



UNIwersYTET KAZIMIERZA WIELKIEGO

KOLEGIUM III

WYDZIAŁ INFORMATYKI

Michał Breczko

nr albumu 92548

TEMAT PRACY INŻYNIERSKIEJ

Projekt i wykonanie programu optymalizującego linię produkcyjną

Praca inżynierska napisana pod kierunkiem

dr. inż. Janusza Dorożyńskiego

BYDGOSZCZ 2023

Streszczenie pracy dyplomowej

Temat pracy dyplomowej

Projekt i wykonanie programu optymalizującego linię produkcyjną.

Imię i nazwisko autora pracy: Michał Breczko

Nr albumu: 92548

Imię i nazwisko promotora pracy: dr. Inż. Janusz Dorożyński

Słowa kluczowe: Python, PyCharm, SQL, Firebird, IBExpert, MS Excel, Analiza, Optymalizacja, Algorytm heurystyczny

Treść streszczenia

Celem niniejszej pracy jest stworzenie programu, który zoptymalizuje organizację pracy, eksploatację maszyn oraz gospodarkę materiałową wykluczając przy tym błędy ludzkie. Program jest napisany w programie Pycharm w języku Python oraz w pewnej części SQL, który pobiera zlecenia z udostępnionej bazy danych, następnie je grupuje, sortuje i za pomocą algorytmu zachłannego przydziela zadania dostępnym maszynom, a po przetworzeniu udostępnia je na lokalnym serwerze w pliku MS Excel. W ostatnim rozdziale opisane zostały wnioski z projektu oraz podsumowanie z osiągniętych korzyści, które daje program.

Podziękowania

Chciałbym złożyć serdeczne podziękowania dr. inż. Januszowi Dorożyńskiemu, wykładowcy Instytutu Informatyki za poświęcony czas oraz cenne uwagi w trakcie tworzenia pracy, kolegom z pracy Rafałowi Krieger za pomoc i motywowanie oraz Wojciechowi Białemu za sprawdzenie zgodności pracy z polityką poufności spółki.

Spis treści

I. WSTĘP	5
1. Wprowadzenie	5
2. Cel pracy	5
3. Teza pracy	6
4. Zakres i układ pracy	6
II. CZĘŚĆ TEORETYCZNA	7
1. Historia algorytmów optymalizacyjnych	7
2. Historia problemu przydziału zadań	7
3. Firebird.....	8
4. SQL	8
5. PyCharm	9
6. Python	9
7. Excel	10
8. Spółka i park maszynowy	11
III. CZĘŚĆ PROJEKTOWA	12
1. Wyliczenie współczynników wydajności poszczególnych maszyn na podstawie udostępnionych danych historycznych.....	12
2. Pobranie od użytkownika dat z których będziemy pobierać zlecenia produkcyjne z bazy danych poprzez odpowiednie zapytanie SQL	13
3. Przygotowanie danych z normami i zleceniami	14
4. Pogrupowanie danych w sposób dopasowany do używanego algorytmu i użytkowanych maszyn.....	16
5. Stworzenie połączenia pliku ze zleceniami i normami	17
6. Przygotowanie wzoru na podstawie współczynników wydajności do wyliczenia mocy przerobowych poszczególnych maszyn	18
7. Pobranie od użytkownika informacji o dostępności poszczególnych maszyn	19

8. Stworzenie algorytmu zachłannego, który będzie przydzielał zadania na podstawie zleceń, norm, dostępnych maszyn oraz ich wydajności	22
9. Zapisanie wyników do pliku	24
10. Wyniki.....	25
IV. PROBLEMY TECHNICZNE	27
1. Wyznaczenie wydajności maszyny	27
2. Tworzenie zapytań do bazy	27
3. Poprawa jakości danych.....	28
4. Nadpisywanie pliku z przydzielonymi zadaniami	29
V. ZAKOŃCZENIE	30
1. Podsumowanie projektu.....	30
2. Kierunki rozwoju i dalszych badań.....	30
VI. LITERATURA	33
VII. OŚWIADCZENIE AUTORA PRACY DYPLOMOWEJ	34

