УДК 519.87

Е.В. Кривцун

Эволюционно-фрагментарный алгоритм поиска минимального множества аксиом

Построена фрагментарная структура задачи поиска минимального множества аксиом. Наличие такой структуры позволяет использовать стандартный эволюционный алгоритм на перестановках для поиска приближенных решений. Для тестирования построенного ЭВФ-алгоритма сгенерированы два набора данных с различными структурами.

Ключевые слова: минимальное множество аксиом, задача выбора аксиом, комбинаторная оптимизация, фрагментарная структура, эволюционный алгоритм.

Побудовано фрагментарну структуру задачі пошуку мінімальної множини аксіом. Наявність такої структури дозволяє використовувати стандартний еволюційний алгоритм на перестановках для пошуку наближених рішень. Для тестування побудованого ЕВФ-алгоритму згенеровано два набори даних з різними структурами.

Ключові слова: мінімальна множина аксіом, задача вибору аксіом, комбінаторна оптимізація, фрагментарна структура, еволюційний алгоритм.

Введение. Поиск минимального множества аксиом (*minimal axiom set*, *MinA*) относится к области математической логики. Эта задача принадлежит к классу *NP*, *Pudlak* [1] показал, что к ней сводится *NP*-полная задача «Точное покрытие 3-множествами (ТП-3)» [2]. Таким образом, данная задача – *NP*-полная. Кроме того, задача поиска минимального множества аксиом имеет огромное прикладное значение. Она возникает в области искусственного интеллекта (ИИ) при решении задачи поиска объяснений для запроса (*the problem of finding explanations for a query*) [3, 4]. В такой постановке задача *MinA* называется задачей выбора аксиом (*axiom pinpointing*) [3] и используется для отслеживания проводимых утверждений и отладки онтологии.

Существующие алгоритмы решения поставленной задачи можно разделить на две основные категории: алгоритмы *стеклянного ящика* (*glass-box*) и алгоритмы *черного ящика* (*black-box*) [5]. Построение алгоритмов стеклянного ящика зависит от описательной логики данной онтологии, и, следовательно, эти алгоритмы не обладают общностью. Преимуществом алгоритмов черного ящика есть то, что они не зависят от описательной логики задачи, поэтому они проще и более универсальны, но при этом может возрастать их вычислительная сложность. Таким образом, оправдано применение различных метаэвристик для решения задачи *MinA*.

Цель работы – это относительно новый подход для решения данной проблемы, основой которого есть комбинация фрагментарного и эволюционного алгоритмов. Этот подход хорошо зарекомендовал себя при поиске оптимальных решений различных задач дискретной оптимизации [6, 7].

Постановка задачи

Рассмотрим множество $T = \{s\}$ «истинных предложений» [2] и множество Z отношений следования вида

$$R = (A, s), \qquad (1)$$

где $A \subseteq T$, $s \in T$.

Назовем *ядром* любое множество $C \subseteq T$, для которого существует конечная последовательность подмножеств $C \equiv T_0 \subseteq T_1 \subseteq \ldots \subseteq T_\ell = T$ такая, что $\forall j = 1, 2, \ldots, \ell$ множество T_j состоит в точности из таких предложений $s \in T$, что или $s \in T_{j-1}$, или существует $A \subseteq T_{j-1}$, для которого $(A, s) \in Z$. Элементы последовательности $T_0, T_1, \ldots, T_\ell$ будем называть *оболочками*.

Из множества всех ядер $\{C_1, \ldots, C_K\}$, $1 \le K \le 2^m$, $m = |T|$ представляют интерес те, у которых мощность минимальна. Соответственно, оптимизационная задача о минимальном множестве аксиом (ММА) имеет вид

$$|C_k| \to \min, \quad k = \overline{1, K}.$$

Структура задачи ММА схематически представлена на рис. 1.

