НАЗНАЧЕНИЕ «СПРАВЕДЛИВЫХ» ПРИОРИТЕТОВВ ТЕХНОЛОГИЧЕСКИХ ХАБАХ НА ОСНОВЕ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ И НЕЧЕТКОЙ ЛОГИКИ

А.А. Емельянов (Москва, Смоленск), О.В. Булыгина (Смоленск), Н.З. Емельянова (Москва), Р.В. Соколов (Санкт-Петербург)

Реализация программных или планируемых действий довольно часто требует назначения приоритетов. Приоритет действий определяет порядок их выполнения.

В математике: приоритет – ранг или старшинство операции , либо оператора. Например, операцию умножения наделяют бóльшим приоритетом , чем операцию сложения, поэтому в выражении $x+y\cdot z$ будет получено сначала произведение y и z, а потом уже сумма.

В теории массового обслуживания: приоритет – свойство, которое меняет порядок обслуживания требований какого-то приоритетного класса и делает его отличающимся от хронологического, определенного моментом появления требований.

Однако в сложных мультипроектных и мультипроизводственных случаях иногда появляются претензии о «несправедливом» назначении приоритетов. Некоторые требования претендуют на «справедливое» переназначение приоритетов в соответствии с их пожеланиями. – Как правило, это заказы на выполнение работ, где срочность влияет на многое, в том числе на доход или ущерб, получаемые в процессе обслуживания. Особенно актуальным стало назначение приоритетов в так называемых технологических хабах – вкоммуникационных узлах производственной сети.

Понятие хаба существует в системной инженерии и в разных областях проектной и оперативно-диспетчерской деятельности:

1. В методах исследования операций, в IT и в моделирующих программах [1].
2. В авиационной промышленности это может быть участок изготовления деталей крыльев самолета из композитных материалов, который связан с технологией производства принципиально разных деталей воздушный судов по их возможностям, по их стоимостям и по сложности технологий изготовления [7].

В авиационном транспорте существуют два типа хабов:

1) узловой аэропорт на территории какого-то региона [4];
2) в крупных аэропортах – службы типа Ground Handling, которые организуют техническое обслуживание, если нужно – небольшой ремонт, и подготовку к вылету воздушных судов разных типов. В этих службах необходимость «справедливых приоритетов» особенно важна, т.к. затраты авиакомпаний по стоимости и по времени на Ground Handling сильно различаются для Boeing–777 и Sukhoi SJ–100 [10]. Существенно различаются и штрафные потери за задержку подготовки самолетов.

В инноватике при решении задач продвижения инновационных проектов в регионы появляются виртуальные экономические нечеткологические хабы [9].

Существуют классические алгоритмы назначения приоритетов:

1) абсолютных, когда необходимо прерывание начатых действий по обслуживанию требования при появлении более приоритетного (в производственных и транспортных системах практически не применяются, в основном в *IT*).
2) относительных, если речь идет о позиции требования в возникающей очереди на обслуживание, когда правило *FIFO* существует только внутри определенного класса требований, имеющего конкретный приоритет в очереди;
3) динамические, если идет речь о соблюдении принципа «справедливости».

Справедливость – понятие древнее и довольно нечеткое, часто зависит и от страны пребывания и других условий. Виды справедливости: социальная, уравнительная и распределительная. Распределительная справедливость (в нашем случае) требует пропорциональности в отношении к людям согласно тому или иному критерию, например, «равное – равным, неравное – неравным», «каждому – своё» и т.п.

Естественным является желание проверять эффективность того или иного алгоритма назначения приоритетов в хабе с помощью имитационной модели. Если речь идет об абсолютных или относительных приоритетах, то современные моделирующие пакеты предоставляют такие средства независимо от того, сколько в узле обслуживания каналов. Например, в Actor Pilgrim достаточно задать операнд abs в операторе serve или операнд prty в операторе queue, соответственно (рис. 1-*а*).

