Моделирование и анализ информационных систем. Т. 25, № 6 (2018), с. 711–725
Modeling and Analysis of Information Systems. Vol. 25, No 6 (2018), pp. 711–725

Модели процессов
Process Modeling

DOI: 10.18255/1818-1015-2018-6-711-725

УДК 004.023

Применение генетического алгоритма для нахождения редакционного расстояния между моделями процессов

Каленкова А. А., Колесников Д. А.

Поступила в редакцию 1 сентября 2018
После доработки 10 ноября 2018
Принята к публикации 20 ноября 2018

Аннотация. Поиск редакционного расстояния между графовыми моделями (определение схожести графовых моделей) является важной задачей в различных областях компьютерных наук, таких как анализ изображений, машинное обучение, химическая информатика. В последнее время, в связи с развитием методов извлечения и анализа процессов, появилась необходимость в адаптации существующих методов сравнения графовых моделей для анализа моделей процессов (аннотированных графов), извлекаемых из логов событий информационных систем. Методы нахождения минимального редакционного расстояния между графами могут быть использованы для обнаружения шаблонов (подпроцессов), а также для сравнения извлекаемых моделей процессов. Как было показано экспериментально и теоретически обосновано, точные методы нахождения минимального редакционного расстояния между извлекаемыми моделями процессов (и графами в общем случае) имеют большую временную сложность и могут быть применены лишь к небольшим моделям процессов. В этой статье мы оцениваем точность и временные характеристики генетического алгоритма, применяемого для нахождения расстояний между моделями процессов, извлекаемых из логов событий. В частности мы находим расстояния между BPMN (Business Process Model and Notation) моделями, извлекаемыми из логов событий с помощью различных алгоритмов синтеза. В этой работе показано, что представленный генетический алгоритм позволяет в значительной степени уменьшить время вычислений, при этом показывая результаты, близкие к оптимальным (минимальным редакционным расстояниям).

Ключевые слова: минимальное редакционное расстояние между графами, извлечение и анализ процессов, BPMN (Business Process Model and Notation), генетический алгоритм

Для цитирования: Каленкова А. А., Колесников Д. А., "Применение генетического алгоритма для нахождения редакционного расстояния между моделями процессов", *Моделирование и анализ информационных систем*, **25**:6 (2018), 711–725.

Об авторах: Каленкова Анна Алексеевна, orcid.org/0000-0002-5088-7602, ст. науч. сотр.,
Национальный исследовательский университет «Высшая школа экономики», лаборатория ПОИС
ул. Мясницкая, 20, г. Москва, 101000 Россия, e-mail: akalenkova@hse.ru

Колесников Данил Александрович, orcid.org/0000-0002-9010-8415, студент
Национальный исследовательский университет «Высшая школа экономики», факультет компьютерных наук
ул. Мясницкая, 20, г. Москва, 101000 Россия, e-mail: dakolesnikov@edu.hse.ru

Благодарности:
Исследование выполнено при поддержке Гранта Президента РФ для молодых российских ученых — кандидатов наук МК-4188.2018.9.

Введение

Информационные системы, автоматизирующие рабочие процессы в различных областях, таких как медицина, образование, предоставление государственных услуг, финансы, сохраняют историю своей работы в виде логов событий. Технологии извлечения и анализа процессов (известные также как process mining) позволяют строить модели процессов, обобщающие поведение систем, представленное в логах событий [1]. Несмотря на то что извлекаемые модели, как правило, достаточно наглядно визуализируют процесс, их структура также может быть более детально проанализирована. Так, например, шаблоны моделирования процессов, такие как «последовательность», «выбор», «параллельное исполнение», «цикл» и другие более сложные структуры, могут быть обнаружены и выделены автоматически. Кроме того, извлекаемые модели процессов могут быть сопоставлены с моделями, определяющими эталонное (ожидаемое) поведение, и, таким образом, отклонения в поведении (использовании) системы могут быть явно определены. В работах [4,6] было предложено использовать редакционное расстояние между графами для сопоставления моделей процессов, извлекаемых из логов событий. Редакционное расстояние показывает степень схожести/различия двух графов. Более точно оно определяется как количество элементарных шагов (добавление/удаление вершины, добавление/удаление дуги), которые необходимо выполнить, чтобы преобразовать один граф в другой. Рассматривая модели процессов как ориентированные графы, методы нахождения минимального редакционного расстояния можно адаптировать для их сопоставления. В этом случае названия вершин и дуг, а также их типы должны быть дополнительно учтены. Точный алгоритм поиска A* [7] был реализован в инструменте [6] для сопоставления BPMN (Business Process Model and Notation) [8] моделей процессов. Этот инструмент был использован для сопоставления BPMN моделей, извлеченных из логов событий сервисов предоставления государственных услуг [4]. Несмотря на то что для сокращения времени нахождения минимального редакционного расстояния были использованы некоторые процессно-ориентированные эвристики [4], точный алгоритм нахождения минимального расстояния был применим лишь к небольшим моделям процессов. Это может быть объяснено в том числе тем обстоятельством, что задача нахождения минимального редакционного расстояния между графами является NP-полной [5].

