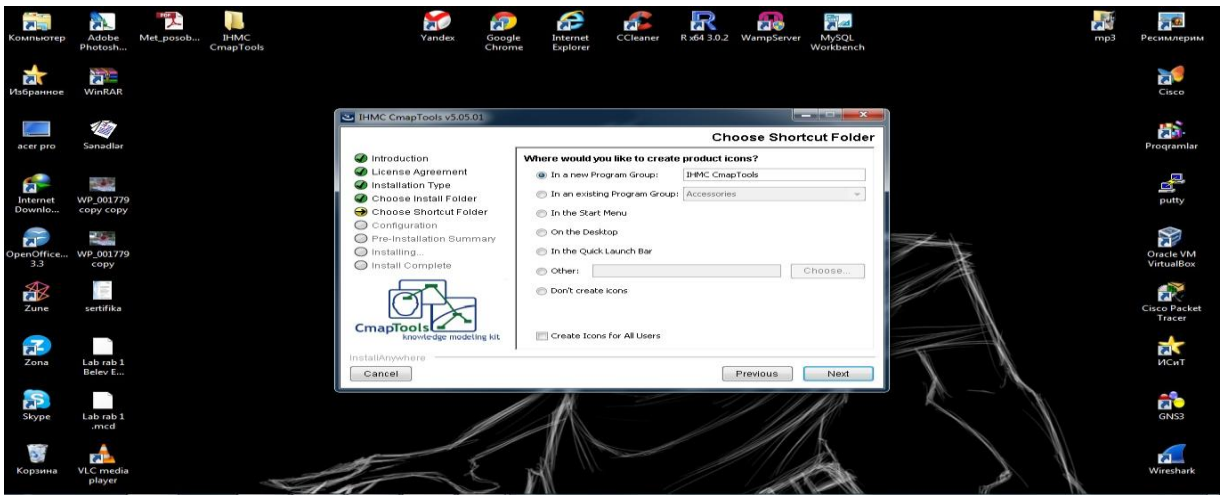


Рис. 7.9

Далее открывается окно предустановочной информации, которая предоставляет возможность проверки правильности введения данных (рис. 7.10).

Предустановочная информация

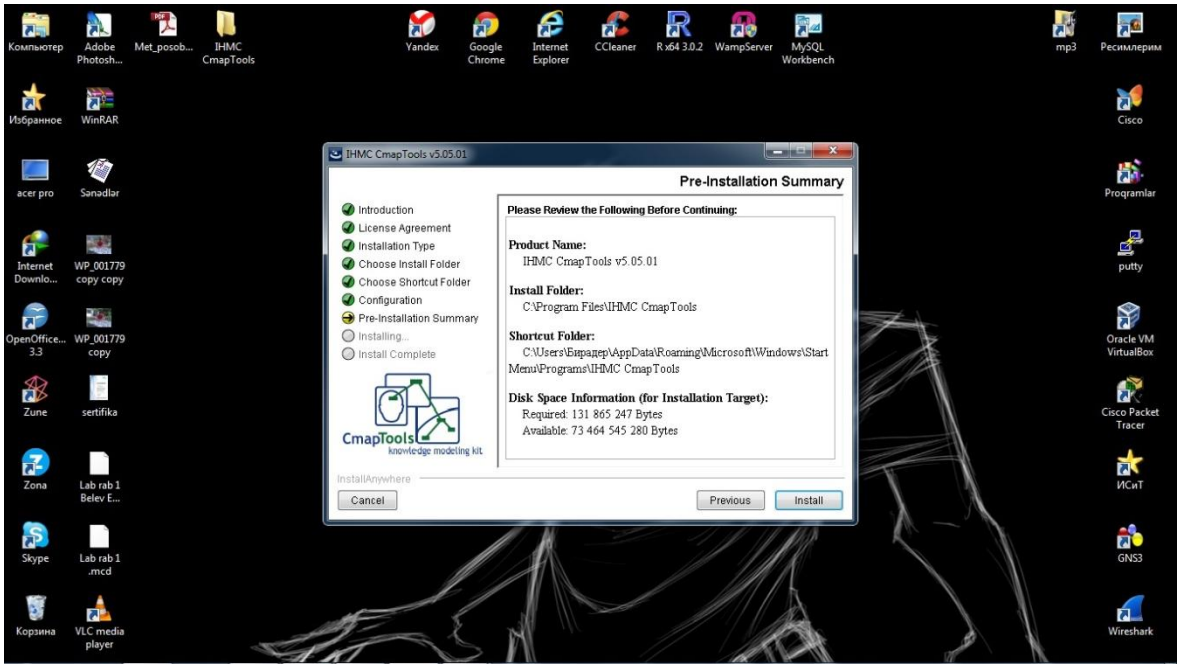


Рис. 7.10

Далее открывается окно установки (рис. 7.11).

Установка

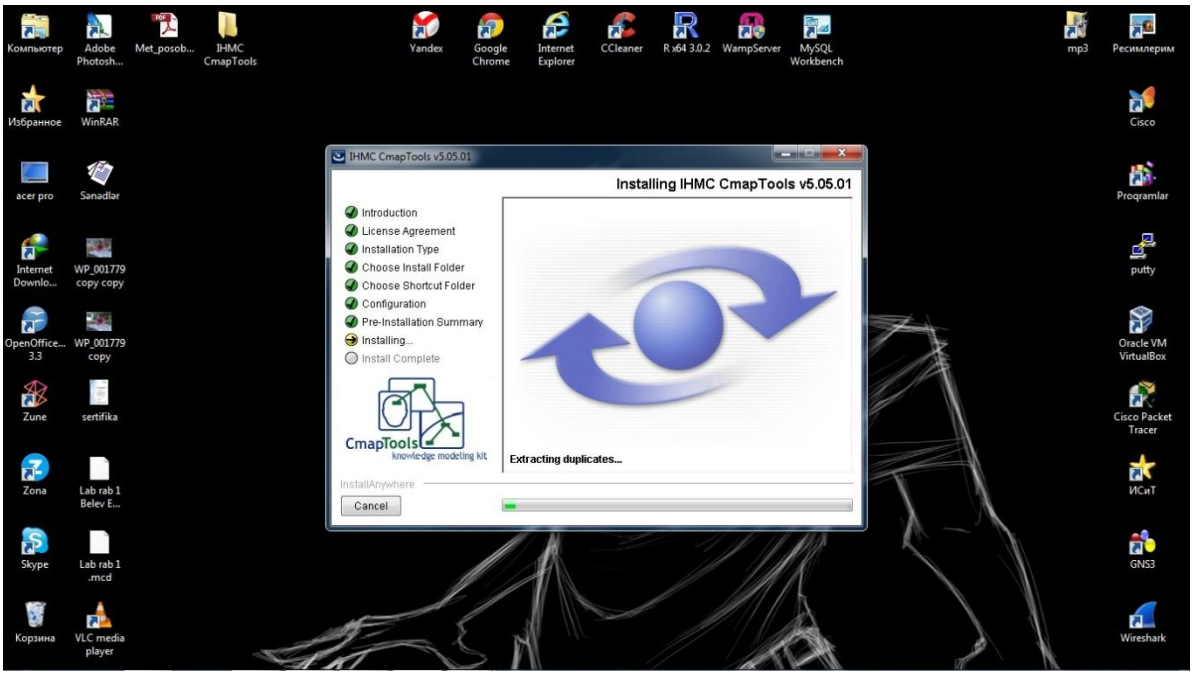


Рис. 7.11

Далее в предпоследнем окне выводится список языков, поддерживаемых программным продуктом. Выбирается нужный язык (рис. 7.12).

Выбор языка

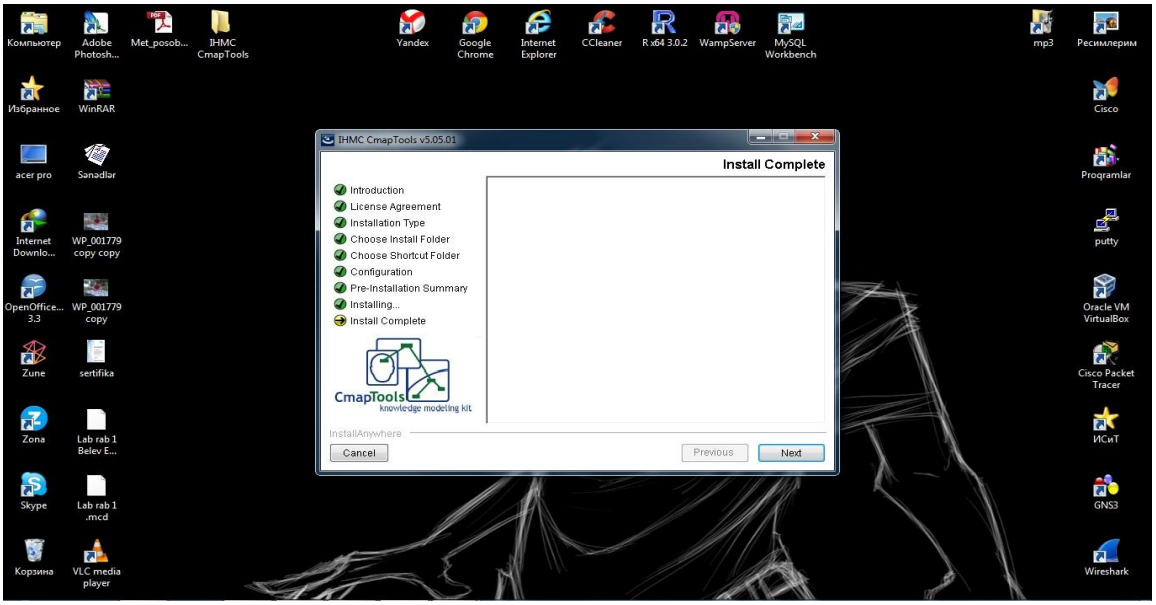


Рис. 7.12

Последнее окно связано с завершением установки программы (рис. 7.13). Успешное завершение сопровождается выводом соответствующего сообщения с указанием расположения файла для запуска и дальнейшей работы.

Завершение установки программы

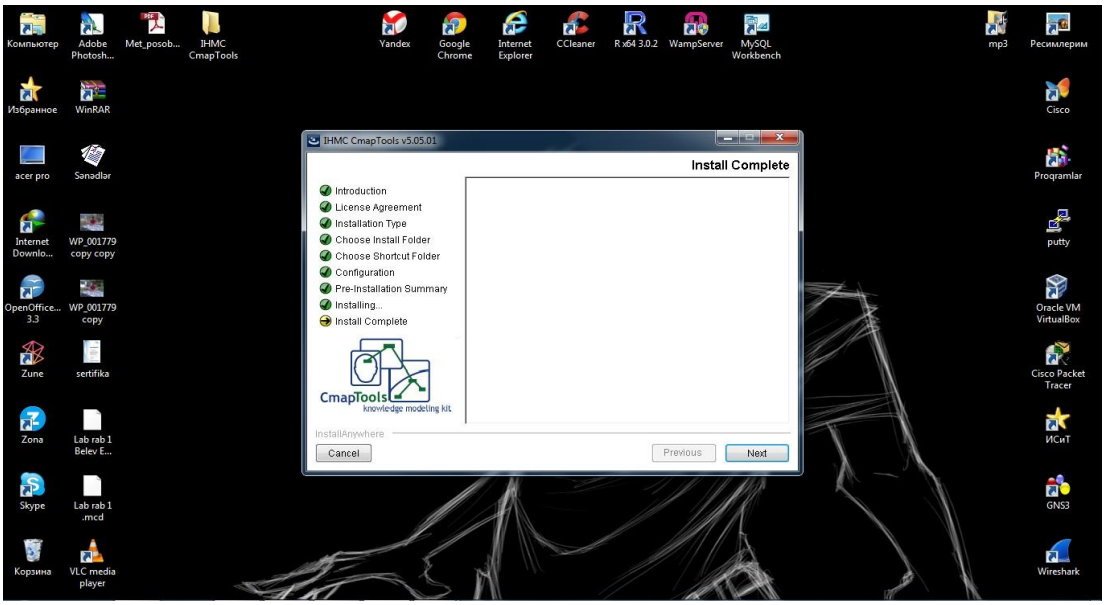


Рис. 7.13

В результате представленных действий приложение Cmap Tools включается в состав программ и может быть использовано для разработки концептуальных моделей.

8 ФОРМИРОВАНИЕ, АНАЛИЗ И ГЕНЕРАЦИЯ ЗНАНИЙ НА ОСНОВЕ МЕТОДОВ ПРЕОБРАЗОВАНИЯ ФУНКЦИОНАЛЬНЫХ МОДЕЛЕЙ

В данном разделе раскрываются формализации, обеспечивающие создание системы генерации знаний о качестве управления потоком запросов к серверной инфраструктуре посредством развития его принципиальной основы. Исходным базисом для развития является концепция автоматизации управления. Концепцией охватываются одноуровневое и двухуровневое управление. Системный контекст выражается в определении динамических характеристик автоматизации управления. Образуемая система генерации знаний становится интеллектуальным ядром управления потоком запросов к серверной инфраструктуре при соблюдении гарантий к качеству её функционирования.

8.1 КОНЦЕПТУАЛЬНАЯ ОСНОВА УПРАВЛЕНИЯ ПОТОКОМ ЗАПРОСОВ К СЕРВЕРНОЙ ИНФРАСТРУКТУРЕ

Концептуальная основа автоматизации управления потоком запросов к серверной инфраструктуре образуется совокупностью следующих принципов:

1. Последовательность запросов к серверной системе является входящим потоком заявок с переменной интенсивностью, которая может как увеличиваться, так и уменьшаться.
2. Во входящем потоке могут содержаться обращения к серверной системе, которые могут размещаться в буферной памяти до принятия решения о выборе конкретного сервера для их обслуживания.
3. Обработка запросов в серверной системе может выполняться с их размещением в буферной памяти и без её использования.
4. Последовательность ответов серверной системы является выходящим потоком с переменной интенсивностью, которая может как увеличиваться, так и уменьшаться за счет вариаций с буферной памятью.
5. Формирование ответа на запрос в серверной системе с использованием буферной памяти характеризуется большей интенсивностью, чем без её применения.
6. Запросы к серверной системе накапливаются в буферной памяти ограниченного объема.
7. Решение относительно использования буферной памяти и альтернативное решение принимаются на основании её заполненности.

Приведённая совокупность принципов является исходным базисом для построения ментальных моделей представления знаний об управлении потоком запросов к серверной инфраструктуре.

В соответствии с концептуальной основой рассматриваются два способа автоматизации управления потоком запросов. Первый способ базируется на одноуровневом управлении, а второй – на двухуровневом управлении.

В целях формализации анализа предлагаемых способов автоматизации управления потоком запросов в дополнение к представленной концептуальной основе выдвигается гипотеза о конкретных статистических свойствах входящего и выходящего потоков серверной инфраструктуры. В рассматриваемом варианте

формализации принимается гипотеза о пуассоновских потоках серверной инфраструктуры.

В случае одноуровневого управления признак использования буферной памяти при обработке запросов изменяется по следующей схеме:

R=0 при $0 \le n \le L$ (работа без буферной памяти),

(8.1.1)

R=1 при $n > L$ (работа с буферной памятью),

где n – число запросов в серверной инфраструктуре;
L – пороговое ограничение.

С учётом четвертого положения концептуальной основы и принятой гипотезы интенсивность выходящего потока серверной инфраструктуры при одноуровневом управлении характеризуется описанием вида:

$\mu = \mu_1$ при $0 \le n \le L$ (работа без буферной памяти),

(8.1.2)

$\mu = \mu_2$ при $n > L$ (работа с буферной памятью).

Сущность одноуровневого управления поясняется на рис. 8.1.1.

Взаимосвязь интенсивности выходящего потока и числа запросов серверной инфраструктуры при одноуровневом управлении

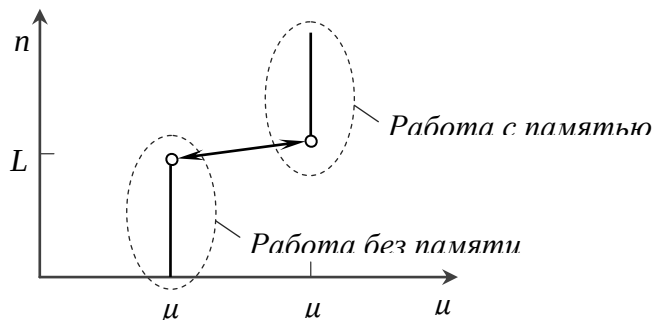


Рис. 8.1.1

При двухуровневом управлении признак использования буферной памяти при обработке запросов формируется по ниже приводимой схеме:

R=0 при $0 \le n \le L_2 - 1$ или при $0 \le n \le L_1$ (работа без буферной памяти),

R=1 при $n \ge L_2$ или при $L_1 < n < L_2$ (работа с буферной памятью), (8.1.3)

где $L_1 < L_2$.

С учётом четвертого положения концептуальной основы и принятой гипотезы интенсивность выходящего потока при двухуровневом (гистерезисном) управлении характеризуется описанием вида:

$\mu = \mu_1$ при $0 \le n \le L_2 - 1$ или при $0 \le n \le L_1$ (работа без буферной памяти),

(8.1.4)

$\mu = \mu_2$ при $n \ge L_2$ или при $L_1 < n < L_2$ (работа с буферной памятью),

Сущность гистерезисного управления потоком запросов поясняется на рис. 8.1.2.

Взаимосвязь интенсивности выходящего потока и числа запросов серверной инфраструктуры при двухуровневом управлении

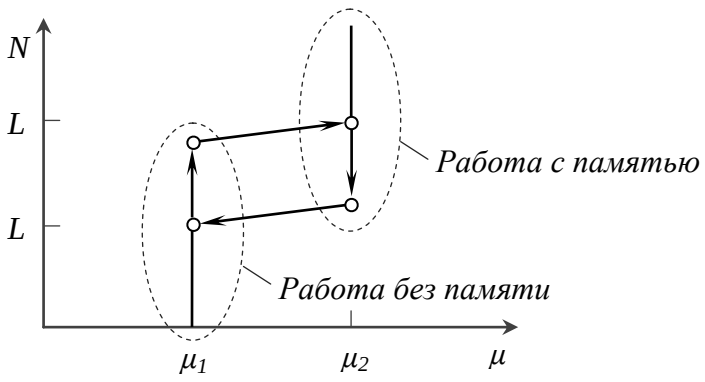


Рис. 8.1.2

Рассмотренная детализация концептуальной основы ориентируется на расширение ментальных моделей представления знаний об управлении потоком запросов к серверной инфраструктуры.

При функционировании серверной инфраструктуры выделяются два направления интересов.

Первое направление касается клиентов, отправляющих запросы к серверной системе, а второе направление – корпорации, в инфраструктуру которой включается серверная система. В связи с этим анализ качества должен ориентироваться на отражение двух направлений интересов.

Согласно концептуальной основе автоматизации управления механизмом отражения атак для анализа интересов корпорации выбирается среднее количество запросов в серверной системе, среднее количество запросов и среднее время простаивания в буферной памяти, а интересов клиентов – среднее время обработки запросов.

8.2. ГЕНЕРАЦИЯ ЗНАНИЙ О КАЧЕСТВЕ ОДНОУРОВНЕВОГО УПРАВЛЕНИЯ ПОТОКОМ ЗАПРОСОВ К СЕРВЕРНОЙ ИНФРАСТРУКТУРЕ

Пусть при одноуровневом управлении интенсивность потока запросов клиентов описывается соотношением (8.2.1):

$$\lambda=\lambda_1 \text{ при } 0 \le n < L, \qquad \lambda=\lambda_2 \text{ при } n \ge L. \tag{8.2.1}$$

В случае одноуровневого управления работа системы характеризуется параметром L, интенсивностью входящего потока λ и интенсивностями выходящего потока для двух режимов работы μ_1 и μ_2.

В режиме работы без буферной памяти при обработке запроса осуществляется обращение к серверной системе и клиенту отправляется ответ, содержащий актуальную информацию.

В режиме работы с памятью при обработке запроса в первую очередь выполняется поиск запрашиваемой информации в памяти.

При обслуживании повторного запроса клиенту отправляется информация из памяти, а при поступлении первичного запроса производится обращение к серверу и клиенту передается ответ, содержащий актуальную информацию.

При передаче ответа соответствующая информация сохраняется в памяти. Поскольку временные затраты на получение информации из памяти значительно меньшие по сравнению с затратами на обращение к серверу, то $\mu_1 < \mu_2$.

Переход из режима без памяти в альтернативный режим выполняется тогда, когда количество запросов, ожидающих обработки либо обрабатываемых в данный момент, превышает пороговое значение L.

Обратный переход в режим без памяти осуществляется в том случае, если длина очереди в буферной памяти вновь уменьшится до значения L.

Одноуровневое управление моделируется одноканальной системой массового обслуживания с источником бесконечного числа заявок и переменными интенсивностями.

Состояние системы определяется числом заявок , находящихся в системе n=0,1,2,…, L, (L+1),….

Граф переходов между состояниями при одноуровневом управлении изображается на рис. 8.2.1.

Граф переходов между состояниями системы при одноуровневом управлении

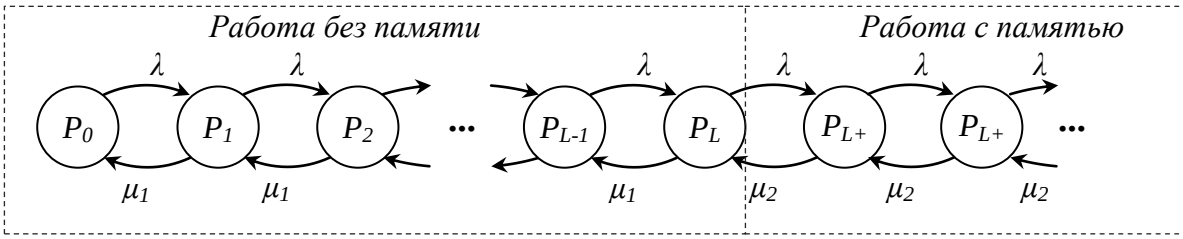


Рис. 8.2.1

После составления и решения системы алгебраических уравнений для определения n=0,1,2,... вероятностей возможных состояний, описывающих механизм отражения атаки при одноуровневом управлении в установившемся режиме при нормирующем условии (8.2.2), представляющем ситуацию полной группы несовместных событий,

$$\sum_{n=0}^{\infty} P_n = 1. \qquad (8.2.2)$$

находятся следующие соотношения

$$P_0 = \frac{(1-\rho_1)(1-\rho_2)}{1-\rho_2-\rho_1^L(\rho_1-\rho_2)}, \qquad (8.2.3)$$

$$P_n = \rho_1^n P_0 \quad при \quad 0 \le n \le L, \qquad (8.2.4)$$

$$P_n = \rho_1^L \rho_2^{n-L} P_0 \quad при \ n > L, \qquad (8.2.5)$$

где $\rho_1 = \lambda/\mu_1$ и $\rho_2 = \lambda/\mu_2$.

На основании (8.2.3), (8.2.4), (8.2.5) определяется производящая функция от стационарного распределения количества запросов в системе при одноуровневом управлении

$$\mathcal{P}_N(z) = \sum_{n=0}^{\infty} P_n z^n = P_0 \left\{ \frac{1-(\rho_1 z)^L}{1-\rho_1 z} + \frac{(\rho_1 z)^L}{1-\rho_2 z} \right\}. \qquad (8.2.6)$$

В соответствии с (8.2.6) среднее количество запросов выражается по формуле (8.2.7)

$$\bar{N} = \left.\frac{d\mathcal{P}_N(z)}{dz}\right|_{z=1} =$$

$$= P_0\left\{\frac{\rho_1}{(1-\rho_1)^2} - \frac{\rho_1^L(\rho_1-\rho_2)}{(1-\rho_1)(1-\rho_2)}\left[L + \frac{1-\rho_1\rho_2}{(1-\rho_1)(1-\rho_2)}\right]\right\}. \qquad (8.2.7)$$

Поскольку из всех пребывающих в системе запросов в любой момент времени один и только один запрос находится на обработке, то производящая функция от стационарного распределения количества запросов, расположенных в буферной памяти при одноуровневом управлении, представляется формулой (8.2.8)

$$\mathcal{P}_Q(z) = P_0 + P_1 + \sum_{n=2}^{\infty} P_n z^{n-1} = P_0\left(1 - \frac{1}{z}\right) + \frac{\mathcal{P}_N(z)}{z}. \qquad (8.2.8)$$

После подстановки в (8.2.8) образуется выражение (8.2.9)

$$\mathcal{P}_Q(z) = P_0\left\{1 - \frac{1}{z} + \frac{1-(\rho_1 z)^L}{z-\rho_1 z^2} + \frac{(\rho_1 z)^L}{z-\rho_2 z^2}\right\}. \qquad (8.2.9)$$

Тогда среднее количество ожидающих обработки в буферной памяти запросов при одноуровневом управлении определяется соотношением (8.2.10)

$$\bar{Q} = \left.\frac{d\mathcal{P}_Q(z)}{dz}\right|_{z=1} =$$

$$= P_0\left\{1 + \frac{2\rho_1 - 1}{(\rho_1 - 1)^2} + \rho_1^L\left[\frac{\rho_2}{(\rho_2 - 1)^2} - \frac{\rho_1}{(\rho_1 - 1)^2} + \frac{(L-1)(\rho_2-\rho_1)}{(\rho_1 - 1)(\rho_2 - 1)}\right]\right\}. \qquad (8.2.10)$$

Согласно вышеприведённым выражениям и формулам Литтла выводятся представления для среднего времени простаивания в буферной памяти (8.2.11) и среднего времени обработки запроса в системе (8.2.12)

$$\bar{W} = \frac{P_0}{\lambda}\left\{1 + \frac{2\rho_1 - 1}{(\rho_1 - 1)^2} + \rho_1^L\left[\frac{\rho_2}{(\rho_2 - 1)^2} - \frac{\rho_1}{(\rho_1 - 1)^2} + \frac{(L-1)(\rho_2 - \rho_1)}{(\rho_1 - 1)(\rho_2 - 1)}\right]\right\}, \qquad (8.2.11)$$

$$\bar{S} = \bar{T} - \bar{W} = \frac{P_0}{\lambda}\left\{\frac{\rho_1}{(1-\rho_1)^2} - \frac{\rho_1^2}{(1-\rho_1)^2} - \frac{\rho_1^L(\rho_1 - \rho_2)}{(1-\rho_1)(1-\rho_2)}\left[L + \frac{1 - \rho_1\rho_2}{(1-\rho_1)(1-\rho_2)}\right]\right.$$
$$\left. - \rho_1^L\left[\frac{\rho_2}{(1-\rho_2)^2} - \frac{\rho_1}{(1-\rho_1)^2} + \frac{(L-1)(\rho_2 - \rho_1)}{(1-\rho_1)(1-\rho_2)}\right]\right\}.$$

(8.2.12)

Таким образом, качество одноуровневого управления отражается системой аналитических выражений.

Перейдем к формированию математического обеспечения системы двухуровневого (гистерезисного) управления потоком запросов к серверной инфраструктуре.

8.3. ГЕНЕРАЦИЯ ЗНАНИЙ О КАЧЕСТВЕ ДВУХУРОВНЕВОГО УПРАВЛЕНИЯ ПОТОКОМ ЗАПРОСОВ К СЕРВЕРНОЙ ИНФРАСТРУКТУРЕ

В случае гистерезисного управления работа системы определяется параметрами L_1 и L_2, $0 < L_1 < L_2$, а также интенсивностью потока запросов λ и интенсивностями потока ответов сервера для двух режимов работы μ_1 и μ_2. Переход из режима работы без памяти в альтернативный режим осуществляется тогда, когда количество запросов достигает значения L_2. Обратный переход в режим работы без памяти выполняется тогда, когда длина очереди в буферной памяти уменьшается до значения L_1.

Пусть $\mu = \mu_1$ $\mu_2 = k\mu$.

Обозначим $\rho = \lambda/\mu$, $\rho_1 = \lambda/k\mu$.

При гистерезисном управлении состояния системы определяются количеством находящихся в ней запросов и режимом работы (с памятью или без памяти).

Модель поведения системы отражается графом с рис. 20.

При работе без памяти каждому состоянию ставится в соответствие число *n*, если $0 \le n < (L_1 + q),\ q = L_2 - L_1 - 1$.

При работе с памятью каждому состоянию при $n \ge (L_1 + 1)$ ставится в соответствие число $n + q,\ q = L_2 - L_1 - 1$. .

Согласно модели с рис. 8.3.1

$$P_n = \begin{cases} P_n^{'} & при\ \ 0 \le n \le L_1, \\ P_n^{'} + P_{n+q}^{'} & при\ \ L_1 < n < L_2, \\ P_{n+q}^{'} & при\ \ n \ge L_2. \end{cases} \qquad (8.3.1)$$

После составления и решения системы алгебраических уравнений для определения $P_j^{'},\ j = 0,1,2,\ldots$ вероятностей возможных состояний, описывающих механизм гистерезисного управления в установившемся режиме при нормирующем условии (18), представляющем ситуацию полной группы несовместных событий

$$\sum_{j=0}^{\infty} P_j^{'} = 1, \qquad (8.3.2)$$

находятся выражения вида

$$\frac{1}{P_0^{'}} = \frac{1}{1-\rho} - \frac{(q+1)\rho^{L_1+q}(\rho-\rho_1)}{(1-\rho^{q+1})(1-\rho_1)}, \qquad (8.3.3)$$

$$P_n = \rho^n P_0, \quad 0 \le n \le L_1;$$

$$P_{L_1+i} = \frac{\rho^{L_1}(1-\rho)}{1-\rho^{q+1}} P_0 \left[\frac{\rho^i - \rho^{q+1}}{1-\rho} + \frac{\rho^q \rho_1 (1-\rho_1^i)}{1-\rho_1} \right], \qquad 1 \le i \le q; \tag{8.3.4}$$

$$P_{L_1+q+i} = \frac{(1-\rho_1^{q+1})(1-\rho)\rho^{L_1+q}}{(1-\rho^{q+1})(1-\rho_1)} \rho_1^i P_0, \qquad i \ge 1.$$

Граф переходов между состояниями системы при гистерезисном управлении

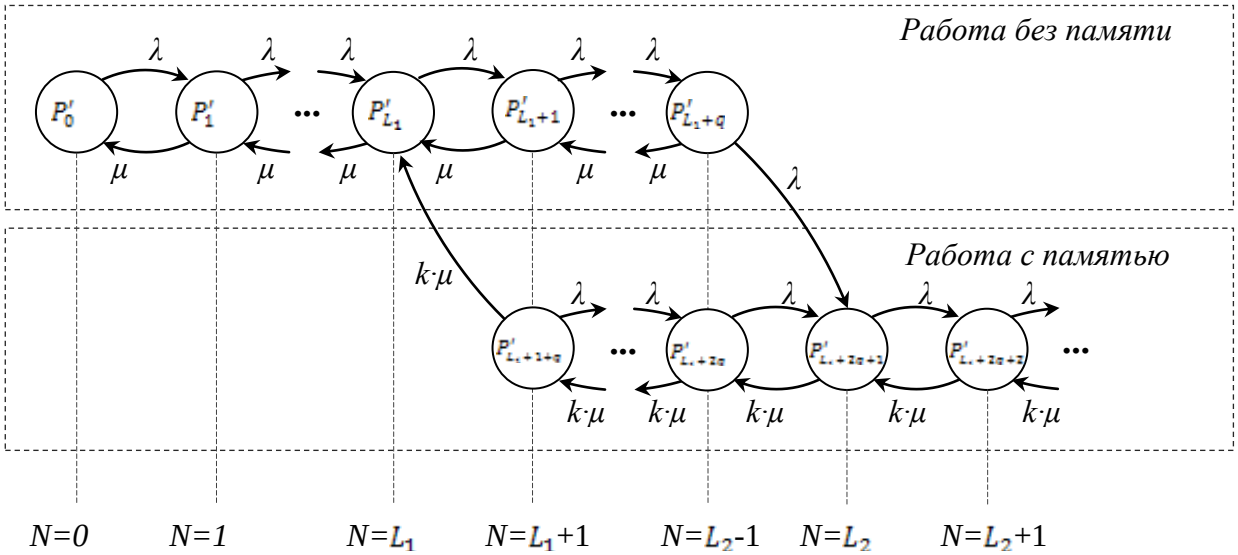


Рис. 8.3.1

На основании вышеприведённых соотношений определяется производящая функция от стационарного распределения количества запросов в системе при гистерезисном управлении

$$\mathcal{P}(z) = \mathcal{P}_1(z) + \mathcal{P}_2(z), \tag{8.3.5}$$

где

$$\mathcal{P}_1(z) = \frac{P_0}{1-\rho z} \left[1 - \frac{\rho^q (\rho z)^{L_1+1}(1-\rho)(1-z^{q+1})}{(1-\rho^{q+1})(1-z)} \right], \tag{8.3.6}$$

$$\mathcal{P}_2(z) = P_0 \left[\frac{\rho^q (\rho z)^{L_1}(\rho_1 z)(1-\rho)(1-z^{q+1})}{(1-\rho^{q+1})(1-\rho_1 z)(1-z)} \right]. \tag{8.3.7}$$

В соответствии с полученными выражениями формируется аналитическое соотношение для определения среднего количества запросов, находящихся в системе при гистерезисном управлении

$$\bar{N} = \left.\frac{d\mathcal{P}(z)}{dz}\right|_{z=1} = P_0\left\{\frac{\rho}{(1-\rho)^2} - \frac{(q+1)\rho^{L_1+q}(\rho-\rho_1)}{(1-\rho^{q+1})(1-\rho_1)}\left[\frac{2L_1+q}{2} + \frac{1-\rho\rho_1}{(1-\rho)(1-\rho_1)}\right]\right\}. \quad (8.3.8)$$

Поскольку из всех пребывающих в системе запросов в любой момент времени один и только один запрос находится на обработке, то производящая функция от стационарного распределения количества запросов, расположенных в буферной памяти при гистерезисном управлении, представляется формулой (8.3.9)

$$\mathcal{P}_Q(z) = P_0 + P_1 + \sum_{n=2}^{\infty} P_n z^{n-1} = P_0\left(1-\frac{1}{z}\right) + \frac{\mathcal{P}_1(z)}{z} + \frac{\mathcal{P}_2(z)}{z}. \quad (8.3.9)$$

Согласно представленным соотношениям получается следующее выражение для производящей функции при гистерезисном управлении

$$\mathcal{P}_Q(z) = P_0\left\{1 - \frac{1}{z} + \frac{\rho^q(\rho z)^{L_1}\rho_1(1-\rho)(1-z^{q+1})}{(1-\rho^{q+1})(1-\rho_1 z)(1-z)} + \frac{1}{z-\rho}\left[1 - \frac{\rho^q(\rho z)^{L_1+1}(1-\rho)(1-z^{q+1})}{(1-\rho^{q+1})(1-z)}\right]\right\},$$

(8.3.10)

С учётом (8.3.10) среднее количество ожидающих обработки в буферной памяти запросов при гистерезисном управлении определяется соотношением

$$\begin{aligned}\bar{Q} = P_0\Bigg\{&\frac{\rho-(L_1+1)\rho^{L_1+1}+L_1\rho^{L_1+2}}{(1-\rho)^2} - \frac{\rho-\rho^{L_1+1}}{1-\rho} \\ &+ \frac{\rho^{L_1}(1-\rho)}{1-\rho^{q+1}}\Bigg[(L_1-1)\left(\frac{\rho-\rho^{q+1}}{(1-\rho)^2} - \frac{q\rho^{q+1}}{1-\rho} + \frac{\rho^q\rho_1}{1-\rho_1}\left(q - \frac{\rho_1-{\rho_1}^{q+1}}{1-\rho_1}\right)\right) \\ &+ \Bigg(\frac{\rho-(q+1)\rho^{q+1}+q\rho^{q+2}}{(1-\rho)^3} - \frac{\rho^{q+1}q(q+1)}{2(1-\rho)} \\ &+ \frac{\rho^q\rho_1}{1-\rho_1}\left(\frac{q(q+1)}{2} - \frac{\rho_1-(q+1){\rho_1}^{q+1}+q{\rho_1}^{q+2}}{(1-\rho_1)^2}\right)\Bigg)\Bigg] \\ &+ \frac{(1-\rho_1^{q+1})(1-\rho)\rho^{L_1+q}}{(1-\rho^{q+1})(1-\rho_1)}\left(\frac{\rho_1(L_1+q-1)}{1-\rho_1} + \frac{\rho_1}{(1-\rho_1)^2}\right)\Bigg\}.\end{aligned}$$

(8.3.11)

По формуле Литтла выражается среднее время обработки запроса при гистерезисном управлении

$$\bar{S} = \frac{\bar{N} - \bar{Q}}{\lambda}. \qquad (8.3.12)$$

Выведенные соотношения образуют аналитическое ядро системы генерации знаний о качестве автоматизации управления потоком запросов к серверной инфраструктуре.

8.4 ИЗВЛЕЧЕНИЕ ЗНАНИЙ О КАЧЕСТВЕ УПРАВЛЕНИЯ ПОТОКОМ ЗАПРОСОВ К СЕРВЕРНОЙ ИНФРАСТРУКТУРЕ

На основе сгенерированных знаний в форме аналитического ядра проанализируем поведение показателей качества управления потоком запросов к серверной инфраструктуре.

В табл. 8.4.1 приводятся параметры исследуемых систем. Показатели качества оценивались с использованием среды Matlab. Оценка проводилась с помощью следующих функций.

Функция `odnourN(lam, m_1, m_2, L)` вычисляет значение $\bar{N}$ – среднее количество запросов, находящихся в системе, при одноуровневом управлении:

```
function N = odnourN(lam, m_1, m_2, L)
%ODNOURN returns average queue length value for one-level control
system

% k          state number = queue length
% lam        input intensity
% m_1        output intensity for 1st mode
% m_2        output intensity for 2nd mode
% L          one-level control parameter

r_1=lam./m_1;
r_2=lam./m_2;
P_0=((1-r_1).*(1-r_2))./(1-r_2-r_1.^L.*(r_1-r_2));
N=(P_0).*(r_1./(1-r_1).^2-((r_1.^L.*(r_1-r_2))./((1-r_1).*(1-
r_2))).*(L+(1-r_1.*r_2)./((1-r_1).*(1-r_2))));
end
```

Функция `odnourQ(lam, m_1, m_2, L)` вычисляет значение $\bar{Q}$ – среднее количество запросов, ожидающих обработки, при одноуровневом управлении:

```
function Q = odnourQ(lam, m_1, m_2, L)
%ODNOURQ returns average Q value for one-level control system

% k          state number = queue length
% lam        input intensity
% m_1        output intensity for 1st mode
% m_2        output intensity for 2nd mode
% L          one-level control parameter

r_1=lam./m_1;
r_2=lam./m_2;
P_0=((1-r_1).*(1-r_2))./(1-r_2-r_1.^L.*(r_1-r_2));
Q=P_0 + P_0.*((r_1.^L.*r_2)./(r_2 - 1).^2 - (r_1.*(r_1.^L - 1))./(r_1
- 1).^2 + (L.*r_1.*r_1.^(L - 1))./(r_1 - 1) - (L.*r_1.*r_1.^(L -
1))./(r_2 - 1)) - P_0.*((r_1.^L - 1)./(r_1 - 1) - r_1.^L./(r_2 - 1));
end
```

Функция `gisterN(lam, m_1, m_2, L_1, L_2)` вычисляет значение $\bar{N}$ – среднее количество запросов, находящихся в системе, при гистерезисном управлении:

```
function N = gisterN(lam, m_1, m_2, L_1, L_2)
%GISTERN returns average queue length value for hysteresis control
system

% k          state number = queue length
% lam        input intensity
% m_1        output intensity for 1st mode
% m_2        output intensity for 2nd mode
% L_1        hysteresis control 1st parameter
% L_2        hysteresis control 2nd parameter

r=lam./m_1;
r_1=lam./m_2;
q=L_2-L_1-1;
P_0=(1./(1-r)-((q+1).*r.^(L_1+q).*(r-r_1))./((1-r.^(q+1)).*(1-
r_1))).^(-1);
N=P_0.*(r./(1-r).^2 -((q+1).*r.^(L_1+q).*(r-r_1 ))./((1-r.^(q+1)
).*(1-r_1 ) ).*((2.*L_1+q)./2+(1-r.*r_1)./((1-r).*(1-r_1 ) )) );
end
```

Функция `gisterQ(lam, m_1, m_2, L_1, L_2)` вычисляет значение $\bar{Q}$ – среднее количество запросов, ожидающих обработки, при гистерезисном управлении:

```
function Q = gisterQ(lam, m_1, m_2, L_1, L_2)
%GISTERP returns average Q value for hysteresys control system

% k          state number = queue length
% lam        input intensity
% m_1        output intensity for 1st mode
% m_2        output intensity for 2nd mode
% L_1        hysteresis control 1st parameter
% L_2        hysteresis control 2nd parameter

r=lam./m_1;
r_1=lam./m_2;
q=L_2-L_1-1;
P_0=(1./(1-r)-((q+1).*r.^(L_1+q).*(r-r_1))./((1-r.^(q+1)).*(1-
r_1))).^(-1);
Q=P_0.*(((r-(L_1+1).*r.^(L_1+1)+L_1.*r.^(L_1+2))./(1-r).^2 -(r-
r.^(L_1+1))./(1-r))+(r.^(L_1 ).*(1-r))./(1-r.^(q+1) ).*((L_1-1).*(((r-
r.^(q+1))./(1-r)-q.*r.^(q+1))./(1-r)+(r.^q.*r_1.*(q-(r_1-r_1.^(q+1)
)./(1-r_1 )))./(1-r_1 ))+(((r-(q+1).*r.^(q+1)+q.*r.^(q+2))./(1-r).^2 -
(r.^(q+1).*q.*(q+1))./2)./(1-r)+(r.^q.*r_1.*((q.*(q+1))./2-(r_1-
(q+1).*r_1.^(q+1)+q.*r_1.^(q+2))./(1-r_1 ).^2 ))./(1-r_1 )) )+((1-
r_1.^(q+1) ).*(1-r).*r.^(L_1+q))./((1-r.^(q+1) ).*(1-r_1 )
).*((r_1.*(L_1+q-1))./(1-r_1 )+r_1./(1-r_1 ).^2 ) );
end
```

Функции, позволяющие построить графики дискретного распределения случайной величины N

Функция `gisterP(k, lam, m_1, m_2, L_1, L_2)` вычисляет вероятность нахождения системы в состоянии k при гистерезисном управлении.

```
function Pn = gisterP(k, lam, m_1, m_2, L_1, L_2)
%GISTERP returns probability distribution for hysteresys control
system

% k          state number = queue length
% lam        input intensity
% m_1        output intensity for 1st mode
% m_2        output intensity for 2nd mode
% L_1        hysteresis control 1st parameter
% L_2        hysteresis control 2nd parameter

k = floor(k);
r=lam./m_1;
r_1=lam./m_2;
q=L_2-L_1-1;
P_0=(1./(1-r)-((q+1).*r.^(L_1+q).*(r-r_1))./((1-r.^(q+1)).*(1-
r_1))).^(-1);
for n=k
    if (n>=0) && (n<=L_1)
        Pn=(r.^n).*P_0;
    elseif (n>L_1) && (n<L_2)
        Pn=(r.^(L_1).*(1-r))./(1-r.^(q+1)).*P_0.*((r.^(n-L_1)-
r.^(q+1))./(1-r)+(r.^q.*r_1.*(1-r_1.^(n-L_1)))./(1-r_1));
    elseif (n>=L_2)
        Pn=((1-r_1.^(q+1)).*(1-r).*r.^(L_1+q))./((1-r.^(q+1) ).*(1-
r_1)).*r_1.^(n-L_1-q).*P_0;
    else
        Pn=0;
    end
end
end
```

Функция `odnourP(k, lam, m_1, m_2, L)` вычисляет вероятность нахождения системы в состоянии k при одноуровневом управлении.

```
function Pn = odnourP(k, lam, m_1, m_2, L)
%ODNOURP returns probability distribution for one-level control system

% k          state number = queue length
% lam        input intensity
% m_1        output intensity for 1st mode
% m_2        output intensity for 2nd mode
% L          one-level control parameter
```

```
k = floor(k);
r_1=lam./m_1;
r_2=lam./m_2;
P_0=((1-r_1).*(1-r_2))./(1-r_2-r_1.^L.*(r_1-r_2));
for n=k
    if (n>=0) && (n<=L)
        Pn=(r_1.^n).*P_0;
    elseif (n>L)
        Pn=(r_1.^L).*(r_2.^(n-L)).*P_0;
    else
        Pn=0;
    end
end
end
```

На рис. 8.4.1, 8.4.2 представляются распределения вероятностей количества находящихся в системе запросов при одноуровневом управлении и разных интенсивностях входящего потока. Для каждого из них выполняются нормирующие условия, что является подтверждением корректности аналитических выкладок.