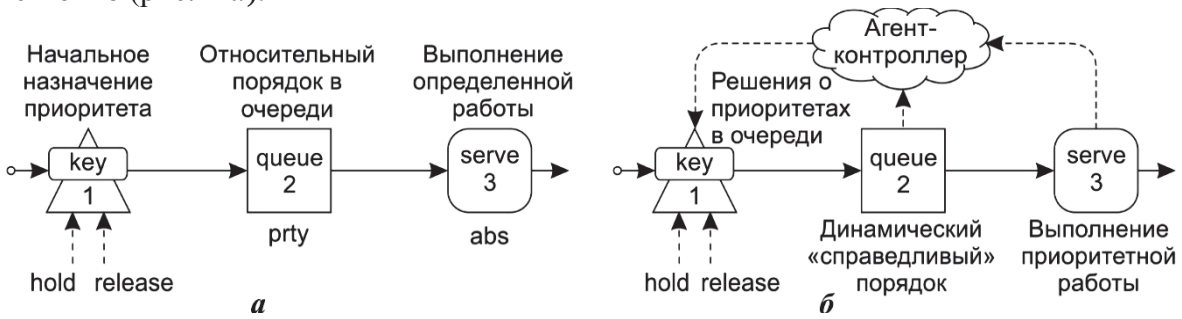

Рис. 1. Управление порядком требований в очереди: *а* – очередь с относительными приоритетами; *б* – управление динамическим «справедливым» порядком

Динамические приоритеты

Одним из первых четких (*crisp*) алгоритмов назначения «справедливых» динамических приоритетов был алгоритм Дж. Р. Джексона [11] в 1960-е годы. Он же впервые предложил методику вычисления последовательности вероятностей времени ожидания в системе с динамическим приоритетом для варианта $M/G/1$. Суть метода:

1. Считается, что время измеряется в дискретных единицах, например, в мин.
2. Исходя из заданного распределения вероятностей, поступающему требованию назначается определенный разряд номера срочности.
3. Вновь поступающее требование получает преимущество перед ожидающим требованием в том и только том случае, если разность между номером разряда срочности вновь поступающего требования и номером разряда предыдущего требования не меньше времени, в течение которого ожидает предыдущее требование.

Этот случай является примером того, как администрация «проявляет заботу» о требованиях тех клиентов, которые ожидали в течение длительного времени. Например, при пиковой загрузке можно «проталкивать» те заявки, которые ожидали исполнения в цехах в течение длительного времени [5]. В настоящее время алгоритм Джексона используется в широкополосных цифровых сетях с интеграцией услуг, и фактически каждый владелец *iPhone* с ним сталкивается.

Однако для программной реализации динамического назначения приоритетов в моделях общих правил нет. Поэтому в модель необходимо «вживить» контроллер – диспетчерскую агентную программу [2], которая перераспределяет требования в очереди (рис. 1-*б*). Причем, кроме временны́х характеристик «справедливости» существуют и другие, связанные с доходами, убытками, штрафными потерями, изменениями рейтинговых и других показателей. Учесть всё это с помощью четких алгоритмов практически невозможно. Но в 1974 г. произошло одно из ярких подтверждений успехов искусственного интеллекта, когда Э. Мамдани в Лондонском Колледже Королевы Мэри применил новое инструментальное средство – нечеткий контроллер, созданный на основе алгоритма нечеткого (*fuzzy*) логического вывода для управления парогенератором в лабораторных условиях, с применением теории нечетких множеств и лингвистических переменных [14].

Получение заключений в системах нечеткого вывода базируется на разделении процесса вывода решения или заключения на ряд последовательных этапов [6], реализуемых на основе нечетких множеств (*fuzzysets*) и нечеткой логики (*fuzzylogic*). Эта последовательность этапов практически стандартная:

а) формирование базы знаний;

б) фаззификация – переход от четких (*crisp*) к нечетким (*fuzzy*) свойствам;

в) поиск правил(а) в базе знаний, которая состоит из базы правил и базы данных функций принадлежности;

г) свертка простых высказываний в «условной» части правил и оценка ее истинности;

д) формирование заключений;

е) дефаззификация – обратный переход от нечетких к четким свойствам.

Ниже не будем углубляться в теорию нечетких множеств, которая работает с квалиметрическими функциями принадлежности, и не будем рассматривать алгоритм нечеткого логического вывода Мамдани. Принцип действия контроллера в системе без обратной связи показан на рис. 2-*а*. В настоящее время контроллер «типа» Мамдани – это хороший инструмент, но только в системах управления с отрицательной обратной связью [8], которая нивелирует его недостатки. Упрощенная схема такого контроллера показана на рис. 2-*б*.