В этой работе мы исследуем возможность нахождения минимального редакционного расстояния между моделями процессов с помощью генетического алгоритма. Методы нахождения минимального редакционного расстояния между моделями процессов, основанные на использовании генетических алгоритмов, были описаны в [9]. Генетические алгоритмы были также использованы в комбинации с другими алгоритмами нахождения минимального редакционного расстояния в [10]. В отличие от работы [9] мы предлагаем алгоритм, который также учитывает специфику графовых моделей процессов (названия и типы вершин). Для оценки времени и точности предложенного алгоритма мы сравниваем модели процессов, извлеченные из логов событий с помощью альфа алгоритма [2] и индукционного алгоритма [3], используя методы приведения извлекаемых моделей к BPMN формату [11].

Статья имеет следующую структуру. В разделе 1 представлены основные понятия и определения, использующиеся в тексте статьи. Раздел 2 содержит описание

точного алгоритма сопоставления BPMN моделей процессов. В разделе 3 представлен новый метод сопоставления BPMN моделей, основанный на принципах генетического алгоритма. Раздел 4 содержит результаты экспериментов (точность и время работы предложенного алгоритма).

1. Основные определения

В этом разделе вводится понятие плоских BPMN (Business Process Model and Notation) моделей, также даются определения графов бизнес-процессов и расстояний между ними. Эти понятия используются далее в статье.

Несмотря на большое количество классов BPMN моделей, предложенных Object Management Group (OMG) в формальной спецификации [8], мы рассматриваем только плоские модели процессов, формализующие поток управления. Эти модели могут быть получены из логов событий в два шага: (1) низкоуровневая модель процесса (сеть Петри, дерево процесса, каузальная сеть) синтезируется из лога событий; (2) эта модель конвертируется в высокоуровневую модель с помощью алгоритмов, предложенных в работе [11].

Плоские BPMN модели, формализующие поток управления и полученные с помощью алгоритмов преобразования из сетей Петри, деревьев процессов и каузальных сетей, представлены следующим базовым набором элементов: *действия*, *исключающие* и *параллельные маршрутизаторы*, *начальные* и *конечные события*, *потоки управления* (Рис. 1).

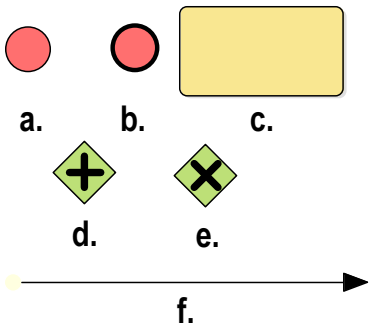

Рис. 1. Элементы плоских BPMN моделей
Fig. 1. Elements of flat BPMN models

Начальные (Рис. 1 a) и *конечные события* (Рис. 1 b) обозначают начало и завершение процесса соответственно. *Действия* (Рис. 1 c) моделируют атомарные шаги процесса. *Параллельные* (Рис. 1 d) и *исключающие маршрутизаторы* (Рис. 1 e) используются для моделирования ветвей процесса, исполняющихся параллельно или взаимоисключающих друг друга.

Пример BPMN модели, описывающей простую процедуру бронирования, приведен на Рис. 2. Сначала пользователь регистрируется, бронирует самолет и гостиницу (эти действия выполняются параллельно и каждое из них может быть пропущено) и оплачивает заказ.

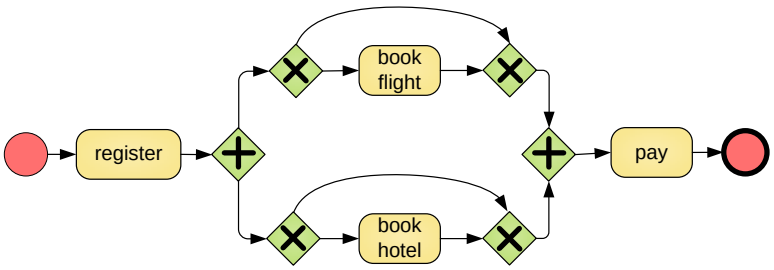

Рис. 2. BPMN модель, описывающая процедуру бронирования
Fig. 2. A BPMN model of a booking procedure

Плоские BPMN модели могут быть рассмотрены как ориентированные графы с типами и именами вершин. Мы будем называть их *графы бизнес-процессов*. Дадим их формальное определение.

Графом бизнес-процесса называется кортеж $G = (N, E, t, l)$, в котором N – это множество вершин, $E \subseteq (N \times N)$ – множество направленных дуг, $t : N \to T$, где $T = \{start\ event, end\ event, task, exclusive\ gateway, parallel\ gateway\}$ – функция, определяющая типы вершин, $l : N \to L$ – функция, задающая имена вершин, L – множество имен.

Рассмотрим два графа бизнес-процессов: $G_k = (N_k, E_k, t_k, l_k)$, $k = 1, 2$. *Расстояние* между ними вычисляется путем построения *редакционного отношения* $R \subseteq (N_1 \cup E_1 \cup \{\epsilon, \delta\}) \times (N_2 \cup E_2 \cup \{\epsilon, \delta\})$, где $\epsilon, \delta \notin N_1 \cup N_2 \cup E_1 \cup E_2$, такого, что выполняются следующие условия:

1. для каждого элемента $i_1 \in N_1 \cup E_1$ ($i_2 \in N_2 \cup E_2$) существует один и только один элемент $i_2 \in N_2 \cup E_2 \cup \{\epsilon, \delta\}$ ($i_1 \in N_1 \cup E_1 \cup \{\epsilon, \delta\}$), такой что $(i_1, i_2) \in R$;