I. WSTĘP

1. Wprowadzenie

W erze cyfryzacji i nieustannie rosnącej złożoności systemów produkcyjnych, efektywne przydzielanie zadań odpowiednim maszynom stanowi jedno z kluczowych wyzwań dla inżynierów i menedżerów. Optymalizacja przydziału zadań nie tylko wpływa na efektywność produkcji, ale również na jakość produktów, koszty operacyjne oraz zadowolenie klienta. W praktyce odpowiednie zarządzanie zadaniami i maszynami może znacząco wpłynąć na konkurencyjność przedsiębiorstwa na rynku.

Przydział zadań maszynom jest problemem, który dotyczy wielu branż - od przemysłu ciężkiego, przez technologię informacyjną, aż po sektor usług. W każdym z tych sektorów specyfika pracy i charakterystyka przyrządów jest inna, jednak podstawowe pytania pozostają te same: jak efektywnie przydzielić zadania, aby zminimalizować czas realizacji, koszty i jednocześnie zwiększyć wydajność?

Niezależnie od branży czy specyfiki procesu produkcyjnego, optymalizacja przydziału zadań maszynom jest krytycznym elementem, który może przynieść znaczące korzyści dla organizacji. Przy prawidłowym podejściu może stać się źródłem trwałej przewagi konkurencyjnej.¹

2. Cel pracy

Celem niniejszej pracy jest opracowanie programu w języku Python, wspomaganego skryptami SQL. Program ma za zadanie pobierać zlecenia z konkretnej bazy danych, następnie grupować i sortować je według ustalonych kryteriów. Kluczową funkcją jest implementacja algorytmu zachłannego, który w inteligentny sposób przydzieli zlecenia dostępnym przyrządom. W wyniku tego procesu przedsiębiorstwo będzie mogło lepiej wykorzystać swoje zasoby, co przekłada się na krótszy czas realizacji zadań i niższe koszty operacyjne. Zakończony proces będzie miał dodatkową wartość dla użytkowników, ponieważ wyniki zostaną udostępnione na lokalnym serwerze w przystępnym formacie MS Excel. Taka formuła pozwoli na łatwy dostęp do danych oraz ich dalszą analizę, co stanowi kolejny krok w kierunku ciągłego doskonalenia procesów w organizacji.

¹ H. Tyszka, Excel Solver w praktyce. Helion 2021, ISBN: 978-83-283-8399-9

3. Teza pracy

Możliwym jest stworzenie programu w języku Python, wspierającego optymalizację przydziału zleceń z bazy danych maszynom za pomocą algorytmu zachłannego.

4. Zakres i układ pracy

Stworzenie programu w języku Python optymalizującego przydział zadań.

Przewidziane zostały następujące zadania:

- Wyliczenie współczynników wydajności poszczególnych maszyn na podstawie udostępnionych danych historycznych;
- Pobranie od użytkownika dat z których będziemy pobierać zlecenia produkcyjne z bazy danych poprzez odpowiednie zapytanie SQL;
- Przygotowanie danych z normami i zleceniami;
- Pogrupowanie danych w sposób dopasowany do używanego algorytmu i użytkowanych maszyn;
- Stworzenie połączenia pliku ze zleceniami i normami;
- Przygotowanie wzoru na podstawie współczynników wydajności do wyliczenia mocy przerobowych poszczególnych maszyn;
- Pobranie od użytkownika informacji o dostępności poszczególnych maszyn;
- Stworzenie algorytmu zachłannego, który będzie przydzielał zadania na podstawie zleceń, norm, dostępnych maszyn oraz ich wydajności;
- Zapisanie wyników do pliku.