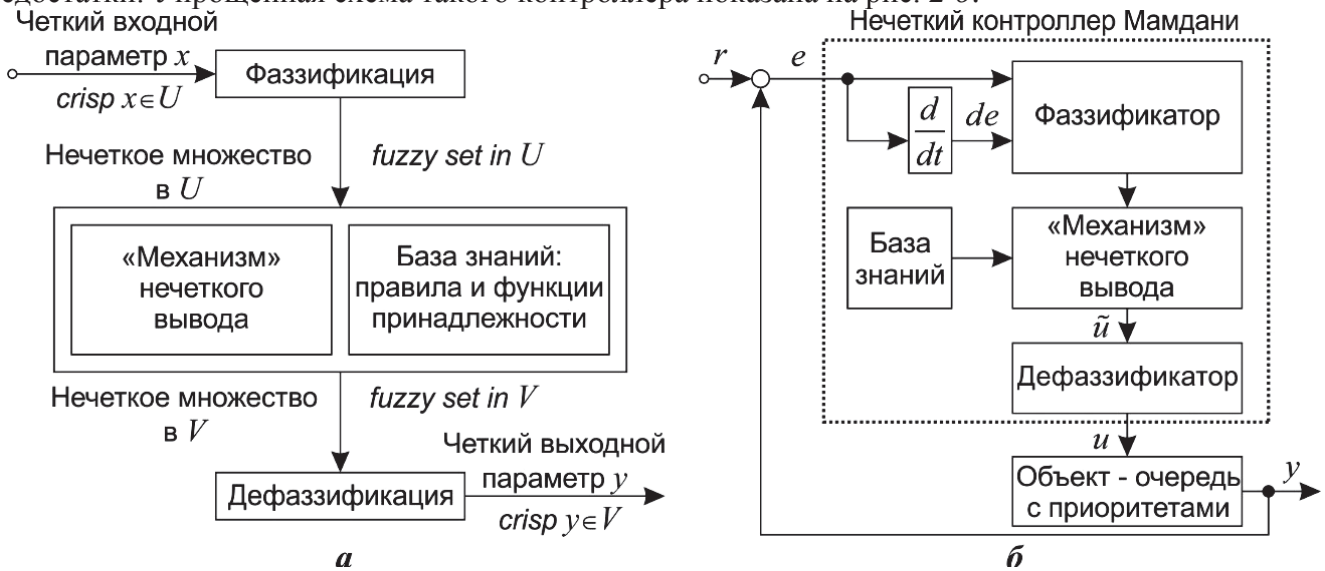

Рис. 2. Нечетки контроллер: *а* – принцип действия, *б* – схема контроллера

При создании поведенческих моделей или моделей-трансформеров полезно применять свойства теории акторных сетей (*ANT – actornetworktheory* [12]). Эта теория является «методологическим мостиком» для соединения метрологических и квалиметрических свойств и параметров в одной имитационной модели [3]. В нашем случае нужен программный контроллер, внедряемый в имитационную модель и работающий не в астрономическом, а в модельном виртуальном времени. Наличие отрицательной обратной связи имеется (см. рис. 1-*б*).

Работа контроллера происходит так. Допустим, что оперативно – при поступлении нового требования в очередь или при выборе очередного требования на обслуживание – в один из каналов узла serve, состав очереди изменяется. Соответственно и назначенный ранее порядок требований и обслуживания меняется.

Причем на входе системы управления с очередью выполняются воздействия r в соответствии с выбранными критериями, и в это же время на входе контроллера возникает ошибка e и вычисляется ее производная по времени de/dt.

Далее обе величины сначала подвергаются операции фаззификации (преобразования в нечеткие переменные). Затем полученные нечеткие переменные используются в блоке нечеткого логического вывода для получения управляющего воздействия на объект, которое после выполнения операции дефаззификации (обратного преобразования нечетких переменных в четкие) поступает на выход контроллера в виде управляющего воздействия u. Регулирование нечеткими переменными строится на основании «высказываний» критерия, сформулированных в виде нечетких правил типа *if–then* [6] и операций нечеткой логики. Совокупность нечетких правил и нечетких переменных используется для осуществления нечеткого логического вывода, результатом которого и является требуемое управляющее воздействие u на объект – очередь с приоритетами, где y – эффект управления [10].

Следует отметить, что квантование времени $\Delta t \to dt$ при дифференцировании в некоторых моделирующих системах технически может серьезно замедлить компьютерное выполнение модели, но только не в *ActorPilgrim*, т.к. две специальные возможности, заложенным при разработке «движка» этой системы, делают *ActorPilgrim* одной из самых быстродействующих систем:

- благодаря особенностям алгоритма управления временем приращения $\Delta t \to dt$ в контроллере виртуально накладываются на общую последовательность событий в дискретных компонентах модели, а при наличии непрерывных компонент типа дифференциальных уравнений эти приращения также виртуально накладываются на реальную последовательность шагов интегрирования, и в действительности контроллер практически не замедляет моделирование;
- ядро системы создавалось с активным использованием адресной арифметики, т.е. фактически с использованием машинных адресов.