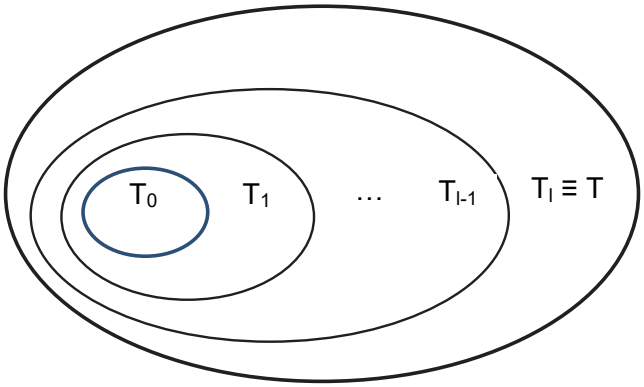

Рис. 1. Структура задачи ММА: $C \equiv T_0$ – ядро, T_j, $j = \overline{1,l}$ – оболочки

В общем виде эта задача, сформулированная как задача распознавания свойств, включена в список *NP*-полных задач в [2] в разделе «Логика».

Фрагментарная модель задачи

Пусть

$$T = \{s_1, s_2, \ldots, s_m\}, \quad |T| = m, \qquad (2)$$

$$Z = \{R_1, R_2, \ldots, R_n\}, \quad |Z| = n. \qquad (3)$$

Определим фрагментарную структуру (Z, E) [8], где Z – множество отношений (3), $E = \{E_0, E_1, \ldots, E_p\}$ – семейство подмножеств множества Z, являющихся допустимыми фрагментами. Элементы $R_i, i = \overline{1,n}$, множества Z выберем в качестве элементарных фрагментов.

Для построения допустимых фрагментов сформулируем рекурсивный алгоритм P, содержащий условия присоединения очередного элементарного фрагмента:

• на начальном шаге $(r = 0)$ выбирается пустое множество $E_0 \equiv \varnothing$;

• $r = 1$: к допустимому фрагменту E_0 можно добавить любой элементарный фрагмент $R_i = (A_i, s_{k_i}) \in Z$ по правилу: $E_1 = E_0 \cup \{R_i\}$, $T_0 = A_i$, $T_1 = T_0 \cup \{s_{k_i}\}$, где A_i – первый элемент упорядоченной пары (1), s_{k_i} – второй элемент этой пары, $1 \le k_i \le m$. Добавленный элемент не участвует в дальнейшем построении. Индекс последнего допустимого фрагмента на $r = 1$ шаге $p_r = p_1 = 1$;

• на r-м шаге, $r = \overline{2,n}$ выбирается очередной элементарный фрагмент $R_i, 1 \le i \le n$, и проводится поиск допустимого фрагмента E_j, $1 \le j \le p_r$, к которому будет присоединяться R_i. Поиск осуществляется по следующему правилу: просматриваем множества T_j, $0 \le j \le p_r$, по порядку, начиная с T_0, и проверяем условие $A_i \subseteq T_j$. Просмотр прекращаем, как только условие выполнилось для некоторого T_j. При этом:

a) если $0 \le j \le p_r - 1$, то выполняем следующие операции: $E_j = E_j \cup \{R_i\}$, $T_{j+u} = T_{j+u} \cup \{s_{k_i}\}$, $u = \overline{0, p_r - j}$;

b) если $j = p_r$, то создаем новый допустимый фрагмент $E_{p_r+1} = E_{p_r} \cup \{R_i\}$, создаем новое множество $T_{p_r+1} = T_{p_r} \cup \{s_{k_i}\}$ и изменяем значение $p_r = p_r + 1$;

Если ни для одного элемента $E_j, j = \overline{1, p_r}$ условие $A_i \subseteq T_j$ не выполнилось, то проводим следующие действия:

c) $E_1 = E_1 \cup \{R_i\}, T_0 = T_0 \cup A_i, T_u = T_u \cup A_i \cup \{s_{k_i}\}$, $u = \overline{1, p_r}$.

Таким образом, всегда выполняется условие $T_j \subset T_{j+1}, \quad j = \overline{0, p_r - 1}$.

Затем переходим к $r+1$-му шагу, для которого $p_{r+1} = p_r$.

• Алгоритм P останавливается при $r = n$, когда все элементы $R_i, i = \overline{1,n}$, множества Z будут включены в фрагментарную структуру, при этом количество p допустимых фрагментов равно p_n.