При $\lambda = 3.7 \cdot 10^4 < \mu_1$ выясняется, что вероятность нахождения системы в режиме работы без памяти выше вероятности пребывания в альтернативном режиме. Указанный эффект объясняется тем, что при указанном значении интенсивности входящего потока переход в режим работы с памятью сопровождается быстрым уменьшением длины очереди и возвращением в альтернативный режим работы. При $\lambda = 2 \cdot 10^5 > \mu_1$ (вероятная ситуация) распределение для случая одноуровневого управления характеризуется ярко выраженным максимумом вблизи значений *n*, равных заданным параметрам *L*. При гистерезисном управлении увеличивается дисперсия и подобного явления не наблюдается.

Таблица 8.4.1

Параметры исследуемых систем

	System #1	System #2	System #3
μ_1	$4 \cdot 10^4$	$4 \cdot 10^4$	$4 \cdot 10^4$
μ_2	$4 \cdot 10^5$	$4 \cdot 10^5$	$4 \cdot 10^5$
Механизм управления	Одноуровневое	Одноуровневое	Гистерезисное
Параметры управления	$L = 5$	$L = 25$	$L_1 = 5$ $L_2 = 25$

В поведении распределения количества находящихся в системе запросов в условиях высокой интенсивности входящего потока проявляется тот факт, что при одноуровневом управлении система большую часть времени находится в состоянии,

при котором длина очереди в буферной памяти приближается к пороговому значению L. Последнее означает, что в системе осуществляется частое переключение режимов, которое негативно сказывается на характеристиках системы из-за временных затрат на переключение.

При гистерезисном управлении указанный недостаток не наблюдается, поскольку переключение режимов выполняется при различных пороговых значениях $L_1 < L_2$.

Распределение вероятности количества находящихся в системе запросов

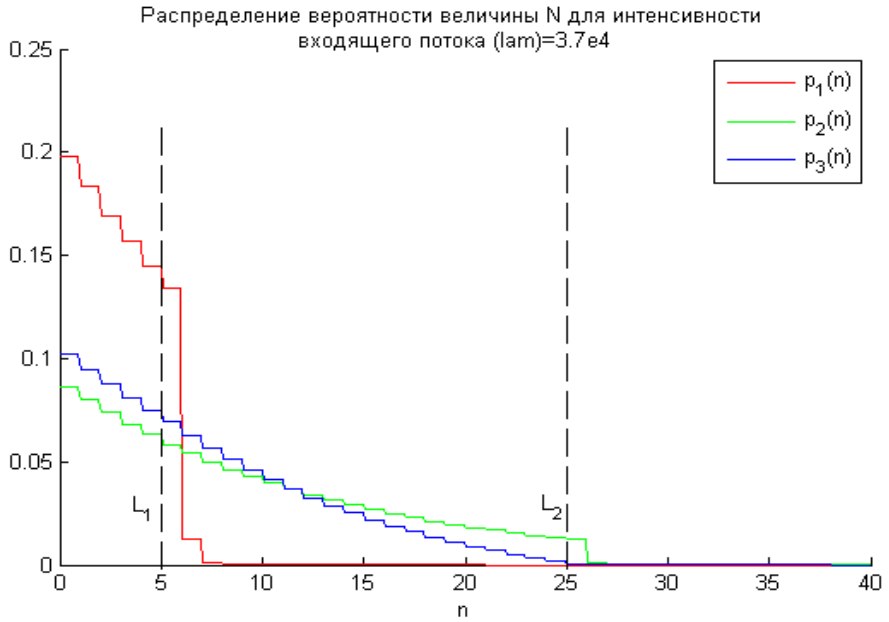


Рис. 8.4.1

Распределение вероятности количества находящихся в системе запросов

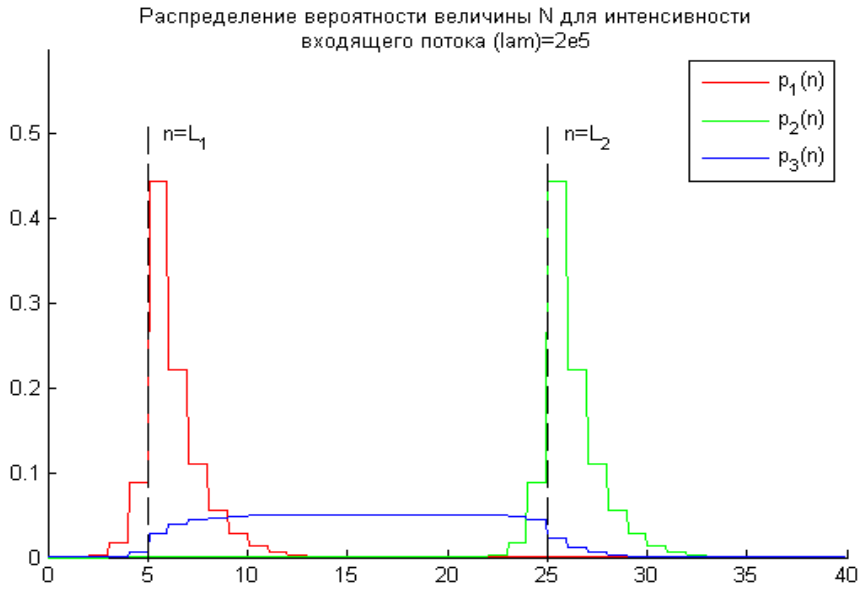


Рис. 8.4.2

Анализ зависимостей показывает, что значения средней длины очереди, приведенные в табл. 8.4.2, находятся в полном соответствии с ранее рассмотренными распределениями.

На рис. 8.4.3 представляются зависимости среднего количества запросов в системе и среднего количества запросов в буферной памяти от интенсивности входящего потока запросов. Значения $\bar{N}$ и $\bar{Q}$ монотонно увеличиваются с возрастанием интенсивности входящего потока. Среднее количество ожидающих обслуживания запросов остается меньшим на единицу по сравнению со средним количеством запросов в системе, что соответствует сущности оцениваемых показателей и подтверждает корректность аналитического ядра систем автоматизации управления потоком запросов к серверной инфраструктуре.

При $\lambda < \mu_1$ длина очереди изменяется в области значений меньших порога L для систем с одноуровневым управлением и порога L_2 для систем с гистерезисным управлением, так как системы функционируют преимущественно в режиме без памяти. При приближении величины λ к значению μ_2 в системе учащается переход к режиму с памятью. На некотором интервале изменения интенсивности входящего потока средняя длина очереди в буферной памяти увеличивается постепенно, а затем наблюдается неограниченное возрастание. Для исследуемых показателей выполняются следующие соотношения:

$$\bar{N}_1 \le \bar{N}_3 \le \bar{N}_2,\ \bar{Q}_1 \le \bar{Q}_3 \le \bar{Q}_2.$$

.

На рис. 8.4.4 показываются графики зависимостей среднего времени простаивания в очереди от интенсивности входящего потока запросов. При этом выясняется, что значение $\bar{W}$ увеличивается по мере возрастания интенсивности входящего потока при $\lambda < \mu_1$.. При λ близком к μ_1 в функции $\bar{W}(\lambda)$ появляется локальный максимум. При приближении величины λ к μ_2 на некотором интервале среднее время простаивания в очереди уменьшается, так как увеличивается вероятность нахождения системы в режиме работы с памятью. При λ близком к μ_2 наблюдается неограниченное возрастание значений W . Для множества рассматриваемых альтернативных вариантов соблюдается соотношение

$$\bar{W}_1 \le \bar{W}_3 \le \bar{W}_2,$$

что обуславливается особенностями выбора параметров сравниваемых способов управления потоком запросов к серверной инфраструктуре.

На рис. 8.4.5 изображаются зависимости среднего времени обработки запросов от интенсивности входящего потока для сравниваемых вариантов управления потоком запросов к серверной инфраструктуре. При $\lambda < \mu_1$ на некотором интервале вариаций интенсивности входящего потока среднее время обработки запросов не изменяется и приблизительно равняется $1/\mu_1$, так как система не нуждается в режиме с памятью.

При приближении величины λ к значению μ_2 среднее время обработки запросов монотонно снижается и приближается к уровню $1/\mu_2$, поскольку

увеличивается вероятность перехода к режиму работы с памятью. В ходе эксперимента выясняется, что

$$\bar{S}_1 \leq \bar{S}_3 \leq \bar{S}_2.$$

Зависимости среднего количества запросов в системе и среднего количества запросов в буферной памяти от интенсивности входящего потока запросов

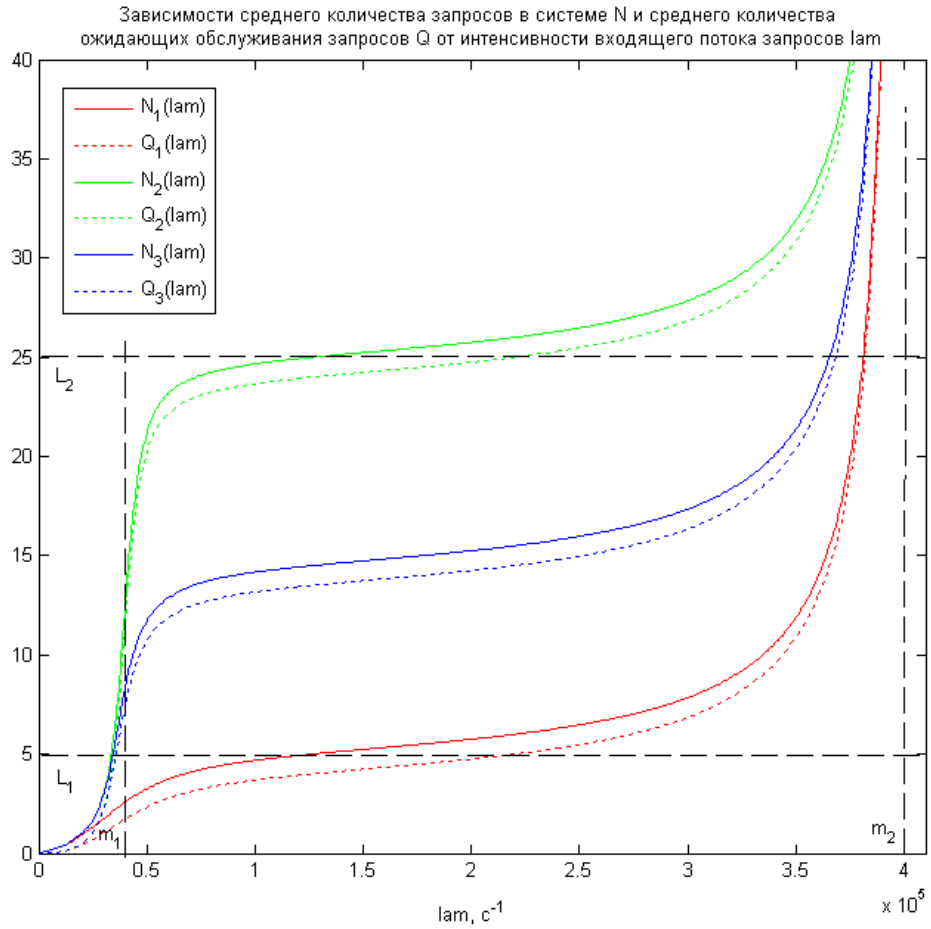


Рис. 8.4.3

В результате применения представленных формализаций:

– разработана концепция автоматизации управления потоком запросов к серверной инфраструктуре;

– раскрыты два способа управления потоком запросов к серверной инфраструктуре;

– формализован процесс определения показателей качества управления потоком запросов к серверной инфраструктуре;

– определены аналитические зависимости показателей качества управления потоком запросов к серверной инфраструктуре от параметров процедур его автоматизации;

– сформирована система генерации знаний о качестве управления потоком запросов к серверной инфраструктуре;

– получены новые знания о влиянии параметров управления потоком запросов к серверной инфраструктуре на его качество;

Зависимости среднего времени простаивания в очереди от интенсивности входящего потока запросов

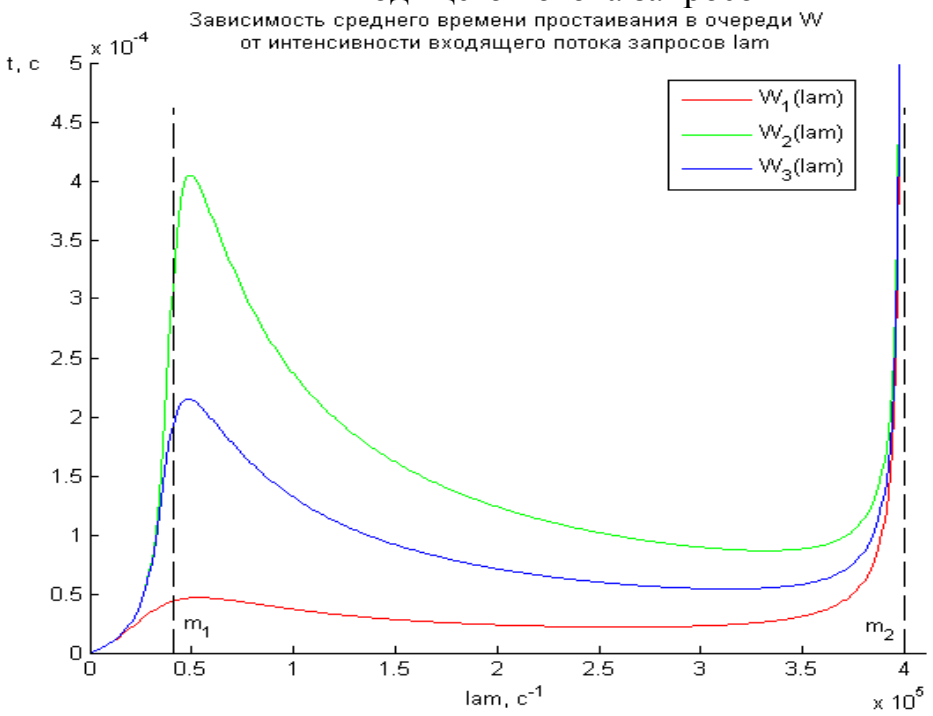


Рис. 8.4.4

Зависимости среднего времени обработки запросов от интенсивности входящего потока

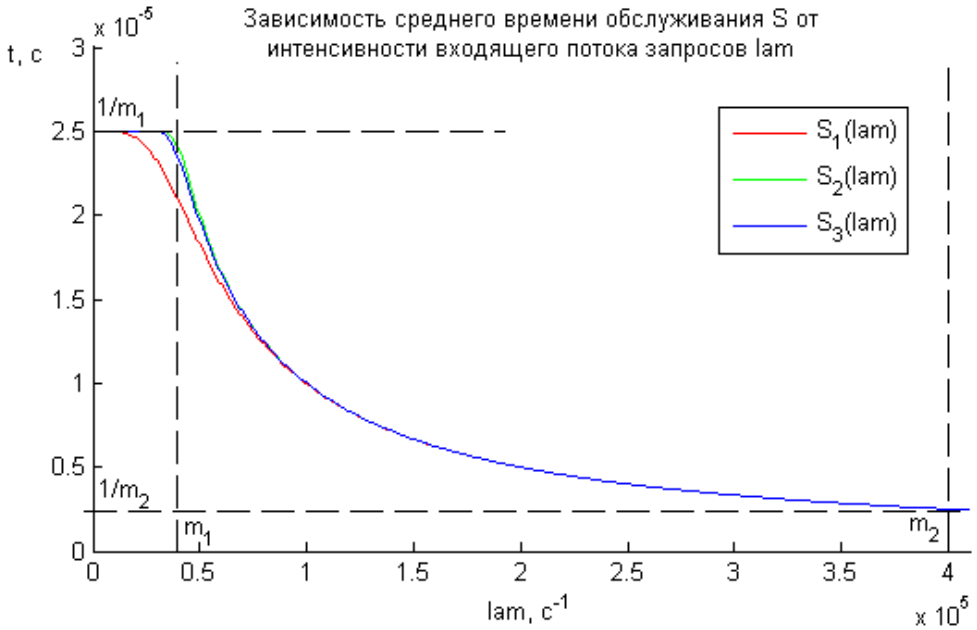


Рис. 8.4.5

Таблица 8.4.2

Показатели качества работы систем при различных способах управления потоком запросов к серверной инфраструктуре

		$\lambda = 3.7 \cdot 10^4$	$\lambda = 2 \cdot 10^5$
System #1	$\overline{N_1}$	2.33	5.75
	$\overline{W_1}$	$4.12 \cdot 10^{-5}$	$2.38 \cdot 10^{-5}$
	$\overline{S_1}$	$2.17 \cdot 10^{-5}$	$5.00 \cdot 10^{-6}$
System #2	$\overline{N_2}$	8.41	25.75
	$\overline{W_2}$	$2.03 \cdot 10^{-4}$	$1.24 \cdot 10^{-4}$
	$\overline{S_2}$	$2.47 \cdot 10^{-5}$	$5.00 \cdot 10^{-6}$
System #3	$\overline{N_3}$	6.61	15.25
	$\overline{W_3}$	$1.55 \cdot 10^{-4}$	$7.13 \cdot 10^{-5}$
	$\overline{S_3}$	$2.43 \cdot 10^{-5}$	$5.00 \cdot 10^{-6}$

– выявлены преимущества гистерезисного управления потоком запросов к серверной инфраструктуре.

9 ПРЕДСТАВЛЕНИЕ ЗНАНИЙ НА ОСНОВЕ ОНТОЛОГИЙ

9.1 ОНТОЛОГИИ В ПРЕДСТАВЛЕНИИ ЗНАНИЙ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

По мнению специалистов по искусственному интеллекту основная роль в описании знаний отводится онтологиям, которые используются при проектировании баз знаний, создании экспертных систем, систем поддержки принятия решений, разработке сред, ориентированных на совместное использование информации несколькими пользователями, и разработке поисковых систем.

В настоящее время онтологии используются в электронной коммерции, как средства обмена информацией; в медицине, как средства классификации и основы построения экспертных систем; в планировании и проектировании, как средства, предлагающие рациональные решения для стандартного набора задач, и в других областях.

В области инженерии знаний онтологии являются средством формирования систем управления знаниями. В области информационных систем онтологии играют важнейшую роль, объединяя основные компоненты любой информационной системы: информационные ресурсы, интерфейс пользователя и прикладные программы.

Особенно широкое распространение онтологии получили благодаря развитию распределённых систем искусственного интеллекта, реализуемых на основе агентных технологий.

На современном уровне развития IT-технологий уже сложился набор средств разработки управления онтологиями.

9.2 ОНТОЛОГИИ КАК СПОСОБ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Онтология рассматривается как универсальный механизм для описания структурированного знания. Онтология включает в себя словарь, средствами которого описываются знания и сама структура. Посредством словаря онтология накладывает ограничения на используемую семантику.

Понятие онтологии заимствовано из философии, где онтологией называлось философское учение об общих категориях и закономерностях бытия.

В рамках искусственного интеллекта онтология используется для описания объектов и явлений мира в формализованном виде, пригодном для компьютерного использования.

Применение аппарата категорий к структуризации знаний привело практически к одновременному появлению теории концептуальных графов, разработанных J. Sowa, и способа представлений в виде технологий, разработанного N. Guarino.

Введение онтологий позволило упростить процесс совместного и многократного использования знаний.

Общепринятым считается определение онтологии, которое сформулировал T. Gruber. Онтология – это точная спецификация концептуализации. Более полное определение сформулировано следующим образом:
онтология – это формальная, точная спецификация совместно используемой концептуализации.

Под концептуализацией понимается абстрактная модель объектов, явлений, процессов в мире, составленная посредством определения существенных для них описаний.

Точность подразумевает то, что типы используемых понятий и ограничений на область применения данных понятий явно определены.

Формальность имеет отношение к тому факту, что онтология должна быть ориентирована на компьютерное представление.

Совместное использование отражает понятие того, что онтология описывает всеобщие знания, т.е. не персональные знания одного человека, а знания, принятые в группе, сообществе.

Определение онтологии, используемое в рамках технологии агентов и соответствующее спецификации FIPA (Foundation for Intelligent Physical Agents), заключается в следующем:

1. Онтология – это подробная спецификация структуры определённой проблемной области;
2. Онтология включает в себя словарь, т.е. список логических констант и предикатных символов, для описания предметной области и набор логических высказываний, формулирующих существующие в данной проблемной области знания и определяющих интерпретацию словаря;
3. Онтология предлагает словарь для представления и обмена знаниями по интересующей проблеме и набор связей и свойств, которые определены между имеющимися в её словаре неделимыми сущностями.

9.3 ВИДЫ ОНТОЛОГИЙ

С точки зрения описываемой концептуализации онтологии делятся на следующие группы:

1. Онтологии представления, определяющие концептуализацию, которая лежит в основе формализации представления.
2. Общие (родовые) онтологии, затрагивающие фундаментальные аспекты концептуализации такие, как часть, причина, участие, представление.
3. Промежуточные онтологии, содержащие общие понятия и отношения, характерные для конкретной предметной области.
4. Онтологии верхнего уровня, являющиеся конкретным назначением понятий общих и промежуточных онтологий.
5. Онтологии предметных областей, содержащие понятия определённой области знаний или входящих в неё областей.
6. Онтологии задач, описывающие определённые задачи или деятельность, используя словарь, введённый в общих, промежуточных онтологиях и онтологиях предметных областей.
7. Онтологии-приложения, являющиеся специализацией онтологий предметных областей и онтологий задач, и опирающиеся на определения, характерные для конкретного приложения.

Описание онтологии в терминологии словаря обычно представляется в форме, предлагаемой в теории логики первого порядка. В этом случае слова из словаря выступают в качестве унарных или бинарных имён предикатов, называемых соответственно понятиями и отношениями.

В самом простом случае онтология описывает иерархию связанных понятий, в более сложных случаях добавляются подходящие аксиомы, выражающие связи между понятиями и ограничивающие их интерпретацию.

Ряд известных специалистов предлагают трёхуровневую схему отношений между онтологиями.

К первому (нижнему) уровню относят онтологии-приложения.

Средний уровень включает онтологии предметных областей и онтологии задач.

На третий уровень выносятся онтологии верхнего уровня. Категории для описания абстракций верхнего уровня и соответствующие им формализмы раскрываются в 11 разделе.

9.4 ОБЩИЕ ПОДХОДЫ К ПОСТРОЕНИЮ ОНТОЛОГИИ

Gruber T. предлагает следующий набор критериев для проектирования онтологий: понятийность, согласованность, расширяемость, минимум ошибок кодирования и минимальное онтологическое соглашение.

Pinto H. расширяет указанный набор требований за счёт таких критериев, как полнота и лаконичность онтологии, исключение двусмысленности в толковании, адекватный уровень детализации и оцениваемость.

Lopez предлагает стандартизировать процесс построения онтологии, опираясь на стандарт IEEE 1074-1995, предназначенный для стандартизации разработки программного обеспечения. Указанный стандарт описывает последовательность следующих этапов процесса разработки программного обеспечения:

1) Процессы управления проектом;
2) Процессы, направленные на разработку программного обеспечения, состоящие из стадии начальной разработки (изучения среды, для которой создаётся программное обеспечение), стадии детальной разработки и стадии внедрения;
3) Процессы по проверке, установке, документированию и обучению.

Kayed A. предлагает процесс создания онтологии, состоящий из следующих этапов:

1) Специализация;
2) Концептуализация;
3) Формализация;
4) Интеграция;
5) Реализация;
6) Эксплуатация.

Перечисленные этапы согласуются с предложенным выше стандартом. Дополненные этапом проверки они могут считаться базовыми для построения онтологий.

Спецификация. На данном этапе составляется план основных заданий, описываются требования к разработке, обосновывается сценарий, интуитивно определяющий возможные области применения онтологии, и задачи, которые могут быть решены.

Концептуализация. Данный этап включает извлечение терминов и распределение их по категориям. На этой базе строится концептуальная модель предметной области.

Формализация. Данный этап зависит от моделируемой предметной области. Известны четыре уровня формализации: уровень с высокой неформальностью (слабо описанная на естественном языке онтология); неформально-структурированный (онтология описана ограниченным набором лексики естественного языка в структурированном виде); полуформальный (онтология описана на специальном формально определённом языке); строго формальный (для описания используются тщательно определённые термины с формальной семантикой, теоремы и доказательства для таких свойств, как надёжность и полнота онтологии).

Реализация. На этом этапе определяется технология, которая будет использоваться для реализации онтологии, интеграции новой онтологии с существующими, и технология поддержки онтологии на этапе её эксплуатации. Требования к разработанной онтологии состоят в том, чтобы пользователь мог просматривать онтологию, искать интересующие его понятия, и оценивать технологию.

Проверка. На данном этапе онтология проверяется на полноту и непротиворечивость. Проверка включает в себя идентификацию и редактирование неопределённых терминов; выявление конфликтов и несогласованностей; проверку онтологии на полноту; идентификацию семантических различий между двумя терминами в различных онтологиях; синтаксический анализ, анализ таксономии, семантическую оценку.

10 МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ ПРИ ОРГАНИЗАЦИОННОМ ПРОЕКТИРОВАНИИ НА ОСНОВЕ ОНТОЛОГИЙ

Модель организационного проектирования описывается в виде кортежа:

$$\mathbf{F} = < \mathbf{O}, \mathbf{Q}, \mathbf{M}, \mathbf{B}, \mathbf{S} > ,$$

$$\mathbf{O} = < \mathbf{C}_o, \mathbf{P}, \mathbf{R}, \mathbf{A}, \mathbf{E} > ,$$

где $\mathbf{O}$ – онтология организации;
$\mathbf{C}_o = \{c_{oi}\}$ – классы – типы;
$\mathbf{P}$ – свойства классов;
$\mathbf{R}$ – связи между классами;
$\mathbf{A}$ – правила онтологии;
$\mathbf{E} = \{\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3\}$ – расширение стандартного формализма логики предикатов для повышения качества онтологии;
$\mathbf{Q}$ – показатели качества онтологии: полнота онтологии; использование единых согласованных принципов построения: когнитивная эргономичность и прозрачность; непротиворечивость, безизбыточность;
$\mathbf{E}_1 = \{\mathbf{C}_R, \{ist(\textit{исполняет роль}(C_i, C_{Rj}), CTX_i)\}, [\{< CTX_i, C_{Rj} >\}]\}$ – представление контекстно-зависимых классов-ролей $\mathbf{C}_R$ и контекста $\mathbf{CTX}$, задающего истинность высказываний;
$\mathbf{E}_2 = \{\textit{используется}\ (CB_y, \{C_{PFi}\}, \{C_{yi}\})\}$ – основания деления классов $(CB_y \subset \mathbf{C}_o)$, типовые подклассы для основания деления $(C_{PFi} \subset \mathbf{C}_o)$ и области их применения $(C_{yi} \subset \mathbf{C}_o)$;
$\mathbf{E}_3$ – дополнительная группировка элементов онтологии организационного проектирования для повышения её когнитивной эргономичности и прозрачности.

11 ХАРАКТЕРИСТИКА СИСТЕМ ПРЕДСТАВЛЕНИЯ И ОБРАБОТКИ ЗНАНИЙ НА ОСНОВЕ ОНТОЛОГИЙ

11.1 КАТЕГОРИИ ДЛЯ ОПИСАНИЯ АБСТРАКЦИЙ ВЕРХНЕГО УРОВНЯ

Основной областью применения онтологий является интеграция информации. Онтологии связывают два существенных аспекта:

1) определяют формальную семантику информации, позволяя обработку информации компьютером;
2) определяют семантику реального мира, позволяя, основываясь на общей терминологии, связывать информацию, представленную в виде, требуемом для компьютерной обработки, с информацией, представленной в удобной для восприятия человеком форме.

Целесообразнее опираться на одну онтологию верхнего уровня вместо того, чтобы полагаться на договорённости, основанные на пересечении различных онтологий. В таблице 11.1.1 представлены высшие категории, определённые различными авторами, и формализмы, задающие принципы описания по этим категориям объектов мира.

Таблица 11.1.1

Высшие категории

Автор, система	Категории для описания абстракций верхнего уровня	Используемый формализм
Guarino N.	Частности: существующие в мире объекты. Универсальные понятия: понятия, необходимые для описания частностей.	Мета-свойства (тождественность, строгость, целостность, зависимость, собственная тождественность) и ограничения, вытекающие из определений перечисленных свойств.
Sowa J.	Независимое: понятия, существующие независимо от наличия каких-либо других объектов. Связанное отношение; понятия, используемые для определения играемых ролей. Посредствующее: среда или контекст, где реализуется предыдущее понятие. Непрерывное. Дискретное.	Примитивные свойства (реальность, форма, понимание, препозиция, звено, намерение) и категории верхнего уровня (объект, процесс, схема сценарий, соединение, участие, описание, история, структура, ситуация, причина, цель)
CYC	Единичный объект: константа, обозначающая единичный объект, в отличие от множества объектов. Неосязаемый объект: набор нефизических объектов (не	CycL – язык исчисления предикатов первого порядка (язык формальной логики). Термы языка: константы, неатомарные термы,

	являющихся сделанными из чего-либо, не результат кодирования, нематериален). Временные объекты: набор понятий, имеющих протяжённость (действия, осязаемые объекты, соглашения, абстрактные единицы времени).	переменные. Термы объединяются в выражения, используемые для вывода утверждений в базу знаний. Каждому понятию из базы знаний соответствует некоторая константа.
GUM (Generalized Upper Model)	Конфигурация: конфигурация элементов, участвующих в некоторой деятельности. Элемент: единичный самостоятельный объект или понятие. Последовательность: сложная ситуация, когда различные виды деятельности или конфигурации связаны отношениями таким образом, что образуют последовательность.	LOOM Knowledge Representation Language Анализ лексико-грамматических систем естественных языков, основанный на использовании метафункций (экспериментальная, логическая, текстовая, взаимодействия высказывающегося и слушателя). Метафункции представляют модели верхнего уровня, набор ограничений относительно того, что модель должна содержать, чтобы описать этими средствами различные грамматические конструкции.
WebKB	Ситуация: объект, происходящий во времени и пространстве. Сущность: объект, который может быть вовлечён в ситуацию.	Resource Description Framework (RDF) формат, включающий 3 базовых метакласса (Class –класс, Property – свойство, Constraint Resource – ограничения) и категорию «отношение» (relation), как экземпляр класса Property. Метаклассы образуют набор классов (Collection).
WordNet – лексическая база данных английского языка	Живой объект: организмы (растения, животные, человек). Неживой объект: вещи (артефакты, вещества, еда).	Лексическая матрица, семантическая сеть. Рассматриваются семантические и морфологические отношения между словами.

Однако разработка общей достаточной онтологии верхнего уровня является серьёзной проблемой, которую пока решить не удалось.

В настоящее время принято придерживаться принципа интеграции частных онтологий в более общие, основываясь на разрабатываемых методах стандартизации таких, как специальные языки моделирования и средства трансляции.

11.2 МЕТОД АВТОМАТИЗИРОВАННОГО ФОРМИРОВАНИЯ ОНТОЛОГИИ

Важной задачей при формировании онтологий для обучающих систем является корректное, полное и непротиворечивое наполнение онтологии информацией. Проектирование онтологии предполагает совместную (в том числе и перманентную) работу группы специалистов по выделению базовых концепций предметной области и постепенному наполнению онтологии понятиями. По мере роста объёма онтологии увеличивается и сложность задачи по объективному определению места нового понятия в общей системе и корректировке структуры онтологии. Авторское («ручное») решение этой задачи, традиционное для широко используемых редакторов онтологий, при работе с объёмной онтологией сопряжено с высокой трудоёмкостью и риском принятия субъективных решений. Инструментом, обеспечивающим повышение качества и интенсивности разработки онтологий, могут стать средства автоматизации онтологического инжиниринга.

Далее раскрывается метод поэтапного автоматизированного формирования онтологии, позволяющий расширить функциональность существующих редакторов онтологий и повысить меру объективности и полноты при дополнении онтологии новыми понятиями. После ввода экспертом нового понятия раскрываемый метод позволяет автоматически определить место включения понятия в онтологическую иерархию и автоматически сформировать обобщённые понятия, порождаемые при вводе нового понятия. Основная идея метода заключается в определении концепта в существующей онтологии, которому новый концепт непосредственно наследует.

Модель онтологии представляется кортежем

$$Q = < C, M, R > ,$$

где $C = \{c_i\}$ – множество понятий (концептов) $i = 1, 2, ..., I$, образующих онтологию Q; I – количество концептов; $M_i = \{m_{1i}, ..., m_{di}\}$ – множество атрибутов концепта c_i (d – количество атрибутов, описывающих концепт c_i; $R \subseteq C \times C$ – отношение непосредственного наследования. Отношение R задаётся матрицей размером $I \times I$. Элементы этой матрицы определяются следующим образом: если концепт c_k в онтологии непосредственно наследует концепту c_i, то элемент матрицы R $r_{ik} = 1$, в противном случае $r_{ik} = 0$.

Эксперт формирует модель нового понятия, указывая набор определяющих понятий атрибутов $M_x = \{m_{1x}, ..., m_{dx}\}$.

Из корневого концепта онтологии и непосредственно наследующих ему концептов для вводимого концепта формируется множество потенциальных концептов родителей $P = \{c_0, c_{j1}, ..., c_{jf}\}$ таких, что $\forall j(j_1, ..., j_f) : r_{0j} = 1$. Корневой концепт c_0 также вводится в формируемое в процессе процедуры отношение RP. Элементы отношения RP определяют потенциально возможные места размещения вводимого концепта $(c_0, c_x) \in RP, rp_{ox} = 1$.

Базовая процедура автоматизации содержит следующие шаги:

1. Организуется просмотр потенциальных концептов-родителей. Если $P=0$ (все потенциальные концепты-родители рассмотрены), то переход к п.5.
2. Из P выбирается следующий по порядку элемент. Пусть из P выбран концепт $c_j, j \in (j_1,..., j_f)$. Тогда возможны три варианта:
2.1. $M_j \not\subset M_x \wedge M_x \not\subset M_j$. В этом случае концепт c_j и все его наследники признаются неперспективными. Концепт c_j исключается из множества P $(c_j \not\subset P)$. Данная операция ускоряет процесс решения задачи по отношению к алгоритму полного перебора.
2.2. $M_j \not\subset M_x \wedge M_x \subset M_j$. Это тот случай, когда вводимый концепт должен быть вставлен между c_k и c_j, где $k: r_{kj}=1$. В этом случае концепт c_j и все его наследники признаются неперспективными. Концепт c_j исключается из множества P $(c_j \not\subset P)$. Данная операция ускоряет процесс решения задачи по отношению к алгоритму полного перебора. Существовавшее в онтологии отношение $(c_k R c_j)$ исключается, т.е. $r_{kj}=0$. Далее вводятся два новых отношения $(c_k R c_x)$ и $(c_x R c_j)$, $r_{kx}=1$и $r_{xj}=1$. Далее необходимо проверить, является ли введённый концепт обобщённым. Для этого процедура должна быть продолжена посредством перехода к п.1.
2.3. $M_j \subset M_x$. Это означает c_x наследует c_j. Вводим в множество RP новый элемент $(c_j, c_x) \in BR$, т.е. $rp_{jx}=1$, тем самым фиксируется потенциальное место для вводимого концепта, и c_j исключается из множества P $(c_j \not\subset P)$. Корневой сегмент также исключается из множества P $(c_0 \not\subset P)$ и из множества RP $rp_{0x}=0$.
3. Из потенциальных мест размещения вводимого концепта RP исключают, если таковой имеется, менее перспективный концепт, c_k, а именно:
$\exists k: (c_k, c_j) \in R \wedge (c_k, c_x) \in RP \to (c_k, c_x) \notin RP, rp_{kx}=0$.
4. Для c_j формируется множество V – множество непосредственно наследуемых ему концептов $V=\{c_{v1},...,c_{vg}\}$ таких, что $\forall v (v \in \{v_1,...,v_g\}: r_{jv}=1$. Множество V объединяют со множеством P $P=P \cup V$. Переходят к п.1.
5. Место для включения вводимого концепта найдено ($|RP| \neq 0$) и он должен включаться в состав онтологии. В том случае, если найдено лишь одно потенциальное место ($|RP|=1$), то $(c_j, c_x) \in RP \to c_x \in C \wedge r_{jx}=1$.
Если в множестве временно образованного отношения RP находится более одного элемента, то соответствующий ему концепт онтологии является обобщением. Тогда

$$\forall j: (c_j, c_x) \in RP \to c_x \in C \wedge \forall (c_j, c_x) \in R, m.e. \forall j: (c_j, c_x) \in RP \to r_{jx}=1.$$

После определения места нового понятия в иерархии могут быть автоматически определены новые понятия боле высоких ровней на основе выявления общих для различных понятий наборов атрибутов:

$$\forall k: M_n = M_x \cap M_k, M_n \neq M_x, M_n \neq M_k, |M_n| \neq 0, c_n \notin C \to c_n \in C,$$

где c_n – новое понятие верхнего уровня по отношению к понятиям c_x и c_k, имеющее набор атрибутов M_n.

12 МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ НА ОСНОВЕ ТЕОРИЙ ЛОГИК

В основе большинства формальных методов используются логические исчисления. Их отличительная особенность заключается в единственности теоретического обоснования и возможности реализации множества формально точных определений и выводов. В этих случаях при представлении знаний определяются формулы и правила соответствующего исчисления.

Логическое исчисление – это система, в которую вводится некоторое число исходных объектов (аксиом) и некоторое число правил вывода новых объектов из исходных и (или) ранее сформированных.

Логика как система базируется на языке, который называется языком исчисления, нескольких теоремах и правилах вывода. Теоремы представляются в терминах языка исчисления. В теоремах описываются отношения базиса логически полной системы. Правила вывода предназначаются для получения новых представлений, отличных от всех известных представлений некоторой группы.

Логика высказываний имеет определённый синтаксис. В синтаксисе языка исчисления проводят определённые параллели с синтаксисом естественных языков. Символами языка логики высказываний являются элементы, составляющие её алфавит. Значениями логических переменных являются логические константы. Правильно построенные формулы должны соответствовать правилам логики.

Специалисты различают следующие группы формальных языков:
- язык пропозициональной (простейшей) логики;
- язык логики первого порядка;
- язык временной (темпоральной) логики;
- язык логики высокого порядка;
- язык теории вероятностей;
- язык нечёткой логики.

Формальные языки вносят различный онтологический и эпистомологический вклад в познание действительности.

В пропозициональной логике предполагается, что существуют лишь факты, которые относятся или не относятся к данному миру. Каждый факт может находиться в одном из двух состояний, быть истинным или ложным.

В логике первого порядка приняты более широкие предположения: мир состоит из объектов, между которыми могут быть или не быть некоторые отношения.

Во временной логике предполагается, что факты имеют место в конкретные интервалы времени и что эти интервалы являются упорядоченными.

В логике высокого порядка как объекты трактуются сами отношения и функции, рассматриваемые в логике первого порядка.

В пропозициональной логике и логике первого порядка любое высказывание представляет собой факт. Интеллектуальный агент может доверять утверждению, что это высказывание истинно, либо доверять утверждению, что оно ложно, либо не иметь своего мнения. В связи с этим рассматриваются три возможных состояния знаний, касающихся высказывания. В системах, где используется теория

вероятностей, может рассматриваться любая степень доверия, начиная от 0 (полное недоверие) и заканчивая 1 (полное доверие).

В таблице 12.1 описаны онтологический и эпистомологический вклады пяти различных рассмотренных вариантов логик.