Пример: модель взаимодействия служб крупного международного аэропорта

Типовым аэропортом выбран АП «Домодедово». Все исходные данные в модели получены из публикаций в СМИ и с сайтов. Дислокация аэропорта показана на рис. 3. Имеются две взлетно-посадочные полосы (*runway*), они не вполне одинаковы. Поэтому запланирован ввод третьей полосы, после чего первая полоса перейдет в резерв.

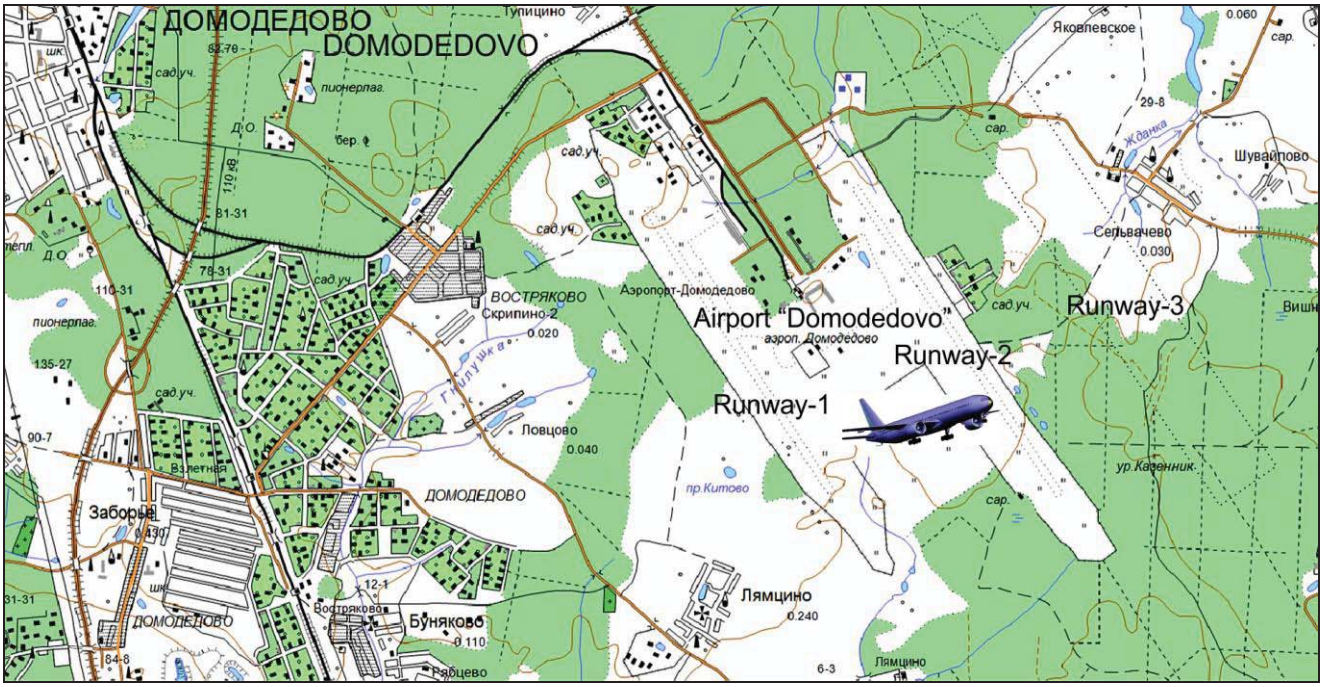

Рис. 3. Дислокация аэропорта «Домодедово»: скриншот из модели

Укрупненная схема имитационной модели показана на рис. 4. Выделены три узла подготовки принятия решений (ППР). Это: ППР посадки, ППР взлета, и ППР аэродрома. Два первых имеют отношение к управлению воздушным движением (УВД) и взаимодействуют через ресурсы: взлетно-посадочные полосы. При этом борт, идущий на посадку, имеет приоритет в отношении готового к взлету воздушного судна.

ППР аэродрома координирует действия наземных служб: заправку воздушных судов и их подготовку к полету службами *GroundHandling*. Процессы обслуживания пассажиров в зданиях АП и на летном поле, а также работа с багажом и грузами (погрузка-выгрузка), в модели не рассматриваются.