II. CZĘŚĆ TEORETYCZNA

1. Historia algorytmów optymalizacyjnych

Historia algorytmów optymalizacyjnych jest bogata i złożona, a jej korzenie sięgają starożytności. W tamtych czasach ludzie rozważali różne problemy usprawnień, takie jak znalezienie najkrótszej drogi czy najbardziej efektywnego planu.

W starożytnym Egipcie matematycy pracowali nad problemem podziału ziemi, a w starożytnym Rzymie inżynierowie projektowali sieci akweduktów, udoskonalać ich przebieg. Te wczesne zagadnienia dały początek rozważaniom nad sposobami rozwiązywania problemów funkcjonalnych.

W XVII i XVIII wieku rozwój rachunku różniczkowego, dzięki Newtonowi i Leibnizowi, pozwolił na formalne podejście do optymalizacji funkcji. Metody oparte na pochodnych stały się podstawą dla wielu klasycznych technik przekształcania.

XX wiek przyniósł rozwój informatyki i pojawienie się komputerów. To otworzyło drzwi dla licznych algorytmów heurystycznych i metaheurystycznych. Algorytmy takie jak algorytmy genetyczne, symulowane wyżarzanie czy roje cząstek stały się popularne dla trudnych problemów optymalizacyjnych, które nie mogły być rozwiązane klasycznymi metodami.

W miarę rozwoju technologii i potrzeby rozwiązania coraz bardziej złożonych problemów, algorytmy optymalizacyjne stały się bardziej zaawansowane. Przykładowo, w drugiej połowie XX wieku rozwinięto metody programowania liniowego, które znalazły zastosowanie w przemyśle i nauce.

Ostatnie dekady przyniosły również rozwój algorytmów opartych na uczeniu maszynowym i sztucznej inteligencji, które są używane do rozwiązywania problemów przydziału zadań w dziedzinach takich jak logistyka, finanse czy medycyna.²

2. Historia problemu przydziału zadań

Problem przydziału zadań jest bardzo powszechny w wielu dziedzinach takich jak transport, handel czy przemysł oraz jest ściśle powiązany z algorytmami, ponieważ jest jednym z najbardziej badanych problemów optymalizacyjnych. Pomijając pierwsze koncepcje jego

² P. Wróblewski, Algorytmy w Pythonie. Helion 2022, ISBN: 978-83-8322-161-8

rozwiązania, które były tworzone jeszcze przed pierwszymi komputerami możemy zaprezentować kilka podejść rozwiązywania tego typu problemów:

- Programowanie liniowe: W latach 40. i 50. XX wieku, z rozwojem programowania liniowego, zaczęły pojawiać się konkretne metody udoskonaleń przydziału. W tym czasie Dantzig wprowadził algorytm sympleksowy, który może być stosowany w sytuacji kłopotliwego przydziału zadań;
- Algorytm Węgierski: W latach 50. XX wieku, matematyk Harold Kuhn opracował tzw. algorytm węgierski - efektywny sposób rozwiązania problemu przydziału w części kosztów jednostkowych. Jest to jedna z pierwszych metod, która była specjalnie dedykowana do tego przypadku i jest ona wciąż używana w wielu aplikacjach;
- Rozwój heurystyk i metaheurystyk: Wraz z rozwojem informatyki w latach 70. i 80. XX wieku, zaczęły pojawiać się heurystyczne i metaheurystyczne podejścia do problemu przydziału, takie jak algorytmy genetyczne, symulowane wyżarzanie czy algorytmy kolonii mrówek. Te metody były używane dla bardziej skomplikowanych wersji przydziału, które nie mogły być efektywnie rozwiązane klasycznymi metodami.

3. Firebird

Firebird to relacyjny system zarządzania bazami danych (RDBMS) oferujący wiele standardowych funkcji RDBMS. Firebird jest open-source i oparty na oryginalnym kodzie źródłowym InterBase od Borland. Przez lata, Firebird został rozbudowany i ulepszony przez społeczność open source. Firebird wspiera procedury przechowywane, triggerzy, funkcje użytkownika, a także oferuje wielowarstwową architekturę. Można go stosować zarówno na serwerach, jak i w aplikacjach klienckich.