2. если $i_1 \in \{\epsilon, \delta\}$ ($i_2 \in \{\epsilon, \delta\}$), тогда $i_2 \notin \{\epsilon, \delta\}$ ($i_1 \notin \{\epsilon, \delta\}$);

3. если $i_1 \in N_1$ и $(i_1, i_2) \in R$, тогда $i_2 \in N_2 \cup \{\epsilon, \delta\}$, и если при этом $i_2 \in N_2$, тогда типы вершин совпадают, т.е. $t_1(i_1) = t_2(i_2)$;

4. если $i_1 \in E_1$ и $(i_1, i_2) \in R$, тогда $i_2 \in E_2 \cup \{\epsilon, \delta\}$;

5. для двух любых дуг $(i_1, i_1') \in E_1$, $(i_2, i_2') \in E_2$ условие $((i_1, i_1'), (i_2, i_2')) \in R$ выполняется тогда и только тогда, когда $(i_1, i_2) \in R$ и $(i_1', i_2') \in R$.

Если для некоторого элемента i выполняется $(i, \epsilon) \in R$ ($(\epsilon, i) \in R$), то мы говорим, что элемент i *удаляется* (*добавляется*). Если $(i, \delta) \in R$ ($(\delta, i) \in R$), то мы говорим, что для i не существует соответствующего элемента (это будет использовано для представления промежуточных результатов сравнения).

Сравним два графа бизнес-процессов $G_k = (N_k, E_k, t_k, l_k)$, $k = 1, 2$, построив редакционное отношение R. Для каждого $r = (i_1, i_2) \in R$ стоимость будет определена следующим образом:

$$cost(r) = \begin{cases} lev(l_1(i_1), l_2(i_2)) \cdot c_{lev}, & i_1 \in N_1, i_2 \in N_2, \\ 0, & i_1 \in E_1, i_2 \in E_2, \\ c_{delete}, & i_2 = \epsilon, \\ c_{insert}, & i_1 = \epsilon, \\ 0, & i_1 = \delta \vee i_2 = \delta. \end{cases}$$

Функция *lev* будет определять расстояние Левенштейна [12] между двумя строками, c_{lev} – специальный коэффициент; c_{delete} и c_{insert} обозначают стоимости операций удаления и добавления соответственно.

Общая стоимость редакционного отношения R определяется как сумма стоимостей всех пар, принадлежащих этому отношению: $cost(R) = \sum_{r \in R} cost(r)$.

Минимальное редакционное отношение между двумя графами бизнес-процессов – это редакционное отношение между этими графами, имеющее минимальную стоимость, такое, что оно не содержит пар с элементом δ (соответствия для всех элементов определены). Стоимость этого отношения называется *минимальным расстоянием редактирования*.

2. Точный алгоритм нахождения минимального редакционного расстояния

В этом разделе приведено описание точного алгоритма нахождения минимального редакционного расстояния между графами бизнес-процессов (Алгоритм 1).

На каждом шаге алгоритм берет из очереди редакционное отношение с минимальной стоимостью, затем из этого отношения создаются все возможные новые отношения, в которых некоторой вершине, пока не имеющей соответствия (соответствующей δ), подбираются в соответствие элементы другой модели того же типа или ϵ. Инцидентные дуги этой вершины также обрабатываются. После этого все новые редакционные отношения добавляются в очередь. Алгоритм останавливает свою работу, когда минимальное по стоимости редакционное отношение не имеет пар, содержащих δ, то есть для всех элементов моделей определены соответствия. Стоимость этого редакционного отношения является минимальным редакционным расстоянием между графами бизнес-процессов, и она возвращается алгоритмом в качестве результата.

На Рис. 3 приведен пример сопоставления графов бизнес-процессов, моделирующих процедуру бронирования. Граф бизнес-процесса, представленный на Рис. 3 a, моделирует процедуру бронирования, представленную ранее (Рис. 2). Модель, представленная на Рис. 3 b, предполагает возможность сброса. Для того чтобы преобразовать первый граф бизнес-процесса (Рис. 3 a) во второй (Рис. 3 b), необходимо удалить две дуги и добавить структуру, соответствующую сбросу. Сама процедура бронирования является единой для обеих графовых моделей.

Для того чтобы сделать поиск минимального редакционного расстояния более быстрым, может быть использована так называемая *эвристическая функция*.

Data: $G_1 = (N_1, E_1, t_1, l_1)$ и $G_2 = (N_2, E_2, t_2, l_2)$ – графы бизнес-процессов;
Result: минимальное редакционное расстояние между G_1 и G_2;
\\R_{init} – *начальное редакционное отношение*;
$R_{init} \leftarrow \{\}$;
for $i_1 \in N_1 \cup E_1$ **do**
  $R_{init} \leftarrow R_{init} \cup \{(i_1, \delta)\}$;
end
for $i_2 \in N_2 \cup E_2$ **do**
  $R_{init} \leftarrow R_{init} \cup \{(\delta, i_2)\}$;
end
\\Q - *очередь редакционных отношений, отсортированных по стоимости*;
$Q \leftarrow \langle R_{init} \rangle$;
while *true* **do**
  взять редакционное отношение с наименьшей стоимостью;
  $R_{min} \leftarrow takeMinCostRelation(Q)$;
  if (R_{min} *содержит пары с* δ) **then**
    $i \leftarrow takeNodeRelatedtoDelta(R_{min})$;
    $Q.remove(R_{min})$;
    добавить все возможные пары для элементы i *в отношении* R_{min};
    $Q.add(expand(R_{min}, i))$;
  else
    return $cost(R_{min})$;
  end
end

Алгоритм 1: Нахождение минимального редакционного расстояния между графами бизнес-процессов.