Таблица 12.1

Формальные языки и показатели их онтологического и эпистомологического вкладов в познание

Язык	Онтологический вклад (что существует в мире)	Эпистомологический вклад (какую степень доверия может выражать интеллектуальный агент в отношении фактов)
Пропозициональная (простейшая) логика	Факты	Истинно / Ложно / Неизвестно
Логика первого порядка	Факты, объекты, отношения	Истинно / Ложно / Неизвестно
Временная логика	Факты, объекты, отношения, интервалы времени	Истинно / Ложно / Неизвестно
Теория вероятностей	Факты	Степень доверия от 0 до 1
Нечёткая логика	Факты со степенью истинности, принадлежащей [0,1]	Известное интервальное значение

В сфере информационных систем и технологий используются и иные исчисления. В них вводятся факты, аксиомы и правила, не относящиеся к элементам классического исчисления. Вводимые факты, аксиомы и правила зависят от конкретной цели и предметной области, для которых разрабатывается формализация постановки задачи. Примерами подобных исчислений являются модальная логика, ситуационная логика, пространственно-временная логика.

Для каждой из рассмотренных логик разработана соответствующая теория.

13 АЛГОРИТМЫ И СИСТЕМЫ ПЛАНИРОВАНИЯ ДЕЙСТВИЙ

13.1 АЛГОРИТМЫ ПЛАНИРОВАНИЯ ДЕЙСТВИЙ В ИСКУССТВЕННОМ ИНТЕЛЛЕКТЕ

Алгоритмы планирования действий призваны обеспечивать непрерывное во времени функционирование агента, направленное на максимизацию его производительности. Функционирование всех интеллектуальных агентов, в том числе и информационных, носит целенаправленный характер. При этом агенту необходимо решать задачи достижения одного или нескольких конечных (целевых) состояний из некоторого множества начальных состояний с помощью выполнения агентом последовательности действий. Результатом решения такой задачи является план действий. Задача алгоритма планирования заключается в том, чтобы сгенерировать частично-упорядоченную совокупность действий (план), которая является возможным решением стоящей перед агентом задачи. Исходная информация представляется посредством описания начальных и конечных (целевых) состояний среды и определения меры оценки производительности плана. При решении задачи планирования могут сформироваться один или несколько планов, удовлетворяющих ограничениям по производительности, исполнение которых агентом переводит среду из начального состояния в конечное. Решение задач с помощью планирования дает агенту три основных преимущества: снижает временные затраты на выполнение агентом заданий; устраняет конфликты целей; обеспечивает возможность исправления ошибок. В зависимости от специфики задачи может потребоваться как поиск удовлетворительных планов (планов, производительность которых выше заданной пороговой величины), так и оптимальных (близких к оптимальным с заданной погрешностью) планов.

Существенное влияние на процесс планирования и исполнения планов оказывает степень доступности среды для наблюдений. По этому критерию принято разделять среды систем планирования на полностью наблюдаемые (на основании результатов наблюдений можно однозначно определить текущее состояние среды) и частично-наблюдаемые (на основании результатов наблюдений можно определить подмножество возможных текущих состояний среды). Сложные реальные среды, к которым относятся гетерогенные сети, как правило, являются частично-наблюдаемыми. Преодолеть проблему частичной наблюдаемости среды (решить задачу восстановления состояния), то есть однозначной идентификации текущего состояния среды возможно, если предусмотреть генерацию плана действий по сбору дополнительной информации. Действия планировщика в условиях частичной наблюдаемости среды называют работой при неполной информации.

По степени динамичности среды систем планирования разделяют на детерминированные и недетерминированные. Среди недетерминированных сред выделяют стохастические, для которых известны распределения вероятностей переходов между состояниями при выполнении действий. При недетерминированных средах стремятся не к нахождению плана, а к вычислению политики. Политики разделяют на универсальные и частичные. В первом случае

определяют полные отображения множества состояний на множество действий, а во втором – частичные отображения множества состояний на множество действий. В дальнейшем, если упоминается просто политика без конкретизации, то подразумевается универсальная политика. Вычисление частичных или универсальных политик дает возможность предусмотреть заранее план действий для соответственно многих или всех состояний среды, в которых может оказаться агент. Среды также разделяют на стационарные (изменяют состояния только под воздействием агента) и нестационарные (могут изменять состояния не только под воздействием агента). Очевидно, что полностью предсказуемыми являются лишь детерминированные среды, изменяющие состояния только под воздействием агента. В зависимости от учета временной составляющей планы разделяют на стационарные (выбор очередного действия не зависит от текущего времени) и зависимые от времени (time variant, выбор очередного действия зависит от текущего времени). Если отображения моментов времени на множество действий однозначные, то такой план является детерминированным. В противном случае план недетерминированный. Среди недетерминированных планов выделяют стохастические планы, для которых отображения моментов времени на множество действий соответствуют распределениям вероятностей выбора действий.

В зависимости от способа исполнения планов агентом их разделяют на планы с обратной связью или замкнутые (условные) планы (closed-loop plans, conditional plans) – исполняющий план механизм для выбора действия в следующий момент времени учитывает результаты наблюдения среды в текущий момент) и планы без обратной связи или открытые (безусловные) планы (open-loop plans, unconditional plans) – исполняющий план механизм для выбора действия в следующий момент времени не учитывает результаты наблюдения среды в текущий момент. Существует также разделение планов по способу исполнения на интерактивные (on-line), чередующие планирование и контроль за исполнением плана, и автономные (off-line), у которых процесс генерации плана завершается до начала исполнения плана.

По наличию механизма декомпозиции действий алгоритмы планирования делятся на иерархические (PR-проблема) и неиерархические (SS-проблема).

В первом случае планирование основывается на неиерархическом представлении плана. Требуется составить последовательности операций для достижения каждой цели. Проблема при таком подходе состоит в том, что планировщик не в состоянии различать наиболее важные цели и менее значимые. Он может потратить значительную часть ресурсов на уточнение малозначительных участков плана, а затем обнаружить, что невозможно найти решение для критического участка плана.

Во втором случае планирование основывается на использовании иерархического представления задачи. План нижнего уровня составляется из элементарных (исполняемых агентом) операций. Задача иерархического планировщика заключается в том, чтобы найти решение задачи для уровней декомпозиции с приемлемой вычислительной сложностью. Найденный план детализируется до уровня элементарных операций, которые могут быть непосредственно исполнены аппаратным обеспечением агента. Для гетерогенных сетей характерна большая размерность пространства состоянии, поэтому необходима его декомпозиция. Требование о возможности декомпозиции пространства состояний интеллектуального

информационного агента для гетерогенных сетей приводит к необходимости прибегнуть к иерархическому планированию.

Классические алгоритмы планирования предполагают детерминированность среды. Для гетерогенных сетей, как отмечалось ранее, характерной особенностью является их стохастический характер, что и препятствует применению в интеллектуальных информационных агентах классических алгоритмов планирования. Вследствие изменчивости условий работы агента (например, агент может получить срочную заявку с высоким приоритетом и существенно пересмотреть выполнение полученных ранее заявок) целесообразно разрабатывать интерактивную систему планирования с обратной связью.

По способу упорядочивания действий планы разделяют на частично-упорядоченные или частичные (partially ordered) и полностью упорядоченные (totally-ordered). Полностью упорядоченным планам соответствуют алгоритмы планирования с ранним подтверждением (eager-commitment), а частично упорядоченным планам соответствуют алгоритмы с поздним (отложенным) подтверждением (least-commitment). При использовании раннего подтверждения планировщик создает полностью упорядоченную последовательность целей и действий, в том числе упорядочиваются и несвязанные между собой действия. При подходе на основе позднего подтверждения решения об упорядочивании действий принимаются только в том случае, если планировщик может определить, что они не будут конфликтовать с предыдущими или последующими решениями. Целью данного подхода является снижение вероятности отката назад при построении плана. Так как многие задачи в распределенной среде могут выполняться информационным агентом параллельно, то целесообразно выбрать частично-упорядоченное представление плана, и, следовательно, задействовать алгоритмы планирования с поздним подтверждением.

По способу упорядочивания целей алгоритмы планирования разделяются на линейные и нелинейные Линейное планирование основано на предположении о том, что все подцели являются независимыми друг от друга. Линейный план состоит из упорядоченной последовательности независимых завершенных планов, достигающих своих подцелей. Нелинейный план состоит из упорядоченной последовательности взаимосвязанных субпланов. Нелинейное планирование предполагает позднее подтверждение и учет временной составляющей. Ранее была установлена целесообразность использования для планирования действий интеллектуальных информационных агентов в гетерогенных сетях алгоритмов с поздним подтверждением. В связи с этим генерируемые для информационных агентов планы должны быть нелинейными. Полученный на выходе нелинейный план может либо линеаризоваться для одного агента, либо исполняться несколькими агентами, что еще раз подчеркивает пригодность предлагаемого подхода для учета специфики работы в распределенных средах.

Для описания и сравнительного анализа механизмов действия различных типов алгоритмов генерации планов необходимо формализовать описание системы планирования. Сначала рассмотрим семантику системы планирования действий в динамической среде.

Окружение системы планирования является динамической средой. Описание окружения в некоторый момент дискретного времени называется состоянием системы.

Допускается конечное множество состояний **S** и конечное множество действий **A**, которые могут быть исполнены в среде и (не обязательно) привести к изменению состояния среды. Рассматриваются марковские среды, в которых новое состояние зависит только от предыдущего состояния и не зависит от предыстории. Состояния описываются векторами переменных состояния, где каждая переменная отображает некоторое свойство среды, изменяющееся во времени (например, доступность и загруженность того или иного информационного источника). Моделируемая динамическая система дискретного времени может быть представлена как детерминированный или недетерминированный (в том числе стохастический) конечный автомат.

В случае детерминированного конечного автомата динамическая система определяется функцией переходов между состояниями $f(s_t, a_t)$:

$$f(s_t, a_t) = s_{t+1},$$

где t – дискретное время, $a \in \mathbf{A}$, $s_t \in \mathbf{S}$, $s_{t+1} \in \mathbf{S}$.

Если переменных состояния N и каждая переменная может принимать два или более значений, то тогда существует не менее чем 2^N состояний. В связи с этим, если N велико, то для снижения вычислительной сложности накладывается следующее ограничение: каждая переменная состояния в момент времени t зависит не более, чем от M переменных состояния в момент времени $t-1$, причем $M<<N$. Такое ограничение позволяет разбить функцию $f(.,.)$ на N функций размерности не более, чем M:

$$f(s_t, a_t) = \{g_1(s_t, a_t), g_2(s_t, a_t),...,g_i(s_t, a_t),...,g_N(s_t\ a_t)\},$$

где $g_i(s_t, a_i)$ отображает i-ю переменную состояния в момент времени $t+1$.

Цель G задается как подмножество множества состояний: $G \subseteq \mathbf{S}$. Планом π обозначается частично-упорядоченная по времени исполнения последовательность действий из множества **A**, целью которой является перемещение системы в одно из целевых состояний $g \in G$. Множество возможных планов обозначается **П** и называется пространством планов. После подготовки плана он передается исполнителю. Для учета связи с окружающей средой необходимо располагать информацией, доступной исполнителю плана. Исполнителю могут быть доступны для наблюдения все переменные состояния, часть переменных или только текущее время. Кроме того, часть информации может быть искажена. Вводится множество возможных наблюдений **O**. Информация, доступная в момент времени t в состоянии s_t, определяется с помощью выходной функции h:

$$o_t = h(s_t), \text{ где } o_t \in \mathbf{O}.$$

У исполнителя плана допускается наличие часов и, следовательно, возможности точно определять текущее время. В общем виде система планирования действий в динамической среде представлена на рис. 13.1.1.

Система планирования действий

в динамической среде

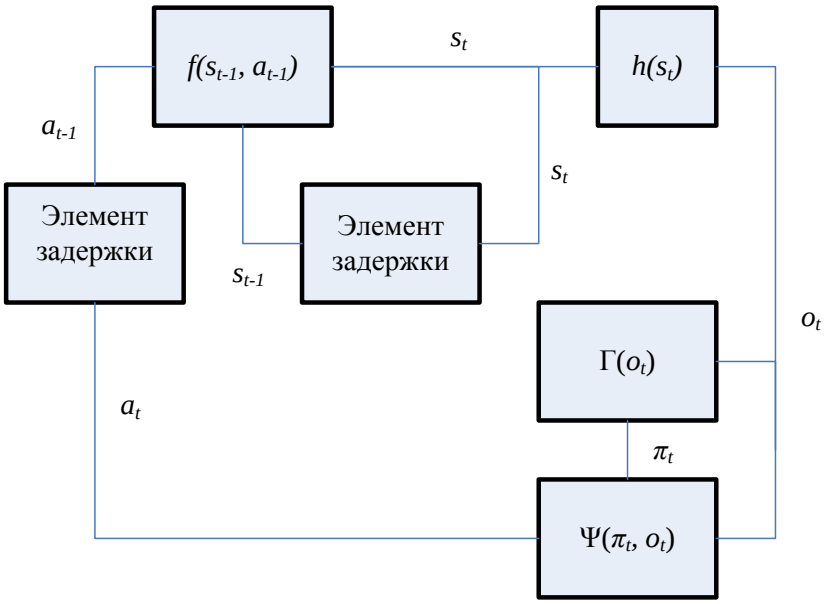


Рис. 13.1.1

На представленном рисунке $\Gamma(o_t)$ – планировщик (planner), который генерирует план на основе текущего наблюдения o_t; π_t – текущий план; $\Psi(\pi_t, o_t)$ – исполнитель плана (plan executor), вырабатывающий на основе текущего плана π_t, и текущего наблюдения o_t текущее действие a_t. Как планировщик, так и исполнитель плана входят в состав интеллектуального агента. Планировщик не обязательно генерирует новый план в ответ на каждое наблюдение среды. Кроме того, планировщик может при необходимости накапливать результаты предыдущих наблюдений.

Для стохастических сред, к которым относятся и гетерогенные сети, вводится функция распределения вероятностей переходов между состояниями:

$$\Pr(f(s_t,o_t) \mid s_t, a_t);$$

и выходная функция распределения вероятностей:

$$\Pr(h(s_t) \mid s_t);$$

В общем случае, для кодирования этих распределений даже в случае булевых переменных состояния требуется машинная память объемом $\underline{O}(2^N)$ бит. Для снижения требуемого объема памяти в случае независимости переменных состояний прибегают к разбиению функции распределения вероятностей переходов между состояниями. Также как и в случае детерминированной среды, накладывается следующее ограничение: каждая переменная состояния в момент времени t зависит не более, чем от M переменных состояния в момент времени t-1, причем $M<<N$. Подмножество переменных состояния в момент времени t-1, от которых зависит i-я

переменная состояния в момент времени t и состоянии s_t, обозначается $Parents(i,s_t)$. Тогда функция распределения вероятностей переходов между состояниями может быть записана следующим образом:

$$\Pr(\{g_1(s_t,a_t),...,g_N(s_t,a_t)\}\,|\,s_t,a_t)=\prod_{i=1}^{N}\Pr(g_i(s_t,a_t)\,|\,Parents(i,s_t),a_t),$$

где $g_i(s_t,a_t)$отображает i-ю переменную состояния в момент времени $t+1$.

В последнем случае для кодирования распределений в случае булевых переменных состояния потребуется память уже объемом $\underline{O}(N*2^M)$ бит.

Для определения производительности плана π вводится понятие траектории пространства состояний или истории h_π Для детерминированных динамических систем h_π представляет собой последовательность состояний (возможно бесконечную), начиная с некоторого состояния s_0, задаваемую планом π. $h_\pi = \{s_0,s_1,\ldots\}$. В общем случае h_π представляет собой случайную величину, распределение которой задается совместно планом и динамической системой. Функция ценности плана V ставит каждой траектории в соответствие вещественное число. В детерминированном случае производительность J плана π равна стоимости истории плана: $J(\pi) = V(h_\pi)$. В общем случае производительность J плана π равна математическому ожиданию стоимости истории плана: $J(\pi) = \mathrm{E}[V(h_\pi)]$. Для задачи планирования большой интерес представляют оценки производительности, ориентированные на достижение планом цели. При этом функция ценности плана может быть определена следующим образом:

$$V(\{s_0,s_1,...\})=\begin{cases}1, \textit{если } \exists i: s_i\in G;\\ 0, \textit{в противном случае.}\end{cases}$$

Альтернативным вариантом является определение ценности плана через минимальное количество шагов до достижения целевого состояния:

$$V(\{s_0,s_1,...\})=\begin{cases}-\min\limits_i s_i\in G, \textit{если } \exists i: s_i\in G;\\ -\infty, \textit{в противном случае.}\end{cases}$$

Для стохастических сред часто вводится ожидаемое время достижения цели как мера производительности плана. В таком случае задачей процесса планирования является минимизация этого времени. Обобщая понятие ожидаемого времени достижения цели, вводят функцию стоимости состояния $C(.)$, которая ставит каждому состоянию в соответствие вещественное число. На основе суммы стоимостей состоянии определяют меру производительности плана:

$$V(\{s_0,s_1,...\})=-\sum_{i=0}^{\infty}C(s_i).$$

При решении некоторых задач (например, при вычислении стационарных политик) для ограничения количества анализируемых шагов вводится дисконтный коэффициент γ:

$0<\gamma<1$, который характеризует, насколько снижается ценность каждого последующего действия. Мера производительности плана с учетом γ называется функцией дисконтной совокупной стоимости плана (discounted cumulative cost) и определяется следующим образом:

$$V(\{s_0, s_1, ...\}) = -\sum_{i=0}^{\infty} \gamma^i C(s_i).$$

В зависимости от конкретных задач системы планирования для оценки производительности плана может быть задействован любой из приведенных выше способов, причем как для определения критерия достаточности при поиске удовлетворительных планов, так и для определения критерия оптимальности при поиске оптимальных (близких к оптимальным) планов.

Перейдем к рассмотрению специфики различных алгоритмов планирования. Если задано пространство планов **П** и функция оценки производительности, то задача планирования сводится к поиску. Предполагается, что вычислительные затраты на оценку плана много меньше затрат на поиск в пространстве планов. Два базовых подхода к решению этой задачи опираются на методы уточнения плана (refinement methods) и методы коррекции плана (repair methods). Любой метод уточнения основывается на наращивании частично определенного плана. Значительная часть исследований по планированию действий в искусственном интеллекте проводилась в области методов уточнения плана. Любой метод коррекции плана пытается трансформировать полностью определенный план в другой полностью определенный план с лучшей производительностью. Методы коррекции зачастую начинают работу с анализа существующего плана с целью определения и улучшения наиболее критичных участков, влияющих на производительность.

Хотя выше упоминалось о том, что для планирования действий интеллектуальных информационных агентов в гетерогенных сетях предпочтительнее использовать стохастическую модель среды, заметим, что методы поиска в пространстве планов для детерминированных моделей могут быть применены после определенных модификаций стохастической модели на основе вероятностных операторов пространства состояний. В связи с этим сначала выделим особенности алгоритмов поиска для детерминированных моделей сред.

Рассмотрим планирование в детерминированных средах в классической постановке. Классическая задача планирования предполагает следующее: дискретность пространства и времени; конечность множества возможных состояний агента и множества действий; детерминированность и полная наблюдаемость среды, изменяющей состояния только под воздействием агента, автономность системы планирования; отсутствие обратной связи.

Перед описанием алгоритмов планирования для детерминированных систем необходимо формализовать представление плана. При формализации рассмотрим динамическую систему в некоторый момент дискретного времени t. Состояние в рассматриваемый момент времени s_t, обозначается для краткости записи s и задается с помощью инициализации множества переменных состояния $\{var_1,...,var_N\}$: $s=\{var_1 = val_1, ..., var_N = val_N\}$, где val_i – значение переменной состояния var_i, $i = 1,2,...,N$. Действие a_t, выполняемое агентом в момент времени t, представляется как оператор пространства состояний α, задаваемый с помощью функции Pre(α) – предусловий или входных

условий (preconditions) и функции Post(or) – постусловий или выходных условий (postconditions). Предусловия и постусловия описываются с помощью инициализации элементов подмножеств множества переменных состояния:

$$\mathrm{Pre}(\alpha)\text{: } pre_1 = pre_val_1,\ldots, pre_K = pre_val_K;$$

$$\mathrm{Post}(\alpha)\text{: } post_1 = post_val_1,\ldots, post_L = post_val_L;$$

где pre_val_i – значение переменной состояния pre_i,

$$pre_i\ \{var_1,\ldots,var_N\}\ ,\ K\leq N,\ i=1,2,\ldots,K,$$

$post_val_j$ – значение переменной состояния $post_j$,

$$post_j \in\{var_1,\ldots,var_N)\ ,\ L\leq N,\ j=1,2,\ldots,L.$$

Если действие выполняется в состоянии, для которого справедливы значения переменных состояния, перечисленные в его предусловиях, то выполняются присвоения, перечисленные в его постусловиях. Если предусловия не выполняются, то состояние не изменяется.

Для формального описания функции переходов между состояниями $f(.,.)$ вводят два оператора инициализации переменных состояния: $\oplus$ и θ. Пусть φ и ν – инициализации двух подмножеств переменных состояния. Выражение $\varphi\ \theta\ \nu$ означает инициализацию подмножества переменных состояния, входящих во множество, соответствующее φ, и не входящих в состав множества, соответствующего ν. Для определения оператора $\oplus$ вводятся понятия совместимости состояний. Инициализации φ и ν несовместимы, если существует такая переменная состояния X, что φ и ν присваивают ей различные значения. В противном случае φ и ν совместимы. Для совместимых φ и ν выражение $\varphi \oplus \nu$ означает инициализацию подмножества переменных, являющегося объединением множеств, соответствующих φ и ν. Теперь можно формально определить функцию переходов между состояниями $f(.,.)$:

$$f(s,\alpha)=\begin{cases} s,\ \textit{если } s \textit{ и } \mathrm{Pre}(\alpha) \textit{ несовместимы}; \\ \mathrm{Post}(\alpha)\oplus(s\ominus\mathrm{Post}(\alpha)),\ \textit{в противном случае} \end{cases}.$$

Для последовательности операторов задается функция $F(.,.)$:

$F(s, [\alpha_1,\alpha_2,\ldots, \alpha_n]) = F(f(s, \alpha_1), [\alpha_2,\ldots, \alpha_n])$,

$F(s, []) = s$.

Для оценки производительности классических алгоритмов планирования наиболее распространена функция ценности, ориентированная на достижение планом цели. Цель представляется как инициализация подмножества переменных состояния. Считается, что состояние s удовлетворяет цели ϕ $(s \Rsh \phi)$ если подмножество состояний, соответствующее инициализации ϕ, входит в состав подмножества, соответствующего инициализации s. Если задано начальное состояние s_0, цель ϕ и множество (библиотека) операторов, то задача планирования

состоит в нахождении такой последовательности операторов пространства состояний $[\alpha_1,\alpha_2,\ldots,\alpha_n]$, для которой справедливо:

$$F(s_0,[\alpha_1,\alpha_2,\ldots,\alpha_n]) \Rightarrow \phi \ .$$

Применение оператора пространства состояний для преобразования одного состояния в другое называется прогрессией. Специфика решаемых задач зачастую требует динамической корректировки целей. Применение оператора для трансформации одной цели в другую называется регрессией. Для формализации описания регрессии вводится символ запрещенной инициализации – $\perp$. Считается, что в том случае, когда для регрессии используется оператор, постусловия которого несовместимы с целью, то трансформированная таким образом цель недостижима.

Функция регрессии цели b описывается следующим образом:

$$b(\phi,\alpha)=\begin{cases}\perp, \textit{ если } \phi \textit{ и } \mathrm{Post}(\alpha) \textit{ несовместимы;}\\ \mathrm{Pre}(\alpha)\oplus(\phi\theta\ \mathrm{Post}(\alpha)), \textit{ в противном случае.}\end{cases}$$

Для представления частично-упорядоченных или частичных планов недостаточно простой последовательности операторов. Вводится множество шагов, каждый из которых ассоциируется с определенным оператором, множество упорядочивающих ограничений (ordering constraints), определяющих порядок выполнения шагов, множество вспомогательных ограничений (auxiliary constraints), блокирующих на определенное количество шагов изменение значений переменных состояний. Каждый шаг обозначается парой (i,α_i), где i – уникальный номер шага, α – оператор, исполняемый на этом шаге. Упорядочивающее ограничение вида $(i \Rightarrow j)$ означает, что шаг i выполняется раньше шага. j. Упорядочивающее ограничение вида $(i \underline{\Rightarrow} j)$ означает, что шаг i непосредственно предшествует шагу j Вспомогательное ограничение в форме $(i\ \underline{P}\ j)$ называется ограничением резервирования интервала (interval preservation constraint, IPC-ограничение) и означает, что между шагами i и j не должно выполняться операторов, изменяющих значение переменной состояния P. Вспомогательное ограничение в форме $(P@j)$ называется ограничением значения (point truth constraint, PTC-ограничение) и означает, что непосредственно перед шагом j не должно выполняться операторов, изменяющих значение переменной состояния P. Весь частичный план представляется следующим образом:

$$[\{\sigma_1, \sigma_2, \ldots, \sigma_d\},$$
$$\{\xi_1, \xi_2, \ldots, \xi_g\},$$
$$\{\zeta_1, \zeta_2, \ldots, \zeta_h\}],$$

где σ_q – элементы множества шагов, $q=1, 2,\ldots, d$;
d – размерность множества шагов;
ξi – элементы множества упорядочивающих ограничений, $i=1, 2,\ldots, g$;
g – размерность множества упорядочивающих ограничений;

ζj – элементы множества вспомогательных ограничений, $j=l, 2,..., h$;
h – размерность множества вспомогательных ограничений.

Множество шагов $\{\delta_1, \delta_2, ..., \delta_m\} \subseteq \{\sigma_1, \sigma_2, ...,\sigma_d\}$ с упорядочивающими ограничениями в форме: $\{(\delta_1 \underline{\Rightarrow} \delta_2), (\delta_2 \underline{\Rightarrow} \delta_3), \ldots, (\delta_{m-1} \underline{\Rightarrow} {}_m)\}$ называется головной частью (header) плана π. Последний элемент δ_m называется головным шагом (head step**).** Состояние, задаваемое $F(s_0, [\beta_1, \beta_2, ..., \beta_m])$, где β_i – оператор, соответствующий шагу δ_i, $i=1,2, ..., m$, называется головным состоянием (head state). Аналогично определяются завершение плана (tail), завершающее состояние (tail state) и завершающий шаг (tail step).

Можно рассматривать уточнение плана как поиск в пространстве частичных планов. Поиск начинается с пустого частичного плана и постепенно наращивает его до завершенного состояния. Алгоритм уточнения плана формирует на основе частичного плана π множество планов $\{\pi_1, \pi_2, ..., \pi_n\}$, хотя бы один из которых включает все решения, возможные на основе исходного плана π. Пустой план, с которого начинаемся поиск представляется как $\left[\{(0,\alpha_0),(\infty,\alpha_\infty)\},\{(0 \Rightarrow \infty),\{\ \}\}\right]$, где α_0 – пустой оператор, непосредственно предшествующий начальному состоянию, α_∞ – пустой оператор, непосредственно следующий за целью. Для α_0 не вводятся предусловия, для α_∞ не вводятся постусловия. В общем виде процедура уточнения плана Refine(π) определяется следующим образом:

1. Если последовательность действий $[\alpha_1, \alpha_2, ..., \alpha_n]$ удовлетворяет упорядочивающим и вспомогательным ограничениям частичного плана π, а также является решением задачи планирования, то завершить выполнение алгоритма и возвратить $[\alpha_1, \alpha_2, ..., \alpha_n]$.
2. Если ограничения нарушены, исключить π из множества рассматриваемых планов.
3. Выбрать стратегию уточнения, применить стратегию к π и включить полученные планы в множество рассматриваемых.
4. Выбрать план π' из множества рассматриваемых и вызвать Refine(π').

Применяемая на третьем этапе стратегия уточнения является классификационным критерием для алгоритмов уточнения плана. Выделяют четыре основные стратегии:

– уточнение в пространстве состояний (state-space refinements);
– уточнение в пространстве планов (plan-space refinements);
– уточнение в пространстве задач (task-reduction refinements);
– смешанное уточнение (hybrid refinements).

Уточнение плана в пространстве состояний является SS-проблемой и основано на наращивании частичного плана – использовании прогрессии для трансформации начального состояния в состояние, удовлетворяющее описанию цели или использованию регрессии для трансформации цели в состояние, удовлетворяющее описанию начального состояния. Формально, уточнение плана в пространстве состояний с использованием прогрессии или прямое уточнение в пространстве состояний состоит в прямом поиске – наращивании головной части плана шагом δ с оператором α_δ , для которого выполняются предусловия в текущем головном состоянии. Шаг δ может быть как уже присутствующим в плане, так и новым, и после размещения в головной части становится новым головным шагом плана. Стратегия уточнения плана в пространстве состояний с использованием

регрессии описывается аналогично и является методом обратного поиска. Недостатком данной стратегии является её высокая вычислительная сложность.

Для снижения вычислительной сложности при использовании уточнения плана в пространстве состояний исследователи часто применяют различные эвристические подходы для выбора наиболее вероятного направления поиска. Наиболее простой подход состоит в выборе направления с минимальной разницей между головным и завершающим состоянием. Однако этот подход пригоден только для стратегий на основе регрессии, так как для сложных сред характерен большой выбор операторов, пригодных для применения в головном состоянии, из которых очень немногие впоследствии приводят к достижению цели. Для снижения вычислительной сложности в этом случае исследователи применяют эвристические подходы, основанные на выявлении перспективных операторов при встречном поиске.

Недостатком стратегии уточнения плана в пространстве состояний является то, что она генерирует только полностью упорядоченные планы. На практике зачастую складываются ситуации, когда несложно определить принадлежность того или иного шага к плану, но трудно заранее сделать вывод о порядке следования шагов. В таких случаях уточнение плана в пространстве состояний может привести к раннему принятию решений о порядке шагов и породить многочисленные возвраты при поиске, а это означает рост вычислительных затрат. Стратегия уточнения в пространстве планов ориентирована на уменьшение возвратов при поиске. Трансформация состояния среды заменяется процедурой подтверждения целей. Предусловие P шага (i, α_i) считается подтвержденным (established), если план содержит некоторый шаг (j, α_i), который предшествует шагу i и постусловия которого удовлетворяют P, а также шаги, которые могут размещаться между i и j, не содержат постусловий, противоречащих P. Доказано, что если подтверждено каждое предусловие каждого шага плана, то такой план является решением задачи планирования. Стратегия уточнения в пространстве планов состоит в выборе предусловия P шага (i, α_i) частичного плана и добавлении шага и ограничений, необходимых для подтверждения P. Недостатком данного подхода является то, что в процессе установки предусловия можно нарушить ранее установленное предусловие, что приводит к увеличению возвратов при поиске. Для избежания этого применяют ограничения резервирования интервала. Когда условие P шага σ установлено с помощью постусловий шага σ', в план добавляется ограничение ($\sigma' \underline{P} \sigma$). Кроме того, вычислительную сложность уменьшают путем сжатия пространства планов, в котором производится поиск. Для этого применяют различные эвристические методы, основанные на специфике среды агента. Например, производится ранжирование постусловий по степени критичности в контексте решаемых агентом задач.

Описанные выше стратегии уточнения плана основаны на одноуровневом представлении действий как примитивных или элементарных операций, которые непосредственно могут исполняться аппаратным обеспечением агента. На практике для сложных сред возникает необходимость оперировать категорией макрозадач или абстрактных действий, которые не могут быть напрямую выполнены аппаратным обеспечением. На работу с задачами при их возможной декомпозиции ориентирована стратегия уточнения в пространстве задач. Уточнение в пространстве задач относится к PR-проблемам. Предполагается наличие у планировщика, наряду со знаниями о

примитивных действиях, описаний абстрактных действий и множества схем приведения (reduction schemas) или фрагментов планов (plan fragments) для замены абстрактных действий. Стратегия уточнения в пространстве состояний получает на входе частичный план π, содержащий как абстрактные, так и примитивные действия, выбирает абстрактное действие σ , и генерирует уточнение плана π для каждой схемы приведения, применимой к σ. Обработка плана проводится до тех пор, пока он не будет состоять только из примитивных действий. Вследствие того, что для многих сред возможна генерация схем приведения, данный подход широко применяется на практике.

Стратегия смешанного уточнения основана на комбинации стратегий уточнения в пространстве планов, пространстве состояний и пространстве задач. Данный подход получил широкое распространение в последнее время в связи с постоянно возрастающей сложностью моделей сред агентов. Разумное сочетание нескольких стратегий уточнения плана позволяет несколько снизить вычислительные затраты. Недостатком данного подхода является отсутствие проработанного механизма выбора стратегий для различных участков плана.

Существенным ограничением области применения методов улучшения плана является то, что они предполагают наличие полной модели динамики системы. Для сложных реальных сред использование такой модели в процессе генерации плана не всегда эффективно. Методы коррекции плана направлены на устранение этого недостатка. Сначала составляется упрощенная модель среды (за счет отбрасывания второстепенных предусловий и постусловий) и к ней применяются методы уточнения плана для получения приблизительного решения. Далее подключаются методы коррекции плана, целью которых является тестирование плана с учетом полной модели динамики системы. Если тестирование проходит успешно, то генерация плана считается завершенной. Если же нет, то план корректируется. Суть корректировки заключается в добавлении и удалении ограничений плана. Методы корректировки используются при планировании на основе примеров (case-based planning), при котором осуществляется накопление решений для наиболее часто встречающихся задач и применение таких библиотечных решений впоследствии для сходных условий.

Рассмотрим специфику алгоритмов планирования действий в стохастических средах. В этом случае план представляется как отображение множества состояний на множество действий: $\pi : S \Rightarrow A$. Для упрощения представления обозначим состояния целыми числами: $S = \{0,1,...,|S|\}$,

где $s_0 = 0$ соответствует начальному состоянию и $|S|$ – размерность множества S. Вероятности переходов между состояниями задаются функцией

$$\Pr(f(i,\pi(i)) = j|i,\pi(i)).$$

Можно привести такое описание к виду, пригодному для использования методов уточнения и коррекции плана в детерминированных средах. Для этого используют вероятностные операторы пространства состояний. Вероятностный оператор пространства состояний α задается как множество элементов в форме (θ,ρ,σ), где θ – множество предусловий, ρ – вероятность перехода, σ – множество постусловий. Если при выполнении оператора справедливы предусловия θ, то с вероятностью ρ производится инициализация σ. Задается пороговая вероятность достижения цели и

ищется план с вероятностью достижения цели выше пороговой. На каждом шаге процедуры уточнения текущая вероятность достижения цели вычисляется с помощью специализированного алгоритма оценки и сравнивается с пороговой. Если текущая оценка превышает пороговую, поиск прекращается, в противном случае производится выбор на основе некоторых критериев очередного действия, приводящего к возрастанию вероятности достижения цели. Существенными ограничениями данного подхода для использования его при планировании действий информационных агентов являются ориентация на построение планов без обратной связи и отсутствие поддержки интерактивного планирования.

При решении задачи планирования в стохастических средах существуют различные методы нахождения оптимального или близкого к оптимальному плана с учетом обратной связи. Для упрощения описания ограничимся полностью обозреваемыми средами, для которых справедливо $h(x_t) = x_t$. В качестве меры производительности плана при этом чаще всего используется функция дисконтной совокупной стоимости плана с дисконтным коэффициентом γ. Вводится производительность плана π; начинающегося в состоянии i как $J(\pi|i)$. Производительность плана описывается $(|S|+1)$ уравнениями с $(|S|+1)$ неизвестными:

$$J(\pi \mid i) = C(i) + \gamma \sum_{j=0}^{|S|} \Pr(f(i,\pi(i)) = j \mid i,\pi(i)) J(\pi \mid j).$$

Целью задачи планирования является нахождение среди множества планов **П** такого плана π, для которого производительность максимальна (оптимального плана):

$$\forall \pi' \in \Pi, J(\pi \mid i) \geq J(\pi'), \quad i = 0,1,...,|S|.$$

Так как искомый план предписывает действия для каждого состояния, то он по определению является политикой. Для поиска оптимальной (близкой к оптимальной) политики существует теоретическая база. Большинство известных алгоритмов поиска опираются на итерационные методы коррекции, которые на каждом шаге изменяют план π на основе вычисленных $J(\pi|i)$ для $i=0,1,...,|S|$. Поскольку, как уже отмечалось ранее, $|S|$ экспоненциально зависит от количества переменных состояния, решение системы $(|S|+1)$ уравнений приводит к большим вычислительным затратам для сложных сред. В связи с этим далее рассмотрим способы преодоления указанного недостатка.

Предположим, что задано начальное состояние s_0 и максимальная стоимость состояния C_{max}:

$$C_{\max} \geq \max(C(i)), \quad i = 0,1,...,|S|.$$

Обозначим $J_\infty(\pi)$ производительность политики π; рассчитанной на бесконечное количество шагов (стационарной политики) и начатой в состоянии s_0. Компромиссной альтернативой по отношению к вычислению производительности стационарной политики является вычисление производительности политики за K шагов $J_K(\pi)$. Доказано, что отклонение производительности, ограниченной K шагами политики от стационарной, может быть оценено следующим соотношением:

$$| J_{\infty}(\pi) - J_K(\pi) | \le \gamma^K C_{\max} /(1-\gamma).$$

Производительность, ограниченной *K* шагами политики, может быть получена на основе усреднения результатов некоторого количества экспериментов по построению планов длины *K*. Пусть необходимо вычислить производительность со следующими ограничениями:

$$\Pr[J_K(\pi)(1-\varepsilon) \le J_K(\pi) \le J_K(\pi)(1+\varepsilon)] > 1-\delta,$$

где ε – точность расчетов, $0 < \varepsilon < 1$;
δ – ненадежность расчетов, $0 < \delta < 1$.

Доказано, что математическое ожидание необходимого количества экспериментов E[*T*] (а следовательно, и объем вычислительных затрат) может регулироваться на основе компромисса между точностью и надежностью:

$$\mathrm{E}[T] < 4 log(2/\delta)(1+\varepsilon)(J_K(\pi)\varepsilon^2)^{-1}.$$

Большинство итерационных методов вычисления политик на основе коррекции плана на каждом шаге оценивает производительность политики и затем пытается улучшить политику. По этой причине для снижения вычислительной сложности требуется устанавливать ограничение и на количество итераций.

Недостатком итерационных методов вычисления политик на основе коррекции плана является то, что для них, как и для метода на основе вероятностных операторов пространства состояний, не проработан вопрос поддержки интерактивного планирования.

Представленная задача планирования действий в общем случае является PSPACE-полной по вычислительной сложности. Это означает экспоненциальное в наихудшем случае время работы планировщика. В связи с этим для представляющих практический интерес задач требуется дополнительная адаптация технологий планирования с целью повышения их эффективности.

Один из способов повышения производительности планировщика основывается на применении методов планирования на основе примеров. Представляется перспективным использовать данный подход для снижения вычислительной сложности задачи планирования действий интеллектуальных информационных агентов в гетерогенных сетях, так как специфика взаимодействия этих агентов с информационными источниками заключается, в том числе, в возможности выполнения запросов по стандартным правилам, известным для каждого источника.

Таким образом, можно выделить следующие перспективные подходы для планирования действий интеллектуального информационного агента в гетерогенных сетях: стремление к нахождению интерактивных частично-упорядоченных универсальных планов (политик) с обратной связью; использование иерархических нелинейных алгоритмов с поздним подтверждением в условиях стохастической среды; планирование с учетом возможностей по распараллеливанию действий.

13.2 ОБОБЩЁННЫЙ АЛГОРИТМ ПЛАНИРОВАНИЯ ДЕЙСТВИЙ ИНТЕЛЛЕКТУАЛЬНЫХ АГЕНТОВ

Классические алгоритмы планирования и их расширения на случай стохастических сред могут быть представлены как частные случаи обобщенного алгоритма поиска методом уточнения плана. Данный обобщенный алгоритм и его вариации являются основой для сравнительного анализа классических алгоритмов планирования. Авторами обобщенного алгоритма акцентируется важность исследований поведения классических алгоритмов планирования и их расширений в различных средах. Подобный сравнительный анализ для гетерогенных информационных сред представляет значительный теоретический и практический интерес. Практическая значимость обусловлена необходимостью разработки для интеллектуальных агентов механизмов планирования сбора информации в гетерогенных средах. Действительно, задача сбора информации в современных гетерогенных сетях требует учета большого количества информационных источников и экономии вычислительных ресурсов, а это приводит к необходимости уделять особое внимание реализации механизма планирования действий, в том числе и обоснованному выбору базовых алгоритмов планирования. Рассмотрим планирование как задачу поиска в пространстве планов (пространстве возможных решений) методом уточнения плана. Пространство, в котором ведется поиск, состоит из кандидатов – возможных решений, представляющих собой полностью упорядоченные последовательности элементарных операций, удовлетворяющие ограничениям частичного плана. При использовании этого метода поиск ведется в пространстве частичных планов с помощью добавления ограничений и разрешения конфликтов ограничений. Классические алгоритмы планирования и их расширения могут быть описаны с помощью обобщенного алгоритма планирования *Refine-Plan,* основанного на методе уточнения плана.

Задача планирования представляется как пара [I,G], где I – множество постусловий, которым должно удовлетворять начальное состояние, G – множество предусловий, которым должно удовлетворять конечное состояние. Планом P является набор [T,O,B,ST,L], где T – множество шагов плана; ST – символьная таблица, ставящая в соответствие каждому шагу выполняемый на этом шаге оператор; O – множество упорядочивающих ограничений, налагаемых на множество шагов T; B – множество ограничений инициализаций переменных, присутствующих в предусловиях и постусловиях операторов; L – множество вспомогательных ограничений. Для задачи планирования вводятся следующие ограничения: модель среды имеет конечное число состояний, для этой модели среды ведется поиск таких решений, которые содержат количество шагов, не превышающее заданной константы l_m.

