AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa inżynierska

Automatyczne odkrywanie procesów biznesowych przy użyciu programowania genetycznego

Automated Business Process Discovery using Genetic Programming

Autor: *Piotr Seemann*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr inż. Krzysztof Kluza*

Kraków, 2020

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję …

Spis treści

1. Wprowadzenie

1.1. Cele pracy

1.2. Przegląd prac

1.3. Zawartość pracy

2. Wstęp teoretyczny

2.1. Procesy biznesowe

2.1.1. Procesy Biznesowe

2.1.2. Dzienniki zdarzeń

2.2. Modelowanie procesów biznesowych

2.3. Algorytmy do wykrywania procesów biznesowych

2.3.1. Alpha algorithm

2.3.2. The ILP Miner

2.3.3. Heuristic Miner

2.3.4. Multi-phase Miner

2.4. Ewolucja genetyczna

2.5. Gramatyka

2.5.1. BNF

2.5.2. Możliwe problemy przy tworzeniu gramatyki

2.6. Metryki

[1]

2.6.1. Stopień złożoności

2.6.2. Odwzorowanie

2.6.3. Precyzja

2.6.4. Generalizacja

3. Projekt i implementacja

3.1. Wykorzystane technologie

3.1.1. Python 3.8.1

3.1.2. PonyGE2

[2]

Rys. 3.1. Przykład użycia

3.2. Projekt systemu

3.3. Tworzenie gramatyki procesu biznesowego

Przy tworzeniu gramatyki procesu biznesowego ważnym jest, żeby znaleźć balans, jeśli chodzi o poziom skomplikowania zaproponowanej gramatyki. [3]

3.4. Implementacja

Ogólny flow [4]:

3.4.0.1. Parsowanie gramatyki

3.4.0.2. Ewaluacja wyniku

3.4.0.3. Wyszukiwanie w modelu logów o określonej długości

3.4.0.4. Obliczanie dopasowania

Pomysł zaczerpnięty z algorytmu Needlessmann-Wunsch [5]

3.4.0.5. Znajdowanie ścieżki w modelu

```
<e> ::= <somethingnoseq>

<somethingnoseq> ::= <somethingnoseq><somethingnoseq> | <andgate> | <xorgate> |
<optgate> | <lopgate> | {<process>}

<something> ::=  <something><something> | <andgate> | <xorgate> | <seqgate> |
<optgate> | <lopgate> | {<process>}

<andgate> ::= and(<something><something>)

<xorgate> ::= xor(<something><something>)

<seqgate> ::= seq(<somethingnoseq><somethingnoseq>)

<optgate> ::= opt(<somethingnosingleopt>)

<optdoublegate> ::= opt(<something><something>)

<somethingnosingleopt> ::= <something><something> | <andgate> | <xorgate> |
<seqgate> | <lopgate> | {<process>}

<lopgate> ::= lop(<somethingnosinglelop>)

<somethingnosinglelop> ::= <somethingnoseq><somethingnoseq> | <andgate> |
<xorgate> | <seqgate> | <optdoublegate> | {<process>}

<process> ::= GE_RANGE:dataset_n_vars
```

Listing 3.1. Gramatyka procesu biznesowego

```
agent Buffer {
  i :: Int = 0;
  proc pop  { out pop i; }
  proc push { in  push i; }
}
```

Listing 3.2. Parser gramatyki

3.4.0.6. Obliczanie innych metryk

3.5. Wybór parametrów algorytmu

```
    def parse(self, expression: str) -> int:

        locally_added_events = []
        numbers = iter(range(len(expression)))
        for i in numbers:
            if expression[i] == "{":
                event = Event(expression[i + 1])
                locally_added_events.append(event)
                self.add_element(event)
                consume(numbers, 2)
            elif expression[i] == ")":
                return i+1
            elif i+4 < len(expression):
                gate_class = getattr(importlib.import_module("processdiscovery.gate." + express
                                     expression[i:i+3].capitalize() + "Gate")
                gate = gate_class()
                consume(numbers, 3)
                processed_characters = gate.parse(expression[i+4:])
                self.add_element(gate)
                consume(numbers, processed_characters)
            else:
                raise Exception
        for event in locally_added_events:
            event.no_branches += 1
```