Она определяется на множестве редакционных отношений и задается следующим образом:

$$H(R) = \sum\nolimits_{t \in T} \begin{cases} |I_1^t| - |I_2^t| \cdot c_{delete}, & |I_1^t| \geq |I_2^t|, \\ |I_2^t| - |I_1^t| \cdot c_{insert}, & |I_2^t| > |I_1^t|. \end{cases}$$

$I_1^t \subseteq N_1$ и $I_2^t \subseteq N_2$ обозначают множества вершин моделей типа t, для которых еще не были определены соответствия.

Тогда общая стоимость определяется как: $cost(R) = \sum_{r \in R} cost(r) + H(R)$. Действительно, используя эвристическую функцию, можно "предугадать" количество вершин, для которых не будет найдено соответствующих вершин в другом графе, поэтому они гарантированно будут удалены или добавлены. Этот подход, основанный на использовании эвристической функции, также известен как алгоритм поиска A* [7].

Другая эвристика, которая может быть использована при реализации алгоритма, заключается в том, что мы сначала можем рассматривать те вершины, которые соединены дугами с вершинами, для которых уже определено соответствие. Эта эвристика позволяет быстрее находить решение при сопоставлении схожих моделей процессов.

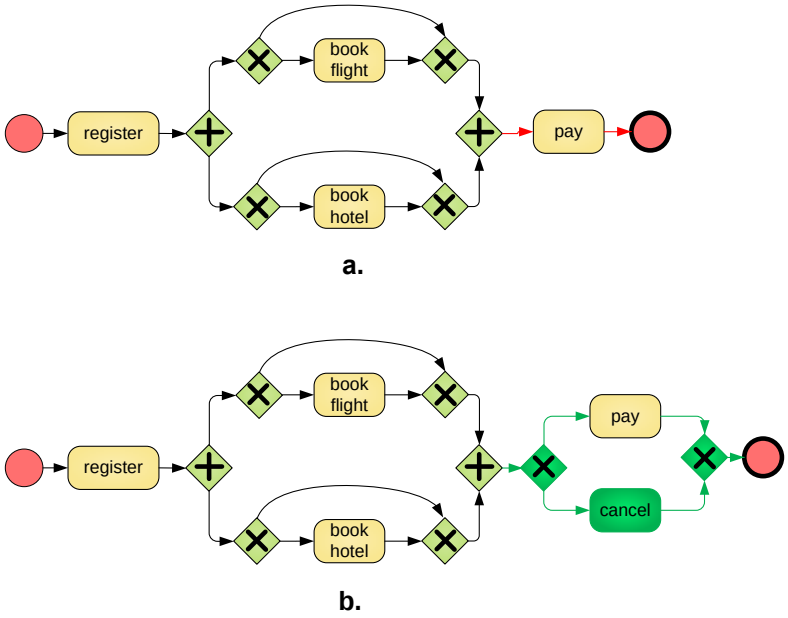

Рис. 3. Результат сопоставления графов бизнес-процессов, моделирующих процедуру бронирования
Fig. 3. A result of the comparison of business process graphs, modeling the booking procedure

Предложенные эвристики были реализованы [6] и протестированы при сопоставлении графов бизнес-процессов, синтезированных по реальным логам событий информационных систем [4]. Несмотря на то что они позволяют сделать процесс сравнения графовых моделей быстрее, сложность задачи, связанная с ее NP-полнотой, остается. Это обусловливает необходимость разработки неточных, но быстрых методов, которые позволят находить минимальное редакционное расстояние между моделями процессов произвольного размера.

3. Применение генетического алгоритма для нахождения минимального редакционного расстояния

В этом разделе будет представлено описание генетического алгоритма [13], позволяющего находить минимальное редакционное расстояние между графами бизнес-процессов.

3.1. Базовый алгоритм и структура гена

Генетический алгоритм основан на идее естественного отбора в природе. Имеется популяция – множество особей, каждая особь однозначно определяется своим геном. Пары особей дают потомство, тем самым порождают новую популяцию. Гены потомков являются продуктом перемешивания генов родителей. Из полученного множества особей выживают только сильнейшие, после чего процесс размножения

продолжается. Таким образом в популяции закрепляются те генетические признаки, которые способствуют выживанию особи.

Предположим, что нам необходимо найти минимальное редакционное расстояние между графами бизнес-процессов $G_1 = (N_1, E_1, t_1, l_1)$ и $G_2 = (N_2, E_2, t_2, l_2)$. Каждое решение (ген) будет определяться некоторым редакционным отношением R между этими графами бизнес-процессов. При этом мы будем рассматривать только те редакционные отношения, в которых для каждого элемента есть соответствие, то есть эти редакционные отношения не содержат пар с δ. Структура и стоимость редакционного отношения однозначно определяются соответствиями элементов первой модели, поэтому в качестве гена возьмем проекцию некоторого редакционного отношения R на первый граф бизнес-процесса. Она будет определена как: $\bar{R} = \{(i_1, i_2) | (i_1, i_2) \in R, i_1 \in N_1\}$. Далее мы будем называть эту проекцию также редакционным отношением.