После выполнения процедуры P будет создано семейство множеств $\{T_0, T_1, \ldots, T_p\}$. Пусть все истинные предложения из T включены в множество R, тогда из построения следует, что $T_p = T$. Содержимое и количество p ос-

тальных множеств $T_j, j=\overline{0,p-1}$ зависит от порядка выбора элементов $R_i, i=\overline{1,n}$, но, в любом случае, выполняется соотношение

$$|T_0| < |T_1| < \ldots < |T_p| = m, \qquad (4)$$

так как по построению $T_0 \subset T_1 \subset \ldots \subset T_p$.

Таким образом, множество T_0 – ядро, т.е. $T_0 = C_k, \quad 1 \le k \le K$.

Теорема. Полученная в результате выполнения алгоритма P фрагментарная структура обладает свойством достижимости [9].

Доказательство. Поскольку при построении фрагментарной структуры, элементарные фрагменты $\{R_i\}$ выбирались произвольным образом, их индексы образуют произвольную перестановку из n элементов. Таким образом, алгоритм P строит сюръективное отображение множества S_n перестановок мощности $n!$ на множество фрагментарных структур. Следовательно, оптимальное решение находится в этом множестве. Таким образом, существует такое упорядочение $Z^s, \quad 1 \le s \le n!$ элементов $R_i, i=\overline{1,n}$ множества Z, что полученное с помощью алгоритма P множество T_0^s равно какому-либо минимальному ядру C_k из множества всех ядер, т.е. $T_0^s = C_k, \quad 1 \le k \le K$ и $|T_0^s| \equiv |C_k| = \min$.

Если не все истинные предложения из (2) участвуют в отношениях из (3), то после выполнения алгоритма P они составят множество $T \setminus T_p$, которое постоянно и не зависит от порядка выбора элементов $R_i, i=\overline{1,n}$. Поэтому, выделив эти предложения один раз, впоследствии их можно исключить из рассмотрения, а на конечном этапе решения задачи получить ядро путем объединения его переменной части T_0 и постоянной части $T \setminus T_p$: $C_k = T_0 \cup T \setminus T_p$. Условие (4) выглядит тогда следующим образом:

$$|T_0| < |T_1| < \ldots < |T_p| < m.$$

Можно сформулировать следующий алгоритм построения ядра $C_k, \quad 1 \le k \le K$:

Ш а г 1. Элементы множества Z линейно упорядочиваются и нумеруются: $(R_1, R_2, \ldots, R_n)$.

Ш а г 2. Выполняем алгоритм P с модификацией для сокращения времени работы. На каждом шаге $r=\overline{1,n}$ выбирается элемент R_r, но, если условие $A_i \subseteq T_j$ не выполнилось ни для одного допустимого фрагмента $E_j \quad j=\overline{1,s_r+1}$, то проверяем его выполнение для последующих элементов $R_{r+1}, R_{r+2} \ldots$. Присоединяем подходящий элемент к фрагментарной структуре согласно правилам *a*) или *b*), если условие не выполнилось ни для одного $R_{r+1}, \; R_{r+2} \ldots, R_n$, то присоединяем к фрагментарной структуре элемент R_r согласно правилу *c*). Присоединенный элемент исключаем из последовательности, оставшиеся перенумеровываем, начиная с $r+1$ до n.

Эволюционно-фрагментарный алгоритм задачи

Эволюционный алгоритм поиска оптимального решения в эволюционной модели строится по следующей стандартной схеме. На начальном этапе случайным образом на множестве допустимых решений выбирается подмножество X_0. Элементы множества X_0 упорядочиваются в соответствии со значениями критерия оптимизации. Это множество будет изменяться в процессе выполнения алгоритма. На шаге с номером s формируется множество X_s – очередная популяция, мощность которой совпадает с мощностью популяции X_0.

На каждом шаге алгоритма с помощью оператора селекции выбирается множество пар текущей популяции, Затем проводятся процедуры скрещивания и мутации, и строится множество потомков. Для каждого потомка также вычисляется значение критерия. С помощью оператора отбора на основе вычисленных значений критерия отбора из объединения текущей популяции и множества потомков формируется новая текущая популяция X_{s+1}, содержащая элементы с наибольшими значе-

ниями критерия отбора. Алгоритм заканчивает работу при выполнении условия остановки.