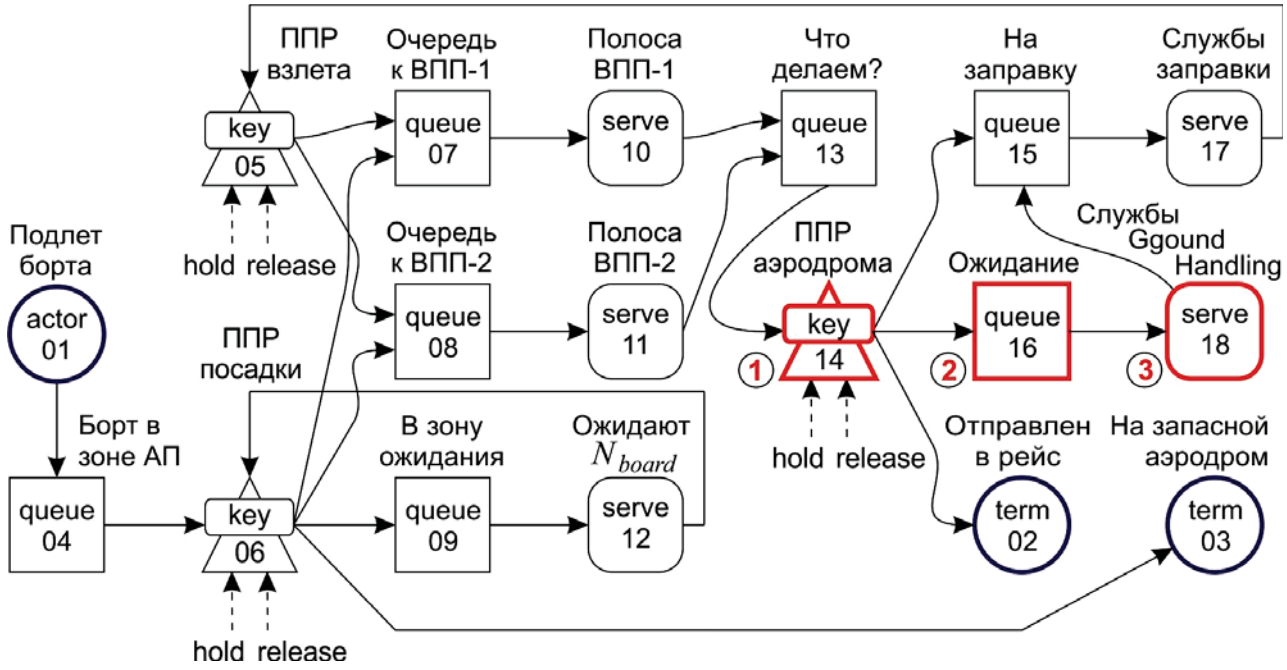

Рис. 4. Имитационная модель взаимодействия служб аэропорта

В качестве первого эксперимента незадолго до приближающегося ЧМ-2018 с помощью такой модели разработчики попытались ответить на два актуальных вопроса:

Насколько увеличит пропускную способность аэропорта ввод в эксплуатацию третьей взлетно-посадочной полосы?

В установившемся режиме работы АП период между взлетами-посадками составляет в среднем 3,5 мин в течение года. Можно ли уменьшить этот период, например, до 2,5 мин на протяжении длительного периода времени, например, года?

После подстановки реальных параметров процессов в АП «Домодедово» были получены ответы на оба вопроса:

- Реальная загрузки полос такова: первой – 6%, второй – 25%. Поэтому ввод третьей полосы практически не повлияет на повышение пропускной способности АП.
- Уменьшить период взлетов-посадок с 3,5 мин до 2,5 мин без других организационных мероприятий, т.е. простым привлечением дополнительных авиакомпаний невозможно. Службы *GroundHandling* могут одновременно готовить до 45 воздушных судов к вылету. Однако соответствующее увеличение воздушного трафика приводит эти службы «в ступор» (рис. 5): если раньше начало подготовка в среднем задерживалось на 0,5 мин (максимально – до 4 мин, несколько раз в год), то после увеличения трафика задержка возрастает до 145 мин (максимально – до 740 мин).

Таких организационных мероприятий может быть несколько: увеличение числа перронов для подготовки воздушных судов (переоборудование аэродрома и увеличение квалифицированного персонала *GroundHandling*– это дорого) или увеличение доли транзитных рейсов через АП (но это несильно увеличивает пассажиропоток). Однако нас по-прежнему интересует «справедливое» назначение приоритетов.