Основная структура генетического алгоритма (Алгоритм 2) приведена ниже. Сначала генерируется начальная популяция особей, представленных редакционными отношениями между графами бизнес-процессов, затем выбираются особи из текущей популяции, они скрещиваются, потомство добавляется в новую популяцию. Далее происходит отбор, и в популяции остаются только наиболее сильные особи (отношения с наименьшей стоимостью). В ходе отбора используется специальный коэффициент $selectionFactor$, который определяет, какую часть от всей популяции особей необходимо оставить. Алгоритм продолжает работу до тех пор, пока в популяции не останется одна особь.

Data: $G_1 = (N_1, E_1, t_1, l_1)$ и $G_2 = (N_2, E_2, t_2, l_2)$ – графы бизнес-процессов;
Result: редакционное отношение между G_1 и G_2;
population $\leftarrow$ *generatrePopulation*(G_1, G_2);
while *population.size*()! = 1 **do**
  while !*population.empty*() **do**
    parent1 $\leftarrow$ population.get(); parent2 $\leftarrow$ population.get();
    children $\leftarrow$ crossingover(parent1, parent2);
    newpopulation.add(children);
  end
  newpopulation.sort(); *сортировка по стоимости;*
  newpopulation.resize(newpopulation.size()$*$ selectionFactor); *отбор;*
  population $\leftarrow$ newpopulation;
end
return population.get();

Алгоритм 2: Нахождение редакционного отношения между графами бизнес-процессов с помощью генетического алгоритма

3.2. Генерация начальной популяции

Приведем алгоритм генерации начальной популяции редакционных отношений между графами бизнес-процессов (Алгоритм 3).

Data: $G_1 = (N_1, E_1, t_1, l_1)$ и $G_2 = (N_2, E_2, t_2, l_2)$ – графы бизнес-процессов;
Result: популяция редакционных отношений между G_1 и G_2;
$\bar{R}_0 \leftarrow \{\}$;
for $i_i \in N_1$ **do**
  $\bar{R}_0 \leftarrow \bar{R}_0 \cup \{(i_1, \epsilon)\}$;
end
population.add($\bar{R}_0$);
for $i_1 \in N_1$ **do**
  for $i_2 \in N_2$ **do**
    if $t_1(i_1) = t_2(i_2)$ **then**
      $\bar{R}_{i_1,i_2} \leftarrow \bar{R}_0 \backslash \{(i_1, \epsilon)\} \cup \{(i_1, i_2)\}$;
      population.add($\bar{R}_{i_1,i_2}$);
    else
    end
  end
end
return population;

Алгоритм 3: Генерация начальной популяции

Сначала создается редакционное отношение $\bar{R}_0$, которое подразумевает удаление всех элементов первой модели и как следствие добавление всех элементов второй модели. Это редакционное отношение добавляется в начальную популяцию. Далее, для каждой вершины первой модели i_1 и для каждой вершины второй модели i_2, если их типы совпадают, создается новая особь $\bar{R}_{i_1,i_2}$ такая, что она отличается от $\bar{R}_0$ лишь тем, что в ней между элементами i_1 и i_2 устанавливается соответствие. Это редакционное отношение также добавляется в начальную популяцию. Для всех особей начальной популяции вычисляется их стоимость и выполняется сортировка начальной популяции.

3.3. Генерация потомков

Кроссинговер – процесс обмена генетической информацией предков, в результате которого получаются гены потомков (Рис. 4).

Из пары генов предков создаются два оппозитных гена потомков. Происходит одновременный перебор редакционных отношений по первому и второму предку. Фактически перебираются вершины первого графа бизнес-процессов $i_1 \in N_1$. Каждая пара $(i_1, i_2) \in \bar{R}_1$ первого предка и каждая соответствующая пара $(i_1, i'_2) \in \bar{R}_2$ второго предка случайным образом распределяются между двумя потомками. При этом, если в результате кроссинговера некоторому потомку принадлежат две различные пары (i_1, i_2) и (i'_1, i_2), где $i_2 \neq \epsilon$ и $i'_2 \neq \epsilon$ (одной вершине второго графа бизнес-процесса соответствуют две различные вершины первого графа), то также случайным образом выбирается одна из них и преобразуется в (i_1, ϵ) (или (i'_1, ϵ) соответственно). После этого вычисляются стоимости новых полученных редакционных отношений.

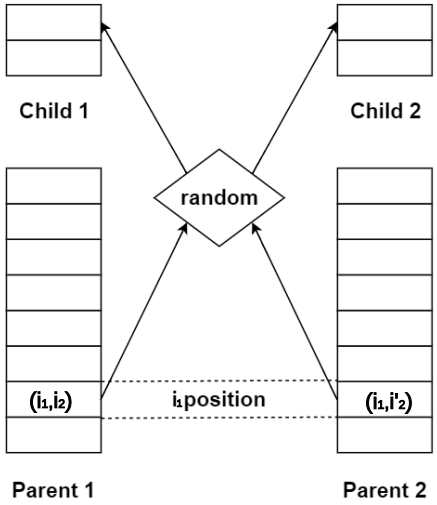

Рис. 4. Кроссинговер
Fig. 4. Crossover

3.4. Параметры генетического алгоритма

Генетический алгоритм имеет набор внутренних параметров, которые отвечают за качество его работы. Далее в ходе экспериментов нами будут рассмотрены следующие параметры: (1) коэффициент отбора популяции ($selectionFactor$), определяющий степень сокращения популяции после каждой итерации размножения; (2) метод выбора из отсортированного массива популяции пар предков для скрещивания. Рассмотрим три основных метода формирования пар из множества предков:

1. Алгоритм первый-второй (FS). Пары образуются последовательно в порядке возрастания стоимости (Рис. 5).