4. SQL

SQL (Structured Query Language) to standardowy język programowania służący do zarządzania i manipulacji bazami danych. Umożliwia on użytkownikom tworzenie, modyfikowanie, zarządzanie i odpypywanie danych w bazach danych. SQL stał się standardem przemysłowym dla relacyjnych baz danych i jest wspierany przez większość systemów RDBMS, w tym Firebird. Operacje w SQL można podzielić na kilka kategorii, takich jak DDL

(Data Definition Language) do zarządzania schematami, DML (Data Manipulation Language) do manipulowania danymi czy DCL (Data Control Language) do zarządzania prawami dostępu.

5. PyCharm

PyCharm to zintegrowane środowisko programistyczne (IDE) stworzone przez firmę JetBrains specjalnie dla języka programowania Python. Jest jednym z najpopularniejszych IDE dla Pythona i oferuje szeroki zakres funkcji, które wspierają programistów w tworzeniu, testowaniu i debugowaniu:

- Zaawansowana analiza kodu, podkreślająca błędy, sugerująca poprawki i zapewniająca szybkie refaktoryzację;
- Narzędzia do debugowania, umożliwiające śledzenie wartości zmiennych, ustawianie punktów przerwania oraz krokowe wykonanie kodu;
- PyCharm oferuje zintegrowane narzędzia do pracy z bazami danych, umożliwiające wykonywanie zapytań SQL, przeglądanie schematów i zarządzanie danymi;
- Umożliwia wykonywanie poleceń bezpośrednio w IDE, bez konieczności przełączania się między oknami.

To tylko kilka przykładów, ponieważ ze względu na rozbudowanie PyCharm nie sposób w krótkim wstępie opisać wszystkich jego możliwości w szczególności, że większość z nich nie jest powiązana z głównym tematem pracy.











6. Python

Python to język programowania wysokiego poziomu, który zdobył ogromną popularność w dziedzinie analizy danych i optymalizacji. Jego elastyczność, łatwość użycia oraz rozbudowane biblioteki czynią go idealnym narzędziem do przetwarzania, analizy i wizualizacji danych, a także do tworzenia modeli optymalizacyjnych.

Dwa kluczowe aspekty Pythona, które przyczyniły się do jego sukcesu w analizie danych, to bogate biblioteki i łatwa integracja z innymi narzędziami. Biblioteka pandas to narzędzie do manipulacji danymi i analizy, które umożliwia łatwe wczytywanie, czyszczenie, transformację oraz analizę strukturalnych zbiorów danych. NumPy pozwala na skomplikowane operacje na danych numerycznych, a matplotlib i seaborn są używane do wizualizacji danych.

Kiedy przychodzi czas na szukanie ulepszeń, Python oferuje narzędzia takie jak SciPy i PuLP do problemów optymalizacyjnych. Dla bardziej zaawansowanych zastosowań, takich jak optymalizacja stochastyczna czy uczenie maszynowe, biblioteki jak scikit-learn czy TensorFlow stają się niezbędne.

W praktyce, bez względu na to, czy analityk danych chce przeprowadzić eksploracyjną analizę danych, wizualizację, czy zbudować zaawansowany model predykcyjny czy optymalizacyjny, Python dostarcza odpowiednie narzędzia, by to osiągnąć w skuteczny i efektywny sposób. Dzięki temu, analitycy i naukowcy danych na całym świecie wybierają Pythona jako swój główny język do analizy danych.³