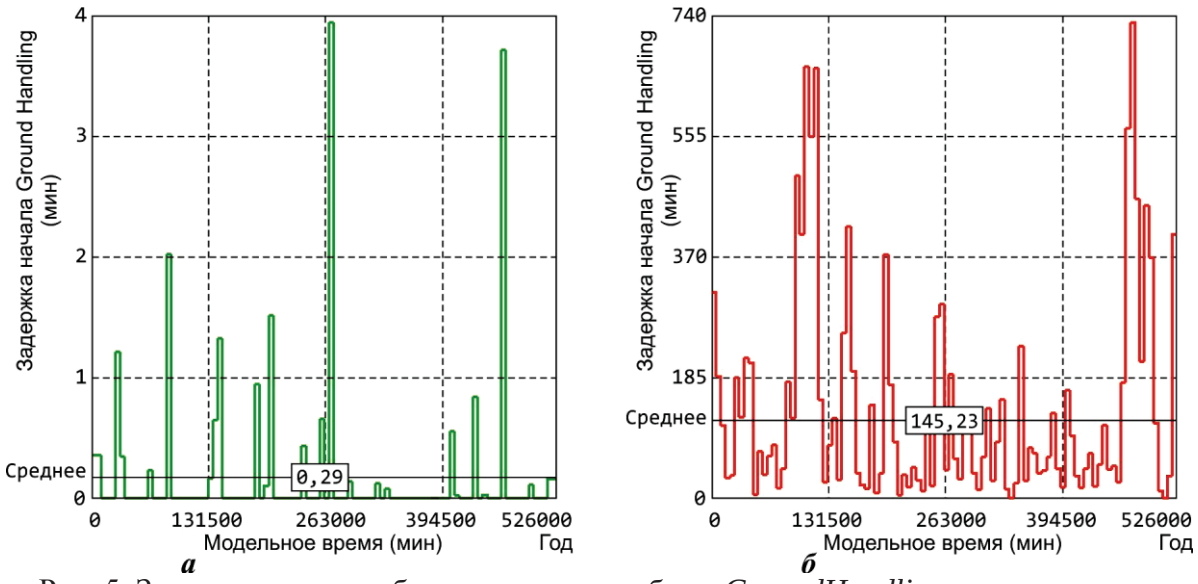

Рис. 5. Задержки начала обслуживания службами *GroundHandling* в течение года: *а* – интервал «взлет-посадка» в среднем 3,5 мин; *б* – тот же интервал в 2,5 мин

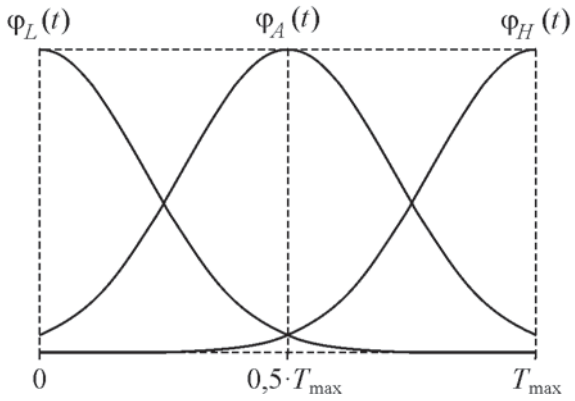

Рис. 6. Вид функции принадлежности

Нечеткий контроллер назначения приоритетов в имитационной модели

Алгоритмы нечеткого логического вывода и правила достаточно известны. Рассмотрим только набор лингвистических переменных и функции принадлежности, которые вводятся в контроллер. Воздушные суда в АП условно разобьем на группы.

Типичные лингвистические переменные: *CG* – стоимость обслуживания группы; *NA* – количество запросов или заявок от соответствующей группы; *RP* – вероятность получения запроса от группы; *ST* – время группового обслуживания в каналах массового обслуживания (в данном случае канал – это перрон); *TL* – время или штрафные потери из-за задержки в очереди.

Обычно используются такие значения лингвистических переменных: *L* – низкие; *A* – среднее; *H* – высокие, но возможны и другие наборы значений.

Функции принадлежности обычно подбираются экспертами. Однако при небольшом числе реализаций (до миллиона) довольно просто использовать, например, симметричные функции гауссова типа, если риски незначительны (рис. 6). Но, если модель создана для фильтрации генеральной совокупности в сотни миллионов реализаций при поиске весьма редких рисковых событий, то в таком случае более корректно использовать функции логнормального типа [1, 4].

Обычно для решения задач методами теории нечетких множеств (алгоритмы Мамдани, Сугэно, Цукамото, Ларсена) используются такие моделирующие системы:

- система *Matlab*, где предусмотрен пакет нечеткой логики *FuzzyLogicToolbox*;
- пакет *FuzzyTech*, который предназначен для проектирования нечетких систем.