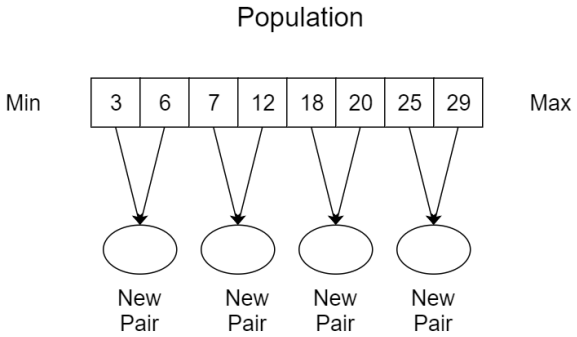

Рис. 5. Алгоритм первый-второй образования пар предков
Fig. 5. An algorithm of first-second parent matching strategy

2. Алгоритм первый-середина (FM). В пару предку, находящемуся в i-й позиции ($i \in [1, \lfloor size/2 \rfloor]$) отсортированного по стоимости массива, ставится предок с номером позиции $i + \lfloor size/2 \rfloor]$, где size – размер массива (Рис. 6).

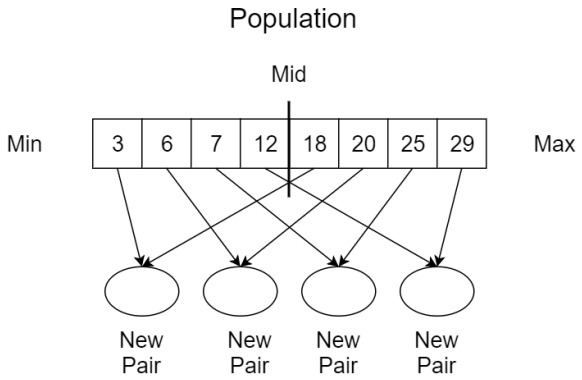

Рис. 6. Алгоритм первый-середина образования пар предков
Fig. 6. An algorithm of first-middle parent matching strategy

3. Алгоритм первый-последний (FL). Предку, находящемуся в i-й позиции ($i \in [1, size - 1]$) отсортированного массива, ставится в пару предок с номером позиции size - i, size – размер массива (Рис. 7).

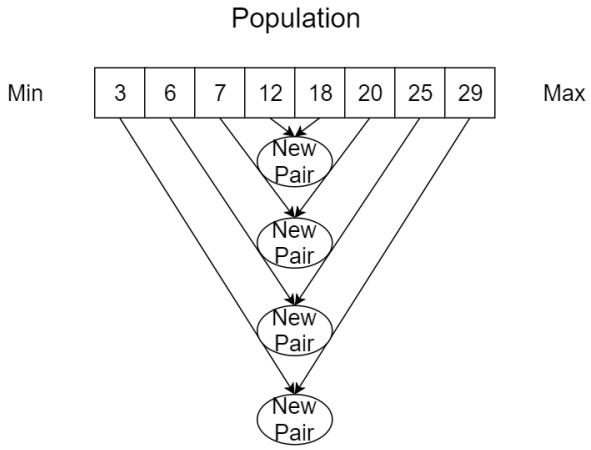

Рис. 7. Алгоритм первый-последний образования пар предков
Fig. 7. An algorithm of first-last parent matching strategy

4. Результаты экспериментов

В этом разделе представлены результаты экспериментов по нахождению минимального редакционного расстояния между BPMN моделями, синтезированными разными алгоритмами [2,3] из одних и тех же логов событий[1]. Были установлены единичные стоимости удаления и добавления элементов, а также символов их названий, т.е. $c_{delete} = 1$, $c_{insert} = 1$ и $c_{lev} = 1$. Только стоимости добавления и удаления действий были определены как 10 для того, чтобы переименования могли конкурировать с удалением и добавлением действий.

[1] http://www.processmining.org.

Все эксперименты были проведены на компьютере Acer Aspire V3, обладающем следующими характеристиками: процессор Intel i5-5200U 2.20GZ, объем оперативной памяти – 8ГБ.

На Рис. 8 показаны зависимости точности решения от размера модели для каждого коэффициента отбора и метода образования пар. Алгоритм A* дает точное минимальное редакционное расстояние. Следует отметить, что нет необходимости сравнивать между собой методы образования пар для одного коэффициента по времени, так как все они имеют одинаковое время работы.