Каждый шаг (кроме начального и конечного) обозначается t_i , где i>0 – уникальный индекс. Начальный шаг обозначается t_0 и непосредственно предшествует начальному состоянию, конечный шаг обозначается t_∞ и непосредственно следует за конечным состоянием. Во множестве ST всегда присутствуют ограничения, связывающие шаг t_0 с псевдооператором *start,* и шаг t_∞ с псевдооператором *fin.* Множество L может содержать два типа ограничений:

ограничения резервирования интервала (IPC-ограничения) и ограничения значений (PTC-ограничения). IPC-ограничения относятся к монотонным вспомогательным ограничениям (то есть ограничениям, которые запрещают появление операторов, меняющих значение ограничиваемой переменной на недопустимое), а PTC-ограничения относятся к немонотонным вспомогательным ограничениям (то есть ограничениям, которые позволяют появление операторов, меняющих значение ограничиваемой переменной на недопустимое, если после них последует оператор, присваивающий этой переменной допустимое значение). Множество ограничений инициализаций переменных *B* состоит из совместных инициализаций (требования одновременного присвоения значений для пары переменных) и несовместных инициализаций (запрещения одновременного присвоения значений для пары переменных).

Обобщенный алгоритм *Refine-Plan* для нахождения плана *P* при заданном множестве целевых предусловий (целей) *A* представляется логической схемой, изображенной на рис.13.2.1 – 13.2.4.

В отдельных реализациях не действуют шаги 1.3. и 3. На шаге 3 осуществляется проверка целостности плана. План считается целостным, если он имеет хотя бы одну допустимую элементарную линеаризацию. Элементарная линеаризация плана (ground linearization) – полностью упорядоченная последовательность шагов плана *P*, удовлетворяющая множествам ограничений *O* и *B*. Элементарная линеаризация считается допустимой (safe), если она удовлетворяет монотонным ограничениям (IPC-ограничениям).

В алгоритме *Refine-Plan* вызываются следующие процедуры:

Sol(P,G) – функция построения решения;

pick-prec – функция выбора предусловий из *A* для удовлетворения (подтверждения);

interacts? – функция для проверки упорядочивающих ограничений.

Для обеспечения принятия решений на различных шагах обобщенного алгоритма исследователи часто используют MTC-критерий – Modal Truth Criterion, модальный критерий истины. При использовании MTC-критерия на шаге 0 обобщенного алгоритма возвращается логическое значение «Истина», если только все допустимые элементарные линеаризации плана являются решениями задачи. При использовании подобной стратегии на шаге 1 обобщенного алгоритма цель выбирается только в том случае, если она еще не истинна согласно MTC-критерию.

На основе представленного алгоритма можно провести сравнительную оценку различных методов поиска решений при частично-упорядоченном планировании. Для оценки производительности алгоритмов планирования выделим следующие критерии:

T_{rp} – среднее время однократного обращения к алгоритму;

N_{rev} – среднее количество обращений к алгоритму;

$T_{sum} = T_{rp} * N_{rev}$ – средние суммарные временные затраты на поиск всех решений.

Отметим, что средние суммарные временные затраты на поиск всех решений являются величиной, обратно пропорциональной производительности алгоритма планирования.

Алгоритм *Refine-Plan*([*P*:[*T*,*O*,*B*,*ST*,*L*],*A*])

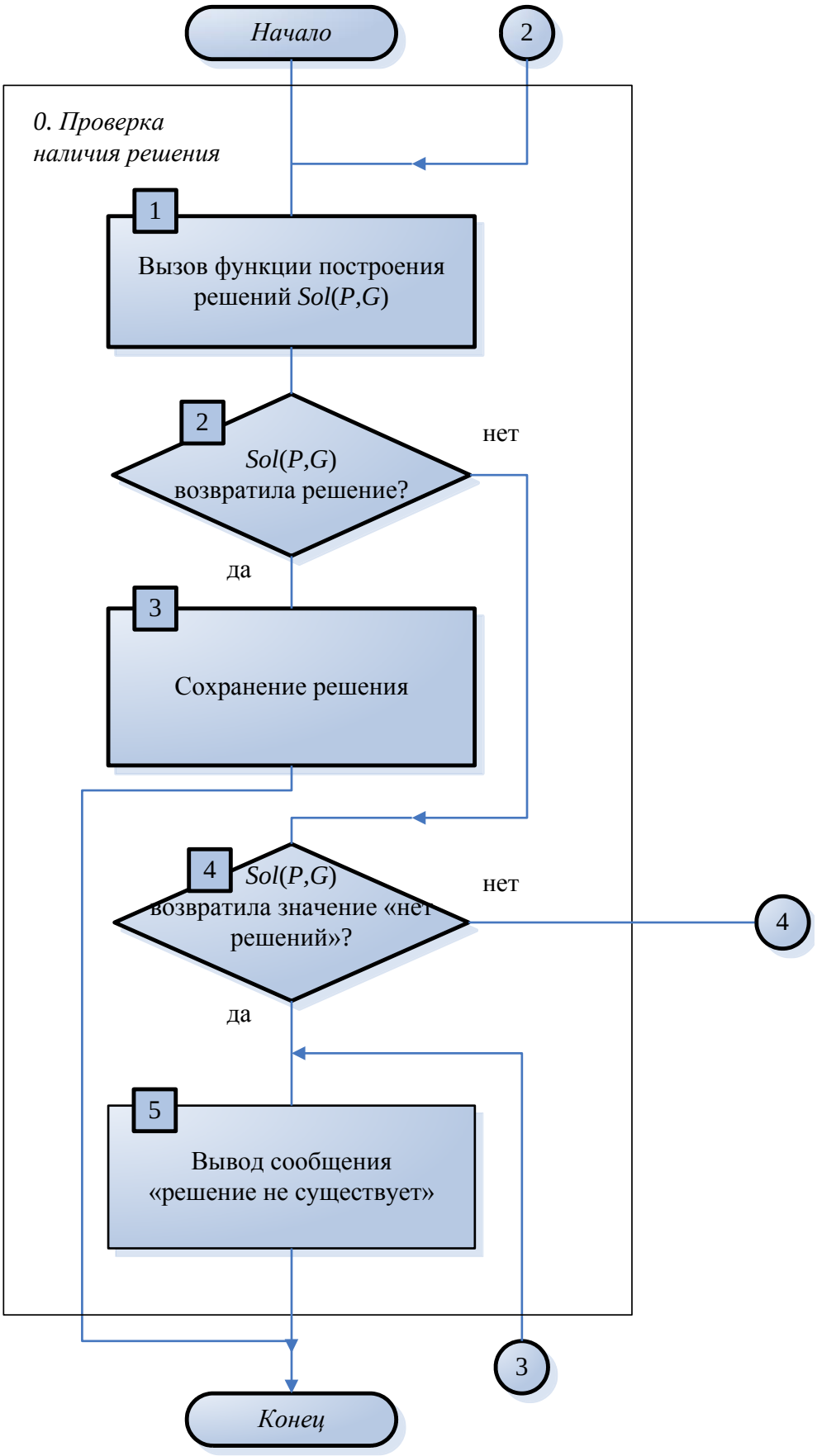


Рис.13.2.1

Алгоритм *Refine-Plan*([*P*:[*T*,*O*,*B*,*ST*,*L*],*A*]) (продолжение)

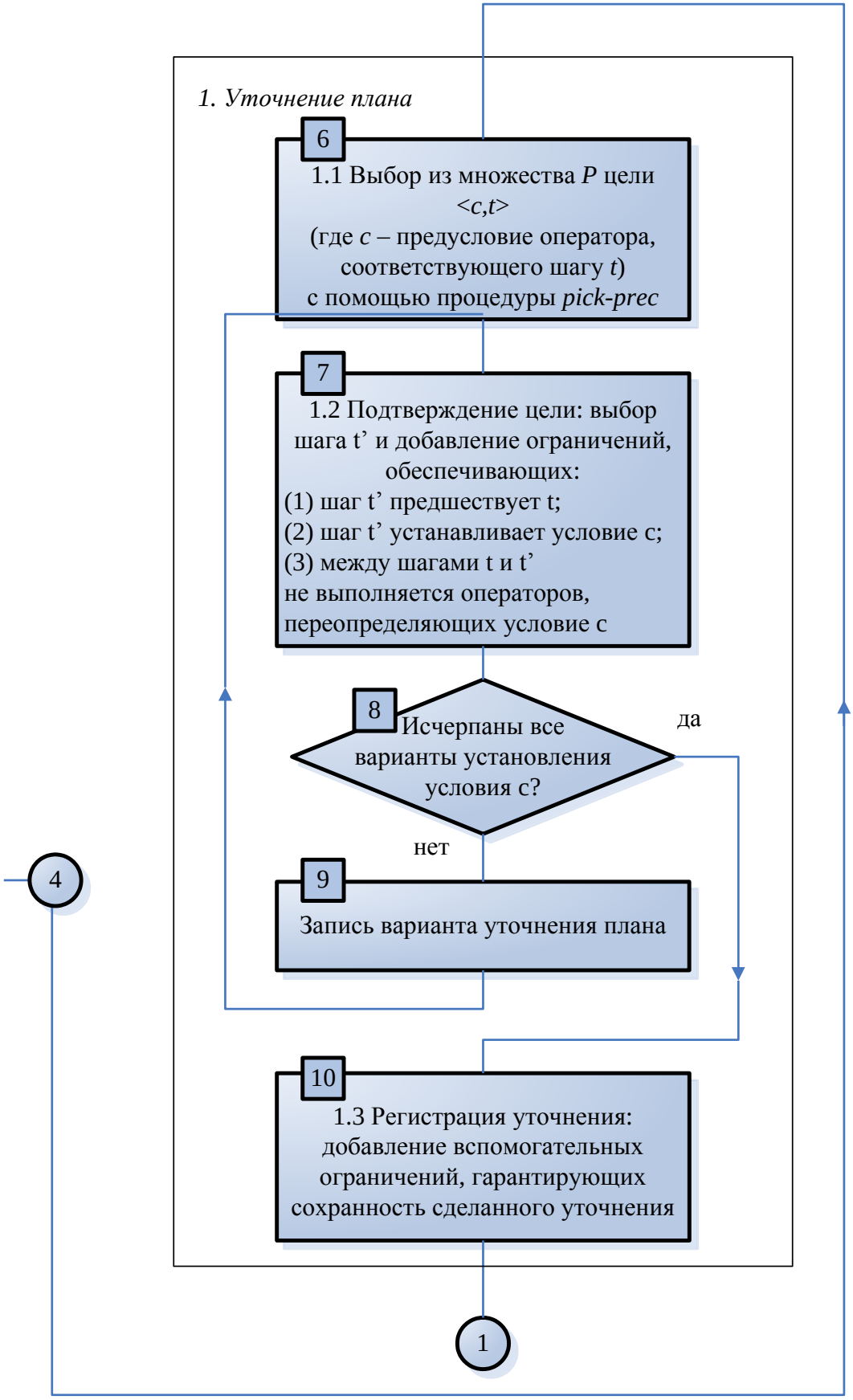


Рис.13.2.2

Алгоритм *Refine-Plan*([*P*:[*T*,*O*,*B*,*ST*,*L*],*A*])

(продолжение)

Рис. 13.2.3

Алгоритм *Refine-Plan*([*P*:[*T*,*O*,*B*,*ST*,*L*],*A*])

(продолжение)

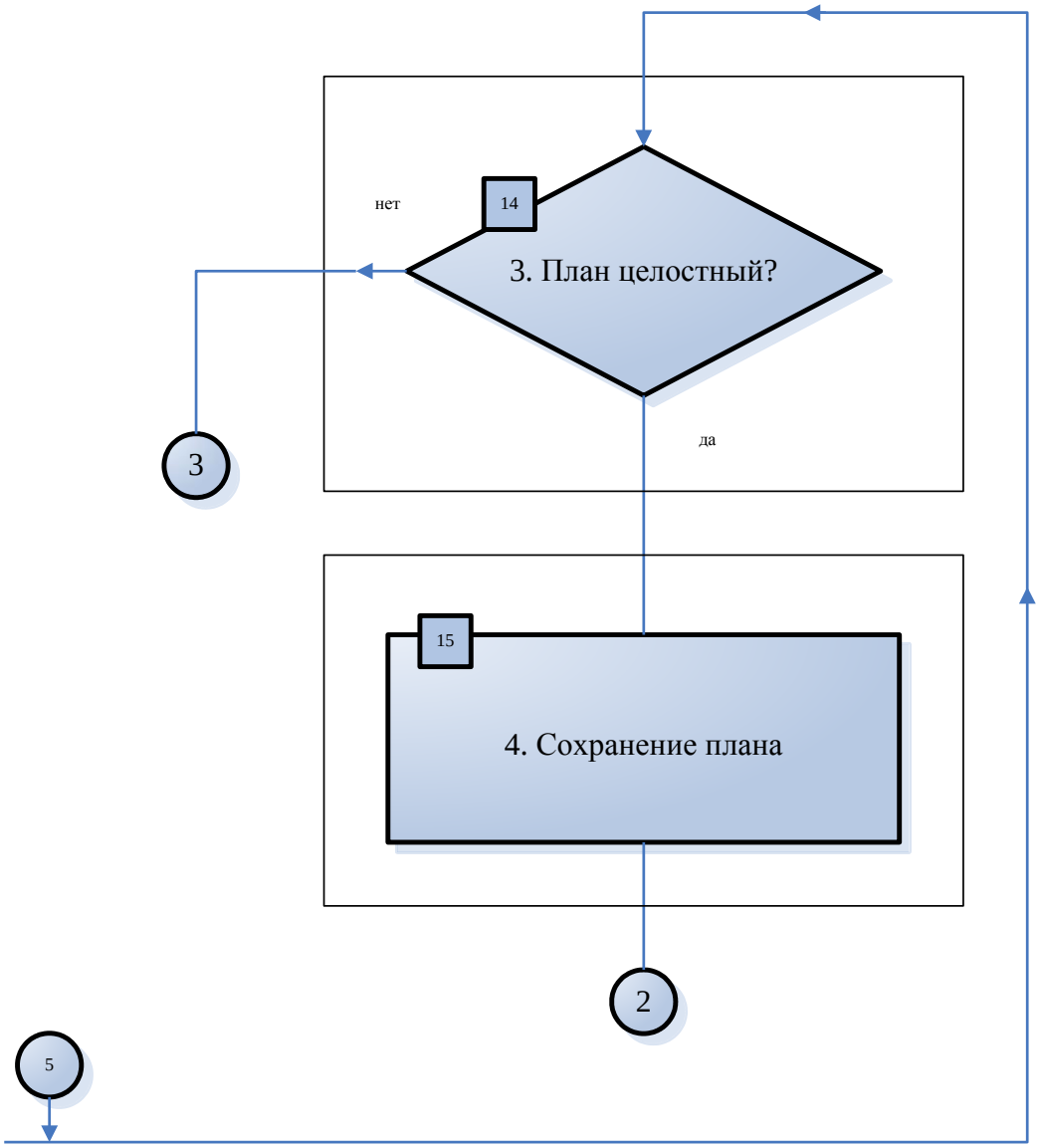


Рис. 13.2.4

13.3 АЛГОРИТМ SNLP

Алгоритм SNLP для нахождения плана *P* при заданном множестве целевых предусловий *A* можно представить на основе обобщенного алгоритма *Refine-Plan*. Схема алгоритма SNLP приведена на рис.13.3.1 и 13.3.2. Алгоритм SNLP вызывает следующие функции и процедуры:

Функция *SNLP_Sol(P,G):*

Если множество *A* пусто, возвратить решение. Если множество *A* не пусто и длина плана не превысила 1_m ,возвратить значение «продолжить». Если множество А не пусто и длина плана превысила l_m , возвратить значение «нет решений».

Процедура *SNLP_pick-prec:*

Произвольным образом выбрать из множества целевых предусловий *A* цель <*c,t*> (где *c* – предусловие оператора, соответствующего шагу *t).* Убрать цель <*c,t*> из множества целевых предусловий *A (A = A* - <*c,t*>).

Процедура *SNLP_establish:*

Выбрать существующий или новый шаг *t'* для <*c,t*>, устанавливающий условие *c* перед шагом *t* (если шаг *t'* не существует и не может быть добавлен, вернуться в предыдущую точку возврата). (1) Дополнить множество шагов *T* шагом *t* $(T = T \cup \{t\})$. Дополнить множество упорядочивающих ограничений *O* ограничением $\{t' \Rightarrow t\}$, гарантирующим выполнение шага *t'* перед шагом *t* $(O = O \cup \{t' \Rightarrow t\})$. (2) Дополнить множество ограничений инициализаций переменных *B* множеством *B',* состоящим из совместных и несовместных инициализаций, содержащихся во множестве *precond(t')* – множестве предусловий выполнения шага *t'* $(B = B \cup B')$. (3) Для каждого шага *t''*, выполняемого между *t'* и *t* и нарушающего условие *c*, выполнить два уточнения плана: первое – с помощью ограничения $O = O \cup \{t \Rightarrow t''\}$, и второе – с помощью ограничения $O = O \cup \{t'' \Rightarrow t'\}$ Если шаг *t'* ранее не встречался в плане, то дополнить множество целевых предусловий *A* и множество дополнительных ограничений *L:*

$$A = A \cup \{< c', t' >\} \mid \forall c' \in precond(t')\};$$

$$L = L \cup \{< c'@t' >\} \mid \forall c' \in precond(t')\}.$$

Процедура *SNLP_save:*

Добавить вспомогательные ограничения, гарантирующие сохранность сделанного уточнения плана с использованием стратегии защиты с помощью уточнений (contributor protection strategy), которая обеспечивает систематичность поиска):

$$L = L \cup < t', c, t > \cup < t', \neg c, t >.$$

Процедура *SNLP_tract:*

Используется способ 2.b, позволяющий разрешать конфликты. (1) Определить конфликты. Шаг t_{conf} считается конфликтующим с IPC-ограничением $<t_1, p, t_2>$, если $t_{conf} \in T, <t_1, p, t_2> \in L$, причем шаг t_{conf} может быть выполнен между шагами t_1 и t_2, и выполнение t_{conf} нарушит условие p.

(2) Разрешить конфликты. Для каждого конфликта, состоящего из шага t_{conf} и IPC-ограничения $<t_1, p, t_2>$, произвести два уточнения плана, первое – с помощью ограничения $O = O \cup \{t \Rightarrow t_1\}$, а второе – с помощью ограничения $O = O \cup \{t_2 \Rightarrow t\}$.

Функция *SNLP_check:*

Если план нецелостный, возвратить значение «нецелостный план», иначе – возвратить значение «целостный план». Нарушением целостности плана является 1) наличие циклов в порядке следования операторов (проверка множества упорядочивающих ограничений O на наличие замкнутых цепочек вида: $\{t_i \Rightarrow t_j \Rightarrow \mathrm{L} \Rightarrow t_i\}$ или 2) наличие для пары переменных одновременно совместной и несовместной инициализации (проверка множества ограничений инициализаций переменных B).

Алгоритм *SNLP*([*P*:[*T*,*O*,*B*,*ST*,*L*],*A*])

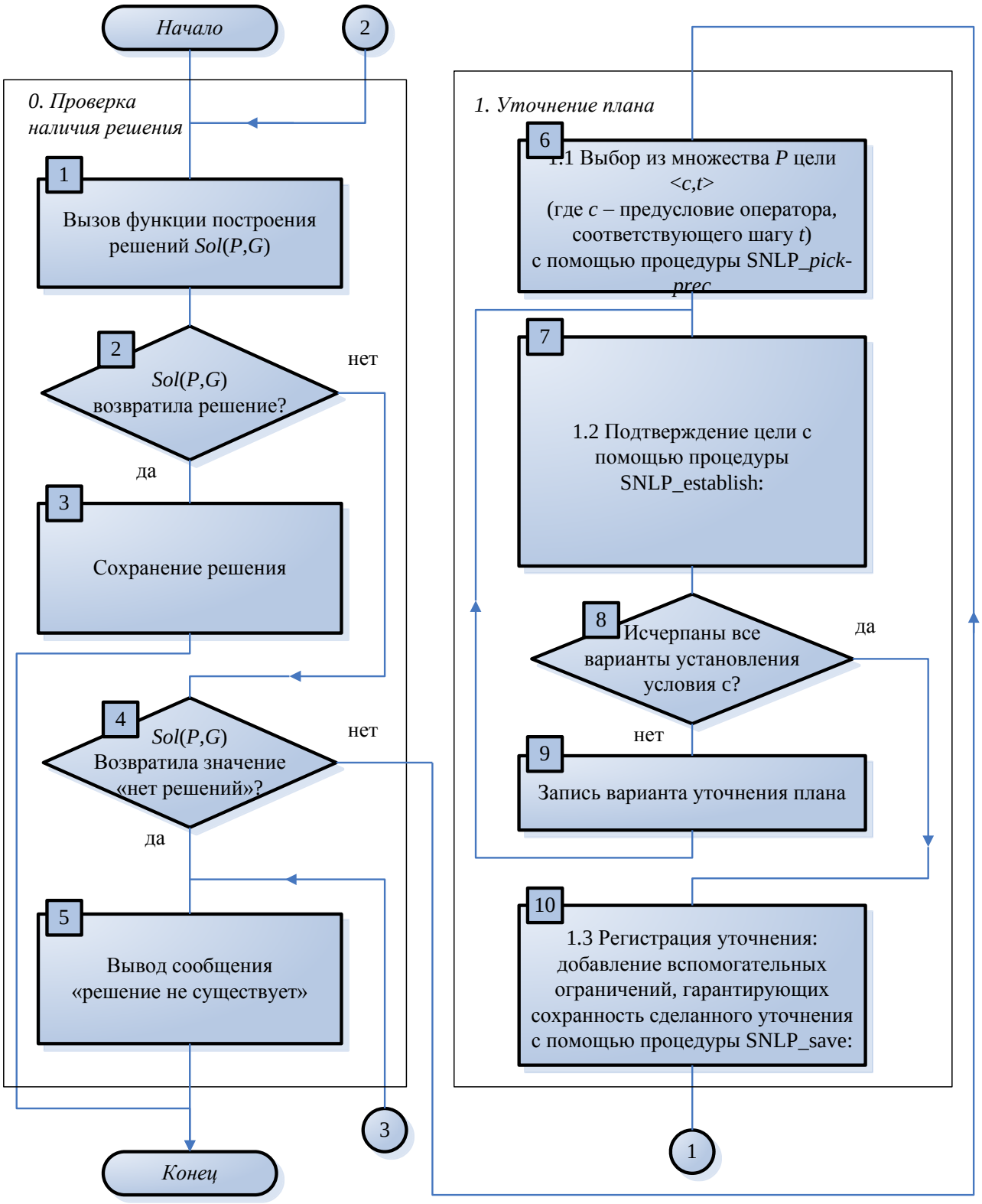


Рис.13.3.1

Алгоритм *SNLP*([*P*:[*T*,*O*,*B*,*ST*,*L*],*A*])
(продолжение)

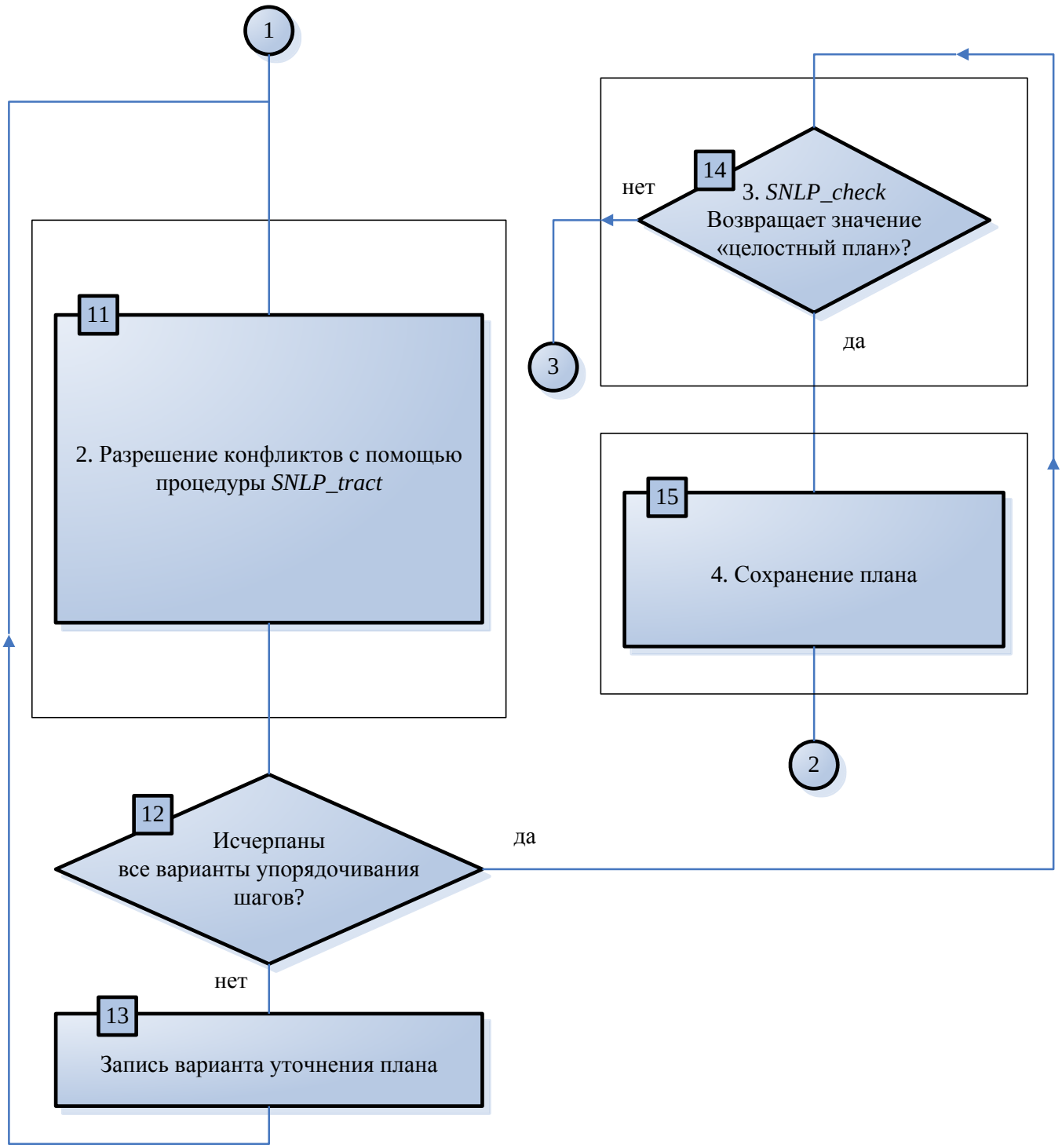


Рис.13.3.2

13.4 АЛГОРИТМ NONLIN

Алгоритм SNLP является развитием семейства алгоритмов NONLIN. Принципиальным различием между NONLIN и SNLP является использование ими различных стратегий при регистрации уточнений (шаг 1.3). В NONLIN используется стратегия защиты интервала (interval preservation strategy). Следует заметить, что используемая в NONLIN стратегия может приводить к более быстрому нахождению решений для некоторых задач, так как стратегия проверки целостности SNLP может приводить к более частым возвратам. Но для других задач поиск с помощью NONLIN может быть более долгим и избыточным (используемая стратегия не гарантирует систематичности поиска). Для анализа отличий алгоритмов при решении задач планирования используется McNONLIN – разновидность алгоритма NONLIN, описание которого на основе обобщенного алгоритма Refine-Plan полностью идентично описанию алгоритма SNLP за исключением шага 1.3. При регистрации уточнений на шаге 1.3. вместо процедуры *SNLP_save* вызывается процедура *NONLIN_save.*

Процедура *NONLIN_save:*

Добавить вспомогательные ограничения, гарантирующие сохранность сделанного уточнения плана с использованием стратегии защиты с помощью уточнений (contributor protection strategy), которая не обеспечивает систематичность поиска:

$$L = L \cup < t', c, t > .$$

13.5 АЛГОРИТМ TWEAK

Основное отличие TWEAK от SNLP и NONLIN заключается в том, что TWEAK не использует ограничения для регистрации уточнений плана. Таким образом, TWEAK не задействует ни шаг 1.3., ни шаг 2 обобщенного алгоритма. Кроме того, TWEAK использует МТС-критерий для оценки того, что является ли целевое предусловие $<c,t> \in A$ необходимо истинным (то есть справедливо для всех элементарных линеаризаций плана). При выборе цели (шаг 1.1.) предпочитаются целевые предусловия, для которых не выполняется МТС-критерий. Проверка наличия решения (шаг 0.) завершается успешно в том случае, если МТС-критерий выполняется для всех целевых предусловий из множества *A*. Схема алгоритма TWEAK на основе обобщенного алгоритма Refine-Plan приведена на рис. 13.5.1 и 13.5.2. Далее приводятся описания процедур и функций, вызываемых алгоритмом TWEAK и отличных от процедур и функций, приведенных в первом случае.

Функция *TWEAK_Sol(P,G):*

Если каждая цель $<c,t>$ (где c – предусловие оператора, соответствующего шагу t) из множества целевых предусловий A необходимо истинна согласно МТС-критерию, то возвратить решение. Если нет и длина плана не превысила l_m, то возвратить значение «продолжить». Если нет и длина плана превысила l_m, то возвратить значение «нет решений».

Процедура *TWEAK_pick-prec:*

Произвольным образом выбрать из множества целевых предусловий A цель $<c,t>$, для которой не выполняется МТС-критерий. Не удалять цель $<c,t>$ из A.

Из описания следует, что TWEAK не исключает рассматриваемые цели из множества целевых предусловий A и поэтому теоретически может рассмотреть одно и то же целевое предусловие более одного раза. Кроме того, хотя TWEAK не уменьшает вычислительную сложность (не задействует шаг 2), за счет использования МТС-критерия алгоритм сохраняет свойство полноты.

Алгоритм *TWEAK*([*P*:[*T*,*O*,*B*,*ST*,*L*],*A*])

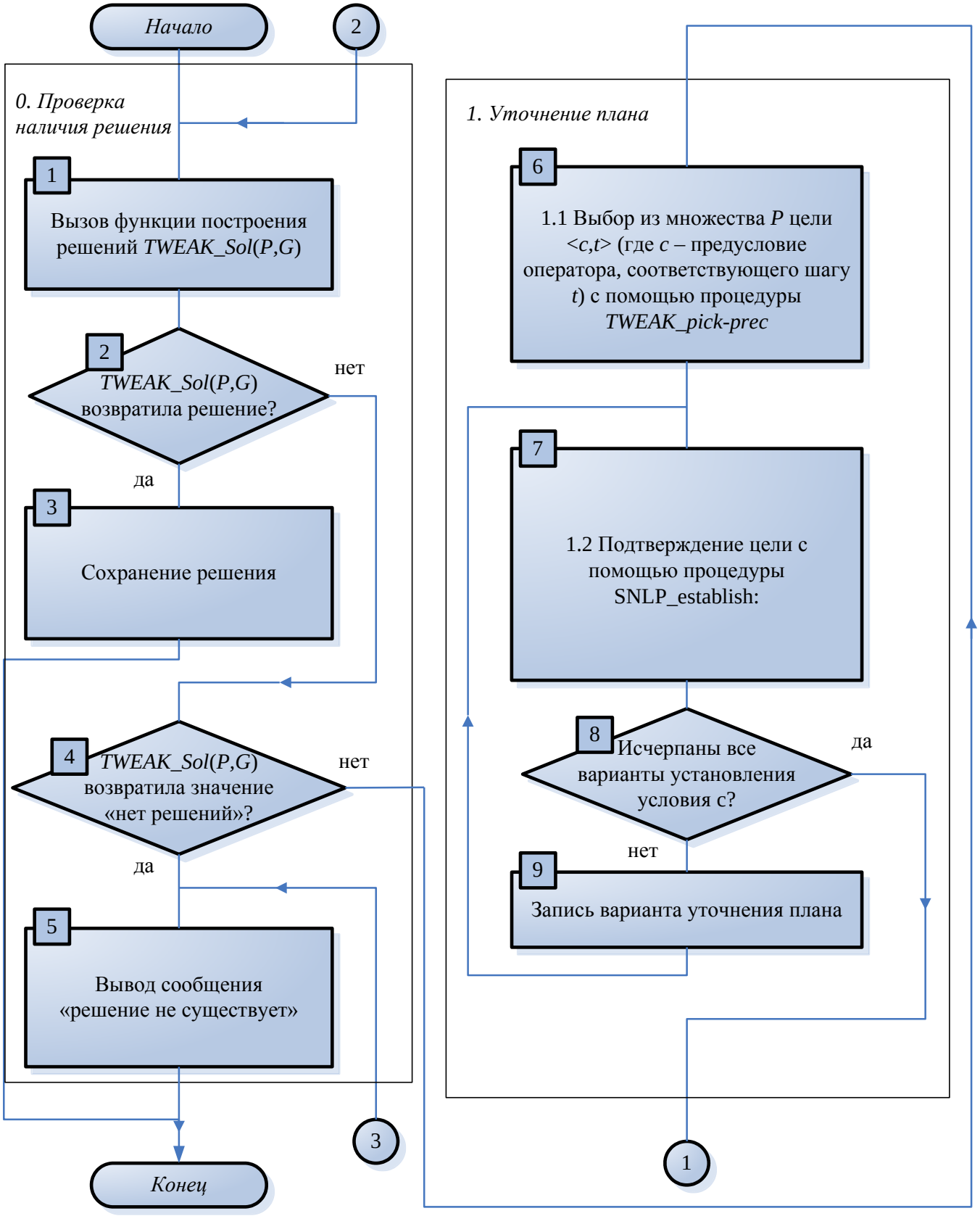


Рис.13.5.1

Алгоритм *TWEAK*([*P*:[*T*,*O*,*B*,*ST*,*L*],*A*])
(продолжение)

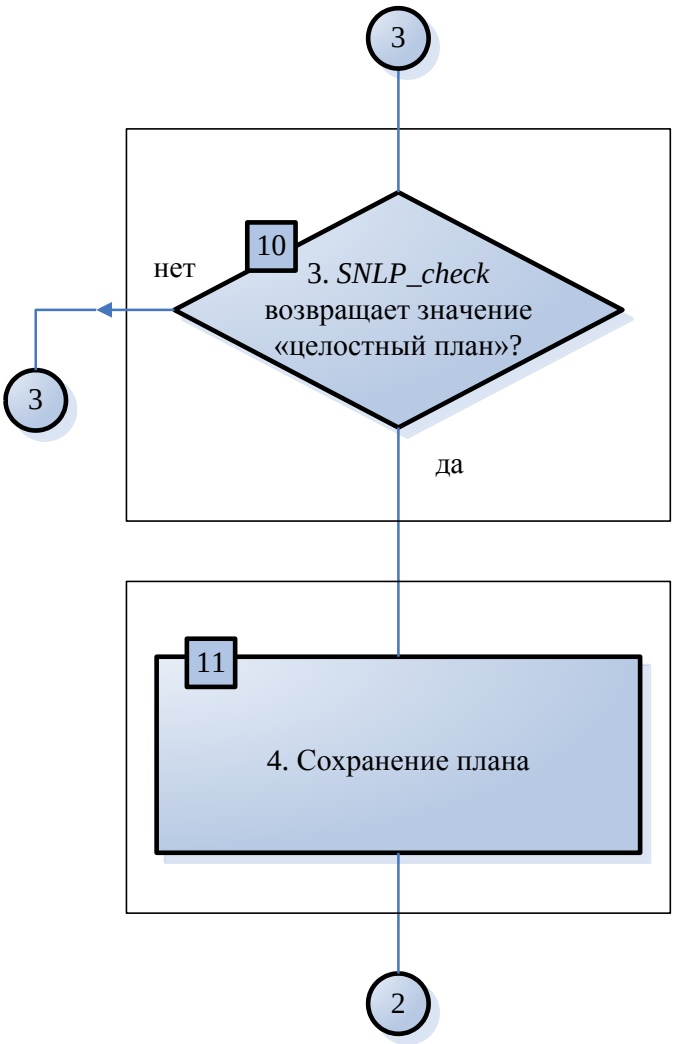


Рис.13.5.2

13.6 АЛГОРИТМ UA

UA во многом аналогичен алгоритму TWEAK, Он также не регистрирует уточнения плана и использует аналогичные стратегии выбора цели и проверки наличия решения. Принципиальным отличием между UA и TWEAK является только наличие в UA шага уменьшения вычислительной сложности (шаг 2.), на котором в UA используется предварительное упорядочивание (шаг 2.а. обобщенного алгоритма). Таким образом, для описания UA на основе обобщенного алгоритма достаточно взять за основу описание алгоритма TWEAK и заменить процедуру, вызываемую на шаге 2 для уменьшения вычислительной сложности, на процедуру *UA_tract.*

Процедура *UA_tract:*

Используется способ 2.а. – предварительное упорядочивание. (1) Определение пересечений. Пара шагов плана t_1 и t_2 считаются пересекающимися, если они не упорядочены относительно друг друга (множество *O* не содержит ни упорядочивающего ограничения $\{t_1 \Rightarrow t_2\}$, ни упорядочивающего ограничения $\{t_2 \Rightarrow t_1\}$ и выполняется одно из следующих трех условий:

t_1содержит предусловие *p*, а t_2 содержит постусловие *p* или $\neg p$;
t_2 содержит предусловие *p,* а t_1 содержит постусловие *p* или $\neg p$;
t_1 содержит постусловие *p*, а t_2 содержит постусловие $\neg p$.

На шаге 2.а. (1) требуется найти все шаги *t*', пересекающиеся с шагом *t.*

(2) Разрешение пересечений. Для каждого шага *t*', пересекающегося с шагом *t,* требуется добавить либо упорядочивающее ограничение $\{t \Rightarrow t'\}$, либо упорядочивающее ограничение $\{t' \Rightarrow t\}$. Для гарантирования полноты поиска необходимо рассмотреть оба варианта.

Из описания стратегии UA на шаге 2. следует, что все частично-упорядоченные планы, генерируемые UA, являются однозначными *(unambiguous),* то есть каждое целевое предусловие $< c,t >$ либо необходимо истинно (то есть справедливо для всех элементарных линеаризаций плана), либо необходимо ложно (то есть не выполняется для всех элементарных линеаризаций плана). В связи с этим для проверки МТС-критерия в UA достаточно исследовать только одну элементарную линеаризацию плана.

13.7 СПЕЦИФИЧЕСКИЕ ПОКАЗАТЕЛИ КАЧЕСТВА АЛГОРИТМОВ ПЛАНИРОВАНИЯ ДЕЙСТВИЙ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ АГЕНТОВ

1. Степень ветвления дерева поиска v_b: среднее количество ветвлений в одной вершине. В общем случае, степень ветвления является произведением двух составляющих:

$$v_b = b_{уточ}\, b_{опт}\ ,$$

где:

$b_{уточ}$ – количество способов достижения выбранной подцели на этапе уточнения частичного плана;

$b_{опт}$ – количество вариантов, получаемых из уточненного частичного плана на этапе оптимизации. Если алгоритм не использует промежуточную оптимизацию, то $b_{опт}$ =1.

2. Глубина дерева поиска v_d, необходимая для нахождения решения. Пусть S – количество шагов в решении задачи планирования, N_{pre} – среднее количество предусловий одного оператора. Тогда общее количество целей, которые необходимо достичь в процессе нахождения решения, будет пропорционально $N_{pre}\, S$. Поскольку алгоритмы планирования, не использующие этап консервации, могут рассматривать одну и ту же цель многократно, необходимо ввести корректирующий коэффициент γ, соответствующий среднему количеству рассмотрений каждой цели в процессе решения задачи. Тогда значение v_d может быть оценено следующим образом:

$$v_d \approx \gamma\, N_{pre}\, S\ .$$

3. Степень избыточности дерева поиска v_r. Как было отмечено, в общем случае алгоритм планирования может рассматривать одно и то же решение в разных ветвях дерева поиска. Степень избыточности v_r определяется как отношение суммы числа потенциальных решений, соответствующих каждой вершине в F, к общему количеству потенциальных решений (количеству элементов в K). При $v_r=1$ каждому потенциальному решению отвечает единственная вершина в F, значение $v_r>1$ означает, что некоторое потенциальное решение может быть получено из нескольких вершин. Если алгоритм планирования обладает свойством систематичности, то для него всегда $v_r=1$.
4. Показатель состоятельных вершин v_g, определяемый как отношение непустых вершин в F к общему количеству элементов в F. Появление пустых вершин на более ранних уровнях дерева поиска способствует значительному сокращению общего числа вершин и, следовательно, размеру пространства поиска. Наибольшее влияние на данный показатель оказывает использование алгоритмов планирования шагов оптимизации.

   Принимая во внимание специфику реализации рассматриваемых алгоритмов планирования, значение v_g может быть определено следующим образом:

$$v_g \approx \frac{v_{B_a}}{v_B},$$

где через v_{Ba} обозначено количество вершин дерева поиска, в которых производится хотя бы один рекурсивный вызов алгоритма планирования, а через v_B – общее число вершин.

5. Вычислительная стоимость (отождествляемая в простейшем случае с временем выполнения) каждого шага алгоритма в отдельности:
   - стоимость выполнения конструктора решений $C_{кон}$;
   - стоимость уточнения частичного плана, состоящая из стоимостей выбора и достижения подцели, а также консервации сделанных изменений: $C_{уточ}=C_{вы} + C_{дост} + C_{конс}$;
   - стоимость оптимизации частичного плана $C_{опт}$;
   - стоимость проверки состоятельности частичного плана $C_{сост}$.

   Вычислительная стоимость одного вызова алгоритма определяется как совокупность стоимостей выполнения отдельных шагов:

$$C_{алг}= C_{кон} + C_{уточ} + C_{опт} + C_{сост}.$$

13.8 УНИВЕРСАЛЬНЫЕ ПОКАЗАТЕЛИ КАЧЕСТВА АЛГОРИТМОВ ПЛАНИРОВАНИЯ ДЕЙСТВИЙ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ АГЕНТОВ

Универсальные показатели оценивают наиболее важные с прикладной точки зрения аспекты работы алгоритмов планирования (а также систем, их реализующих) и являются применимыми практически ко всем системам планирования. К подобным показателям относятся показатели, отражающие функциональность, качество решения, вычислительную сложность используемого алгоритма, быстродействие, предсказуемость.