Listing 3.3. Parser gramatyki

```
    n = round(log_average_length)
    i = 1
    best_result = 0
    # should be change later
    while not n < calculate_min_allowed_length(log_average_length) and \
            not n > calculate_max_allowed_length(log_average_length):
        if min_length <= n <= max_length:
            routes = gate.get_all_n_length_routes(n)
            if len(routes) > 10:
                print(10)
            model_events_list_with_parents = gate.get_events_with_parents()
            model_events_list = [x[1] for x in model_events_list_with_parents]
            model_parents_list = [x[0] for x in model_events_list_with_parents]
            reset_executions(model_events_list)
            # fix_routes to strings inside gate
            if routes is not None and not is_struct_empty(routes):
                best_local_error = 0
                for elem in log:
                    min_local = 1023
                    for event_group in routes:
                        value, events = calculate_best_alignment(event_group, elem)
                        if value < min_local:
                            min_local = value
                            events_global = events
                    add_executions(model_events_list, events_global)
                    best_local_error += min_local

                best_local_alignment = calculate_fitness_metric(best_local_error, log_length, log, n
                best_local_generalization = calculate_generalization_metric(model_events_list)
                best_local_result = (best_local_alignment + best_local_generalization) / 2
                if best_local_result > best_result:
                    best_result = best_local_result
        if i % 2 == 1:
            n -= i
        else:
            n += i
        i += 1
    return best_result
```

Listing 3.4. Parser gramatyki

```
    n = round(log_average_length)
```

Listing 3.5. Parser gramatyki

```
agent Buffer {
  i :: Int = 0;
  proc pop  { out pop i; }
  proc push { in  push i; }
}
```

Listing 3.6. Parser gramatyki

```
def traceback(al_mat, penalty_gap, model, log_global, model_results_local):
    array = copy(al_mat)
    log = copy(log_global)
    model_result = []
    i = len(model)  #The dimension of the matrix rows.
    j = len(log)  #The dimension of the matrix columns.

    while i != 0:
        event_group_full_length = len(model[i - 1])
        if model_results_local[i] is not None:
            matched_flag = False
            if array[i][j] == array[i - 1][j] + event_group_full_length * penalty_gap:
                [model_result.append(None) for _ in range(event_group_full_length)]
                array[i][j] = 0
                i -= 1

            else:
                for k in range(j):
                    processes = get_not_none(model_results_local[i][k][len(model_results_local[i][k]
                    if array[i][j] == array[i - 1][k] + (event_group_full_length + (j-k) - 2 * len(p
                        [model_result.append(x) for x in processes]
                        for x in processes:
                            log = log.replace(x.name, "", 1)
                        [model_result.append(None) for _ in range(event_group_full_length - len(proc
                        array[i][j] = 0
                        i -= 1
                        j = k
                        matched_flag = True
                        break

                if not matched_flag:
                    if array[i][j] == array[i][j - 1] + penalty_gap:
                        array[i][j] = 0
                        j -= 1

        else:
            if array[i][j] == array[i - 1][j] + penalty_gap:
                model_result.append(None)
                array[i][j] = 0
                i -= 1
            elif array[i][j] == array[i][j - 1] + penalty_gap:
                array[i][j] = 0
                j -= 1
            elif array[i][j] == array[i - 1][j - 1]:
                model_result.append(model[i-1])
                log = log.replace(model[i-1].name, "", 1)
                array[i][j] = 0
                i -= 1
                j -= 1

    return model_result
```


Listing 3.7. Znajdowanie ścieżki w modelu

```
agent Buffer {
  i :: Int = 0;
  proc pop  { out pop i; }
  proc push { in  push i; }
}
```

Listing 3.8. Parser gramatyki

4. Dyskusja rezultatów

4.1. Przykładowe wyniki

4.2. Porównanie z innymi algorytmami

4.3. Wyniki w zależności od przyjętych metryk

4.4. Wnioski

5. Podsumowanie

Bibliografia

[1] J. C. A. M. Buijs, B. F. van Dongen i W. M. P. van der Aalst. „Quality Dimensions in Process Discovery: The Importance of Fitness, Precision, Generalization and Simplicity". W: *International Journal of Cooperative Information Systems* 23.01 (2014), s. 1440001. DOI: *10.1142/S0218843014400012*. eprint: *https://doi.org/10.1142/S0218843014400012*.

[2] Michael Fenton i in. „PonyGE2". W: *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2017). DOI: *10.1145/3067695.3082469*.

[3] Michael zur Muehlen i Jan Recker. „How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation". W: *Advanced Information Systems Engineering*. Red. Zohra Bellahsène i Michel Léonard. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, s. 465–479. ISBN: 978-3-540-69534-9.

[4] Wil van der Aalst, Arya Adriansyah i Boudewijn van Dongen. „Replaying history on process models for conformance checking and performance analysis". W: *WIREs Data Mining and Knowledge Discovery* 2.2 (2012), s. 182–192. DOI: *https://doi.org/10.1002/widm.1045*. eprint: *https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1045*.

[5] Saul B. Needleman i Christian D. Wunsch. „A general method applicable to the search for similarities in the amino acid sequence of two proteins". English (US). W: *Journal of Molecular Biology* 48.3 (mar. 1970), s. 443–453. ISSN: 0022-2836. DOI: *10.1016/0022-2836(70)90057-4*.