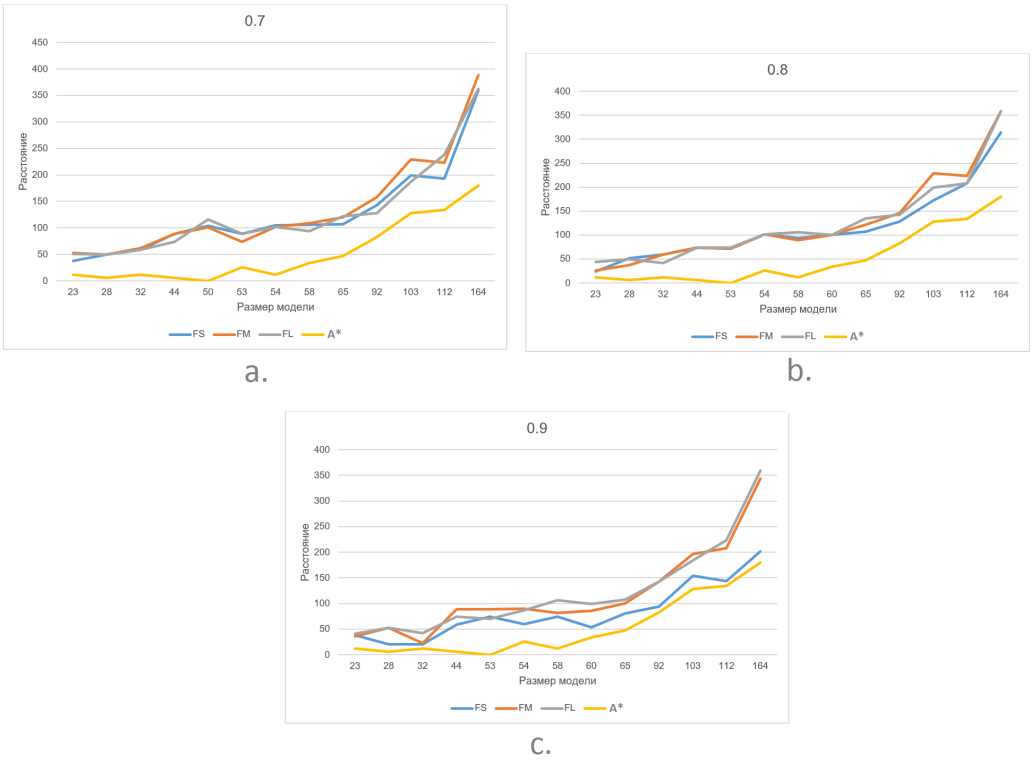

Рис. 8. Зависимость точности результата от характеристик алгоритма
Fig. 8. A dependence of a result accuracy from the characteristics of the algorithm

Как видно из Рис. 8, на небольших размерах моделей разные методы образования пар дают примерно одинаковый результат, а для больших размеров наиболее близкий к точному ответу результат получен для метода FS независимо от коэффициента отбора. Далее будем анализировать только алгоритм FS.

На Рис. 9 видна зависимость точности решения от значения коэффициента отбора: чем больше коэффициент, тем точнее ответ. Действительно, при «строгом» отборе вероятность того, что «полезные качества» успеют зафиксироваться в решении, намного меньше, чем при большем коэффициенте отбора.

Наконец, проанализируем время работы алгоритма для различных коэффициентов (Рис. 10).

Как видно из Рис. 10, коэффициент отбора влияет на время работы алгоритма. При большем значении коэффициента популяция сокращается медленнее, что

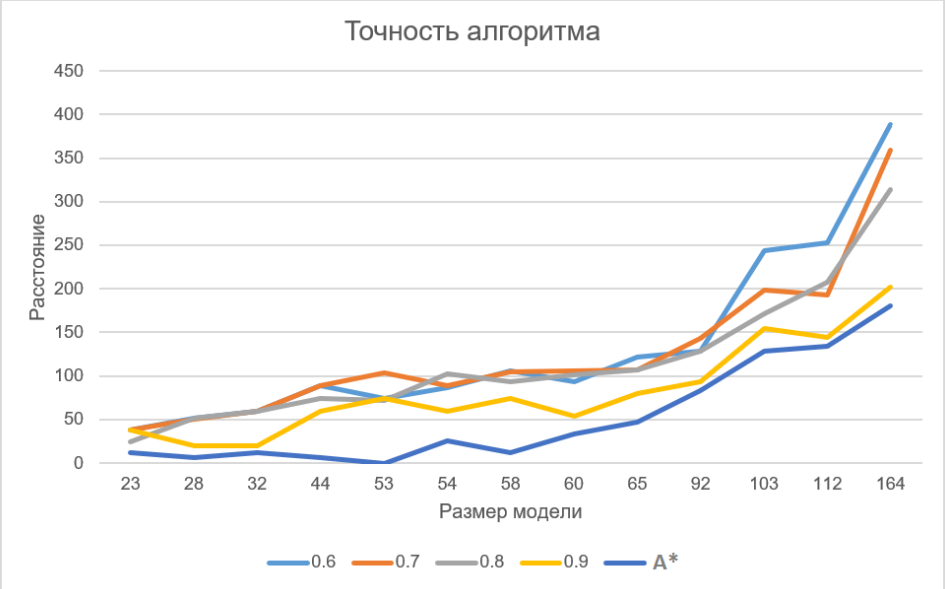

Рис. 9. Зависимость точности результата от коэффициента отбора
Fig. 9. A dependence of a result accuracy from the selection factor

Рис. 10. Зависимость времени работы алгоритма от коэффициента отбора
Fig. 10. A dependence of a computation time from the selection factor

сказывается на времени работы. Следует заметить, что алгоритм A* работает на порядок медленнее.

Итак, наиболее эффективным является метод образования пар первый-второй. Обнаружилась прямая зависимость времени работы и точности алгоритма от коэффициента отбора, его значение устанавливается в зависимости от необходимых пользователю критериев.