1. Функциональность трактуется как способность системы находить решения поставленных перед ней задач планирования. Показатель v_F, представляющий функциональность, можно определить как отношение числа задач, правильно решенных системой, к общему числу поставленных задач.
2. Качество предлагаемого решения v_Q представляется группой показателей, отражающих свойства найденного плана, среди которых:
   - Длина плана v_L (количество шагов в плане): дополнительные очки могут присуждаться в случае, если план, предложенный системой в качестве решения, является кратчайшим из возможных решений.
   - Показатель степени достижения целей v_G (если к данной группе задач применима концепция частичного достижения целей). Для численного представления указанного свойства может быть использовано несколько вариантов. Например, в случае множественных целей отношение количества достигнутых целей к общему количеству целей в задаче. Иногда в предметной области может быть введена некоторая метрика, отражающая близость или удаленность состояний друг от друга (например, пропорционально количеству совпадающих или несовпадающих переменных). В этом случае показатель степени достижения цели может быть определен через близость состояния, в котором среда оказывается в результате выполнения плана, к заданному целевому состоянию.
   - Показатель рациональности v_U использования доступных ресурсов на этапе исполнения плана. Для оценки эффективности использования ресурсов обычно используется стоимостная функция, которая может включать в себя:
     - Стоимости выполнения отдельных действий, определяемых через потребляемые при этом ресурсы.
     - Штрафные и премиальные очки за прохождение некоторых нежелательных или предпочтительных состояний.

   Необходимо отметить, что представленные в предыдущей главе алгоритмы планирования не содержат в явном виде специальных процедур повышения качества решения.
3. Показатель вычислительной сложности v_C используемого алгоритма определяется через теоретические оценки временных (гарантированность нахождения решения за линейное, полиномиальное или экспоненциальное время)

и емкостных (аналогичные требования к вычислительное памяти) ограничений для данного алгоритма.

4. Показатель быстродействия v_T связан со скоростью решения поставленной задачи. В отличие от вычислительной сложности, получаемой на основе аналитического исследования используемого алгоритма, быстродействие определяется, как правило, экспериментальным путем посредством решения значительного количества тестовых задач. В рамках данной категории обычно выделяют такие параметры как:
   - среднее время решения одной задачи данной системой;
   - минимальное и максимальное время решения одной задачи данной системой.

   Аналогично, показатель $v_{\text{М}}$ соответствует фактической утилизации вычислительной памяти.
5. Показатель предсказуемости v_P характеризует возможность прогнозирования поведения системы при решении заданной задачи с высокой степенью достоверности. В частности, для практических приложений особую важность имеет возможность априорной оценки следующих основных показателей для заданной пары <*П* , *S*>, где *П* - задача планирования, а *S* - система планирования:
   - вероятность успешного решения задачи планирования *П* системой *S*;
   - ожидаемое время решения задачи *П* системой *S*;
   - минимальное и максимальное время, необходимое для решения задачи *П* системой *S*.;
   - вероятность нахождения оптимального (или субоптимального) в некотором смысле решения (минимальное время генерации плана, наименьшее количество шагов и т.д.) задачи *П* системой *S*.

Помимо численной оценки значения интересующего параметра, важную роль играют другие статистические показатели, отражающие качество полученной оценки, а также стабильность поведения системы. Для большинства перечисленных показателей смысл имеют не абсолютные, а относительные значения, позволяющие оценить, насколько одна система превосходит другую на заданном базовом множестве.

13.9 ФОРМИРОВАНИЕ КРИТЕРИЕВ ЭФФЕКТИВНОСТИ АЛГОРИТМА

Рассмотрим возможные способы построения критериев эффективности с использованием выделенных ранее основных показателей качества алгоритмов планирования. Пусть v – некоторый показатель эффективности алгоритма, для заданной задачи планирования v: $\Omega \to R$. Введем v'– относительную величину показателя v, спроецированную на промежуток $[1,+\infty)$ таким образом, что $v'(\omega)=1$ тогда и только тогда, когда ω является наилучшим алгоритмом в Ω относительно показателя v. В большинстве случаев v' может быть получено как:

$$v'(\omega) = \frac{v(\omega)}{\min\limits_{\psi \in \Omega} v(\psi)}.$$

Использование относительных значений показателей для построения критериев эффективности обеспечивает выполнение условий применимости теоремы Карлина (соизмеримость и однородность частных критериев).

Простые (частные) критерии эффективности могут быть построены на базе $v'(\omega)$ в соответствии с описанием, представленным в предыдущем разделе. Однако с прикладной точки зрения наибольший интерес представляют интегральные оценки эффективности алгоритмов на некотором множестве задач планирования. Принимая во внимание тот факт, что производимый алгоритмами планирования поиск является в общем случае недетерминированным, имеет смысл перейти к вероятностным оценкам эффективности. При рассмотрении относительного показателя v' можно использовать представление $v'(\omega)=1+\xi_\omega$, где ξ_ω является случайной величиной, обладающей следующими свойствами:

1. ξ_ω является неотрицательной: $P(\xi_\omega < 0) = 0 \quad \forall \omega \in \Omega$.
2. Вероятность того, что алгоритм ω является наилучшим: $P(\xi_\omega = 0) = P_\omega^o$; причем в общем случае выполняется:

$$\sum_{\omega \in \Omega} P_\omega^o \geq 1,$$

а при выборе v, гарантирующем единственность минимума (наилучшего алгоритма), справедливо

$$\sum_{\omega \in \Omega} P_\omega^o = 1.$$

В силу определения ξ_ω ее функция распределения может быть представлена в виде:

$$F_\omega(x) = P(\xi_\omega < x) = \left(P_\omega^0 + (1 - P_\omega^0) F_\omega^{\,+}(x)\right) I_{R^+}(x),$$

где

$P^0{}_\omega$ – вероятность того, что алгоритм ω является наилучшим относительно выбранного показателя;

I_{R+} – индикаторная функция множества R^+;

F_ω^+ – функция распределения случайной величины ξ_ω^+, состоящей из положительных значений ξ_ω.

Для каждого из выделенных показателей качества функционирования (и основанных на них частных критериев) могут быть построены следующие вероятностные критерии эффективности:

1. ε-оптимальный алгоритм: $\varphi_1(\omega)=P(\xi_\omega\leq\varepsilon)$.
2. Алгоритм, наилучший в среднем: $\varphi_2(\omega)=E\xi_\omega$.
3. Алгоритм, наилучший в среднем для однокритериального выбора: $\varphi_3(\omega)=Ev(\omega)$
4. Абсолютно наилучший алгоритм: $\varphi_1(\omega)=P(\xi_\omega=0)$.
5. Алгоритм, миинимизирующий риск: $\varphi_4(\omega)=P(\xi_\omega>\varepsilon)$.

Легко видеть, что в такой постановке с формальной точки зрения критерии ε-оптимальности и минимизации риска эквиваленты. Использование той или иной формулировки определяется значением пороговой величины: при малых ε говорят об ε-оптимальности, при больших – о минимизации риска. Также следует отметить, что при выборе алгоритма исключительно на основании простого критерия допустимо построение критерия с использованием абсолютных значений (т.е. v вместо v'), причем в качестве критерия обычно используется среднее значение рассматриваемого показателя, что соответствует критерию $\varphi_3(\omega)$.

Для выбора алгоритма планирования используется минимизация (или максимизация) соответствующего критерия:

$$a^* = \arg\max_{\omega\in\Omega} \varphi_j(\omega), \quad j = 1,4;$$
$$a^* = \arg\min_{\omega\in\Omega} \varphi_j(\omega), \quad j = 2,3,5.$$

Поскольку функция распределения случайной величины ξ_a является неизвестной, приходится использовать выборочные аналоги вероятностных критериев (через Φ_i обозначен выборочный критерий, соответствующий вероятностному критерию φ_i, $i=1,2,..5$):

$\Phi_1(\omega)$ – доля задач, при решении которых алгоритм ω оказался ε-оптимальным с заданным пороговым значением;

$\Phi_2(\omega)$ – относительное значение показателя производительности алгоритма ω, усредненное на множестве задач планирования;

$\Phi_3(\omega)$ – абсолютное значение показателя эффективности алгоритма ω, усредненное на заданном множестве задач планирования;

$\Phi_4(\omega)$ – доля задач, при решении которых алгоритм ω оказался наилучшим;

$\Phi_5(\omega)$ – доля задач, при решении которых относительное значение показателя производительности для алгоритма ω оказалось хуже заданного порогового значения.

Поскольку Φ_4 и Φ_5 могут быть легко представлены через Φ_1 (Φ_4 является частным случаем при $\varepsilon=0$, а $\Phi_5=1 - \Phi_1$ при одинаковых пороговых значениях), в дальнейшим будут использоваться только критерии Φ_1, Φ_2 и Φ_3.

В случае недостаточной адекватности простых критериев эффективности может быть использован многокритериальный выбор алгоритма планирования. Как следует из основных теорем ТИО, при построении частных критериев на основе относительных значений показателей в качестве свертки частных критериев может быть использована их линейная комбинация:

$$V = \sum_{i=1}^{n} \lambda_i v_i, \quad \sum_{i=1}^{n} \lambda_i = 1.$$

Очевидно, что при использовании подобного подхода особое внимание требуется уделить выбору весовых коэффициентов $\Lambda=\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, адекватных условиям исходной задачи. Одно из возможных решений данной задачи состоит в разбиении задач планирования на классы на основе особенностей функционирования интеллектуального агента. Пример построения агрегированного критерия на основе показателей качества алгоритма, представляющих в большинстве случаев наибольший интерес с прикладной точки зрения:

1. Функциональность v_F, определяемая как вероятность нахождения алгоритмом минимального решения данной задачи планирования.
2. Использование вычислительных ресурсов в процессе решения задачи, в частности процессорное время v_T и объем оперативной памяти v_M.
3. Эффективность поиска, выполняемого алгоритмом планирования, выражаемая через общее количество ветвлений дерева поиска v_B и общее количество рекуррентных вызовов алгоритма v_{Ba}.

Для оценки качества алгоритма планирования $\omega \in \Omega$ используются относительные значения соответствующих показателей:

$$v'_F(\omega) = 1 + \frac{\max\limits_{\zeta\in\Omega} v_F(\zeta) - v_F(\omega)}{\max\limits_{\zeta\in\Omega} v_F(\zeta) - \min\limits_{\psi\in\Omega} v_F(\psi)},$$

$$v'_x(\omega) = \frac{v_x(\omega)}{\min\limits_{\psi\in\Omega} v(\psi)}, \quad v_x \in \{v_T, v_M, v_B, v_{B_a}\}.$$

Оптимальность алгоритма определяется с помощью целевого функционала, состоящего из суммы относительных значений представленных показателей, умноженных на весовые коэффициенты, отражающие важность каждого показателя в конкретной задаче планирования:

$$v(\Lambda, \omega) = \Lambda V'(a)^T, \quad a \in \Omega;$$

$$\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_5), \sum_{i=1}^{5} \lambda_i = 1;$$

$$V' = (v'_F, v'_T, v'_M, v'_B, v'_{B_a}).$$

В качестве примера практического использования предложенного критерия эффективности можно рассмотреть три наиболее распространенных функциональных типа задач планирования:

1. Автономное планирование: построение плана происходит независимо от текущего функционирования агента, в силу чего основными требованиями

являются высокая вероятность нахождения решения при умеренном быстродействии и несущественных емкостных ограничениях.

2. Оперативное планирование: время построения плана ограничено из-за функциональных особенностей агента и постоянных изменений в среде или условиях задачи. Поскольку по истечении определенного промежутка времени текущая задача планирования перестает быть актуальной, основным показателем является скорость нахождения плана.
3. Распределенное планирование: имеется возможность параллельного поиска по нескольким ветвям дерева решений (многопроцессорный агент или мультиагентная система), поэтому основными показателями являются полнота поиска и эффективность каждого вызова алгоритма.

Таблица 13.9.1

Пример выбора весовых коэффициентов в зависимости от типа задачи

Тип задачи					
Автономное планирование	0.60	0.20	0.10	0.05	0.05
Оперативное планирование	0.20	0.50	0.20	0.05	0.05
Распределенное планирование	0.30	0.15	0.05	0.20	0.30

На рис. 13.9.1 изображены оценки ε-оптимальности рассматриваемых алгоритмов для каждого функционального типа задачи планирования в смысле выбранного критерия с использованием весовых коэффициентов, представленных в табл. 13.9.1.

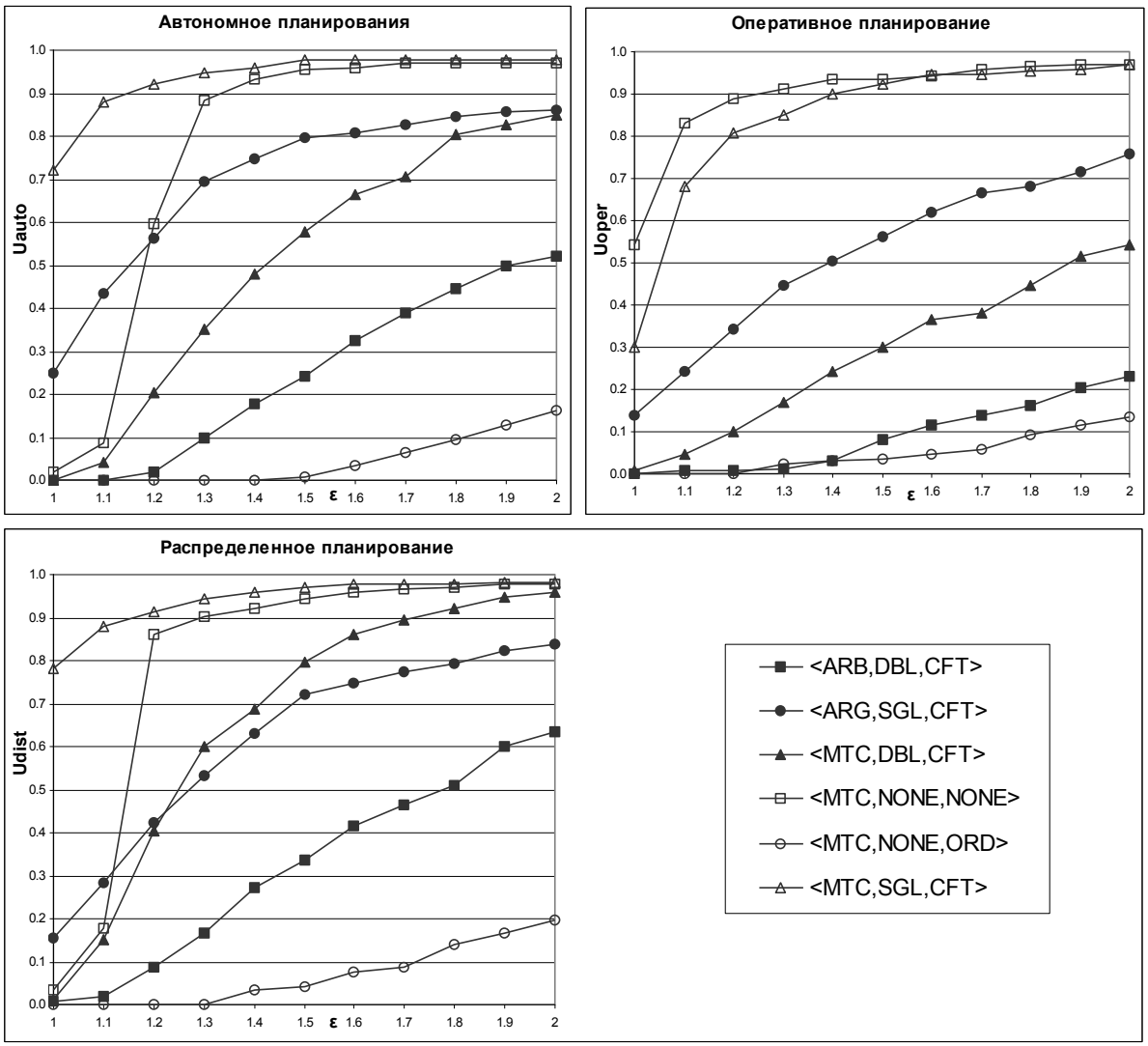

Автономное планирования
Uauto
ε
Оперативное планирование
Uoper
ε
Распределенное планирование
Udist
ε
<ARB,DBL,CFT>
<ARG,SGL,CFT>
<MTC,DBL,CFT>
<MTC,NONE,NONE>
<MTC,NONE,ORD>
<MTC,SGL,CFT>

Рис. 13.9.1

14 АНАЛИЗ СИСТЕМ ПЛАНИРОВАНИЯ ДЕЙСТВИЙ

14.1 РАЗРАБОТКА МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ ПЛАНОВ ДЕЙСТВИЙ ИНФОРМАЦИОННЫХ АГЕНТОВ ГЕТЕРОГЕННЫХ СЕТЕЙ ДЛЯ ПРЕОДОЛЕНИЯ АПРИОРНОЙ НЕОПРЕДЕЛЕННОСТИ

Рассмотрим очередной этап приобретения знаний об интеллектуальных информационных агентов посредством построения математических моделей для анализа временных и стоимостных характеристик последовательно-параллельных планов для стохастических сред. Построим модели для анализа временных характеристик устранения неопределенности при выполнении типовых опросов информационных источников (то есть характеристик времени, за которое станет известно, выполнился ли запрос успешно или неуспешно) и стоимостных характеристик устранения неопределенности в течение указанного времени. При построении аналитических моделей необходимо предусмотреть ограничения типовых запросов по времени и стоимости. Как показывает опыт сопровождения информационных систем, типовыми запросами считаются опрос одного информационного источника, последовательный опрос двух и более реплицированных информационных источников (содержащих одну и ту же информацию), последовательный опрос двух и более нереплицированных информационных источников, параллельный опрос двух и более реплицированных информационных источников и параллельный опрос двух и более нереплицированных информационных источников. Для построения логических моделей и получения аналитических соотношений воспользуемся методом индукции.

Посредством анализа построенных моделей определим оценки временных и стоимостных характеристик устранения неопределенности при выполнении типовых опросов информационных источников с ограничениями.

Первоначально построим модель запроса к одному информационному источнику. Пусть запрос к информационному источнику выполняется успешно с вероятностью p, а неуспешно – с вероятностью $(1-p)$. Причем временной интервал $t_{success}$, через который станет известен результат выполнения успешного запроса, является случайной величиной, распределенной по некоторому закону. Аналогично временной интервал $t_{failure}$, через который станет известен результат выполнения неуспешного запроса, также является случайной величиной, распределенной по некоторому закону. Для учета временных ограничений проведем дискретизацию времени ожидания исхода запроса с некоторым квантом Δt, установим максимальное время наблюдений равным $K_{\max}^{s}$ квантов для успешного запроса и $K_{\max}^{f}$ квантов для неуспешного запроса. Введем дискретную случайную величину (ДСВ) k_s – ожидаемое количество квантов до завершения успешного запроса и ДСВ k_f – ожидаемое количество квантов до завершения неуспешного запроса. Пусть известно дискретное распределение вероятностей времени завершения успешного запроса $f_s(k_s)$ и известно дискретное распределение вероятностей времени завершения неуспешного запроса $f_f(k_f)$. Допустим, что ДСВ k_s и k_f – независимые.

Пусть время наблюдений ограничено N_{max} квантами. Для распределений $f_s(k_s)$ и $f_f(k_f)$ должны выполняться следующие условия:

$$\sum_{k_s} f_s(k_s) = 1, k_s = 1, 2, ..., K^s_{\max};$$

$$\sum_{k_f} f_f(k_f) = 1, k_f = 1, 2, ..., K^f_{\max}.$$

Аналогично для учета стоимостных ограничений введем квантование стоимости ожидания исхода запроса с некоторым шагом ΔC, установим максимальную стоимость равной $M^s_{\max}$ квантам для успешного запроса и $M^f_{\max}$ квантам для неуспешного запроса. Введем ДСВ C_s – ожидаемую стоимость успешного запроса и ДСВ C_f – ожидаемую стоимость неуспешного запроса. Допустим, что ДСВ C_s и C_f – независимые. Пусть известны дискретные распределения вероятностей стоимости успешного запроса $f_s(C_s)$ и неуспешного запроса $f_f(C_f)$. Пусть стоимость выполнения запроса ограничена C_{max} квантами. Для распределений $f_s(C_s)$ и $f_f(C_f)$ должны выполняться условия:

$$\sum_{C_s} f_s(C_s) = 1, C_s = 1, 2, ..., M^s_{\max};$$

$$\sum_{C_f} f_f(C_f) = 1, C_f = 1, 2, ..., M^f_{\max}.$$

Процесс запроса к информационному источнику можно описать в виде логической модели, представляемой ориентированным графом с двумя неузловыми вершинами и двумя последовательными подпроцессами, соответствующими успешному и неуспешному запросам. Каждая неузловая вершина соответствует фрагменту кода программы, выполняемому последовательно. Расхождение и схождение последовательных подпроцессов описывается с помощью логической функции $\oplus$ – «Исключающее ИЛИ» (рис. 14.1.1). Время выполнения последовательного подпроцесса, представляющего успешный запрос, характеризуется распределением вероятностей $f_s(k_s)$, а время выполнения последовательного подпроцесса, соответствующего неуспешному запросу, характеризуется распределением вероятностей $f_f(C_f)$.

Модель запроса к информационному источнику

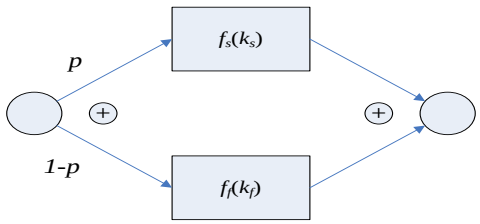


Рис. 14.1.1

Представленный на рис. 14.1.1 процесс запроса к информационному источнику является последовательным и характеризуется некоторым распределением вероятностей времени устранения неопределенности $f(k_0)$, где $k_0 = min(k_s, k_f),\ldots,max(k_s,k_f)$. Для распределения $f(k_0)$ должно выполняться:

$$\sum_{k_0} f(k_0) = 1, k_0 = \min(k_s,\ k_f),...,\max(k_s, k_f). \qquad (14.1.1)$$

Проведем анализ предлагаемой модели запроса к одному информационному источнику с целью определения времени устранения неопределенности. Для анализа воспользуемся явным методом свертки. В соответствии с построенной моделью:

$$f(k_0) = p * f_s(k_s = k_0) + (1-p) * f_f(k_f = k_0); \qquad (14.1.2)$$
$$k_0 \ = \ \min(k_s,\ k_f),...,\max(k_s, k_f).$$

Для проверки подставим выражение (14.1.2) в соотношение (14.1.1):

$$\sum_{k_0} f(k_0) = p * \sum_{k_0} f_s(k_s) + (1-p) * \sum_{k_0} f_f(k_f) = p + (1-p) = 1.$$

При последовательном опросе двух реплицированных информационных источников в случае успешного обращения к первому информационному источнику опрос завершается, иначе – опрашивается второй информационный источник. На основе модели опроса одного информационного источника построим модель последовательного опроса двух реплицированных информационных источников. Пусть вероятность успешного обращения к первому информационному источнику равна p_1, а ко второму – p_2. Пусть известны распределения вероятностей времени успешного и неуспешного обращения к каждому информационному источнику – $f1_s(k1_s)$, $f2_s(k2_s)$ и $f1_f(k1_f)$, $f2_f(k2_f)$, а также стоимости успешного и неуспешного обращения к каждому информационному источнику $f1(C1_s)$, $f2(C2_s)$ и $f1(C1_f)$, $f2(C2_f)$. Обозначим распределение вероятностей времени устранения неопределенности первого информационного источника как $f_1(k_{01})$, а второго информационного источника – $f_2(k_{02})$. Отметим, что $f_1(k_{01})$ и $f_2(k_{02})$ могут быть найдены на основе соотношений, полученных для распределения вероятностей времени устранения неопределенности одного информационного источника. Обозначим итоговое распределение вероятностей времени устранения неопределенности при последовательном опросе двух реплицированных информационных источников как *f_(k_)*. Тогда процесс последовательного опроса двух реплицированных информационных источников можно представить в виде следующей логической модели (рис. 14.1.2.):

Модель последовательного опроса двух реплицированных информационных источников

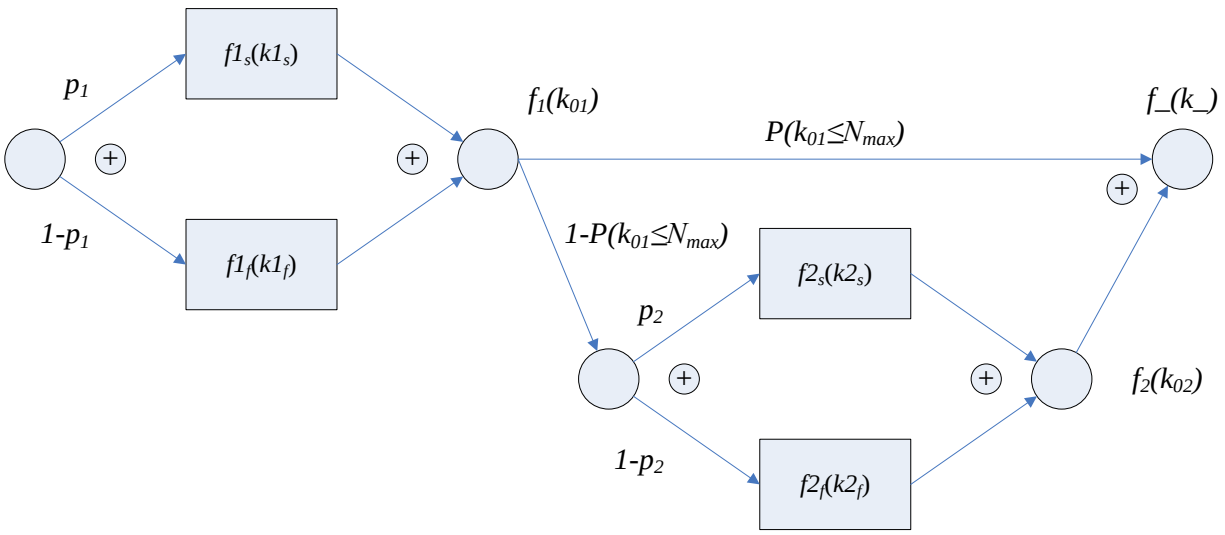


Рис. 14.1.2

Представленный на рис. 14.1.2 процесс опроса информационных источников является последовательным, т.к. узлы используют только логические функции «Исключающее ИЛИ». Найдем итоговое распределение вероятностей времени устранения неопределенности $f_(k_)$.

В соответствии с представленной на рис. 14.1.2 моделью, а также соотношениями, полученными для запроса к одному информационному источнику, имеем:

$$f_1(k_{01}) = p_1 * f1_s(k1_s = k_{01}) + (1 - p_1) * f1_f(k1_f = k_{01});$$

$$k_{01} = \min(k1_s,\ k1_f),\ldots,\max(k1_s, k1_f);$$

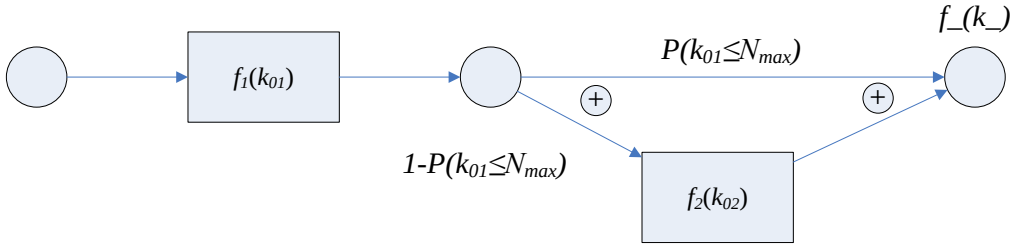
$$P(k_{01} \le N_{\max}) = \sum_{k_{01} \le N_{\max}} f_1(k_{01}) = p_1 * \sum_{k1_s \le N_{\max}} f1_s(k1_s) + (1 - p_1) * \sum_{k1_f \le N_{\max}} f1_f(k1_f);$$

$$f_2(k_{02}) = p_2 * f2_s(k2_s = k_{02}) + (1 - p_2) * f2_f(k2_f = k_{02});$$

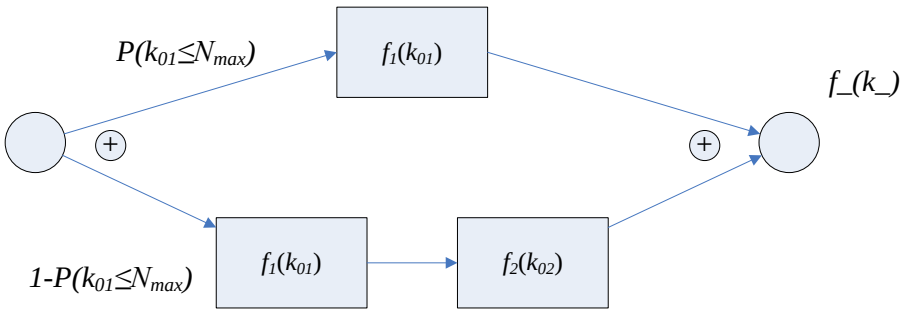
$$k_{02} = \min(k2_s,\ k2_f),\ldots,\max(k2_s, k2_f).$$

Обозначим как $f_{12}(k_{012})$ распределение вероятностей времени обязательного последовательного опроса двух информационных источников (то есть опроса, при котором вне зависимости от успешности первого обращения опрашивается второй информационный источник). Тогда модель последовательного опроса двух информационных источников, представленную на рис. 14.1.2, можно преобразовать следующим образом (рис. 14.1.3):

Преобразование модели последовательного опроса двух реплицированных информационных источников



откуда следует:



и в окончательном виде имеем:

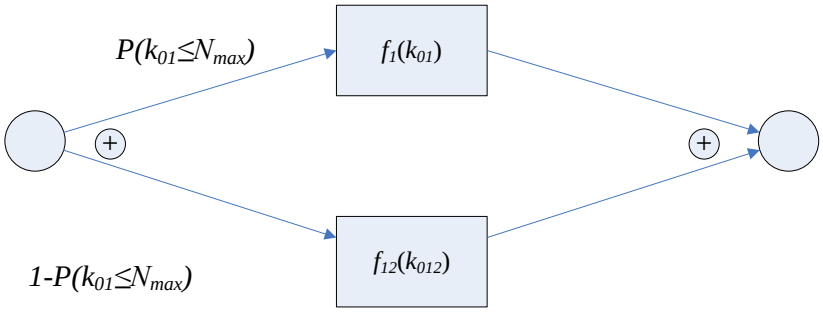


Рис. 14.1.3

Согласно представленной на рис. 14.1.13 модели, а также соотношениям, полученным для запроса к одному информационному источнику, определим итоговое распределение вероятностей времени устранения неопределенности $f_(k_)$. Примем времена начала последовательных подпроцессов независимыми от времен их выполнения. Тогда будет справедливо следующее соотношение:

$$f_{12}(k_{012}) = \sum_{k_{01}} f_1(k_{01}) * f_2(k_{012} - k_{01});$$

$$k_{012} = \min(k_{01} + k_{02}),...,\max(k_{01} + k_{02});$$

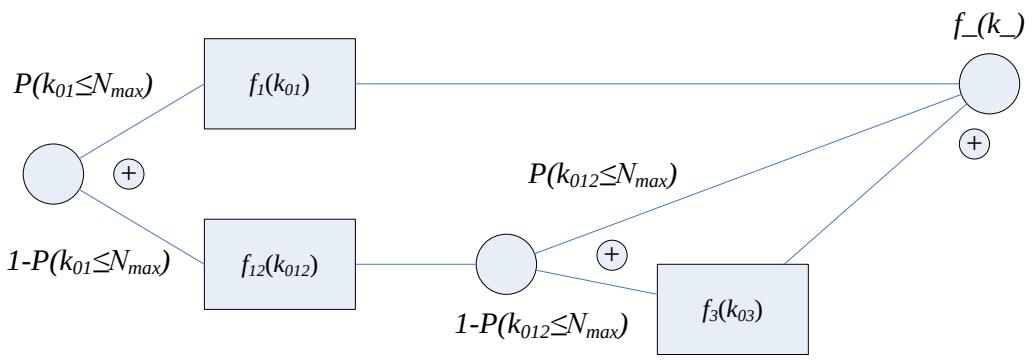
$$f_(k_) = P(k_{01} \le N_{\max}) * f_1(k_{01} = k_) + (1 - P(k_{01} \le N_{\max})) * f_{12}(k_{012} = k_);$$

$$k_ = \min(k_{01}, k_{012}),...,\max(k_{01}, k_{012}).$$

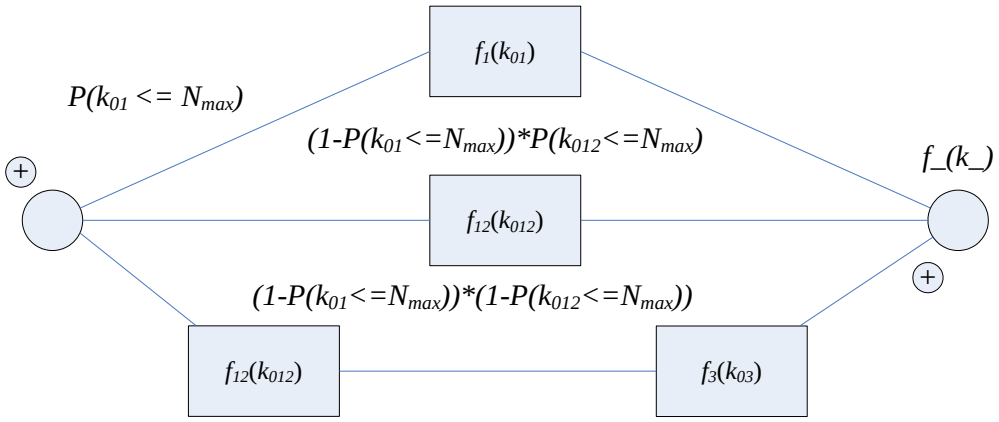
При построении моделей для обобщенных случаев (опрос *I* источников) введем следующие обозначения: пусть для каждого информационного источника *i* (*i*=1,...,*I*) известны: (1) p_i – вероятность успешного обращения к *i*-му информационному источнику; (2) распределения вероятностей времени успешного и неуспешного обращения к *i*-му информационному источнику – $fi_s(ki_s)$ и $fi_f(ki_f)$; (3) стоимости успешного и неуспешного обращения к *i*-му информационному источнику $fi(Ci_s)$ и $fi(Ci_f)$. Обозначим распределение вероятностей времени устранения неопределенности *i*-го информационного источника как $f_i(k_{0i})$. $f_i(k_{0i})$ может быть найдено на основе соотношений, полученных для распределения вероятностей времени устранения неопределенности одного информационного источника. Обозначим как $f_{123...j}(k_{123...j})$ распределение вероятностей времени обязательного последовательного опроса j информационных источников (j=1,...,*I*) (то есть опроса, при котором каждый последующий информационный источник опрашиваются вне зависимости от успешности обращения к предыдущему).

Воспользуемся методом индукции. Рассмотрим случай последовательного опроса трех реплицированных информационных источников. Модель последовательного опроса трех информационных источников можно представить следующим образом (рис. 14.1.4):

Преобразование модели последовательного опроса трех реплицированных информационных источников



откуда следует:



и в окончательном виде имеем:

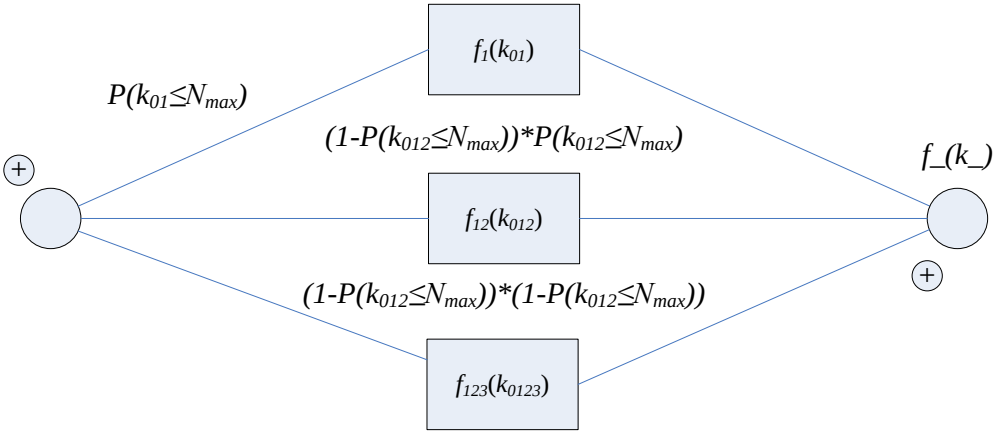


Рис. 14.1.4

Определим итоговое распределение вероятностей устранения неопределенности *f_(k_)*:

$$f_j(k_{0j}) = p_j * fj_s(kj_s = k_{0j}) + (1 - p_j) * fj_f(kj_f = k_{0j});\ j = 1,2,3;$$

$$k_{0j} = \min(kj_s, kj_f), \ldots, \max(kj_s, kj_f);\ j = 1,2,3;$$

$$P(k_{012} \le N_{\max}) = \sum_{k_{012} \le N_{\max}} f_{12\ldots j}(k_{012\ldots j});\ j = 1,2,3;$$

$$f_{12}(k_{012}) = \sum_{k_{01}} f_1(k_{01}) * f_2(k_{012} - k_{01});$$

$$k_{012} = \min(k_{01} + k_{02}), \ldots, \max(k_{01} + k_{02});$$

$$f_{123}(k_{0123}) = \sum_{k_{012}} f_{12}(k_{012}) * f_3(k_{0123} - k_{012});$$

$$k_{0123} = \min(k_{01} + k_{02} + k_{03}), \ldots, \max(k_{01} + k_{02} + k_{03});$$

$$f_(k_) = P(k_{01} \le N_{\max}) * f_1(k_{01} = k_) +$$
$$(1 - P(k_{01} \le N_{\max})) * P(k_{012} \le N_{\max}) * f_{12}(k_{012} = k_) + (1 -$$
$$P(k_{01} \le N_{\max})) * (1 - P(k_{012} \le N_{\max})) * f_{123}(k_{0123} = k_);$$

$$k_ = \min(k_{01}, k_{012}, k_{0123}), \ldots, \max(k_{01}, k_{012}, k_{0123}).$$

Рассмотрим случай последовательного опроса I реплицированных информационных источников. Для удобства записи введем следующие обозначения:

$$\Pi'_i = (1 - P(k_{01} \le N_{\max})) * (1 - P(k_{012} \le N_{\max})) * \ldots * (1 - P(k_{012\ldots i} \le N_{\max})); i = 1, \ldots, I.$$

Преобразованная по аналогии со случаем последовательного опроса трех реплицированных информационных источников логическая модель такого опроса представлена на рис. 14.1.5.