Для задачи минимального множества аксиом в качестве базового множества выберем подмножество перестановок множества S_n, где n – мощность множества отношений Z. Размером задачи есть длина перестановки. В качестве оператора кроссовера определен геометрический кроссовер на перестановках [8], сохраняющий относительный порядок элементов. Оператор мутации с заданной вероятностью осуществляет произвольную транспозицию двух элементов перестановки. Построенный таким образом эволюционный алгоритм называется эволюционно-фрагментарным алгоритмом задачи.

Генерирование наборов тестовых примеров

Для тестирования эволюционно-фрагментарного (ЭВФ) алгоритма были сгенерированы наборы примеров, различающиеся как по размеру, так и по структуре данных.

Используем схематическое обозначение для отношения *R*, приведенное на рис. 2.

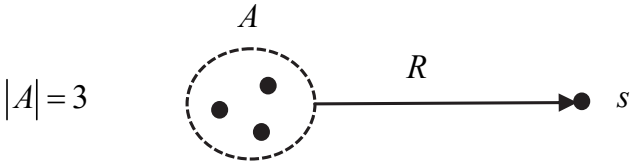

Рис. 2. Схематическое изображение отношения *R* = (*A*, *s*)

Примеры набора A представляют собой сильно упорядоченную структуру, в которой для $i=\overline{1,n}$ каждое множество A_i из отношения $R_i=(A_i,s_{k_i})\in Z$ состоит из одного предложения, все множества A_i различны, все предложения s_{k_i} различны, и каждое предложение $s\in T$ входит в Z. Следовательно, одно и то же предложение не может участвовать в более чем двух отношениях.

Для генерации примера задавались два параметра: мощность ядра m_0 и количество оболочек l, а также случайная перестановка $<\pi_1\ \pi_2\ldots\pi_n>$ из $n=m_0\times l$ номеров отношений R_i. Такие примеры имеют единственное точное заранее известное решение, которому соответствует единственная фрагментарная структура, но не единственная перестановка, среди которых всегда есть тождественная. По номеру отношения можно однозначно восстановить номера двух предложений, образующих его, и, следовательно, номера предложений, образующих ядро и оболочки.

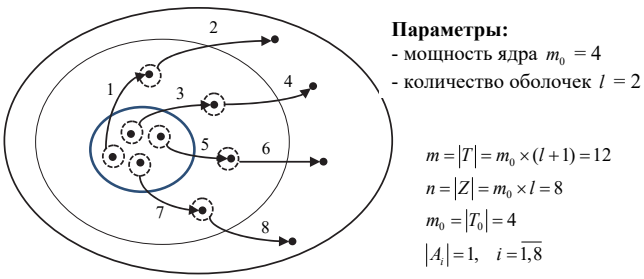

Рис. 3. Точное решение для примера из набора *A*

Пример задачи с размером входа $n=8$ приведен на рис. 3. Пронумеруем подряд отношения, образующие цепочки от ядра до последней оболочки. Тогда данной фрагментарной структуре, кроме тождественной, соответствуют, например, перестановки $<1\,3\,5\,7\,4\,8\,6\,2>$, $<1\,2\,7\,8\,5\,3\,4\,6>$ и др. Схематически построение фрагментарной структуры, приведенной на рис. 3, по тождественной перестановке можно представить следующим образом:

$$m_0 \rightarrow \begin{bmatrix} 1 \xrightarrow{1} 2 \xrightarrow{2} 3 \\ 4 \xrightarrow{3} 5 \xrightarrow{4} 6 \\ 7 \xrightarrow{5} 8 \xrightarrow{6} 9 \\ 10 \xrightarrow{7} 11 \xrightarrow{8} 12 \end{bmatrix}.$$

Здесь количество строк равно мощности ядра m_0, длина цепочки в каждой строке равна количеству оболочек l. Узлы цепочки – это номера предложений, связанных отношением, их в каждой строке $l+1$. Для данного примера ядро $T_0=\{1,4,7,10\}$, оболочка $T_1=\{1,4,7,10,2,5,8,11\}$, оболочка $T_2\equiv T=\{1,4,7,10,2,5,8,11,3,6,9,12\}$.