Однако программа, реализующая контроллер, довольно компактная, если ее написать, например, на C++, и нет необходимости интегрировать в модель другую моделирующую систему. Поэтому для создания контроллера использовалась открытая библиотека исходных модулей для нечетких приложений на C++ [13]. Контроллер создан в виде типовой агентной программы [2], включаемой в модель. Внутреннее взаимодействие контроллера с ядром модели на уровне адресов показано на рис. 7.

На этом рисунке состав структур случайный и меняется во времени. Показаны дескрипторы kcb_1, kcb_2 и kcb_3 узлов, соответствующих узлам на рис. 1 и рис. 4. Стрелки обозначают адресные ссылки из одних структур данных в другие.

Заключение

Погоня за «справедливым» назначением приоритетов не приводит к снижению среднего времени ожидания в очереди для всего суммарного потока требований: это очевидно и следует из теоремы сохранения, известной из теории очередей. Однако принятие «справедливых» решений влечет за собой иные положительные эффекты, в основном – экономические. В связи с этим можно сделать следующие выводы.

Элементом научной новизны представленной работы является включение в состав моделирующей системы, предназначенной для получения метрологических результатов, квалиметрического компонента из области искусственного интеллекта, позволяющего снизить влияние факторов неопределенности в исходных данных.

С помощью расширения данной модели наблюдались рейтинговые эффекты: например, рейтинг Boeing–777 «удается приподнять» в модели без какого-либо ущерба для авиакомпаний, использующих воздушные суда типа SukhoiSJ–100. В целом получение дополнительных возможностей опосредованного управления рейтингами – это практически сильный эффект назначения «справедливых» приоритетов.

«Справедливое» назначение приоритетов позволяет в среднем понизить штрафные санкции и потери, т.к. цены на услуги Ground Handling и размеры штрафов в случае задержек работ сильно зависят от типов воздушных судов и рейсов.