Преобразованная логическая модель последовательного опроса I реплицированных информационных источников

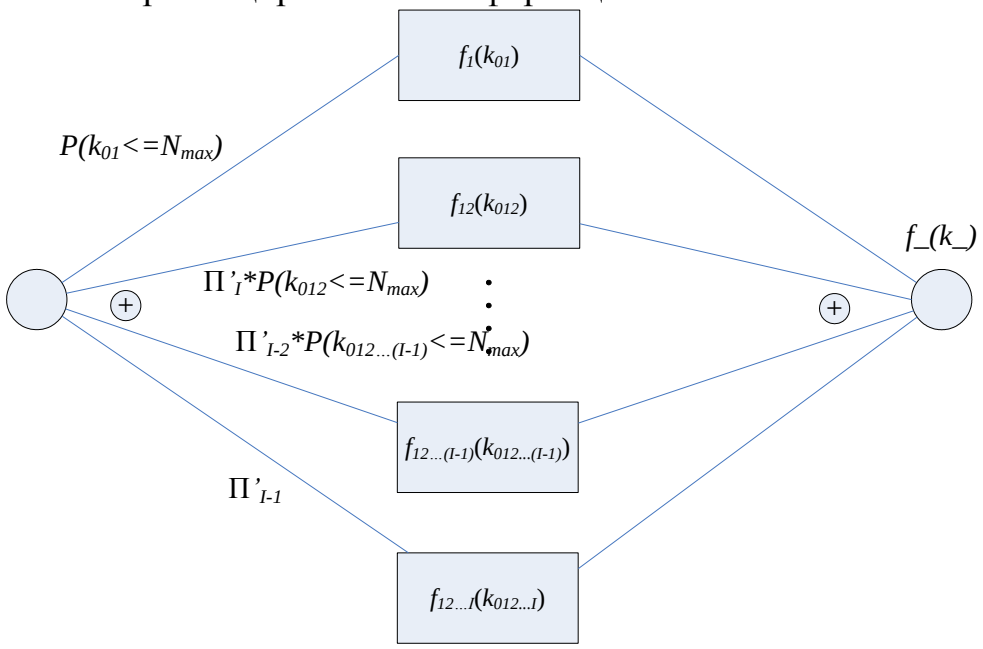


Рис. 14.1.5

Согласно представленной на рис. 14.1.5 логической модели, итоговое распределение вероятностей устранения неопределенности при последовательном опросе I реплицированных информационных источников $f_(k_)$ определяется следующим образом:

$$f_j(k_{0j}) = p_j * fj_s(kj_s = k_{0j}) + (1 - p_j) * fj_f(kj_f = k_{0j});\ j = 1,...I;$$

$$k_{0j} = \min(kj_s, kj_f),...,\max(kj_s, kj_f);\ j = 1,...,I;$$

$$f_{12...j}(k_{012...j}) = \sum_{k_{012...(j-1)}} f_{12...(j-1)}(k_{012...(j-1)}) * f_j(k_{012...j} - k_{012...(j-1)});\ j = 1,...,I;$$

$$k_{012...j} = \min(k_{01} + k_{02} + ... + k_{0j}),...,\max(k_{01} + k_{02} + ... + k_{0j});\ j = 1,...,I;$$

(14.1.3)

$$P(k_{012...j} \le N_{max}) = \sum_{k_{012...j} \le N_{max}} f_{12...j}(k_{012...j});\ j = 1,...,I;$$

$$f_(k_) = P(k_{01} \le N_{\max}) * f_1(k_{01} = k_) +$$
$$\Pi'_1 * P(k_{012} \le N_{\max}) * f_{12}(k_{012} = k_) + ... +$$
$$\Pi'_{I-2} * P(k_{012...(I-1)} \le N_{\max}) * f_{12...(I-1)}(k_{012...(I-1)} = k_) +$$
$$\Pi'_{I-1} * f_{12...I}(k_{012...I} = k_);$$

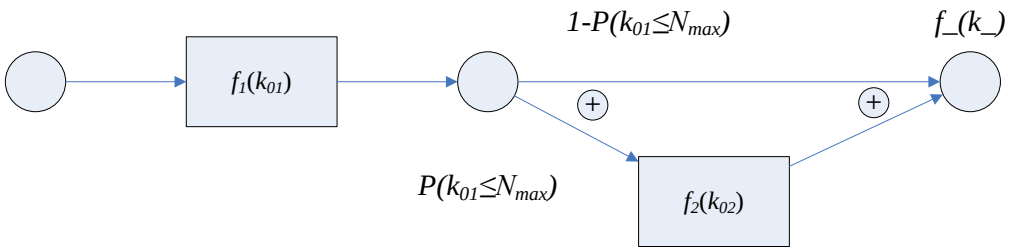
$$k_ = \min(k_{01}, k_{012},..., k_{012...I}),...,\max(k_{01}, k_{012},..., k_{012...I}).$$

Соответственно, среднее время устранения неопределенности при последовательном опросе I реплицированных информационных источников можно вычислить как математическое ожидание случайной величины $k_$:

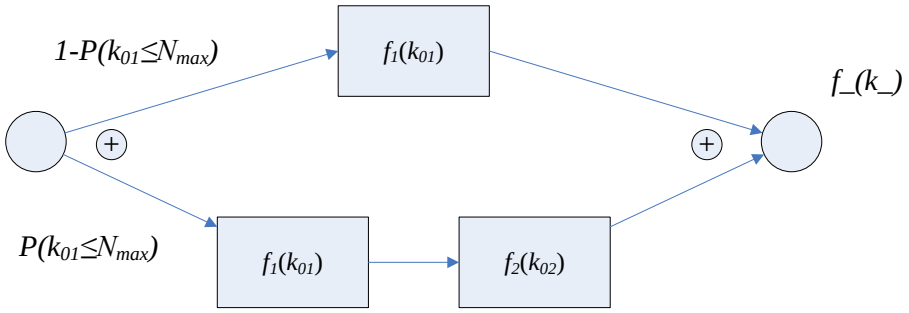
$$\text{МО}[k_] = \sum_{k_} k_ * f_(k_). \qquad (14.1.4)$$

Теперь определим распределение вероятностей времени устранения неопределенности для случая последовательного опроса двух нереплицированных информационных источников. По сути, эта задача аналогична случаю последовательного опроса реплицированных информационных источников и единственным отличием логической модели будут инверсные вероятности переходов для одной из вершин (рис. 14.1.6):

Модель последовательного опроса двух нереплицированных информационных источников



откуда следует:



и в окончательном виде имеем:

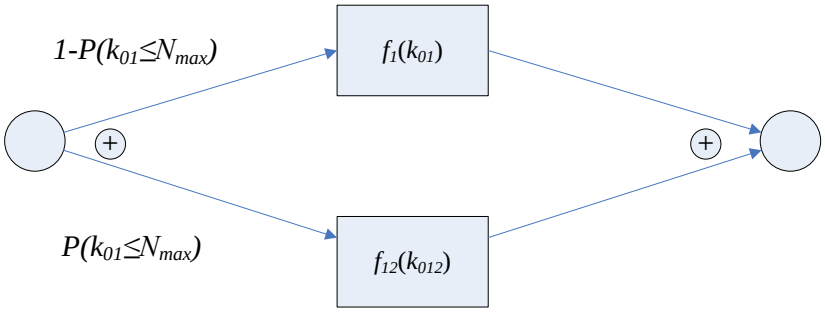


Рис. 14.1.6

Поэтому по аналогии со случаем последовательного опроса реплицированных информационных источников определим вероятность $P(k_<N_{max})$ того, что к моменту N_{max} неопределенность по последовательному опросу двух нереплицированных информационных источников будет устранена:

$$f_{12}(k_{012}) = \sum_{k_{01}} f_1(k_{01}) * f_2(k_{012} - k_{01});$$

$$k_{012} = \min(k_{01} + k_{02}), \ldots, \max(k_{01} + k_{02});$$

$$f_(k_) = (1 - P(k_{01} \le N_{\max})) * f_1(k_{01} = k_) + P(k_{01} \le N_{\max}) * f_{12}(k_{012} = k_);$$

$$k_ = \min(k_{01}, k_{012}), \ldots, \max(k_{01}, k_{012});$$

$$P(k_ \le N_{\max}) = \sum_{k_ \le N_{\max}} f_(k_).$$

Рассмотрим случай последовательного опроса I нереплицированных информационных источников. Преобразованная по аналогии со случаем последовательного опроса I реплицированных информационных источников логическая модель такого опроса представлена на рис. 14.1.7.

Преобразованная логическая модель последовательного опроса I нереплицированных информационных источников

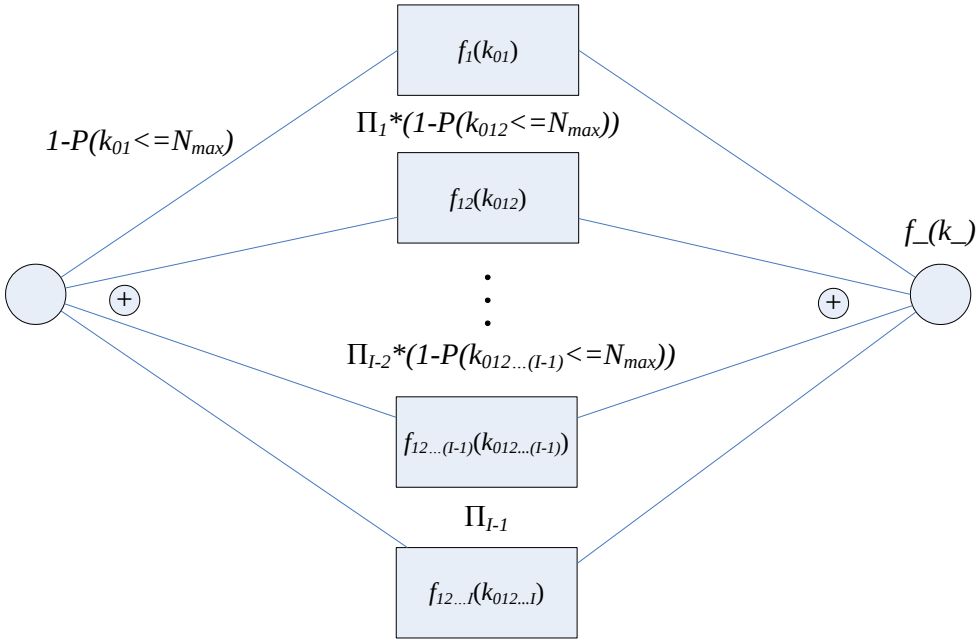


Рис. 14.1.7

При построении приведенной модели использованы следующие обозначения:

$$\Pi_i = P(k_{01} \le N_{\max}) * P(k_{012} \le N_{\max}) * \ldots * P(k_{012\ldots i} \le N_{\max}); \quad i = 1, \ldots, I\ .$$

В соответствии с представленной на рис. 14.1.7 логической моделью, вероятность устранения неопределенности при последовательном опросе I нереплицированных информационных источников $P(k_ \le N_{max})$ за время наблюдений N_{max} определяется следующим образом:

$$f_j(k_{0j}) = p_j * fj_s(kj_s = k_{0j}) + (1 - p_j) * fj_f(kj_f = k_{0j}); \quad j = 1,...,I;$$

$$k_{0j} = \min(kj_s, kj_f),...,\max(kj_s, kj_f); \quad j = 1,...,I;$$

$$f_{12...j}(k_{012...j}) = \sum_{k_{012...(j-1)}} f_{12...(j-1)}(k_{012...(j-1)}) * f_j(k_{012...j} - k_{012...(j-1)}); \quad j = 1,...,I;$$

$$k_{012...j} = \min(k_{01} + k_{02} + ... + k_{0j}),...,\max(k_{01} + k_{02} + ... + k_{0j}); \qquad (14.1.5)$$

$$P(k_{012...j} \le N_{\max}) = \sum_{k_{012...j} \le N_{\max}} f_{012...j}(k_{012...j}); \quad j = 1,...,I;$$

$$\begin{aligned} f_(k_) = {} & (1 - P(k_{01} \le N_{\max})) * f_1(k_{01} = k_) + \\ & \Pi_1 * (1 - P(k_{012} \le N_{\max})) * f_{12}(k_{012} = k_) + ... + \\ & \Pi_{I-2} * (1 - P(k_{012...(I-1)} \le N_{\max})) * f_{12...(I-1)}(k_{012...(I-1)} = k_) + \\ & \Pi_{I-1} * f_{12...I}(k_{012...I} = k_); \end{aligned}$$

$$P(k_ \le N_{\max}) = \sum_{k_ \le N_{\max}} f_(k_). \qquad (14.1.6)$$

$$k_ = \min(k_{01}, k_{012},..., k_{012...I}),...,\max(k_{01}, k_{012},..., k_{012...I}).$$

Соответственно, среднее время устранения неопределенности при последовательном опросе I нереплицированных информационных источников можно вычислить как математическое ожидание случайной величины $k_$ согласно соотношениям (14.1.4), (14.1.5).

Теперь рассмотрим случай параллельного опроса двух реплицированных информационных источников. Как и в рассмотренных ранее случаях последовательного опроса информационных источников, обозначим распределение вероятностей времени устранения неопределенности первого информационного источника как $f_1(k_{01})$, а второго информационного источника – $f_2(k_{02})$. Соотношения для них уже получены. Обозначим итоговое распределение вероятностей времени выполнения параллельного опроса двух реплицированных информационных источников как $f_(k_)$. Определим вероятность $P(k_ \le N_{max})$ того, что к моменту N_{max} неопределенность по параллельному опросу двух реплицированных информационных источников будет устранена. Логическая модель, соответствующая такому опросу, приведена на рис. 14.1.8.

Логическая модель параллельного опроса
двух реплицированных информационных источников

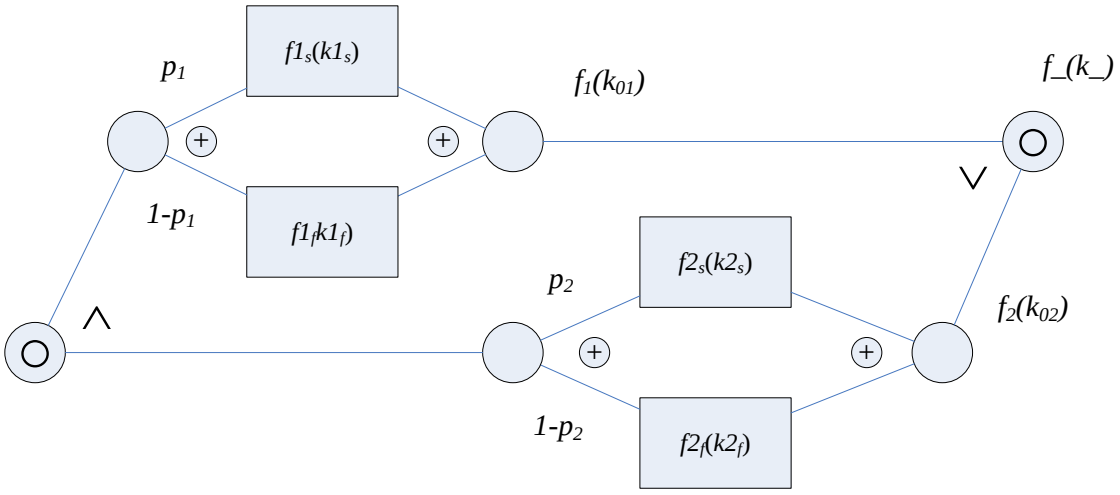


Рис. 14.1.8

Для представления построенной модели введены следующие обозначения:
– узловые вершины, символизирующие объединение и разветвление последовательных подпроцессов;
$\wedge$ – логическая функция «И» (логическое умножение);
$\vee$ – логическая функция «ИЛИ» (логическое сложение),

Присутствие в построенной логический модели узловых вершин отражает тот факт, что модель относится к разряду логических моделей параллельных процессов. С учетом соотношений, полученных для описания распределения вероятностей времени выполнения запроса к одному информационному источнику, и аналогично рассмотренным ситуациям с последовательным опросом информационных источников произведем преобразование логической модели (рис. 14.1.9):

$$f_(k_) = f_1(k_{01} = k_) * (1 - \sum_{k_{02} \le k_} f_2(k_{02})) + f_2(k_{02} = k_) * (1 - \sum_{k_{01} < k_} f_1(k_{01}));$$

$$k_ = \min\{\min(k_{01}), \min(k_{02})\}, ..., \min\{\max(k_{01}), \max(k_{02})\};$$

$$P(k_ \le N_{\max}) = \sum_{k_ \le N_{\max}} f_(k_).$$

Преобразованная логическая модель параллельного опроса
двух реплицированных информационных источников

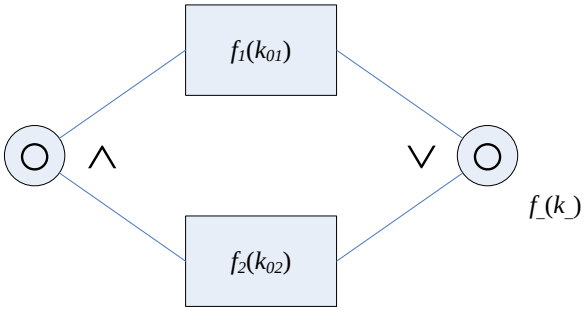


Рис. 14.1.9

Проанализируем случай параллельного опроса I реплицированных информационных источников. Логическая модель анализируемого опроса представлена на рис. 14.1.10. После преобразования модель принимает вид, приведенный на рис. 14.1.11.

Логическая модель параллельного опроса
I реплицированных информационных источников

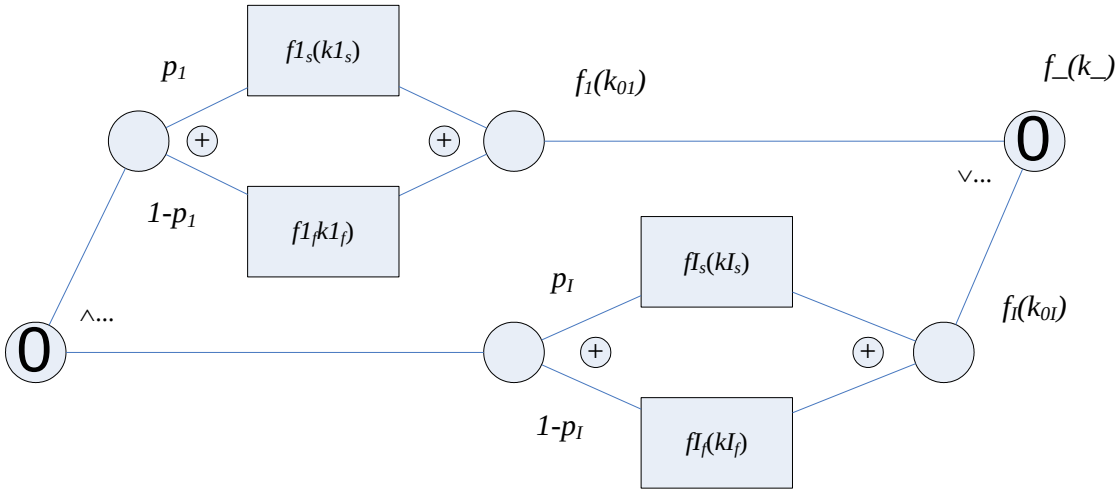


Рис. 14.1.10

Согласно представленной на рис. 14.1.11 преобразованной логической модели, искомые характеристики находятся по следующим формулам:

$$f_j(k_{0j}) = p_j * fj_s(kj_s = k_{0j}) + (1 - p_j) * fj_f(kj_f = k_{0j}); \quad j = 1,...,I;$$

$$k_{0j} = \min(kj_s, kj_f),...,\max(kj_s, kj_f); \quad j = 1,...,I;$$

Преобразованная логическая модель параллельного опроса I реплицированных информационных источников

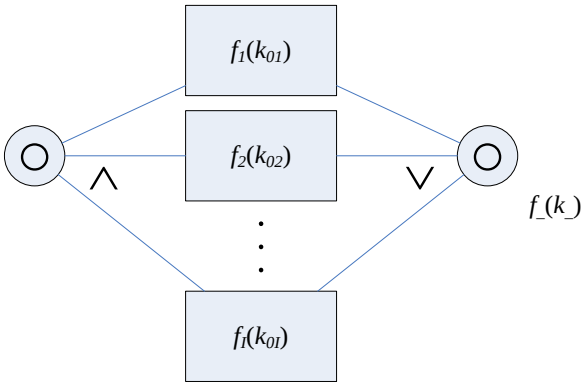


Рис. 14.1.11

$$f_(k_) = f_1(k_{01} = k_) * (1 - \sum_{k_{02} \le k_} f_2(k_{02})) * \text{K} * (1 - \sum_{k_{0I} \le k_} f_I(k_{0I})) +$$
$$+ f_2(k_{02} = k_) * (1 - \sum_{k_{01} < k_} f_1(k_{01})) * (1 - \sum_{k_{03} \le k_} f_3(k_{03})) * \text{K} * (1 - \sum_{k_{0I} \le k_} f_I(k_{0I})) + \text{K} +$$
$$+ f_I(k_{0I} = k_) * (1 - \sum_{k_{01} < k_} f_1(k_{01})) * \text{K} * (1 - \sum_{k_{0(I-1)} < k_} f_{I-1}(k_{0(I-1)}));$$

(14.1.7)

$$k_ = \min\{\min(k_{01}), \min(k_{02}), ..., \min(k_{0I})\}, ...,$$
$$\min\{\max(k_{01}), \max(k_{02}), ..., \max(k_{0I})\}.$$

Теперь рассмотрим случай параллельного опроса I нереплицированных информационных источников. Все обозначения аналогичны введенным для случая параллельного опроса I реплицированных информационных источников. Логическая модель для такого опроса представлена на рис. 14.1.12.

Согласно представленной на рис. 14.1.12 логической модели, вероятность устранения неопределенности при параллельном опросе I нереплицированных информационных источников $P(k_ \le N_{max})$ за время наблюдений N_{max} определяется следующим образом:

$$f_j(k_{0j}) = p_j * fj_s(kj_s = k_{0j}) + (1 - p_j) * fj_f(kj_f = k_{0j}); \quad j = 1, ..., I;$$

$$k_{0j} = \min(kj_s, kj_f), ..., \max(kj_s, kj_f); \quad j = 1, ..., I;$$

Логическая модель параллельного опроса I нереплицированных информационных источников

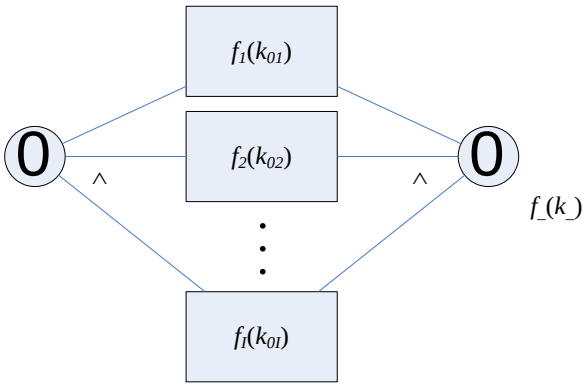


Рис. 14.1.12

$$f_(k_) = f_1(k_{01} = k_) * \sum_{k_{02} \le k_} \mathrm{K} \sum_{k_{0I} \le k_} f_2(k_{02}) * \mathrm{K} * f_I(k_{0I}) +$$
$$+ f_2(k_{02} = k_) * \sum_{k_{01} < k_} \sum_{k_{03} \le k_} \mathrm{K} \sum_{k_{0I} \le k_} f_1(k_{01}) * f_3(k_{03}) * \mathrm{K} * f_I(k_{0I}) + \mathrm{K} + \quad (14.1.8)$$
$$+ f_I(k_{0I} = k_) * \sum_{k_{01} < k_} \mathrm{K} \sum_{k_{0(I-1)} < k_} f_1(k_{01}) * f_2(k_{02}) * \mathrm{K} * f_{I-1}(k_{0(I-1)});$$

$$k_ = \max\{\min(k_{01}), \min(k_{02}), \ldots, \min(k_{0I})\}, \ldots,$$
$$\max\{\max(k_{01}), \max(k_{02}), \ldots, \max(k_{0I})\}.$$

Среднее время устранения неопределенности при параллельном опросе I нереплицированных информационных источников можно вычислить как математическое ожидание случайной величины $k_$ путем подстановки (14.1.8) в (14.1.4).

Итоговые стоимостные характеристики получаются путем подстановки в соотношения (14.1.13) – (14.1.8) распределений стоимости выполнения успешных (неуспешных) запросов к информационным источникам вместо соответствующих временных распределений. На основе полученных соотношений можно определить вероятность успешного (неуспешного) выполнения запроса за C_{max} квантов и математическое ожидание стоимости успешного (неуспешного) выполнения запроса по аналогии с временными характеристиками:

$$P(C_ \le C_{\max}) = \sum_{C_ \le C_{\max}} f_(C_);$$

$$\mathrm{MO}[C_] = \sum_{C_} C_ * f_(C_).$$

Возникает вопрос, каким образом вычислить распределения случайных величин дискретного типа k_s, k_f, C_s и C_f ? Так как на момент первого выполнения запроса у агента нет никакой информации о нем, то изначально распределения можно положить равновероятными и равными соответственно $1/K^s_{max}$ $(1/K^f_{max})$ для временных характеристик и $1/M^s_{max}$ $(1/M^f_{max})$ для стоимостных характеристик успешного (неуспешного запроса). В процессе работы агента распределения ДСВ k_s, k_f, C_s и C_f можно уточнять для каждого запроса на основе накапливаемой статистики. Подобный подход обеспечит повышение достоверности информационной базы агента, качества генерируемых им планов, и, соответственно, приведет к повышению его производительности. Показатель обучаемости определяется как прирост производительности при повторном выполнении аналогичных информационных запросов. Поэтому предлагаемый механизм обеспечит автономную обучаемость информационного агента.

Таким образом, построены логические модели типовых опросов информационных источников при преодолении априорной неопределенности. Для построенных моделей получены оценки временных и стоимостных характеристик устранения неопределенности при ограничениях по времени и стоимости. Далее целесообразно перейти к построению моделей для оценки временных и стоимостных характеристик достижимости цели.

14.2. РАЗРАБОТКА МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ ПЛАНОВ ДЕЙСТВИЙ ИНФОРМАЦИОННЫХ АГЕНТОВ ГЕТЕРОГЕННЫХ СЕТЕЙ ДЛЯ ОЦЕНКИ ДОСТИЖИМОСТИ ЦЕЛИ

Выведем соотношения для оценки времени достижения цели каждым из типовых запросов. Оценки стоимости достижения цели, как и в случае преодоления априорной неопределенности, могут быть получены при замене в соответствующих соотношениях временных характеристик информационных источников на их стоимостные характеристики.

Для оценки временных характеристик достижимости цели введем две итоговые функции распределения вероятностей времен исполнения: $f_s(k_s)$ и $f_f(k_f)$ – функции распределения вероятностей времени выполнения успешного и неуспешного запроса соответственно.

Посредством анализа явных представлений по методу свертки получим аналитические соотношения для оценки достижимости цели при последовательном опросе двух реплицированных источников. Логическая модель такого опроса приведена на рис. 14.2.1.

Логическая модель последовательного опроса двух реплицированных источников
для оценки достижимости цели

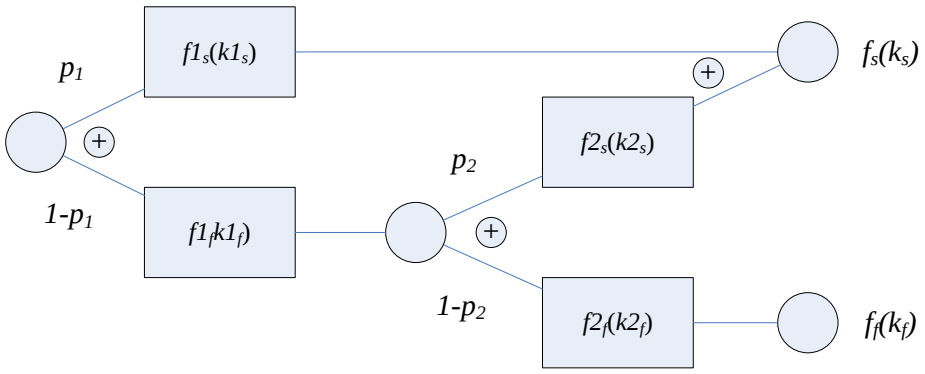


Рис. 14.2.1.

Для данной модели целесообразно найти выражение для функции распределения вероятностей неуспешного запроса. Согласно представленной на рис. 14. 2.1. модели, выражение $f_f(k_f)$ может быть найдено следующим образом:

$$f_f(k_f) = (1-p_1)*(1-p_2)*\sum_{k1_f} f1_f(k1_f)*f2_f(k_f - k1_f),$$

$$k_f = \min(k1_f + k2_f),...,\max(k1_f + k2_f).$$

Выражение $f_s(k_s)$ может быть найдено следующим образом:

$$f_s(k_s) = p_1 * f1_s(k1_s = k_s) + (1 - p_1) * p_2 * \sum_{k1_f} f1_f(k1_f) * f2_s(k_s - k1_f),$$

$$k_s = \min(k1_s), ..., \max(k1_f + k2_s).$$

По индукции перейдем к рассмотрению достижимости цели при последовательном опросе I реплицированных источников. Логическая модель такого запроса представлена на рис. 14.2.2.

Логическая модель последовательного опроса I реплицированных источников для оценки достижимости цели

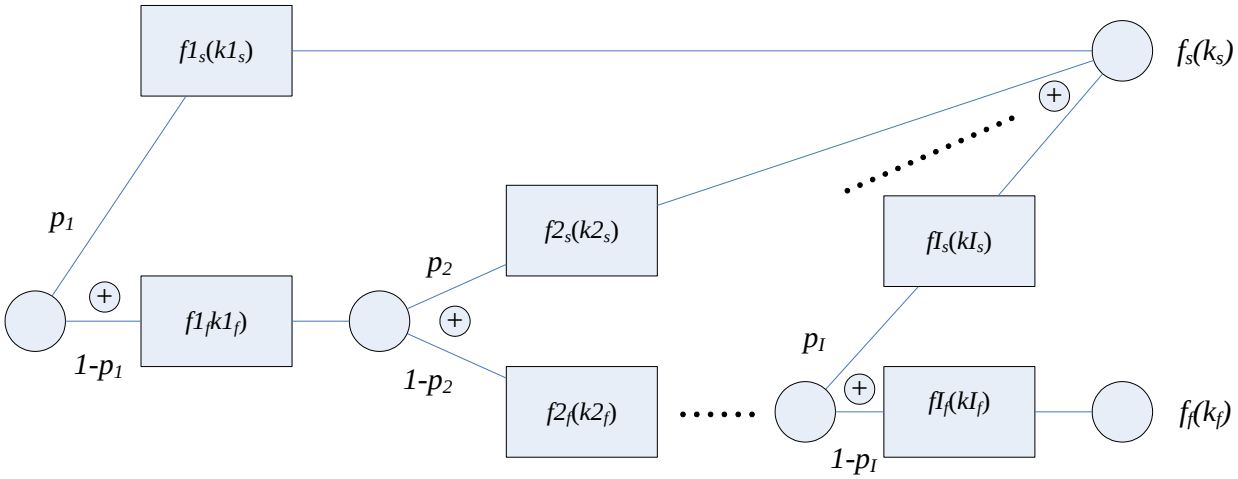


Рис. 14.2.2.

Согласно представленной на рис. 14.2.2. логической модели, выражение $f_f(k_f)$ может быть найдено следующим образом:

$$f_f(k_f) = (1 - p_1) * (1 - p_2) * \text{K} * (1 - p_I) * f^f_{1...j}(k_{1...j} = k_f);$$

$$f^f_{1...j}(k_{1...j} = k_f) = \sum_{k_{1...(j-1)}} f^f_{1...(j-1)}(k_{1...(j-1)}) * fj_f(k_f - k_{1...(j-1)});$$

$$k_{12...j} = \min(k1_f + k2_f + ... + kj_f), ..., \max(k1_f + k2_f + ... + kj_f); \quad (14.2.1)$$

$$k_f = \min(k_1 + k_{12} + ... + k_{12...I}), ..., \max(k_1 + k_{12} + ... + k_{12...I}).$$

Выражение $f_s(k_s)$ может быть найдено следующим образом:

$$f_s(k_s) = p_1 * f^s_1(k_1 = k_s) + (1 - p_1) * p_2 * f^s_{12}(k_{12} = k_s) + \text{K}$$
$$+ (1 - p_1) * \text{K} * (1 - p_{I-1}) * p_I * f^s_{12...I}(k_{12...I} = k_s)$$

$$f^{s}_{1\ldots j}(k_{1\ldots j}) = \sum_{k_{1\ldots(j-1)}} f^{f}_{1\ldots(j-1)}(k_{1\ldots(j-1)}) * fj_s(k_{1\ldots j} - k_{1\ldots(j-1)})$$

$$f^{s}_{1}(k_1) = f1_s(k_1);$$

$$j = 1, 2, \ldots, I; \qquad (14.2.2)$$

$$k_1 = k1_s;$$

$$k_{12\ldots j} = \min(k1_s, k1_f + k2_s, \ldots, k1_f + \ldots k(j-1)_f + kj_s)\ldots$$
$$\max(k1_s, k1_f + k2_s, \ldots, k1_f + \ldots k(j-1)_f + kj_s);$$

$$k_s = \min(k_1, k_{12}, \ldots, k_{12\ldots I}), \ldots, \max(k_1, k_{12}, \ldots, k_{12\ldots I}).$$

Теперь выведем аналитические соотношения для оценки достижимости цели при последовательном опросе двух нереплицированных источников. Логическая модель такого опроса приведена на рис. 14.2.3.

Логическая модель последовательного опроса двух нереплицированных источников для оценки достижимости цели

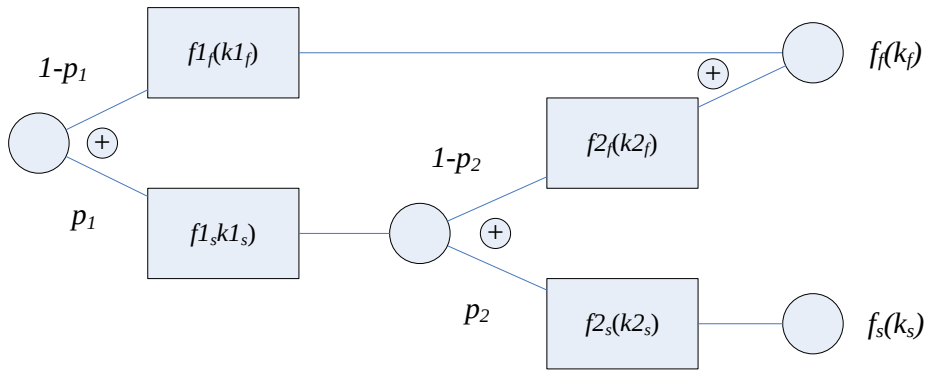


Рис. 14.2.3.

Согласно представленной на рис. 14.2.3. модели, выражение $f_s(k_s)$ может быть найдено следующим образом:

$$f_s(k_s) = p_1 * p_2 * \sum_{k1_s} f1_s(k1_s) * f2_s(k_s - k1_s),$$

$$k_s = \min(k1_s + k2_s), \ldots, \max(k1_s + k2_s).$$

Выражение $f_f(k_f)$ может быть найдено следующим образом:

$$f_f(k_f) = (1-p_1)*f1_f(k1_f = k_f) + p_1*(1-p_2)*\sum_{k1_s} f1_s(k1_s)*f2_f(k_f - k1_s),$$

$$k_f = \min(k1_f),...,\max(k1_s + k2_f).$$

По индукции рассмотрим достижимость цели при последовательном опросе I нереплицированных источников. Логическая модель такого запроса представлена на рис. 14.2.4.

Логическая модель последовательного опроса I нереплицированных источников для оценки достижимости цели

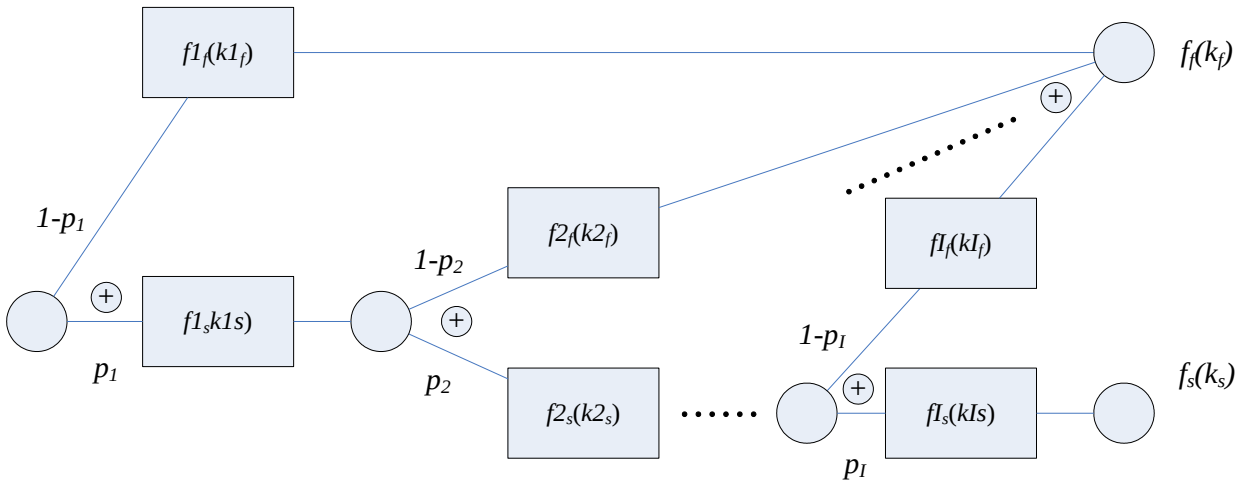


Рис. 14.2.4.

Согласно представленной на рис. 14.2.4. логической модели, выражение $f_s(k_s)$ может быть найдено следующим образом:

$$f_s(k_s) = p_1 * p_2 * ... * p_I * f^s_{1...I}(k_{1...I} = k_s);$$

$$f^s_{1...j}(k_{1...j}) = \sum_{k_{1...(j-1)}} f^s_{1...(j-1)}(k_{1...(j-1)}) * fj_s(k_{1...j} - k_{1...(j-1)});$$

(14.2.3)

$$k_{12..j} = \min(k1_s + k2_s + ... + kj_s),...,\max(k1_s + k2_s + ... + kj_s);$$

$$j = 1,...,I;$$

$$k_s = \min(k_1 + k_{12} + ... + k_{12...I}),...,\max(k_1 + k_{12} + ... + k_{12...I});.$$

Выражение $f_f(k_f)$ может быть найдено следующим образом:

$$f_f(k_f) = (1-p_1)*f_1^f(k_1 = k_f) + p_1*(1-p_2)*f_{12}^f(k_{12} = k_f) + \text{L}$$

$$+ p_1 * \text{L} * p_{I-1} * (1-p_I) * f^f_{12...I}(k_{12...I} = k_f);$$

$$f_{1...j}^{f}(k_{1...j}) = \sum_{k_{1...(j-1)}} f_{1...(j-1)}^{f}(k_{1...(j-1)}) * fj_f(k_{1...j} - k_{1...(j-1)});$$

$$f_1^f(k_1) = f1_f(k_1); \quad j = 1, 2, ..., I;$$

(14.2.4)

$$k_1 = k1_f;$$

$$k_{12..j} = \min(k1_f, k1_s + k2_f, ..., k1_s + ... + k(j-1)_s + kj_f)...$$
$$\max(k1_f, k1_s + k2_f, ..., k1_s + ... + k(j-1)_s + kj_f);$$

$$k_f = \min(k_1, k_{12}, ..., k_{12...I}), ..., \max(k_1, k_{12}, ..., k_{12...I}).$$

Теперь рассмотрим случай параллельного опроса двух реплицированных информационных источников. Логическая модель, соответствующая такому опросу, приведена на рис. 14.2.5.

Логическая модель параллельного опроса двух реплицированных информационных источников для оценки достижимости цели

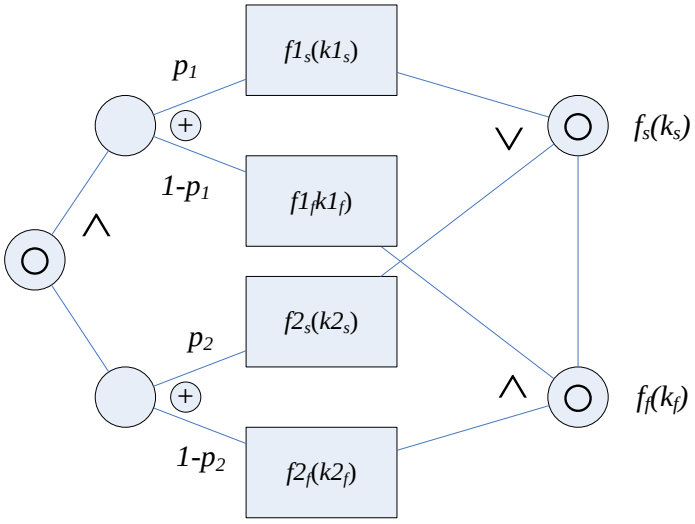


Рис.14.2.5

Присутствие в построенной модели узловых вершин с указанными функциями отражает тот факт, что модель относится к разряду логических моделей параллельных процессов.

Согласно представленной на рис. 14.2.5. логической модели, выражение $f_s(k_5)$ может быть найдено следующим образом:

$$f_s(k_s) = p_1 * p_2 * f1_s(k1_s = k_s) * (1 - \sum_{k2_s \le k_s} f2_s(k2_s)) + p_1 * p_2 * f2_s(k2_s = k_s) * (1 - \sum_{k1_s < k_s} f1_s(k1_s));$$

$$k_s = \min\{\min(k1_s), \min(k2_s)\}, ..., \min\{\max(k1_s), \max(k2_s)\}.$$

Выражение для $f_f(k_f)$ принимает следующий вид:

$$f_f(k_f) = (1-p_1)*(1-p_2)*\left[f1_f(k1_f = k_f)* \sum_{k2_f \le k_f} f2_f(k2_f) + f2_f(k2_f = k_f)* \sum_{k1_f < k_f} f1_f(k1_f) \right];$$

$$k_f = \max\left\{\min(k1_f), \min(k2_f)\right\}, ..., \max\left\{\max(k1_f), \max(k2_f)\right\}.$$

По индукции рассмотрим случай параллельного опроса *I* реплицированных информационных источников. Логическая модель, соответствующая такому опросу, приведена на рис. 14.2.6. Согласно представленной логической модели, выражение $f_f(k_f)$ может быть найдено следующим образом:

$$f_f(k_f) = (1-p_1)*(1-p_2)*\text{L} *(1-p_I)*\left[f1_f(k1_f = k_f)* \sum_{k2_f \le k_f} \text{L} \sum_{kI_f \le k_f} f2_f(k2_f)*\text{L} * fI_f(kI_f) + \right.$$

$$+ f2_f(k2_f = k_f)* \sum_{k1_f < k_f} \sum_{k3_f \le k_f} \sum_{kI_f \le k_f} f1_f(k1_f)* f3_f(k3_f)*\text{L} * fI_f(kI_f) + \text{L} + \qquad ($$

$$\left. + fI_f(kI_f = k_f)* \sum_{k1_f < k_f} \sum_{k2_f < k_f} \text{L} \sum_{k(I-1)_f < k_f} f1_f(k1_f)* f2_f(k2_f)*\text{L} * f(I-1)_f(k(I-1)_f) \right];$$

14.2.5)

$$k_f = \max\left\{\min(k1_f), \min(k2_f), ..., \min(kI_f)\right\}, ...,$$
$$\max\left\{\max(k1_f), \max(k2_f), ..., \max(kI_f)\right\}.$$

Выражение $f_s(k_s)$ может быть найдено следующим образом:

$$f_s(k_s) = p_1 * p_2 *\text{L} * p_I *\left[f1_s(k1_s = k_s)*(1- \sum_{k2_s \le k_s} f2_s(k2_s))*\text{L} *(1- \sum_{kI_s \le k_s} fI_s(kI_s)) + \right.$$

$$+ f2_s(k2_s = k_s)*(1- \sum_{k1_s < k_s} f1_s(k1_s))*(1- \sum_{k3_s \le k_s} f3_s(k3_s))*\text{L} *(1- \sum_{kI_f \le k_f} fI_s(kI_s)) + \text{L} +$$

$$\left. + fI_s(kI_s = k_s)*(1- \sum_{k1_s < k_s} f1_s(k1_s))*(1- \sum_{k2_s < k_s} f2_s(k2_s))*\text{L} *(1- \sum_{k(I-1)_s < k_s} f(I-1)_s(k(I-1)_s)) \right];$$

(14.2.6)

$$k_s = \min\left\{\min(k1_s), \min(k2_s), ..., \min(kI_s)\right\}, ...,$$
$$\min\left\{\max(k1_s), \max(k2_s), ..., \max(kI_s)\right\}.$$

Перейдем к рассмотрению случая параллельного опроса двух нереплицированных информационных источников. Логическая модель, соответствующая такому опросу, приведена на рис. 14.2.7.