Sep 2023	Sep 2022	Change	Programming Language	Ratings	Change
1	1		 Python	14.16%	-1.58%
2	2		 C	11.27%	-2.70%
3	4	▲	 C++	10.65%	+0.90%
4	3	▼	 Java	9.49%	-2.23%
5	5		 C#	7.31%	+2.42%
6	7	▲	 JavaScript	3.30%	+0.48%
7	6	▼	 Visual Basic	2.22%	-2.18%
8	10	▲	 PHP	1.55%	-0.13%
9	8	▼	 Assembly language	1.53%	-0.96%
10	9	▼	 SQL	1.44%	-0.57%

Rysunek 1 Porównanie popularności języków programowania na świecie 09.2023, Dostępny w internecie:

<https://www.tiobe.com/tiobe-index/>

7. Excel

MS Excel to niezwykle intuicyjne i szeroko dostępne narzędzie, które od lat stanowi fundament w zakresie przetwarzania i analizy danych w wielu firmach i organizacjach. Ze względu na jego prostotę i uniwersalność jest idealnym wyborem dla użytkowników, którzy nie

³ M. Gągolewski, M. Bartoszek, A. Cena, Przetwarzanie i analiza danych w języku Python. Wydanie I, PWN 2016, ISBN: 978-83-01-18940-2

mają głębokiego doświadczenia w korzystaniu z bardziej zaawansowanych systemów analizy danych.

Jednym z kluczowych atutów Excela jest jego elastyczność. Umożliwia on tworzenie złożonych arkuszy kalkulacyjnych, wykresów oraz raportów w sposób stosunkowo prosty i szybki. Dla tych, którzy potrzebują głębszej analizy i optymalizacji oferuje różnorodne dodatki i narzędzia, jednym z nich jest Solver. Dzięki przystępności Solvera użytkownicy mogą przeprowadzać zaawansowane analizy i testy optymalizacyjne na swoich danych, co czyni Excela jeszcze potężniejszym narzędziem nawet w nieco bardziej specjalistycznych zastosowaniach.

Co więcej, wymieniony arkusz kalkulacyjny radzi sobie doskonale z zestawieniami i analizami na mniejszych zbiorach danych. Choć może nie być idealny do przetwarzania gigantycznych baz danych, w wielu przypadkach, zwłaszcza w mniejszych i średnich przedsiębiorstwach, jest więcej niż wystarczający.

Podsumowując, choć na rynku dostępne są bardziej specjalistyczne i skomplikowane narzędzia do analizy danych, MS Excel pozostaje jednym z najbardziej uniwersalnych i przyjaznych dla użytkownika rozwiązań. Dla wielu firm i indywidualnych użytkowników stanowi on "złoty środek" między funkcjonalnością a prostotą obsługi.⁴

8. Spółka i park maszynowy

Dostarczone dane pochodzą z firmy produkującej meble tapicerowane i dla niej też tworzony jest niniejszy program. Mimo posiadania nowoczesnych maszyn nadal polega na doświadczeniu i intuicji swoich pracowników w procesie przydzielania zleceń na maszyny, który odbywa się w sposób ręczny. W skład parku maszynowego firmy wchodzi urządzenia typu "TopSpin" oraz "Cutter". Maszyna "TopSpin" to wyspecjalizowane urządzenie do cięcia tkanin i innych materiałów tapicerskich. Jego główną cechą charakterystyczną jest możliwość obsługi tylko jednej warstwy krojonego materiału na raz, co pozwala na nadzwyczaj precyzyjne cięcie szczególnie delikatnych tkanin. Z kolei "Cutter" to potężne urządzenie wielowarstwowe, zdolne do cięcia aż 20 warstw materiału jednocześnie. Dzięki temu jest niezastąpiony w produkcji masowej oraz w przypadkach, kiedy konieczne jest szybkie i jednorodne przetworzenie większej ilości surowca.