Заключение

В настоящее время большую популярность приобретают алгоритмы анализа моделей процессов по логам событий информационных систем (process mining). Существует множество коммерческих программ построения и анализа моделей процессов по логам событий информационных систем, которые широко используются в индустрии, в том числе в крупных компаниях. В связи с развитием теории process mining задача сопоставления модель-модель становится крайне актуальной и важной, так как не только помогает найти различия в ожидаемом и реальном поведении, но и наглядно визуализировать результат (это качество алгоритмов анализа процессов особенно ценится пользователями). В силу того что в общем случае задача сопоставления графовых моделей процессов является NP-полной и точное сравнение моделей занимает значительное время, эвристические методы сопоставления моделей процессов становятся особенно ценными. В этой работе мы адаптировали генетический алгоритм сравнения графов для задачи сопоставления моделей процессов, извлекаемых из логов событий. Этот алгоритм был реализован и протестирован на BPMN моделях, синтезированных по логам событий. Была проведена как оценка точности алгоритма в зависимости от параметров, так и его временных характеристик.

Список литературы / References

[1] Van der Aalst W. M. P., *Process Mining — Data Science in Action*, Second Edition, Springer, 2016.

[2] Van der Aalst W. M. P., Weijters T., Maruster L., “Workflow Mining: Discovering Process Models from Event Logs”, *IEEE Transactions on Knowledge and Data Engineering*, **16**:9 (2004), 1128–1142.

[3] Leemans S. J. J., Fahland D., van der Aalst W. M. P., “Discovering Block-Structured Process Models from Incomplete Event Logs”, *Application and Theory of Petri Nets and Concurrency*, Lecture Notes in Computer Science, **8489**, Springer, 2014, 91–110.

[4] Kalenkova A. A., Ageev A. A., Lomazova I. A., van der Aalst W. M. P., “E-Government Services: Comparing Real and Expected User Behavior”, *Business Process Management Workshops*, Springer International Publishing, 2018, 484–496.

[5] Garey M. R., Johnson D. S., *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1990, ISBN: 0716710455.

[6] Ivanov S. Y., Kalenkova A. A., van der Aalst W. M. P., “BPMNDiffViz: A Tool for BPMN Models Comparison”, Proceedings of the BPM Demo Session 2015 Co-located with the 13th International Conference on Business Process Management (BPM 2015) (Innsbruck, Austria, September 2, 2015), 2015, 35–39.

[7] Hart P. E, Nilsson N. J, Raphael B., “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”, *IEEE Transactions on Systems Science and Cybernetics*, **4**:2 (1968), 100–107.

[8] *Business Process Model and Notation (BPMN)*, Object Management Group, formal/2013-12-09, 2013.

[9] Cross A. D. J., Wilson R. C., Hancock E. R., “Inexact Graph Matching Using Genetic Search”, *Pattern Recognition*, **30**:6 (1997), 953 – 970.

[10] Riesen K., Fischer A., Bunke H., “Improving Approximate Graph Edit Distance Using Genetic Algorithms”, *Structural, Syntactic, and Statistical Pattern Recognition*, Springer Berlin Heidelberg, 2014, 63–72.

[11] Kalenkova A. A., van der Aalst W. M. P., Lomazova I. A., Rubin V. A., "Process Mining Using BPMN: Relating Event Logs and Process Models", *Software & Systems Modeling*, **16**:4 (2017), 1019–1048.

[12] Levenshtein V. I., "Binary Codes Capable of Correcting Deletions, Insertions and Reversals", *Soviet Physics Doklady*, **10** (1966), 707.

[13] Гладков Л.А., Курейчик В.В, Курейчик В.М., *Генетические алгоритмы*, Физматлит, М., 2010, 368 с.; [Gladkov L. A., Kureichik V. V., Kureichik V. M., *Genetic Algorithms*, Fizmatlit, Moscow, 2010, 368 pp., (in Russian).]

Kalenkova A. A., Kolesnikov D. A., "Application of Genetic Algorithms for Finding Edit Distance between Process Models", *Modeling and Analysis of Information Systems*, **25**:6 (2018), 711–725.

DOI: 10.18255/1818-1015-2018-6-711-725

Abstract. Finding graph-edit distance (graph similarity) is an important task in many computer science areas, such as image analysis, machine learning, chemicalinformatics. Recently, with the development of process mining techniques, it became important to adapt and apply existing graph analysis methods to examine process models (annotated graphs) discovered from event data. In particular, finding graph-edit distance techniques can be used to reveal patterns (subprocesses), compare discovered process models. As it was shown experimentally and theoretically justified, exact methods for finding graph-edit distances between discovered process models (and graphs in general) are computationally expensive and can be applied to small models only. In this paper, we present and assess accuracy and performance characteristics of an inexact genetic algorithm applied to find distances between process models discovered from event logs. In particular, we find distances between BPMN (Business Process Model and Notation) models discovered from event logs by using different process discovery algorithms. We show that the genetic algorithm allows us to dramatically reduce the time of comparison and produces results which are close to the optimal solutions (minimal graph edit distances calculated by the exact search algorithm).

Keywords: minimal graph edit distance, process mining, BPMN (Business Process Model and Notation), genetic algorithm

On the authors:
Anna A. Kalenkova, orcid.org/0000-0002-5088-7602, senior research fellow,
National Research University Higher School of Economics, Laboratory of Process-Aware Information Systems,
20 Myasnitskaya St., Moscow 101000, Russia, e-mail: akalenkova@hse.ru

Danil A. Kolesnikov, orcid.org/0000-0002-9010-8415, student
National Research University Higher School of Economics, Faculty of Computer Science
20 Myasnitskaya St., Moscow 101000, Russia, e-mail: dakolesnikov@edu.hse.ru

Acknowledgments:
This work was funded by the President Grant MK-4188.2018.9.