Логическая модель параллельного опроса I реплицированных информационных источников для оценки достижимости цели

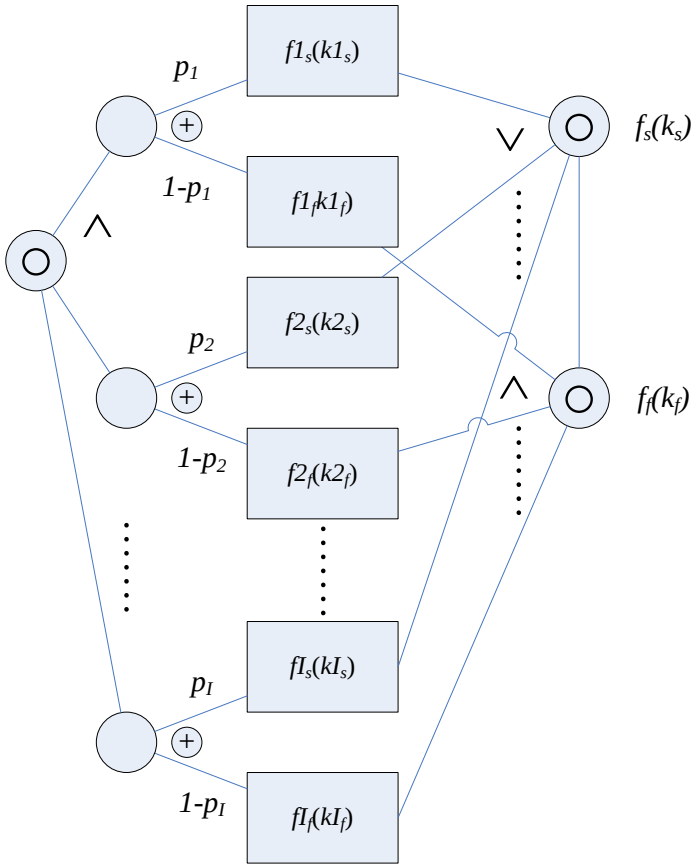


Рис. 14.2.6

Логическая модель параллельного опроса двух нереплицированных информационных источников для оценки достижимости цели

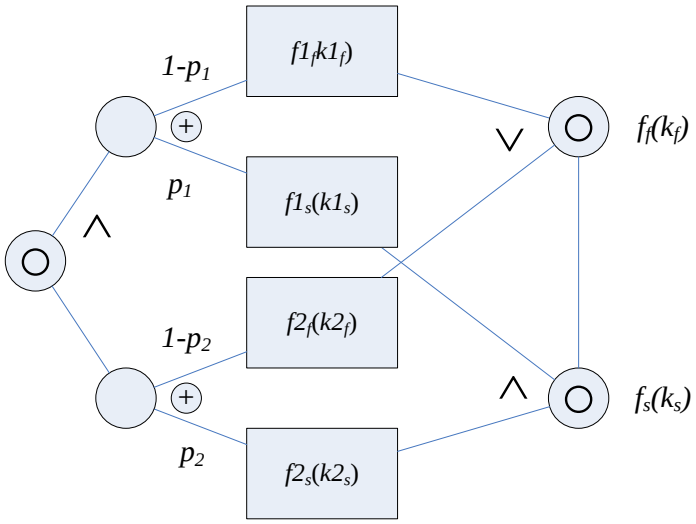


Рис. 14.2.7

Согласно представленной на рис. 14.2.7. логической модели, выражение $f_s(k_s)$ может быть найдено следующим образом:

$$f_s(k_s) = p_1 * p_2 * \left[f1_s(k1_s = k_s) * \sum_{k2_s \le k_s} f2_s(k2_s) + f2_s(k2_s = k_s) * \sum_{k1_s < k_s} f1_s(k1_s) \right];$$

$$k_s = \max\{\min(k1_s), \min(k2_s)\}, ..., \max\{\max(k1_s), \max(k2_s)\}.$$

Выражение для $f_f(k_f)$ принимает следующий вид:

$$f_f(k_f) = (1 - p_1) * (1 - p_2) * \left[f1_f(k1_f = k_f) * (1 - \sum_{k2_f \le k_f} f2_f(k2_f)) + f2_f(k2_f = k_f) * (1 - \sum_{k1_f < k_f} f1_f(k1_f)) \right]$$

$$k_f = \min\{\min(k1_f), \min(k2_f)\}, ..., \min\{\max(k1_f), \max(k2_f)\}.$$

По индукции рассмотрим случай параллельного опроса I нереплицированных информационных источников. Логическая модель, соответствующая такому опросу, приведена на рис. 14.2.8.

Согласно представленной модели, выражение $f_s(k_s)$ может быть определено следующим образом:

$$f_s(k_s) = p_1 * p_2 * \mathrm{L} * p_I * \left[f1_s(k1_s = k_s) * \sum_{k2_s \le k_s} \mathrm{L} \sum_{kI_s \le k_s} f2_s(k2_s) * \mathrm{L} * fI_s(kI_s) + \right.$$

$$+ f2_s(k2_s = k_s) * \sum_{k1_s < k_s} \sum_{k3_s \le k_s} \mathrm{L} \sum_{kI_f \le k_f} f1_s(k1_s) * f3_s(k3_s) * \mathrm{L} * fI_s(kI_s) + \mathrm{L} +$$

$$\left. + fI_s(kI_s = k_s) * \sum_{k1_s < k_s} \mathrm{L} \sum_{k(I-1)_s < k_s} f1_s(k1_s) * f2_s(k2_s) * \mathrm{L} * f(I-1)_s(k(I-1)_s) \right];$$

$$k_s = \max\{\min(k1_s), \min(k2_s), ..., \min(kI_s)\}, ...,$$
$$\max\{\max(k1_s), \max(k2_s), ..., \max(kI_s)\}.$$

(14.2.7)

Выражение $f_f(k_f)$ может быть найдено следующим образом:

$$k_f = \min\{\min(k1_f), \min(k2_f), ..., \min(kI_f)\}, ...,$$
$$\min\{\max(k1_f), \max(k2_f), ..., \max(kI_f)\}.$$

Логическая модель параллельного опроса I нереплицированных информационных источников для оценки достижимости цели

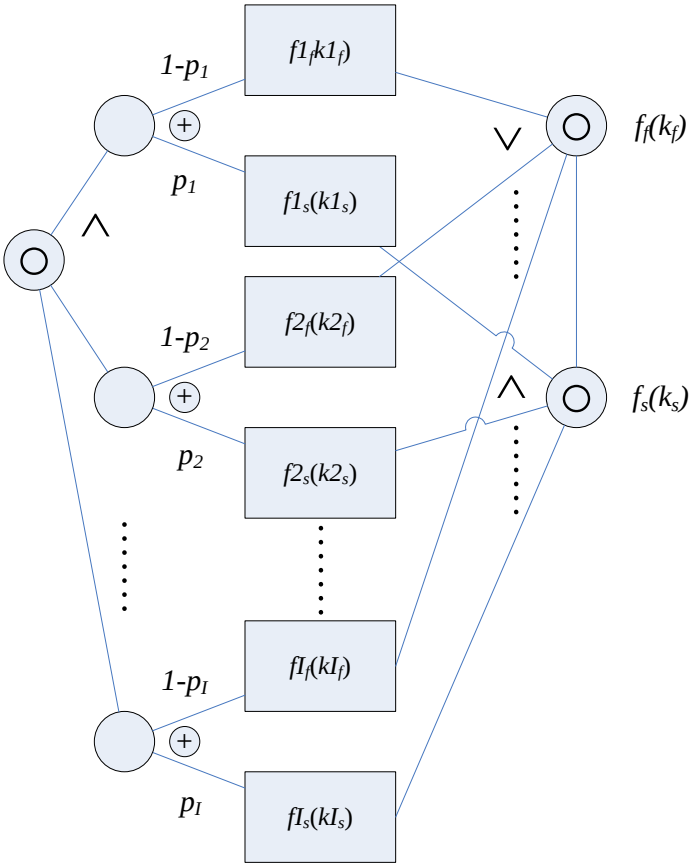


Рис. 14.2.8

$$f_f(k_f) = (1-p_1)*(1-p_2)*\text{L}\ *(1-p_I)*$$

$$*\Bigg[f1_f(k1_f = k_f)*(1-\sum_{k2_f \le k_f} f2_f(k2_f))*(1-\sum_{k3_f \le k_f} f3_f(k3_f))*\text{L}\ *(1-\sum_{kI_f \le k_f} fI_f(kI_f)) +$$

$$+ f2_f(k2_f = k_f)*(1-\sum_{k1_f < k_f} f1_f(k1_f))*(1-\sum_{k3_f \le k_f} f3_f(k3_f))*\text{L}\ *(1-\sum_{kI_f \le k_f} fI_f(kI_f)) + \text{L}\ +$$

$$+ fI_s(kI_f = k_f)*(1-\sum_{k1_f < k_f} f1_f(k1_f))*(1-\sum_{k2_f < k_f} f2_f(k2_f))*\text{L}\ *(1-\sum_{k(I-1)_f < k_f} f(I-1)_f(k(I-1)_f))\Bigg];$$

(14.2.8)

На основе полученных соотношений (14.2.1) – (14.2.8) можно определить вероятность успешного (неуспешного) выполнения запроса за N_{max} квантов и математическое ожидание времени успешного (неуспешного) выполнения запроса:

$$P(k_s \le N_{\max}) = \sum_{k_s \le N_{\max}} f_s(k_s); \qquad (14.2.9)$$

$$\text{MO}[k_s] = \sum_{k_s} k_s * f_s(k_s) \qquad (14.2.10)$$

$$P(k_f \le N_{\max}) = \sum_{k_f \le N_{\max}} f_f(k_f) \qquad (14.2.11)$$

$$\text{MO}[k_f] = \sum_{k_f} k_f * f_f(k_f). \qquad (14.2.12)$$

Итак, построены логические модели типовых опросов информационных источников для оценки достижимости цели. Для построенных моделей получены оценки временных и стоимостных характеристик достижимости цели при ограничениях по времени и стоимости. Логические модели типовых опросов информационных источников являются базовым набором для построения более сложных моделей опросов (произвольных опросов). Логические модели произвольных опросов декомпозируются на составляющие, соответствующие типовым опросам, и методом свёртки для произвольных опросов определяются оценки временных и стоимостных характеристик устранения неопределенности при ограничениях по времени и стоимости. Построена система приобретения знаний о качестве функционирования интеллектуальных информационных агентов.

15. АПРИОРНЫЙ ВЫБОР ОПТИМАЛЬНОГО АЛГОРИТМА ПЛАНИРОВАНИЯ ДЕЙСТВИЙ

15.1 СТРУКТУРНОЕ ОПИСАНИЕ АЛГОРИТМОВ ПЛАНИРОВАНИЯ ДЕЙСТВИЙ ПРОГРАММНЫХ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ АГЕНТОВ

Компоненты обобщенного алгоритма планирования (в скобках приведены используемые в дальнейшем сокращения) с учетом взаимно-однозначного соответствия между типом конструктора решений и способом выбора подцели:

- Конструктор решений и выбор подцели:
  - Проверка условия завершения и выбор подцели на основе МКИ (*MTC*).
  - Проверка условия завершения на основе пустоты текущего множества подцелей и произвольный выбор подцели из текущего множества подцелей (*ARB*).
- Консервация:
  - Не используется (*NC*).
  - Односторонняя защита казуальных связей (*SGL*).
  - Двусторонняя защита казуальных связей (*DB*L).
- Оптимизация:
  - Не используется (*NO*).
  - Разрешение конфликтов (*CFT*).
  - Дополнительное упорядочивание шагов (*ORD*).

Таким образом, можно определить пространство алгоритмов планирования Ω как множество элементов вида $\langle \omega^S, \omega^C, \omega^O \rangle$, где:

$\omega^S \in \Omega^S = \{ MTC, ARB \}$ – способ выбора подцели;

$\omega^C \in \Omega^C = \{ NC, SGL, DBL \}$ – используемая стратегия консервации;

$\omega^O \in \Omega^O = \{ NO, CFT, ORD \}$ – стратегия оптимизации.

Рассмотренные выше существующие алгоритмы нелинейного планирования соответствуют следующим элементам пространства алгоритмов: $NONLIN = \langle ARB, SGL, CFT \rangle$, $TWEAK = \langle MTC, NC, NO \rangle$, $SNLP = \langle ARB, DBL, CFT \rangle$, $UA = \langle MTC, NCE, ORD \rangle$.

Необходимо отметить, что не все компоненты алгоритма планирования являются независимыми – например, оптимизация на основе разрешения конфликтов имеет смысл только при использовании стратегии оптимизации. В связи с этим далеко не каждой комбинации элементов базиса пространства алгоритмов соответствует состоятельный алгоритм планирования.

Таблица 15.1.1

Состоятельные алгоритмы планирования действий ИИА

Известные	Новые
<ARB, DBL, CFT>	<MTC, DBL, CFT>
<ARB, SGL, CFT>	*<MTC, SGL, CFT>*
<MTC, NC, NO>	
<MTC, NC, ORD>	

15.2 КРИТЕРИЙ ОТБОРА ДОСТУПНОЙ ИНФОРМАЦИИ

Информация, доступная в рамках рассматриваемой модели перед решением задачи планирования, может быть разделена на две группы:

1. Характеристики среды функционирования агента I_E.
2. Параметры задачи планирования $I_П$.

При рассмотрении обобщенного информационного множества $I_A=I_E \cup I_П$ с точки зрения задачи выбора оптимального алгоритма возникают следующие вопросы:

1. Оценка влияния отдельных элементов I_A на показатели качества алгоритма.
2. Возможность использования некоторого подмножества $I_A^* \subset I_A$ для априорного определения оптимального алгоритма.
3. Критерий выбора подмножества I_A^* (при условии его существования).

Для ответа на поставленные вопросы предлагается использовать подход, основанный на оценке влияния имеющейся информации на эффективность алгоритма с позиции статистической проверки гипотез. Для заданного показателя v эффективности алгоритма и каждого показателя $z \in I_A$ требуется проверить гипотезу H_0: случайные величины ξ_v и z независимы против альтернативы H_1: H_0 не верна. Если на основании статистического критерия гипотеза H_0 отклоняется, то z включается в множество I^*. Для оценки адекватности выбора I_A на основании предложенного критерия необходимо провести сравнение результатов решения задачи выбора оптимального алгоритма с раздельным использованием множеств I_A и I_A^*.

Наиболее распространенным статистическим критерием, используемым для проверки гипотезы о независимости совместно наблюдаемых случайных величин на основе экспериментально полученных выборок, является критерий независимости хи-квадрат, связанный с построением таблицы сопряженности признаков:

$$\varepsilon_{z,v} = P(\chi^2_{(n-1)(m-1)} \geq \rho_{z,v}),$$

$$\rho_{z,v} = n\left(\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\eta_{i,j}^2}{\eta_{i,\cdot}\ \eta_{\cdot,j}} - 1\right), \qquad (15.2.1)$$

где $\rho_{z,v}$ – значение статистики хи-квадрат, вычисляемое с помощью эмпирических частот $\eta_{i,j}$, $\eta_{i,\cdot}$ и $\eta_{\cdot,j}$, а m и n - количества интервалов разбиений по z и v соответственно.

Таким образом, для фиксированного v и заданного уровня значимости $0<\alpha<1$ в качестве множества I_A^* может быть использовано множество I_V^*, определяемое в соответствии со следующим правилом:

$$I_V^* = \{z \in I_A: \varepsilon_{z,v} < \alpha\}. \qquad (15.2.2)$$

Выделенные в предыдущей главе показатели позволяют сформировать множества I_E и $I_П$ следующим образом: $I_E = \{N_{op}, N_c, N_{pre}, N_{add}, N_{del}, n_+, n_-, H_+, H_-, h_+, h_-\}$ и $I_П = \{N_g\}$. В табл. 4.1 представлены пороговые значения $\varepsilon_{z,v}$ (усредненные по пространству алгоритмов Ω) для каждого $z \in I_A$ при $v=v_T$, полученные в соответствии с (4.1) при различном количестве интервалов разбиений по z и v.

Таблица 15.2.1

Значения $\varepsilon_{z,v}$ для проверки гипотезы о независимости характеристик среды/задачи планирования и показателей эффективности

m x n / *z*	4x4	6x6	8x8	10x10	Среднее
N_{op}	0.760258	0.824000	0.815223	0.808100	0.801895
N_{pre}	0.535418	0.627935	0.826786	0.995571	0.746427
N_c	0.893463	0.766051	0.993789	0.884025	0.884332
N_{add}	0.569658	0.580158	0.761449	0.846754	0.689505
N_{del}	0.676744	0.708288	0.824942	0.701268	0.727811
n_+	0.661404	0.530194	0.611172	0.750947	0.638429
n_-	0.802594	0.664192	0.856104	0.758671	0.770390
H_+	0.667174	0.683628	0.761479	0.892179	0.751115
H_-	0.830277	0.970451	0.999993	0.999997	0.950179
h_+	0.424103	0.515976	0.632593	0.607729	0.545100
h_-	0.805540	0.523800	0.652462	0.845662	0.706866
N_g	0.534625	0.565111	0.721379	0.993233	0.703587

Например, при уровне значимости $\alpha=0.75$ в соответствии с (15.2.2) имеем: $I_V^*=\{N_{pre}, N_{add}, N_{del}, n_+, h_+, h_-, N_g\}$, что сокращает размер информационного множества почти в два раза. При этом наибольшее значение среди элементов I_A приобретают усредненные показатели, характеризующие различные соотношения между операторами и условиями предметной области, в то время как зависимость от непосредственного размера предметной области (определяемого количеством доступных действий (операторов) N_{op} и количеством условий N_c, используемых для описания среды) проявляется гораздо слабее. Следует также обратить особое внимание на введенную в предыдущей главе и не отраженную до сих пор ни в одной из известных работ характеристику плотности высокочастотных условий h, которая, как показывают представленные данные, оказывает существенное влияние на значения показателей эффективности алгоритмов планирования.

Для экспериментальной проверки применимости предложенного подхода к построению I_A* следует провести сравнение результатов оценки эффективности алгоритмов планирования и качества выбора оптимального алгоритма, произведенных на основании множеств I_A и I_A*.

15.3 ВЫБОР ОПТИМАЛЬНОГО АЛГОРИТМА НА ОСНОВЕ РЕГРЕССИОННОГО АНАЛИЗА

Ввиду новизны решаемой задачи выбора оптимального алгоритма планирования действий, следствием которой является отсутствие специализированных регрессионных моделей, представляется целесообразным использовать известные проблемно-независимые методы анализа данных.

Ниже каждая из задач будет решена с использованием как канонических регрессионных моделей математической статистики (линейная и квадратичная регрессии), так и с помощью относительно новых средств анализа данных, таких как искусственные нейронные сети. Для проверки адекватности предложенного метода построения информационного множества I_A* будет проведено сравнение результатов, полученных в случае $X=I_A$*, с результатами использования аналогичных регрессионных моделей на множестве всех информационных показателей ($X=I_A$).

Для решения обеих задач использованы две хорошо известные полиномиальные регрессионные модели, которые в n-мерном случае (n – размерность используемого в качестве аргумента множества I_A или I_A*) имеют вид:

1. Линейная регрессия:

$$R_L(X) = \sum_{i=1}^{n} a_i x_i + c, \quad x_i \in X \ . \tag{15.3.4}$$

2. Квадратичная регрессия:

$$R_S(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} x_i x_j + \sum_{i=1}^{n} a_i x_i + c, \quad x_i \in X \ . \tag{15.3.5}$$

При проведении эксперимента множество задач планирования Π, полученное с помощью описанного в предыдущей главе генератора задач, было разбито на два непересекающихся подмножества:

1. Обучающее Π_L, служащее для оценки параметров регрессии a_i, b_{ij} и c по методу наименьших квадратов.
2. Тестовое Π_V, используемое для проверки адекватности регрессионной модели. Как отмечено выше, выполняется: $\Pi = \Pi_L \cup \Pi_V$, $\Pi_L \cap \Pi_V = \varnothing$.

В качестве показателей, позволяющих оценить эффективность рассматриваемой модели, использованы величины максимальной и средней ошибки регрессионной функции на заданном множестве задач планирования:

$$E_{\max} = \max_{\pi \in \Pi'} \left| R(\pi) - \xi_\pi \right|, \quad \Pi' \subset \Pi \, ; \tag{15.3.6}$$

$$E_{av} = \frac{\sum_{\pi \in \Pi'} \left| R(\pi) - \xi_\pi \right|}{\left| \Pi' \right|}, \quad \Pi' \subset \Pi \, , \tag{15.3.7}$$

где:

$R(\pi)$ – оценка значения ξ для задачи π, получаемое на основе заданной регрессионной модели;

ξ_π – значение ξ, экспериментально определенное в процессе решения задачи π;

$|\Pi'|$ – мощность подмножества Π'.

В зависимости от решаемой подзадачи (обучение или оценка адекватности) в качестве подмножества Π' выбиралось соответственно Π_L или Π_V.

Размер множества Π_L (и, следовательно, Π_V) определялся необходимостью обеспечения достаточного объема данных для корректного определения регрессионных множителей. Общий размер рассматриваемого множества ($|\Pi|$) составил 500 задач; принимая во внимание случай R_S, выбрано $|\Pi_L|=450$ и $|\Pi_V|=50$.

Численное моделирование произведено как с использованием специально разработанного программного кода, так и с применением библиотеки статистических процедур Regressional Toolbox для пакета Matlab.

В целях нормализации к исходным данным применена препроцессинговая процедура, состоящая в нахождении относительных значений используемых показателей качества алгоритма и масштабировании включенных в регрессионную модель показателей в промежуте [0, 1].

В табл. 15.3.2 представлены результаты использования регрессионных моделей, построенных для задач типа 1, при использовании обоих наборов входных параметров и критерия эффективности, основанном на времени T решения задачи планирования.

Полученные данные носят стабильный характер, в большинстве случаев ошибки при использовании модели $R_S(x)$ в среднем на 20% ниже, чем при $R_L(x)$ для обоих входных наборов (I_V* и I_A), при этом сохраняется относительная разница между E_{av} на тестовом и обучающем множествах.

Большие значения E_{max} (в том числе и на обучающем множестве) свидетельствуют о наличии отдельных задач, для которых ни одна из канонических регрессионных моделей не смогла получить удовлетворительный результат.

При решении задач типа 2 необходимость в дополнительном преобразовании значений аппроксимируемой функции отсутствует.

Для строгого соответствия слабой постановке задачи (аппроксимируемая случайная величина является бинарной) необходимо построить отображения значений функции регрессии на множество {0, 1}. Однако поскольку для решения исходной задачи выбора необходимо проводить сравнение значений критерия для различных алгоритмов, наличие непрерывного спектра оказывается более удобным несмотря на потенциально превосходящие абсолютные ошибки.

Таблица 15.3.2

Результаты решения задачи типа 1 с использованием регрессионных моделей

Алгоритм	Вход. набор	Регресс. модель	Π_L		Π_V	
			E_{max}	E_{av}	E_{max}	E_{av}
<*ARB,DBL,CFT*>	I_V*	$R_L(x)$	0.6088	0.2044	0.6227	0.2347
<*ARB,SGL,CFT*>	I_V*	$R_L(x)$	0.6552	0.1816	0.4921	0.1872
<*MTC,DBL,CFT*>	I_V*	$R_L(x)$	0.5310	0.1497	0.7202	0.1886
<*MTC,NC,NO*>	I_V*	$R_L(x)$	0.6684	0.0559	0.8523	0.0838
<*MTC,NC,ORD*>	I_V*	$R_L(x)$	0.5151	0.1823	0.6254	0.2054
<*MTC,SGL,CFT*>	I_V*	$R_L(x)$	0.8494	0.0996	0.7256	0.1136
<*ARB,DBL,CFT*>	I_V*	$R_S(x)$	0.6287	0.1571	0.5535	0.2047
<*ARB,SGL,CFT*>	I_V*	$R_S(x)$	0.6043	0.1628	0.4821	0.1700
<*MTC,DBL,CFT*>	I_V*	$R_S(x)$	0.5370	0.1344	0.6803	0.1749
<*MTC,NC,NO*>	I_V*	$R_S(x)$	0.6082	0.0587	0.6756	0.0707
<*MTC,NC,ORD*>	I_V*	$R_S(x)$	0.5113	0.1216	0.6107	0.1749
<*MTC,SGL,CFT*>	I_V*	$R_S(x)$	0.7311	0.0939	0.6412	0.0943
<*ARB,DBL,CFT*>	I_A	$R_L(x)$	0.6113	0.2024	0.8136	0.2622
<*ARB,SGL,CFT*>	I_A	$R_L(x)$	0.6607	0.1799	0.5657	0.1847
<*MTC,DBL,CFT*>	I_A	$R_L(x)$	0.5163	0.1459	0.8179	0.1984
<*MTC,NC,NO*>	I_A	$R_L(x)$	0.6591	0.0572	0.8525	0.0942
<*MTC,NC,ORD*>	I_A	$R_L(x)$	0.3766	0.1174	0.6332	0.1277
<*MTC,SGL,CFT*>	I_A	$R_L(x)$	0.8544	0.0986	0.7542	0.1136
<*ARB,DBL,CFT*>	I_A	$R_S(x)$	0.6177	0.1365	0.5287	0.2064
<*ARB,SGL,CFT*>	I_A	$R_S(x)$	0.5911	0.1464	0.5134	0.1679
<*MTC,DBL,CFT*>	I_A	$R_S(x)$	0.5184	0.1259	0.7923	0.1777
<*MTC,NC,NO*>	I_A	$R_S(x)$	0.5788	0.0554	0.8843	0.0712
<*MTC,NC,ORD*>	I_A	$R_S(x)$	0.5185	0.1164	0.4436	0.1351
<*MTC,SGL,CFT*>	I_A	$R_S(x)$	0.7204	0.0930	0.6229	0.0945

Для оценки эффективности рассматриваемой модели использовалась доля задач D, для которых регрессионная модель правильно определила выполнение условия ε-оптимальности:

$$D = \frac{|\Pi^*|}{|\Pi|}, \Pi^* = \{\pi \in \Pi : R(\pi) \geq \frac{1}{2} \mid \xi' = 1\} \cup \{\pi \in \Pi : R(\pi) < \frac{1}{2} \mid \xi' = 0\}. \quad (15.3.8)$$

В табл. 15.3.3 представлены результаты построения и апробации регрессионной модели для задачи типа 2 для аналогичного критерия

эффективности при уровне $\varepsilon=0.1$ (по аналогии с предыдущим экспериментом также приведены значения E_{max} и E_{av} на обучающем множестве Π_L, однако следует иметь в виду, что непосредственное сравнение ошибок обучения в задачах типов 1 и 2 является некорректным по причине различного характера целевых данных).

При использовании модели $R_S(x)$ доля успешно решенных тестовых задач (D_V) оказалась выше 75% на всем множестве алгоритмов, а для половины алгоритмов превзошла 90%. Как и следовало ожидать, в случае $R_L(x)$ качество результатов оказалось в целом несколько ниже, но и в этом случае значение D_V не опустилось ниже 70% уровня ни для одного из рассмотренных алгоритмов.

На основании результатов проведенных экспериментов с каноническими регрессионными моделями можно сделать следующие предварительные выводы:

1. Рассмотренный подход применим для оценки значений критерия эффективности, однако точность полученных оценок невелика.
2. Ввиду невысокой точности оценок, предпочтительной является слабая постановка регрессионной задачи, которая определяет лишь выполняемость условия ε-оптимальности без непосредственной оценки значения рассматриваемого критерия.
3. Экспериментальные данные подтверждают адекватность выбора множества I_A* на основе предложенного статистического критерия.

Как и можно было предположить, практически во всех случаях качество результатов, полученных при использовании квадратичной функции регрессии, оказалось выше, чем при использовании линейной функции.

Однако для формулирования окончательного вывода о применимости классических регрессионных моделей для решения исходной задачи априорного выбора оптимального алгоритма планирования необходимо провести оценку качества выбора на основании обеих моделей, что и будет сделано далее.

Таблица 15.3.3

Результаты решения задачи типа 2 с использованием регрессионных моделей

Алгоритм	Вход. набор	Регресс. модель	Π_L		D_L	D_V
			E_{max}	E_{av}		
<ARB,DBL,CFT>	I_V*	$R_L(x)$	0.9724	0.0272	0.985	1.00
<ARB,SGL,CFT>	I_V*	$R_L(x)$	1.0344	0.3143	0.794	0.80
<MTC,DBL,CFT>	I_V*	$R_L(x)$	0.9986	0.1347	0.927	0.94
<MTC,NC,NO>	I_V*	$R_L(x)$	1.0084	0.3985	0.743	0.82
<MTC,NC,ORD>	I_V*	$R_L(x)$	0.7801	0.1542	0.966	0.92
<MTC,SGL,CFT>	I_V*	$R_L(x)$	0.9851	0.2806	0.840	0.82

<*ARB,DBL,CFT*>	I_V*	$R_S(x)$	0.9921	0.0451	0.985	1.00
<*ARB,SGL,CFT*>	I_V*	$R_S(x)$	1.1081	0.2617	0.875	0.82
<*MTC,DBL,CFT*>	I_V*	$R_S(x)$	0.8807	0.1371	0.941	0.98
<*MTC,NC,NO*>	I_V*	$R_S(x)$	0.9171	0.2824	0.824	0.76
<*MTC,NC,ORD*>	I_V*	$R_S(x)$	0.4506	0.1194	1.000	0.96
<*MTC,SGL,CFT*>	I_V*	$R_S(x)$	0.9593	0.3311	0.772	0.80
<*ARB,DBL,CFT*>	I_A	$R_L(x)$	0.8990	0.0481	0.985	1.00
<*ARB,SGL,CFT*>	I_A	$R_L(x)$	0.8816	0.3012	0.816	0.84
<*MTC,DBL,CFT*>	I_A	$R_L(x)$	0.9928	0.1382	0.927	0.94
<*MTC,NC,NO*>	I_A	$R_L(x)$	0.8828	0.3456	0.787	0.82
<*MTC,NC,ORD*>	I_A	$R_L(x)$	0.8134	0.1215	0.978	0.94
<*MTC,SGL,CFT*>	I_A	$R_L(x)$	0.9594	0.3808	0.772	0.70
<*ARB,DBL,CFT*>	I_A	$R_S(x)$	0.3543	0.0335	1.000	1.00
<*ARB,SGL,CFT*>	I_A	$R_S(x)$	0.7189	0.1639	0.955	0.84
<*MTC,DBL,CFT*>	I_A	$R_S(x)$	0.6365	0.0982	0.992	0.96
<*MTC,NC,NO*>	I_A	$R_S(x)$	0.7492	0.1847	0.955	0.80
<*MTC,NC,ORD*>	I_A	$R_S(x)$	0.4387	0.0934	1.000	0.98
<*MTC,SGL,CFT*>	I_A	$R_S(x)$	0.7974	0.2070	0.914	0.82

15.4 ПРИМЕНЕНИЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ЭКСПЕРТНОЙ СИСТЕМЫ ВЫБОРА ОПТИМАЛЬНОЙ ПОДСИСТЕМЫ ПЛАНИРОВАНИЯ ДЕЙСТВИЙ ИНТЕЛЛЕКТУАЛЬНОГО ИНФОРМАЦИОННОГО АГЕНТА

Основное внимание в данном разделе уделяется результатам практического применения аппарата искусственных нейронных систем. Для проверки применимости данного подхода для оценки значения заданного критерия эффективности алгоритма планирования использовалась простейшая нейросетевая модель $R_N(x)$ на базе персептрона (сети прямого распространения) с одним скрытым слоем и единственным выходным элементом (рис. 15.4.1), состоящая из следующих компонентов:

1. Входной слой $\{P_k\}$, выполняющий в случае необходимости препроцессинг исходных данных $\{I_k\}$. Количество узлов N_{in} равно количеству входных параметров.
2. Скрытый слой, содержащий узлы с заданной функцией активации $A(x)$. Ввиду специфики используемого программного обеспечения в качестве функции активации использовался гиперболический тангенс:

$$A(x) = th(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \qquad (15.4.9)$$

Следует отметить, уже доказано, что в общем случае единственным существенным требованием к виду функции активации искусственного нейрона является нелинейность, при выполнении этого условия для любой непрерывной функции нескольких переменных можно построить ИНС, которая вычисляет данную функцию с любой заданной точностью. При этом использование нечетной активационной функции, какой и является $th(x)$, ускоряет процесс обучения.

Схема используемой ИНС (двухслойный персептрон)

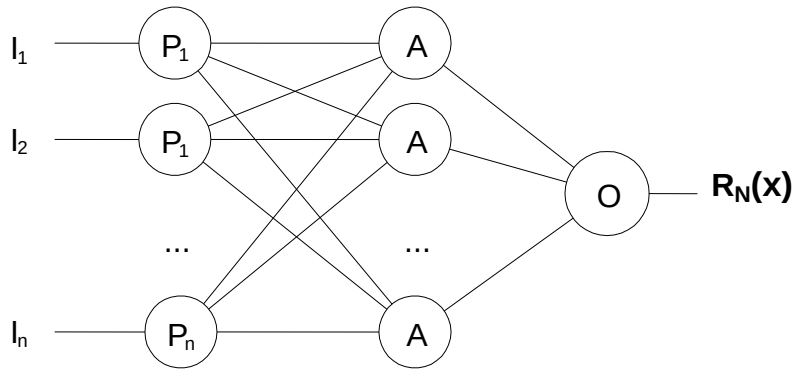


Рис. 15.4.1.

3. Выходной слой, обеспечивающий постпроцессинг результатов в зависимости от решаемой задачи (дискретизация, восстановление исходного масштаба и т.д.). Размер выходного слоя $N_{out} = 1$.

Одним из ключевых вопросов, возникающих при построении систем с использованием двухслойного персептрона, является выбор оптимального размера скрытого слоя N_{hid}. Некоторые известные практические руководства в области искусственных нейронных сетей предлагают формальные правила по определению N_{hid}, в частности:

1. N_{hid} определяется на основании размеров входного и выходного слоев, например: $N_{hid} = N_{in}$ или $N_{hid}=[0.5*(N_{in}+ N_{out})]$, где $[X]$ – операция определения ближайшего к X целого числа.
2. N_{hid} определяется на основании мощности обучающего множества, например $N_{hid}=[0.1\ |\Pi_L|]$.

К сожалению, подобные рекомендации не подкреплены достаточными обоснованиями, они не учитывают специфику конкретной задачи и особенности обучающего множества, хотя во втором случае частично адресуется проблема переобучения ИНС. В литературе приводятся примеры задач (например, классическая задача с двумя спиралевидными входами или задача о шахматной доске), для которых перечисленные правила заведомо не работают.

Существует также ряд методов, основанных на обучении набора сетей с различными размерами скрытого слоя и последующем определении оптимального размера скрытого слоя посредством использования статистических критериев на обучающем множестве, в частности, байесовского критерия Шварца, оригинального и скорректированного информационных критериев Акайке. Однако проведенное экспериментальное сравнение этих методов на группе шаблонных задач продемонстрировало оправданность их применения лишь в случае очень большого размера обучающего множества.

Принимая во внимание вышеизложенное, в данной работе применена адаптивная процедура выбора размера скрытого слоя, заключающаяся в обучении набора сетей с разными размерами скрытого слоя и выборе N_{hid} из условия минимизации контрольной ошибки E'_{av} , вычисляемой в соответствии с (15.4.7) на некотором сравнительном множестве задач. Для корректного сравнения нейросетевой и регрессионной моделей, при определении N_{hid} использовалось альтернативное тестовое множество, не совпадающее с Π_V. На рис. 15.4.2 представлена зависимость E'_{av} от узлов в скрытом слое для $1 \le N_{hid} \le 12$ (верхняя граница выбрана как соответствующая максимальному рассматриваемому значению N_{in}, достигаемому при входном наборе I_A). Как видно из полученных данных, в обоих случаях следует признать оптимальным значением $N_{hid}=5$, причем тот факт, что оптимальное значение N_{hid} не зависит от размера входного множества, может рассматриваться как дополнительное подтверждение корректности процедуры выбора множества I_A*.

Множество I_A * рассматривается как некоторое подмножество обобщённого информационного множества факторов I_A , оказывающих влияние на качество планирования, при априорном определении оптимального алгоритма

$$I_A^* \subset I_A .$$

Обобщённое информационное множество факторов, оказывающих влияние на качество планирования, формируется посредством объединения двух множеств $I_A = I_E \cup I_П$, где множество I_E представляет характеристики среды функционирования агента, а множество $I_П$ – параметры задачи планирования.

Множества I_E и $I_П$ образуются следующим образом: $I_E = \{N_{op}, N_c, N_{pre}, N_{add}, N_{del}, n_+, n_-, H_+, H_-, h_+, h_-\}$ и $I_П = \{N_g\}$.

Элементы множества I_E выделяются в результате параметризации предметной области, при которой различаются следующих два множества:

O – множество операторов предметной области;

Q – множество условий предметной области.

Каждому оператору $o \in O$ ставятся в соответствие следующие характеристики:

n^o_{pre} – количество предусловий;

n^o_{add} – количество добавляемых условий;

n^o_{del} – количество удаляемых условий.

С помощью введенных обозначений можно определить основные параметры описания предметной области, к которым в первую очередь относятся:

$N_{op} = |O|$ – общее количество операторов предметной области;

$N_c = |Q|$ – общее количество условий предметной области;

N_{pre} – среднее количество предусловий одного оператора:

$$N_{pre} = \frac{\sum_{o \in O} n^o_{pre}}{/O/}; \qquad (15.4.10)$$

N_{add} и N_{del} – среднее количество добавляемых и удаляемых условий для одного оператора, определяемые аналогично N_{pre}.

N^*_{pre} – максимальное количество предусловий одного оператора:

$$N^*_{pre} = \max_{o \in O} n^o_{pre}. \qquad (15.4.11)$$

Использование N^*_{pre} позволяет получить гарантированную оценку сверху для глубины дерева поиска, степени ветвления и, следовательно, количества узлов в дереве поиска. При этом очевидно, что в общем случае данная оценка является сильно завышенной, поэтому для более адекватной

оценки может быть использовано среднее количество предусловий одного оператора.

Для каждого из условий предметной области наибольший интерес представляют их частотные характеристики:

n_{q+} – количество операторов, добавляющих условие q;

n_{q-} – количество операторов, удаляющих условие q.

Поскольку для задач с большим размером предметной области учет и практическое использование характеристик каждого из условий является затруднительным, то в дальнейшем будут использоваться их усредненные значения n_+ и n_-:

$$n_+ = \frac{\sum_{q \in Q} n_{q+}}{|Q|} \; ;$$

$$n_- = \frac{\sum_{q \in Q} n_{q-}}{|Q|} \; . \qquad (15.4.12)$$

Значения n_+ и n_- позволяют с некоторой степенью достоверности судить о такой неявной характеристике предметной области как “плотность решений”. Очевидно, что при высокой плотности решений и требовании найти хотя бы одного решение задачи планирования наиболее перспективным режимом поиска будет “поиск в глубину”, поскольку велика вероятность найти решение задачи уже среди первых рассмотренных ветвей. При невысокой плотности решений или требовании найти минимальное (или суб-минимальное) решение предпочтительнее использовать режим “поиска в ширину”.

Дополнительным фактором, характеризующим предметную область, является существование условий с высокой частотой вхождения в постусловия операторов, то есть таких условий q, у которых $n_{q+} >> 1$ или $n_{q-} >> 1$. Попадание подобного условия в текущее множество подцелей на каком-либо этапе ведет к значительному росту степени ветвления дерева поиска из-за большого количества способов достижения для больших n_{q+} и потенциально большого количества конфликтов при больших n_{q-}. Кроме того, в этом случае важное значение начинает играть порядок, в котором подцели выбираются для достижения. Для численного описания данного свойства будут использоваться два показателя:

$H = H_+ + H_-$ – общее количество высокочастотных условий (H_+ и H_- – количество условий с большими n_{q+} и n_{q-} соответственно);

$h = \frac{H}{|Q|} = \frac{H_+ + H_-}{|Q|} = h_+ + h_-$ – плотность высокочастотных условий.

Для исследования влияния существования высокочастотных условий на решение задачи планирования интересно рассмотреть следующие типы предметных областей:

- Отсутствие высокочастотных условий: $h = 0$.
- Небольшая часть условий является высокочастотными: $0 < h << 1$.
- Почти все условия являются высокочастотными: $(1 - h) << 1$.

Для получения более детальной картины может потребоваться исследование влияния каждой составляющей h (то есть h_+ и h_- в отдельности).

На основании определения задачи планирования в качестве ее характеристик может использоваться совокупность факторов, отражающих специфику начального и целевого состояний в отдельности, а также их совокупные характеристики. Для целевого состояния такими факторами являются:

- Общее количество условий в целевом множестве $N_g=|G|$, в том числе подцелей, требующих достижения, и изначально заданных ограничений, если таковые имеются.
- Наличие в целевом множестве высокочастотных условий, поскольку это автоматически влечет вхождение таких условий в текущее множество подцелей, причем на самых верхних уровнях поиска.
- Существование потенциальных конфликтов в целевом множестве, например, наличие подцелей с большим значением n_{q-}, или существование в предметной области операторов, содержащих условия из целевого множества одновременно в *ADD* (добавляющих) и *DEL* (удаляющих) списках ADL-представления.

Зависимость E'_{av} от размера скрытого слоя N_{hid}

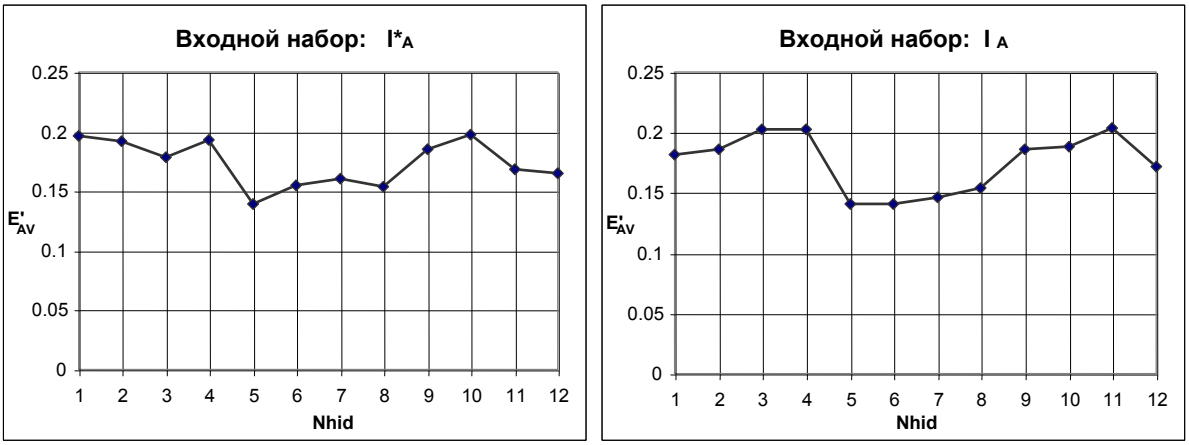


Рис. 15.4.2

Численное моделирование проводилось с использованием пакета Matlab версии 6.0 и библиотеки Netlab 3.2, предоставляющей набор базовых

процедур для симуляции ИНС. Ввиду относительно небольших размеров сети (определяемых количеством узлов и синаптических связей между ними) и обучающего набора данных (множество задач Π_L) вместо стандартного для персептрона метода обучения с обратным распространением ошибки применялась процедура обучения ИНС на основе минимизации функции ошибки с использованием обобщенных методов оптимизации, в частности, квази-ньютоновского алгоритма. Процесс обучения заканчивался в случае, когда относительное изменение функции ошибки не превосходило 10^{-8} на протяжении 10 последовательных итераций. Для оценки качества построенных нейросетевых моделей использовался тот же самый набор показателей, что и в случае канонических регрессионных моделей: E_{max} и E_{av} для задач типа 1, D_L и D_V для задач типа 2.