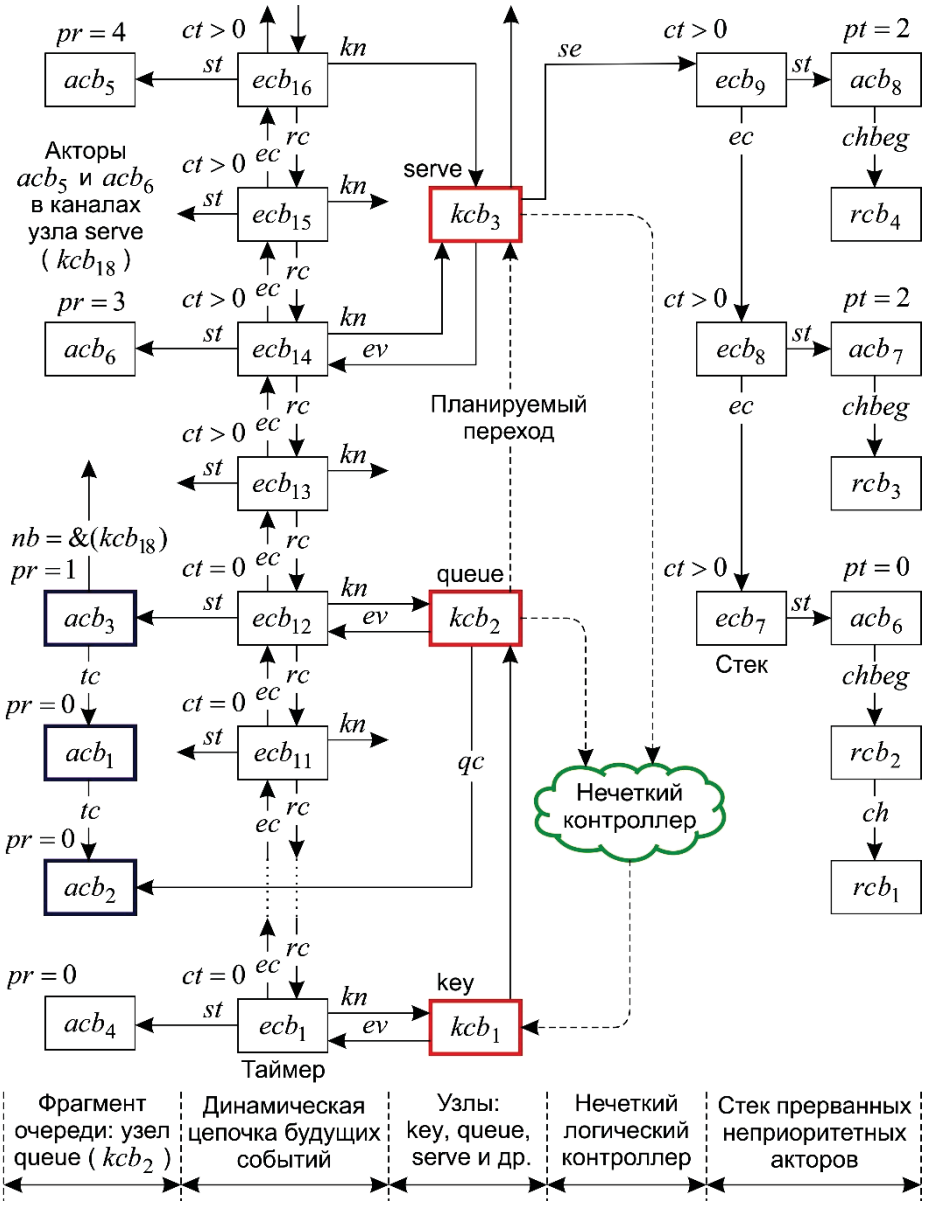

Рис. 7. Динамические структуры данных управления временем в *ActorPilgrim*

Литература

1. Булыгина О.В., Емельянов А.А., Емельянова Н.З. Имитационное моделирование в экономике и управлении / Под ред. А.А. Емельянова. М.: ИНФРА-М, 2019. – 592 с.

2. Булыгина О.В., Емельянов А.А., Емельянова Н.З. Назначение приоритетов в технологических хабах на основе имитационного моделирования и нечеткой логики // Прикладная информатика. 2017. Т.12. № 5. С. 71–92.

3. Булыгина О.В., Емельянов А.А., Росс Г.В. Гибридное кибермоделирование в экономике: теория акторных сетей, симуляция, НЕ-факторы и сверхнечеткая логика // Прикладная информатика. 2018. Т. 13. № 6 (78). С. 78–90.

4. Емельянов А.А., Булыгина О.В., Халин В.Г. Экономико-имитационное моделирование с элементами искусственного интеллекта. М.: НЕОЛИТ, 2018. – 160 с.

5. Саати Т.Л. Элементы теории массового обслуживания и ее приложения. М.: Либроком, 2010. – 512 с.

6. Чернов В.Г. Основы теории нечетких множеств. Владимир: ВлГУ им. А.Г. и Н.Г. Столетовых, 2010. – 96 с.

7. Чижов М. И., Скрипченко Ю. С., Гусев П. Ю. Имитационное моделирование производства деталей из полимерных композиционных материалов // Компьютерные исследования и моделирование. 2014 Т. 6. № 2. С. 245−252.

8. Aceves-Lopez A., Aguilar-Martin J. A simplified version of Mamdani's fuzzy controller: the natural logic controller // IEEE Transactions on Fuzzy Systems. 2006. Vol. 14. No. 1. P. 16–30.
9. Emelyanov A.A., Bulygina O.V., Emelyanova N.Z. Complex swarm-simulation modeling of innovative projects promotion into the regions // 2018 IV International Conference on Information Technologies in Engineering Education (Inforino). Moscow, Russia: 2018. P. 1–4. – DOI: 10.1109/INFORINO.2018.8581762 (Scopus).
10. Emelyanov A., Khalin V., Tukaev D., Morozov A. Decision-making by airport ground services by means of math-economic simulation and fuzzy logic // Journal of Engineering and Applied Sciences. 2018. Vol. 13. No. 16. P. 6748–6753. – DOI: 10.3923/JEASCI.2018.6748.6753 (Scopus).
11. Jackson J.R. Queues with dynamic priority discipline // Management Science. 1961. Vol. 8. No. 1. P. 18–34.
12. Latour B. Reassembling the social: an introduction to actor network theory. Oxford: Oxford University Press Inc., 2005. – 302 p.
13. Madarich S. C++ Fuzzy Logic API + Simple DSL. Code Project. 2012. No. 1. Maribor: Slovenia University FERI. URL: http://www.codeproject.com/Articles/ 316668 /Cplusplus-Fuzzy-Logic-API-plus-Simple-DSL.
14. Mamdani E.H. Application of fuzzy algorithms for control of simple dynamic plant // Proceedings of the Institution of Electrical Engineers. 1974. Vol. 121. No. 12. P. 1585–1588.