Таким образом, отношения $R_i, i=\overline{1,n}$ в примерах набора А имеют структуру $(\{s_{k_i}\},s_{k_i+1})$, где

$$k_i = \left\lfloor \frac{i}{l} \right\rfloor * (l+1) + \begin{cases} i \bmod l, & i \bmod l > 0, \\ -1, & i \bmod l = 0. \end{cases}$$

Структура примеров из набора Б гораздо более хаотична. Для генерации примера задавались четыре параметра. Первый – количество предложений в множестве T, второй – количество отношений в множестве Z. Размер $|A_i|$ множества *утверждений* здесь – случайная величина с треугольным распределением и минимальным значением, равным единице. Средний размер и максимальный размер множества A_i – оставшиеся два параметра. Случайное сочетание $C_n^{|A_i|+1}$ производило номера предложений для множества утверждений и предложения–*следствия* отношения $R_i = (A_i, s_{k_i})$, $i = \overline{1,n}$. Таким образом, не все предложения из T обязательно участвуют в отношениях $R_i = (A_i, s_{k_i})$, $i = \overline{1,n}$, и одни и те же предложения могут входить в более чем два отношения.

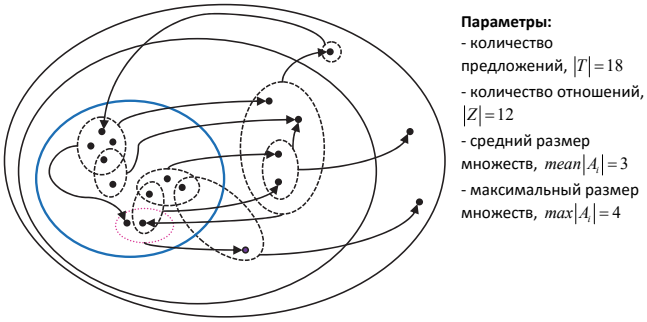

Рис. 4. Неоптимальное решение для примера из набора Б

На рис. 4 приведен пример из набора Б с неоптимальным ядром. Множество, отмеченное выделенным пунктиром, может быть вынесено из ядра, при этом количество оболочек увеличиться на одну.

Результаты работы ЭВФ-алгоритма

Сравнение работы ЭВФ-алгоритма проводилось с двумя приближенными алгоритмами. Первый алгоритм – случайный поиск с равномерным распределением точек. Второй – метод локального поиска со случайным выбором начальной точки. На каждой итерации локального поиска ограничиваемся шаром радиуса единица в метрике Кендалла. Точка локального оптимума в этом шаре становится центром шара следующей итерации. Локальный поиск останавливается на итерации, на которой решение не было улучшено.

Для разных алгоритмов качество полученных решений сравнивалось при равном или почти равном числе просчетов целевой функции (ЦФ). Количество N_{evf} оценок ЦФ ЭВФ-алгоритма есть

N_{evf} = размер_популяции + количество_пар_для_кроссовера * число_поколений,

где размер популяции равен $|X_0|$, количество пар для кроссовера равно $|X_s|$. Метод отбора пар – пропорциональный, вероятность мутации равна 0,1.

Для случайного поиска количество N_{rs} просчетов ЦФ, т.е. количество генерируемых случайных перестановок, задавалось равным N_{evf}. Параметры для локального поиска подбирались так, чтобы среднее по набору число оценок ЦФ приблизительно равнялось N_{evf}.

Примеры набора А генерировались со следующими параметрами: мощность ядра $m_0 = 20$, количество оболочек $l = 4$. Таким образом, количество отношений, т.е. длина перестановки, в каждой задаче $n = 80$, количество предложений $m = 100$. При решении задач из набора А для локального поиска на начальном этапе выбирались две случайные перестановки. Это приводило к тому, что количество N_{ls} оценок ЦФ для каждой из 100 задач изменялось в диапазоне от 85 323 до 132 723 со средним значением 108 425.

Параметры ЭВФ-алгоритма задавались такие:

$размер\_популяции = 1000$,

$количество\_пар\_для\_кроссовера = 200$,

$число\_поколений = 625$.

Таким образом, $N_{evf} = N_{rs} = 126\,000$.