В зависимости от способа оценки значения критерия эффективности выделяются два типа задач:

- Задача типа 1 (сильная постановка): оценивается непосредственно значение информативной составляющей показателя эффективности v: $Y=\xi$.
- Задача типа 2 (слабая постановка): оценивается выполняемость условия ε-оптимальности алгоритма планирования относительно соответствующего показателя эффективности v: $Y=\xi'$, где:

$$\xi' = \begin{cases} 1, & v \le 1+\varepsilon; \\ 0, & v > 1+\varepsilon. \end{cases}$$

Табл. 15.4.4 содержит результаты обучения и апробации описанной нейросетевой модели $R_N(x)$ для задач типа 1. Ввиду непрерывности аппроксимируемой функции использовался выходной нейрон с линейной функцией активации. Препроцессинг исходных данных проводился так же как и при использовании аналогичных канонических регрессионных моделей. Как видно из представленных данных, результаты применения нейросетевого подхода оказываются сопоставимы с результатами, полученными при использовании квадратичной регрессии, несмотря на значительную разницу в количестве степеней свободы N_f (табл. 4.5 содержит значения N_f, для каждой из рассматриваемых моделей, определяемые как общее число коэффициентов регрессии для регрессионной модели и общее число весовых коэффициентов и коэффициентов смещения для ИНС), что свидетельствует о более высокой эффективности нейросетевой модели.

Таблица 15.4.4

Результаты решения задачи типа 1 с использованием ИНС

Алгоритм планирования действий ИИА	Входной набор	Π_L		Π_V	
		E_{max}	E_{av}	E_{max}	E_{av}

<*ARB,DBL,CFT*>	I_V*	0.3880	0.0976	0.7035	0.1983
<*ARB,SGL,CFT*>	I_V*	0.6281	0.1127	0.9233	0.1826
<*MTC,DBL,CFT*>	I_V*	0.4860	0.1601	0.5693	
<*MTC,NC,NO*>	I_V*	0.5660	0.0426	0.1106	0.1480
<*MTC,NC,ORD*>	I_V*	0.2453	0.0806	0.9633	0.0161
<*MTC,SGL,CFT*>	I_V*	0.7043	0.0998	0.3424	0.2277
					0.0737
<*ARB,DBL,CFT*>	I_A	0.2668	0.0622	0.9328	0.1990
<*ARB,SGL,CFT*>	I_A	0.6104	0.1053	0.7732	0.1724
<*MTC,DBL,CFT*>	I_A	0.4338	0.1234	0.8364	0.1391
<*MTC,NC,NO*>	I_A	0.3077	0.0342	0.0692	0.0280
<*MTC,NC,ORD*>	I_A	0.1755	0.0403	0.8401	0.2381
<*MTC,SGL,CFT*>	I_A	0.5723	0.0909	0.4383	0.0372

Как и следовало ожидать, ошибки обучения при использования множества I_A заметно ниже, чем в случае I_V*, что объясняется более чем 50% разницей в числе синаптических связей. При этом не наблюдается существенной разницы в величине ошибок на тестовом множестве, что может рассматриваться как подтверждение адекватности предложенного критерия выбора I_A*. Большие значения E_{max} для некоторых алгоритмов свидетельствуют о существовании задач, в которых используемая модель не смогла произвести адекватную оценку показателя. В то же время сравнительно низкие значения E_{av} указывают на незначительную долю подобных случаев, что позволяет признать данную модель удовлетворительной.

Таблица 15.4.5

Количество степеней свободы N_f

Входное множество	Размер входного множества	Модель		
		$R_L(x)$	$R_S(x)$	$R_N(x)$
I_V*	7	8	57	46
I_A	12	13	157	71

В табл. 15.4.6 представлены результаты использования нейросетевого подхода для решения задач типа 2. Поскольку задача данного типа может быть рассмотрена как стандартная задача бинарной классификации, в качестве функции активации выходного нейрона была использована логистическая функция:

$$y = \frac{1}{1+e^{-x}}. \qquad (15.4.10)$$

Таблица 4.6

Результаты решения задачи типа 2 с использованием ИНС

Алгоритм	Входной набор	$П_L$		D_L	D_V
		E_{max}	E_{av}		
<ARB,DBL,CFT>	I_V*	0.1275	0.0022	1.00	1.00
<ARB,SGL,CFT>	I_V*	0.2294	0.0286	1.00	0.80
<MTC,DBL,CFT>	I_V*	0.1279	0.0041	1.00	0.96
<MTC,NC,NO>	I_V*	0.1292	0.0141	1.00	1.00
<MTC,NC,ORD>	I_V*	0.2094	0.0359	1.00	1.00
<MTC,SGL,CFT>	I_V*	0.9700	0.0706	0.97	0.86
<ARB,DBL,CFT>	I_A	0.1397	0.0019	1.00	1.00
<ARB,SGL,CFT>	I_A	0.5086	0.0201	0.99	0.82
<MTC,DBL,CFT>	I_A	0.0840	0.0018	1.00	0.88
<MTC,NC,NO>	I_A	0.0503	0.0062	1.00	0.84
<MTC,NC,ORD>	I_A	0.1056	0.0140	1.00	1.00
<MTC,SGL,CFT>	I_A	0.5024	0.0206	0.99	0.84

Как следует из представленных данных, для задачи типа 2, нейросетевые модели превосходят по качеству полученных результатов канонические регрессионные модели, что в некоторой степени может быть объяснено использованием специализированной активационной функции в выходном слое. В частности, для алгоритмов *<ARB,DBL,CFT>* и *<MTC,NC,ORD>* обе нейросетевые модели корректно отработали на всем тестовом множестве, а первая модель также успешно решила задачу для алгоритма *<MTC,NC,NO>*. В то же время, для алгоритма *<MTC,SGL,CFT>* качество результатов оказалось на одном уровне с $R_S(x)$, несмотря на более низкие ошибки обучения.

Суммируя результаты применения нейросетевого подхода для решения задач обоих типов, можно сделать заключение о том, что основные выводы относительно канонических регрессионных моделей применимы и к ИНС-моделям, а именно:

1. Предпочтительная область применения: качественные оценки эффективности алгоритмов планирования.
2. Расширение набора входных данных с I_A* до I_A не оказывает существенного влияния на качество оценки.

При этом, в отличие от регрессионных моделей, аналогичное качество результатов получено при использовании моделей значительно меньшего размера, что свидетельствует о предпочтительности использования ИНС-моделей для оценки значений критерия качества алгоритма планирования.

16. ХАРАКТЕРИСТИКА СИСТЕМ ПРЕДСТАВЛЕНИЯ И ПРИОБРЕТЕНИЯ ЗНАНИЙ НА ОСНОВЕ АГЕНТНЫХ ТЕХНОЛОГИЙ

16.1 ТИПЫ ИНТЕЛЛЕКТУАЛЬНЫХ АГЕНТОВ

В результате анализа известных прикладных систем можно выделить следующие типы интеллектуальных агентов:

1. Кооперативные агенты, способные не только к автономному изолированному функционированию, но и к совместной деятельности с другими агентами, в частности координации действий, разработке общих планов и разрешению конфликтов. Примерами агентов являются проект Pleiades университета Карнеги-Меллон, системы MII и ADEPT.
2. Интерфейсные агенты, задачей которых является взаимодействие с пользователем (а не с другими агентами) и помощь ему в выполнении некоторой деятельности. Данный тип агентов также иногда называют персональными ассистентами. Существующие реализации включают различные справочные системы, торговые помощники, системы поддержки документооборота и развлекательные системы.
3. Мобильные агенты, обладающие способностью выполнять свои функции при различном расположении внутри среды обитания. Наиболее естественной средой функционирования для таких агентов являются различные вычислительные сети или системы связи. Необходимо также заметить, что сама по себе мобильность не является ни необходимым, ни достаточным свойством интеллектуального агента.
4. Информационные агенты возникли в виде отдельного класса в результате резко возникшей необходимости поиска, сбора и переработки большого количества информации с относительно простым доступом. В первую очередь к данной группе относятся системы поиска в Интернет, например поиск в WWW (Jasper, Webwatcher) и фильтрация архивов телеконференций (NewT).
5. Реактивные агенты составляют специальную группу агентов, которые не располагают какой-либо внутренней моделью среды, а действуют лишь в ответ на определенное состояние окружающей среды или изменение состояния. Примерами таких систем являются «ситуационный автомат», различные системы моделирования общественного поведения, игровые приложения.
6. Гибридные агенты, совмещающие в себе особенности, присущие различным вышеперечисленным классам. К данной группе относятся, в частности, InteRRaP, сочетающий в себе реактивный и кооперативный модули, система мониторинга пациентов Guardian, а также различные мобильные информационные агенты.
7. Гетерогенные агентные системы, в отличие от гибридных агентов, состоят из нескольких агентов, принадлежащих к разным классам. Основной мотивацией при создании таких систем является построение интеграции

существующих специализированных систем (ARCHON), при этом одним из основных вопросов оказывается организация взаимодействия между агентами.

Несмотря на очевидные преимущества и перспективы применения интеллектуальных агентов в различных научных и практических областях, данному подходу присущ также ряд ограничений, в частности:

- Отсутствие общего контроля над интеллектуальными агентами может повлечь значительные сложности при необходимости учета глобальных ограничений, требований гарантированного ответа в реальном времени и обхода возможных тупиковых ситуаций.
- Отсутствие глобальной перспективы: поскольку в реальности агентные системы не могут обладать полной системой знаний об окружающем мире, возникает вопрос о возможности нахождения оптимальных или субоптимальных решений на основе локальной базы знаний.
- Проблема доверия: насколько можно доверять интеллектуальным агентам при автономном выполнении возложенных на них функций, особенно когда такой агент действует в реальном мире от имени некоего физического лица или организации.

Тем не менее, в настоящее время наблюдается повышенный спрос на технологию интеллектуальных агентов со стороны ведущих мировых корпораций, что стимулирует большой объем научно-исследовательской работы в данной области, которая ведется в нескольких основных направлениях:

- Направление по теории агентов занимается исследованием и разработкой спецификаций, концептуализацией агентов, определением свойств и способов их формализованного представления. Исследуются признаки, отличающие интеллектуальных агентов от обычного программного обеспечения.
- Направление по архитектуре агентов изучает вопросы реализации спецификаций, аппаратные и программные аспекты проблемы построения вычислительной системы, удовлетворяющей заданным свойствам. Примерами наиболее известных архитектур интеллектуальных агентов являются: GISA (Generic Intelligent Software Agent), BDI (Belief-Desire-Intention) и FIPA (Foundation for Intelligent Physical Agents).
- Направление по языкам программирования агентов исследует способы формального описания теоретических принципов, вопросы поиска оптимальных примитивов при кодировании агентов, эффективной компиляции и выполнении программ. К данной группе, в частности, относятся работы, посвященные реализации агентов для Интернет и корпоративных сетей, исследованию особенностей применения объектно-ориентированного подхода и концепции агентно-ориентированного программирования.

16.2 ПРЕДСТАВЛЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ АГЕНТОВ В ОБЛАСТИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Одно из основных направлений развития искусственного интеллекта базируется на идеи интеллектуального агента. Многие исследователи описывают искусственный интеллект как науку об агентах, которые выполняют заданные действия на основании результатов актов восприятия среды.

Под агентом понимают систему, имеющую возможность принимать воздействие из внешнего мира, определять свою реакцию на это воздействие и осуществлять его.

В общем случае агент (рис. 16.2.1) воспринимает среду при помощи датчиков и воздействует на нее с помощью исполнительных механизмов. (Например, человек, рассматриваемый в роли агента, имеет различные органы чувств – датчики; руки, ноги – исполнительные механизмы. Робот, выступающий в роли агента, в качестве датчиков может иметь видеокамеры, инфракрасные дальномеры, а его исполнительными механизмами являются различные двигатели. Программное обеспечение, выступающее в роли агента, в качестве входных сенсорных данных получает коды нажатия клавиш, содержимое файлов, сетевые пакеты, а воздействие на среду выражается в том, что программное обеспечение записывает файлы, осуществляет некоторые процедуры информационно-аналитического характера, передает сетевые пакеты или выполняет иные предусматриваемые действия).

Каждый агент реализует функцию, отражающую математическое описание последовательности от актов восприятия среды к действиям. Под восприятием понимают полученные агентом данные в какой-то конкретный момент времени. Функция реализуется программой агента.

Общая структура агента

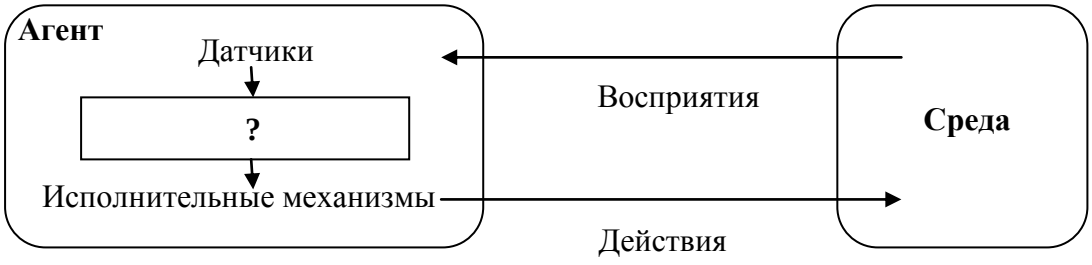


Рис. 16.2.1

Выделяется понятие рационального агента, то есть агента, который выполняет действия, обеспечивающие наиболее успешное его функционирование, следовательно, для каждого последовательного набора

актов восприятия агенту на основании его встроенных знаний необходимо выбрать действие, которое максимизирует его показатели производительности. Для оценки рациональности требуется определенный способ измерения успеха. Критерии успеха, наряду с описанием среды, а также датчиков и исполнительных механизмов агента, представляют полную спецификацию задачи, которую решает агент.

Оценка рациональности действий агента зависит от ряда факторов, среди которых:

– показатели производительности, на основе которых определяются критерии успеха;

– знания о среде, ранее приобретенные агентом;

– действия, выполнение которых возможно агентом;

– дошедшая до настоящего времени последовательность актов восприятия агента.

Рациональный агент является автономным, то есть должен обучаться всему, что может освоить для компенсации неполных априорных знаний. После достаточного опыта существования в своей среде поведение рационального агента может по сути стать независимым от его априорных знаний, поэтому включение в проект способности к обучению позволяет проектировать простых рациональных агентов, которые могут действовать успешно в исключительно разнообразных вариантах среды.

Описание проблемной среды, в которой функционируют агенты, можно классифицировать по нескольким параметрам:

– полностью наблюдаемая или частично наблюдаемая (в зависимости от степени доступа датчиков агента к полной информации о состоянии среды);

– детерминированная или стохастическая (в зависимости от связи последующего состояния среды с текущим);

– эпизодическая, когда агент принимает независимые решения в каждом эпизоде своей деятельности, или последовательная, когда последующее функционирование зависит от решения предыдущего этапа;

– статическая или динамическая (в зависимости от изменчивости среды в процессе выбора агентов своей реакции);

– дискретная или непрерывная (в зависимости от способа учета времени и восприятия агента);

– одноагентная или мультиагентная.

Наиболее сложными для работы являются частично наблюдаемые, стохастические, последовательные, динамические, непрерывные, мультиагентные среды, когда агенту приходится действовать в условиях неопределенности и изменчивости среды.

Основной задачей искусственного интеллекта в области агентов признана разработка программной реализации функции агента, направляющей восприятия к действиям. Данная программа должна работать на вычислительной машине с определенными датчиками и исполнительными механизмами, то есть с определенной архитектурой. Исходя из этого архитектура агента рассматривается как совокупность программного и

аппаратного обеспечения.

Существует набор основных видом программ агентов:

– простейшие рефлексные агенты, отвечающие непосредственно за акт восприятия;

– рефлексные агенты, основанные на модели, поддерживающие внутреннее состояние;

– агенты, выполняющие действия на основе цели, ставят задачу достигнуть определенного результата;

– агенты, действующие, опираясь на безопасность и полезность, пытаются максимально эффективно провести свои действия.

Каждый агент способен повышать эффективность своей работы за счет обучения.

Для случая, когда возникает необходимость использования агентов для решения сложных масштабных задач реального мира, необходимо ввести систему планирования, иначе агент может быть подавлен большим количеством действий, не относящихся напрямую к решаемой задаче, или не сможет справиться из-за отсутствия развитой эвристической функции. Так как большинство реальных задач поддаются лишь частичной декомпозиции, то планировщик должен работать над подцелями независимо и выполнять дополнительную работу по объединению результирующего решения.

Задачи планирования действий агентов могут решаться в условиях временных ограничений, неопределенных вариантах среды, недетерминированных проблемных областях, когда агентам приходится работать с неполной, неправильной информацией.

Система планирования должна координировать действия кооперирующих агентов и не допускать их конкуренции друг с другом в борьбе за выполнение функций полезности. Любой агент в среде должен учитывать наличие других агентов, прогнозировать их возможные планы, определять влияние других на его собственные планы, определять наиболее эффективные действия с учетом описанных ограничений.

В большинстве случаев агенты не имеют доступа ко всей необходимой информации о своей среде, поэтому вынуждены действовать в условиях неопределенности. При наличии неопределенности невозможно использовать упрощения, допустимые в дедуктивном логическом выводе. Для описания знаний агента используется теория вероятностей, позволяющая сформировать относящиеся к решаемой проблеме высказывания с определенной степенью уверенности. Вероятности отображают метод суммарного учета неопределенности, которая возникает из-за экономии усилий или отсутствия знаний, а также во всех сложных, динамических, труднодоступных средах. В процессе получения агентом результатов восприятия вероятностные оценки обновляются таким образом, чтобы отражать полученные свидетельства. Вероятности, оцененные до получения свидетельства, являются априорными, после получения – апостериорными.

Агенты называют интеллектуальными, если они обладают рядом особенностей:

– имеют возможность действовать на объекты в своей среде, на других агентов и на самих себя (действие);

– могут общаться с остальными агентами в системе (общение);

– действуют на основании цели и обладают рядом интенциональных качеств (убеждение, желание, намерение и пр.);

– несут обязанности при предоставлении услуг (наличие обязательств);

– обладают собственными ресурсами, позволяющими обеспечить их автономию (автономия);

– способны воспринимать окружающую среду (восприятие с ограниченным разрешением) и выстраивать частичное представление этой среды на основе ее восприятия (локальное представление среды);

– прогнозируют возможные изменения среды (предвидение);

– обучаются, эволюционируют (эволюционный и адаптационный потенциал);

– самоорганизуются и самовоспроизводятся (самосохранение);

– обладают системой планирования действий;

– способны накапливать знания и анализировать опыт, применять полученные знания для достижения новых целей.

Интеллектуальные агенты содержат базу знаний, механизмы решения и анализа действий, что позволяет им построить развитую модель внешнего мира. Они автономны, характеризуются целесообразным поведением с группе агентов, обладают когнитивной, рассуждающей, коммуникативной и ресурсной функциями.

Изначально для решения глобальной проблемы, опираясь на классическую теорию искусственного интеллекта, создавался агент, обладающий необходимыми знаниями, способностями, вычислительными ресурсами. Позднее получило развитие направления, связанного с созданием многоагентных (мультиагентных) систем, представляющих собой множество взаимодействующих агентов. В подобных системах каждый агент владеет лишь частичной информацией о решаемой задаче и может выполнить действия, которые внесут вклад в преодоление общей проблемы. В многоагентных системах имеет значение понятие кооперации между агентами, означающее их коллективную работу, разделение функций и обязанностей с целью достижения совместного результата. Кооперация агентов приводит к синергетическому эффекту при создании многоагентных систем, предоставляет возможность совместного решения задачи, которую сложно решить одному агенту, повышает эффективность и производительность каждого агента, максимизирует качество общего решения за счет детальной проработки частных решений различными агентами и обмена информацией между ними.

Эффективность применения многоагентной системы и ее преимущество над централизованными экспертными системами убедительно проявляется при преодолении сложной задачи, решение которой получается объединением нескольких точек зрения различных агентов. На рисунке 16.2.2 представлена типичная последовательность распределенного решения задачи

кооперацией агентов.

Агент-субординатор на данной схеме отвечает за декомпозицию исходной проблемы на отдельные задачи, которые распределяются между агентами-исполнителями. В процессе решения своей задачи агенты-исполнители могут ее разделить на ряд подзадач. Агент-интегратор отвечает за общий результат, проводит композицию частных решений, соответствующих выделенным задачам.

Типовая схема распределенного решения задачи

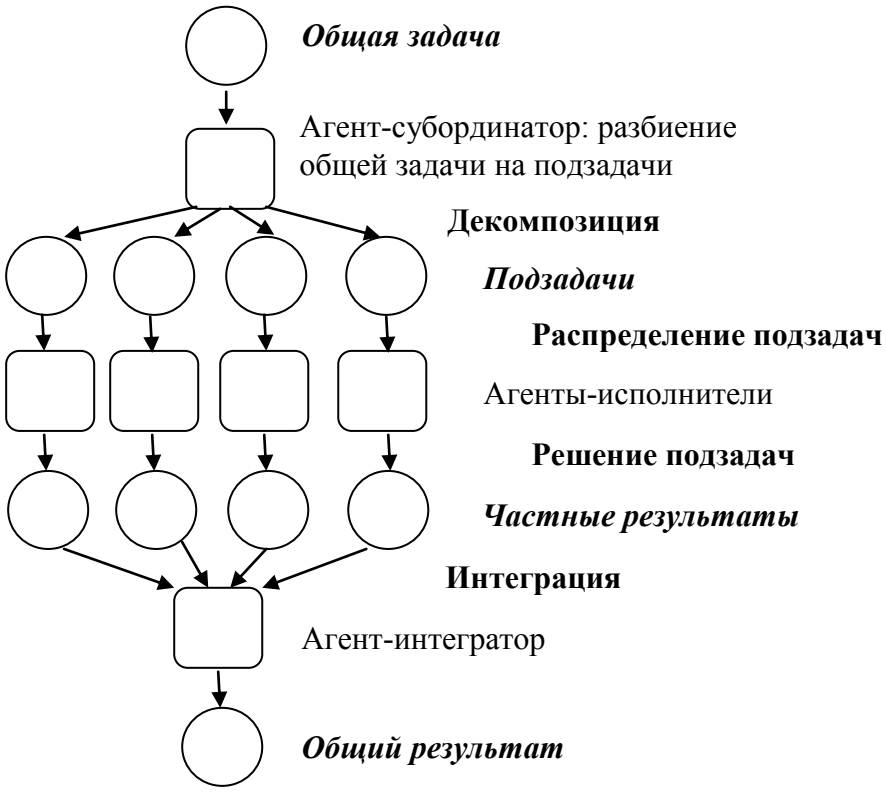


Рис. 16.2.2

Агент-субординатор на данной схеме отвечает за декомпозицию исходной проблемы на отдельные задачи, которые распределяются между агентами-исполнителями. В процессе решения своей задачи агенты-исполнители могут ее разделить на ряд подзадач. Агент-интегратор отвечает за общий результат, проводит композицию частных решений, соответствующих выделенным задачам.

16.3 ДОСТИЖЕНИЯ В ОБЛАСТИ РАЗРАБОТКИ МЕТОДОЛОГИЙ И ПЛАТФОРМ ДЛЯ СОЗДАНИЯ МНОГОАГЕНТНЫХ СИСТЕМ

При анализе проблем жизненного цикла многоагентных систем особое внимание обращается на этапы их разработки. В связи с этим в последующем изложении описываются достижения в области разработки методологий и платформ для создания многоагентных систем.

Исследователями представляются результаты развития методологии и платформы для разработки мультиагентных систем. Развитие базируется на использовании формально-логических методов. Основными целями развития является разработка и экспериментальная реализация методов формально-логической спецификации мультиагентных систем с ограничениями на время реакции. При разработке предусматривается построение алгоритмов верификации и планирования с темпорально расширенными целями, а также методов накопления и анализа опыта для определения рациональных вариантов поведения системы. Для решения поставленной задачи предлагаются:

• метод формально-логической спецификации мультиагентных систем с временными ограничениями. Далее соответствующий формализм именуется MASL;

• алгоритм верификации систем по спецификации MASL;

•алгоритм построения мультиагентных планов с темпорально расширенными целями, выраженными в ограниченном варианте логики MASL;

• метод накопления и анализа опыта для мультиагентных систем.

Многими авторами рассматриваются программные решения и технологии, позволяющие создавать мультиагентные системы для решения различных задач в рамках парадигмы Semantic Web, на платформе Jade.

Во многих работах описывается алгоритм работы агента эксперта, функционирующего в рамках модуля многоагентной системы (МАС), реализующего базовый класс для решения задач в определенной предметной области. При детализации описания дается общая характеристика модуля МАС и перечисляются его основные свойства, представляется базовая модель класса агента эксперта, основанная на модели FIPA, рассматривается общая схема обработки запроса пользователя сообществом агентов, описывается пошаговый алгоритм работы агента эксперта по обработке запроса пользователя, перечисляются реализованные агенты эксперты в прототипе системы для апробации полученных результатов, и делается вывод о том, что предложенный алгоритм работы агента эксперта позволяет выполнять запросы в рамках модуля МАС, а также предоставляет возможность адаптации модуля МАС к решению новых задач путем добавления нового агента эксперта.

Рядом авторов анализируются современные принципы самоорганизации МАС и рассматриваются причины, по которым их технологическая интеграция обеспечивает новый импульс развития

информационных технологий в контексте самых сложных современных приложений. При анализе проводится систематизация и даётся краткая характеристика моделей и механизмов самоорганизации, реализуемых в архитектуре МАС. Обращается особое внимание на биологические механизмы самоорганизации. Новые возможности самоорганизующихся МАС демонстрируются на примере самообучающейся системы распределенного обнаружения вторжений в компьютерные сети, в которой решается задача самоконфигурирования оверлейной сети. Возможности самоорганизующейся МАС для решения крупномасштабных задач управления в реальном времени представляются на примере задачи адаптивного управления дорожным движением в мегаполисах.

В ряде работ рассматривается архитектура базовых агентов многоагентной системы комплексной защиты информации в компьютерных сетях.

Отдельными авторами предлагается подход к построению систем защиты информации в виде комплексных интеллектуальных многоагентных систем. Основное внимание уделяется предлагаемой модели многоагентной системы защиты информации (МАСЗИ), описывается новая информационная технология создания интеллектуальных прикладных систем, обеспечивающая резкое сокращение трудоемкости разработки и эксплуатации этих систем.

Ряд работ посвящается описанию процесса управления конфигурацией как части инженерии программного обеспечения. При анализе различных реализаций систем управления конфигурацией показывается необходимость применения одноранговой распределенной архитектуры.

В периодических публикациях описывается процесс разработки модели генерации интеллектуальных мультиагентных систем. Идея универсального генератора агентов появляется в связи с потребностью разработчиков в оптимизации и ускорении процесса разработки и изменения систем, основанных на модели взаимодействующих интеллектуальных агентов. Генератор ориентируется на использование широкого спектра интеллектуальных методов для создания агентов мультиагентных систем. Производится апробация системы генерации мультиагентных систем на задачах поддержки технического обслуживания и ремонта оборудования.

В некоторых статьях затрагиваются вопросы разработки архитектуры многоагентной системы защиты информации (СЗИ), создания моделей представления распределенных СЗИ классов атак и действий агентов по их обнаружению. Описывается один из наиболее распространенных классов атак на компьютерные сети и рассматриваются механизмы их обнаружения. Приводится архитектура многоагентной системы защиты информации.

В научных публикациях раскрываются процессы разработки и реализации мультиагентной системы поддержки принятия решений, позволяющей в масштабе реального времени осуществлять оценку и прогноз эксплуатационной надежности аппаратной части систем управления с учетом ряда факторов, влияющих на надежность

Исследователями обращается внимание на разработку алгоритмов адаптивного поведения интеллектуальных агентов в задаче исследования проходимости естественной среды функционирования, методы построения карт априори неизвестной области, проблемы автоматической корректировки заранее известных карт, а также способы использования составленных карт местности для повышения эффективности выполнения поставленных перед агентом задач. Экспериментально обосновывается продуктивность разработанных алгоритмов исследования естественной среды функционирования интеллектуальных агентов на базе специализированного программного средства, предназначенного для моделирования их адаптивного поведения.

В ряде публикаций описывается исследование моделей, методов и алгоритмов адаптивной фильтрации потоков информации в группах пользователей и разработка многопользовательской многоагентной обучаемой пользователями автоматизированной системы адаптивной фильтрации, обеспечивающей автоматическое выявление и блокирование незапрашиваемой рассылки в потоках сообщений систем электронной почты глобальной информационной сети Интернет.

В научных исследованиях представляются определения и характеристики интеллектуальных агентов, сферы их применения и трудности, существующие в развитии данной концепции. В процессе представления рассматривается схема мультиагентной системы для решения задачи маркетинга жизненного цикла наукоемкой продукции. Предлагаемое решение основывается на построении информационно-логистической системы поддержки жизненного цикла наукоемкой продукции на основе мультиагентных систем.

16.4 КОНЦЕПТУАЛЬНЫЕ МОДЕЛИ МНОГОАГЕНТНЫХ СИСТЕМ

В результате анализа вербальных представлений функциональных спецификаций многоагентных систем разработана типовая концептуальная модель интеллектуального информационного агента из состава многоагентной системы в инфокоммуникационной среде, приведённая на рис. 16.4.1. Построенная модель рассматривается как опорный базис для последующей разработки расширенной объектно-ориентированной модели, предназначаемой для определения показателей качества многоагентной системы.

Концептуальная модель интеллектуального информационного агента из состава многоагентной системы в инфокоммуникационной среде

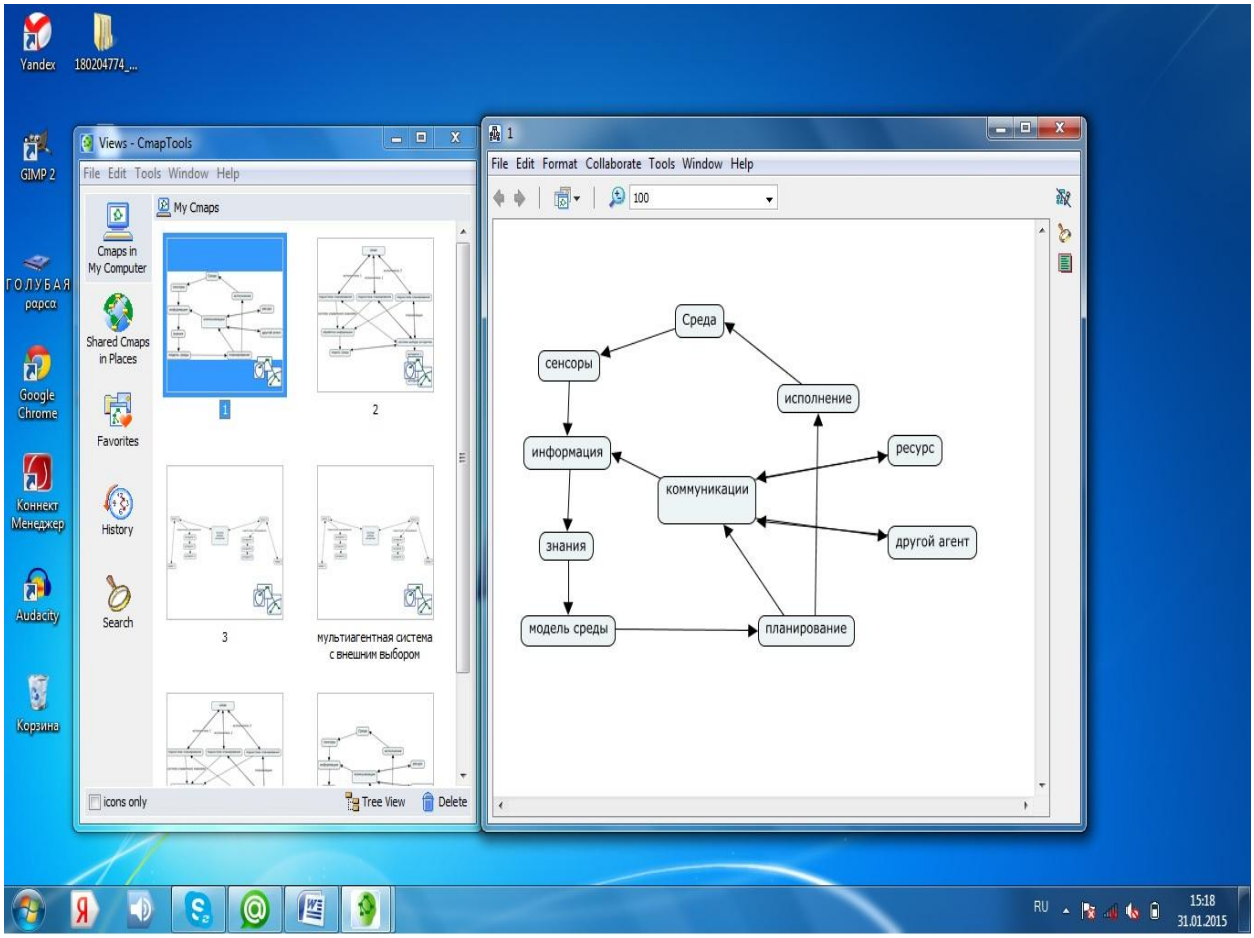


Рис. 16.4.1

Последующая формализация развёрнута в виде типовой концептуальной модели многоагентной системы, имеющей субагентную архитектуру (рис. 16.4.2).

Концептуальная модель многоагентной системы с субагентной архитектурой

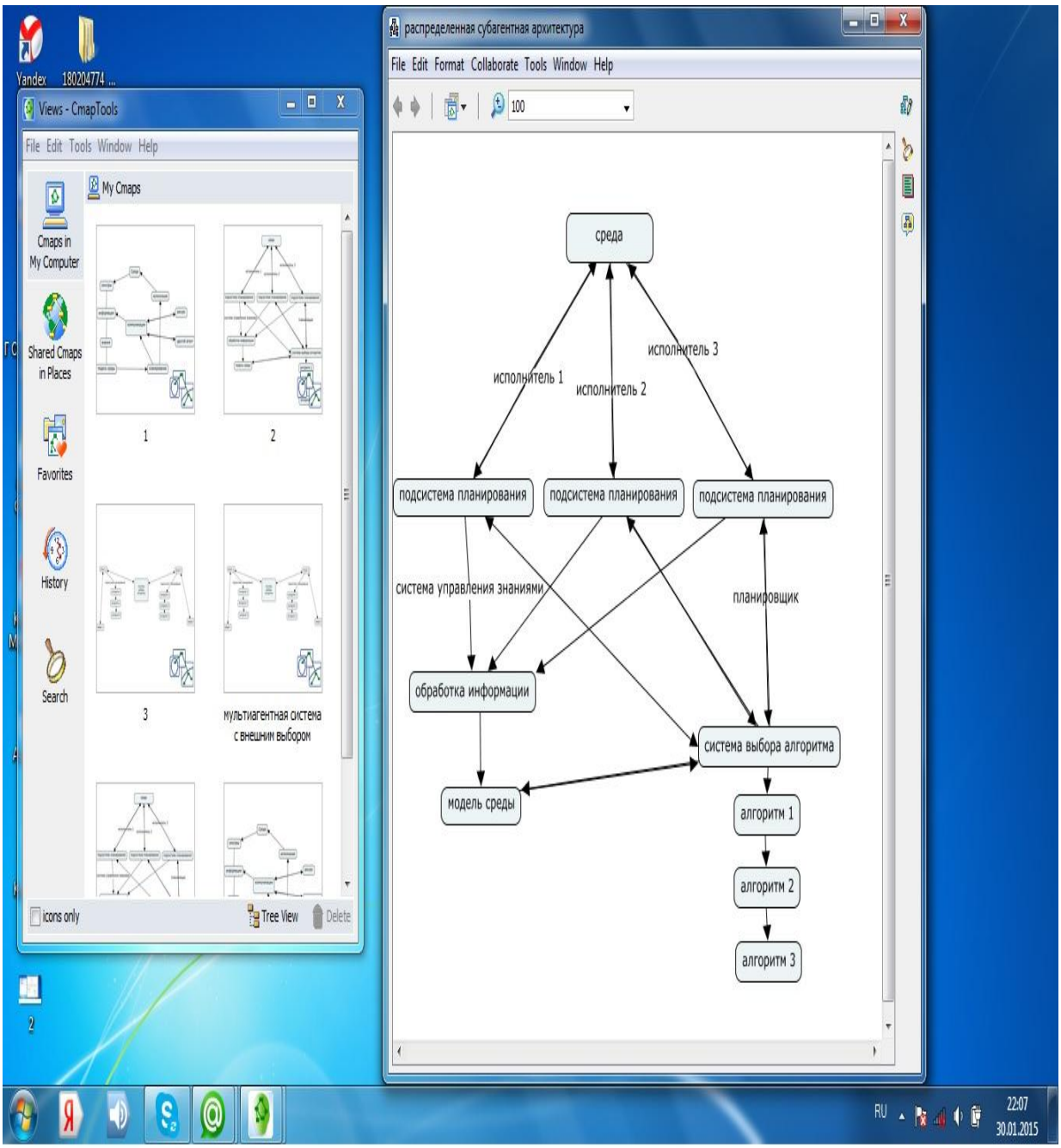


Рис. 16.4.2

Специфика многоагентных систем с внешней системой выбора учтена в разработанной типовой концептуальной модели, приведённой на рис. 16.4.3.

Концептуальная модель многоагентной системы с внешним выбором

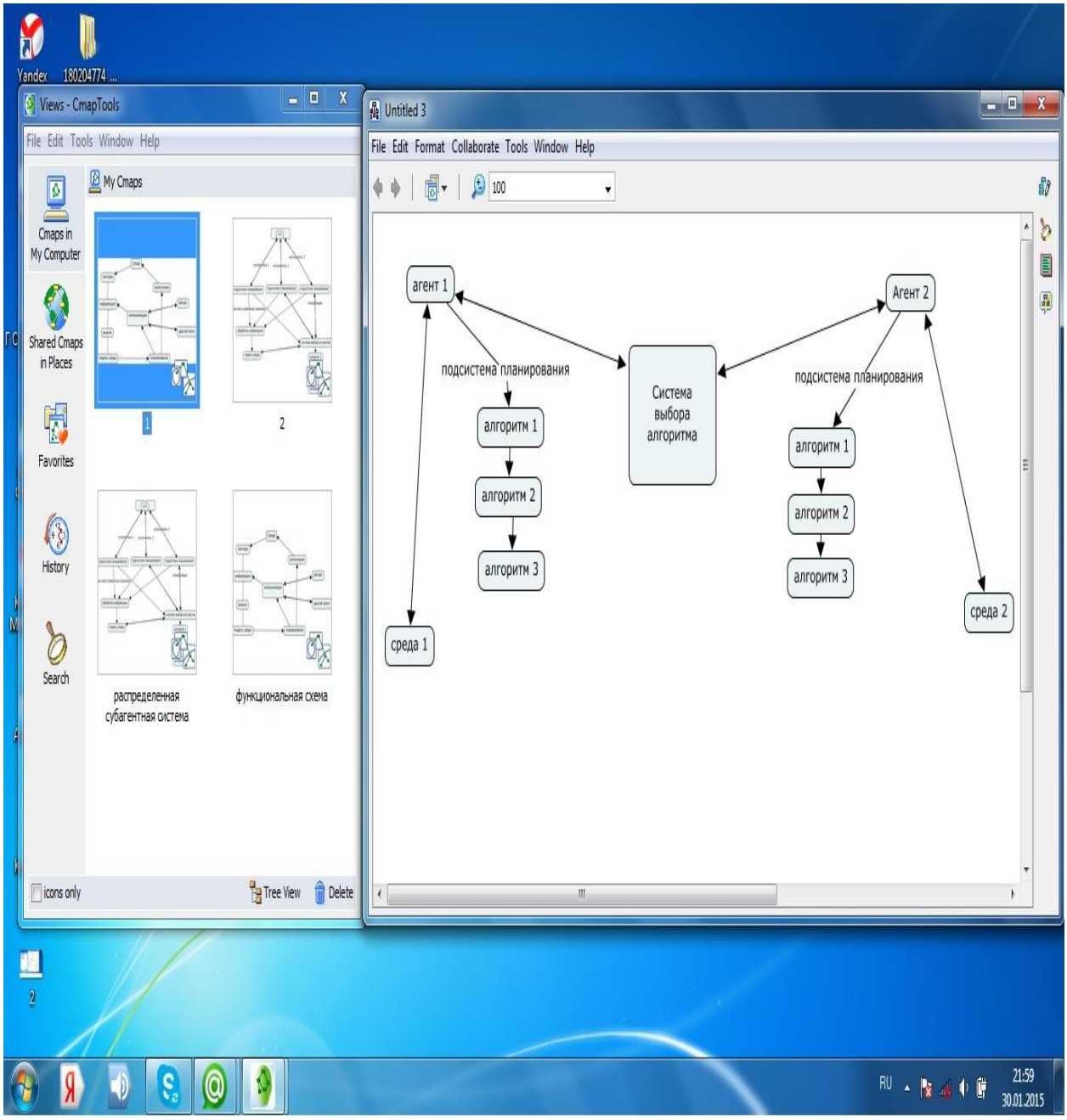


Рис. 16.4.3

Каждая из построенных моделей может использоваться для последующей разработки расширенных объектно-ориентированных моделей многоагентных систем.