Для генерации примеров из набора Б задавались следующие параметры: количество отношений $n = 50$, количество предложений $m = 100$, для $i = \overline{1,n}$ средний размер множества A_i равнялся трем, максимальный размер множества A_i

равнялся шести. Здесь при решении задач для локального поиска выбирались четыре случайные начальные перестановки, что приводило к следующим характеристикам N_{ls}: минимальное значение 26 955, максимальное 56 355, среднее по набору 43 076. Параметры ЭВФ-алгоритма:

$размер\_популяции = 600$,

$количество\_пар\_для\_кроссовера = 100$,

$число\_поколений = 384$.

Таким образом, $N_{evf} = N_{rs} = 40\,000$.

Все программы реализованы на языке *C*++98 в ИСР *Code::Blocks* 13.12. Расчеты проводились на компьютере с процессором *Intel Core*™ *i*5-4440 *CPU @* 3.40 *GHz* ОЗУ 8 *Gb* и 64-разрядной *Windows*.

Каждый набор А и Б содержал по 100 примеров. В табл. 1 приведены результаты сравнения работы ЭВФ-алгоритма и случайного поиска для наборов А и Б.

Таблица 1

Набор тестовых примеров	Количество			
	Примеров в наборе	Предложений в примере, *m*	Отношений в примере, *n*	Побед ЭВФ
А	100	100	80	100
Б	100	100	50	100

Из табл. 1 видно, что ЭВФ-алгоритм абсолютно превосходит случайный поиск на обоих наборах данных. Отметим, что алгоритм случайного поиска более чувствителен к качеству генератора случайных чисел.

В табл. 2 приведены результаты сравнения работы ЭВФ-алгоритма и локального поиска для наборов А и *Б*.

Таблица 2

Набор тестовых примеров	Количество				
	Предложений в примере, *m*	Отношений в примере, *n*	Побед ЭВФ	Одинаковых решений	Проигрышей ЭВФ
А (100)	100	80	33	18	49
Б (100)	100	50	39	43	18

Из табл. 2 следует, что ЭВФ-алгоритм работает гораздо эффективнее на не сильно упорядоченных данных (набор Б). В то время как на сильно упорядоченных данных (набор А) эффективность работы практически одинакова и очень чувствительна к количеству проводимых оценок ЦФ и качеству генератора случайных чисел. К тому же, как видно из табл. 3, время работы ЭВФ-алгоритма из-за сортировок и других операций, выполняемых на каждой генерации, больше, чем время работы локального поиска.

Таблица 3. Время расчета для всего набора примеров

Набор тестовых примеров	Время работы, с		
	ЭВФ-алгоритма	Случайного поиска	Локального поиска
А (100)	58,4	58,8	8,1
Б (100)	15,3	18,7	3,3

Разница во времени счета между наборами А и Б объясняется тем, что количество оценок ЦФ для ЭВФ-алгоритма и случайного поиска в наборе Б было строго в 3,15 раз меньше, чем в наборе А. Для локального поиска это отношение – случайная величина со средним значением того же порядка.

Заключение. Предложенная в данной статье метаэвристика для решения задачи поиска минимального множества аксиом относится к классу алгоритмов *черного ящика*, т.е. не требует изменений при решении задачи для любой дескрипционной логики.

Возможности данного эволюционно-фрагментарного алгоритма были протестированы на большой базе специально сгенерированных тестовых примеров. Сравнение проводилось с известными приближенными алгоритмами: случайного и локального поиска. Исследована эффективность алгоритма для различных структур входных данных. Результаты исследования выявили преимущество ЭВФ-алгоритма в сравнении со случайным поиском для обеих структур входных данных. В сравнении с алгоритмом локального поиска ЭВФ-алгоритм проявил большую эффективность для входных данных с менее упорядоченной структурой, в то время как для данных с сильно упорядоченной структурой эффективность этих алгоритмов практически одинакова.

Результаты тестирования показали достаточно высокую эффективность эволюционно-

фрагментарного алгоритма при решении задач о минимальном множестве аксиом.

1. *Pudlak P.* Polynomially complete problems in the logic of automated discovery // Mathematical Foundations of Comp. Sci. 1975, Lecture Notes in Comp. Sci. – Berlin: Springer, 1975. – **32**. – P. 358–361. – http://link.springer.com/chapter/ 10.1007%2F3-540-07389-2_221#page-1
2. *Гэри М.*, *Джонсон Д.* Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982. – 416 с.
3. *Schlobach S.*, *Cornet R.* Non-standard reasoning services for the debugging of description logic terminologies // Int. Joint Conf. on Artificial Intelligence. – 2003. – N 3. – P. 355–362. – http://ijcai.org/Proceedings/03/Papers/053.pdf
4. *Kalyanpur A.*, *Horridge M.*, *Sirin E.* Finding All Justifications of OWL DL Entailments // In The Semantic Web. 6th Int. Semantic Web Conf., 2nd Asian Semantic Web Conf., ISWC 2007, ASWC 2007, Busan, Korea, Nov. 11–15, 2007. Proc. – Berlin Heidelberg: Springer, 2007. – P. 267–280. – http://link.springer.com/chapter/10.1007%2F978-3-540-76298-0_20
5. *Penaloza R.*, *Sertkaya B.* On the Complexity of Axiom Pinpointing in the EL Family of Description Logics // Proc. of the Twelfth Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2010). – AAAI Press, 2010. – P. 289–289. – https://www.aaai.org/ocs/index.php/KR/KR2010/ paper/viewFile/1345/1629
6. *Козин И.В.*, *Кривцун Е.В.* Моделирование однослойных и двухслойных трассировок // УСиМ. – 2016. – № 2. – С. 58–64.
7. *Козин И.В.*, *Кривцун Е.В.*, *Полюга С.И.* Фрагментарная структура и эволюционный алгоритм для задач прямоугольного раскроя // Вісн. Запорізьк. нац. ун-ту: Зб. наук. праць. – Запоріжжя: ЗНУ, 2014.– № 2. – С. 65–72.
8. *Козин И.В.* Фрагментарные структуры и эволюционные алгоритмы // Питання прикладної математики і математичного моделювання: Зб. наук. праць. – Д.: Вид-во Дніпропетр. нац. ун-ту ім. Олеся Гончара, 2008. – С. 138–146.
9. *Козин И.В.*, Полюга С.И. О свойствах фрагментарных структур // Вісн. Запорізьк. нац. ун-ту: Зб. наук. праць. Фізико-математичні науки. – Запоріжжя: ЗНУ, 2012. – № 1. – С. 99–106.

Поступила 25.07.2016

Тел. для справок: +38 061 228-7641, 213-41-69 (Запорожье)
E-mail: *kryvtsun@ukr.net*

UDC 519.87

Ye.V. Kryvtsun

Evolutionary-Fragmentary Algorithm of Finding the Minimal Axiom Set

Keywords: minimal axiom set, axiom pinpointing, combinatorial optimization, fragmentary structure, evolutional algorithm.

Introduction. The problem of finding the minimal axiom set (*MinA*) underlies in searching a set of sentences with minimal cardinality that, by applying to it, a given set of relations can be obtain. Any set of sentences, from which, we can obtain the full set of sentences in such a way, is called a core. The cardinality of the core is an objective function of the problem minimization.

The purpose of the work is to develop and to test the hybrid evolutionary-fragmentary algorithm for approximate solving *MinA*.

Methods. A fragmentary structure for the problem is proposed, a given set of relations is defined on, where the elementary fragments are distinct as the relations from that set. The rules of the elementary fragment joining for forming the next feasible fragment are presented. The "greedy" algorithm that allows for any numbers of permutation to build the core is formulated. The built fragmentary model of the problem allows the use of the standard evolutionary algorithm on permutations to find the approximate solution. As the basic set, the subset of permutations is used. The size of the problem is the length of the permutation, which is cardinality of the set of relations. As a crossover operator, geometric crossover on permutations is assumed. It preserves the relative order of elements. The mutation operator performs random transposition of two elements of permutation with a given probability. The hybrid algorithm constructed in this way is called the evolutionary-fragmentary algorithm of the problem.

Results. Performance of evolutionary-fragmentary (EVF) algorithm is compared with two approximate algorithms: the random search and the local search. For tests we generated collections of the instances, which differ both in size and in the data structure. Results of the study show the advantage of EVF-algorithm compared to the random search for both structures of input data. In comparison with the algorithm of the local search EVF-algorithm shows the higher efficiency for the input data with less ordered structure, while for the data with highly ordered structure performances of these algorithms are almost the same.

Conclusions. The proposed algorithm shows high efficiency for finding *MinA* on data with the different structures.

●