⁴ Excel [dostęp 14.09.2023], <https://support.microsoft.com/excel>

III. CZĘŚĆ PROJEKTOWA

1. Wyliczenie współczynników wydajności poszczególnych maszyn na podstawie udostępnionych danych historycznych

Badanie wydajności maszyny stanowiło kluczowy element analizy w kontekście przydziału zadań. Zastosowano zestaw danych obejmujący kilkadziesiąt tysięcy zaraportowanych czynności zgromadzonych przez kilka tygodni. Te dane pochodziły bezpośrednio z operacji na przyrządach, oferując szeroki przegląd jej działalności w różnych warunkach i okolicznościach. Zbieranie tak szerokiego zakresu informacji miało na celu zapewnienie, że analiza była jak najbardziej reprezentatywna dla rzeczywistego funkcjonowania parku maszynowego. Wszystkie czynności zostały poddane szczegółowej analizie, aby zidentyfikować wzory działania, potencjalne zakłócenia oraz inne istotne czynniki, które mogłyby wpływać na wydajność urządzeń. Na podstawie tych danych wyznaczono stopień wydajności maszyn. Ustalenie tego wskaźnika było niezbędne do zaprojektowania i wdrożenia skutecznego algorytmu przydziału zadań.

Artykuł opis	Raportujący	Imię	Nazwisko	Czynności opis	Zlec.prod.	Godzina	Data	Czas [min]
1	TOPSPIN6	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	10:50:48	06.02.2023	8,936
1	TOPSPIN6	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	10:50:49	06.02.2023	8,936
2	TOPSPIN6	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	10:50:51	06.02.2023	6,156
3	TOPSPIN6	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	10:50:52	06.02.2023	5,393
4	TOPSPIN6	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	10:50:54	06.02.2023	8,028
5	TOPSPIN1	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	20:52:32	06.02.2023	12,943
5	TOPSPIN1	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	20:52:32	06.02.2023	12,943
6	TOPSPIN6	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	12:51:25	07.02.2023	9,553
7	TOPSPIN6	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	12:51:25	07.02.2023	9,115
8	TOPSPIN6	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	20:53:04	27.02.2023	8,002
9	TOPSPIN6	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	23.01.2023	20:53:04	27.02.2023	8,028
10	TOPSPIN3	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	07:42:58	01.02.2023	38,239
11	TOPSPIN5	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	07:57:32	01.02.2023	3,007
10	TOPSPIN3	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	08:24:38	01.02.2023	38,239
12	TOPSPIN5	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	08:30:46	01.02.2023	13,514
13	TOPSPIN3	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	09:25:18	01.02.2023	19,775
14	TOPSPIN5	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	09:40:35	01.02.2023	15,773
15	TOPSPIN3	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	09:49:55	01.02.2023	33,57
16	TOPSPIN5	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	09:50:40	01.02.2023	1,986
16	TOPSPIN5	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	09:50:41	01.02.2023	1,986
17	TOPSPIN1	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	10:07:05	01.02.2023	11,694
17	TOPSPIN1	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	10:10:56	01.02.2023	11,694
17	TOPSPIN1	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	10:10:57	01.02.2023	11,694
17	TOPSPIN1	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	10:10:58	01.02.2023	11,694
18	TOPSPIN3	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	10:16:29	01.02.2023	33,57
18	TOPSPIN3	XYZ	XYZ	KROJENIE TKANINY GŁÓWNEJ	24.01.2023	10:48:41	01.02.2023	33,57

Rysunek 2 Dane pobrane do wyznaczenia współczynnika wydajności.

Na czerwono zostały zaznaczone pozycje, które ze względu na specyfikację maszyn powinny zostać przypisane do innego urządzenia, jednakże nie wpływa to na wyliczenia, które bazują na powyższej tabeli (poglądowo został załączony tylko kawałek całego zestawienia).

2. Pobranie od użytkownika dat z których będziemy pobierać zlecenia produkcyjne z bazy danych poprzez odpowiednie zapytanie SQL

Aby pobierać zlecenia produkcyjne z bazy danych w oparciu o konkretne daty, najpierw konieczne jest nawiązanie połączenia z bazą. W tym celu używamy funkcji `fdb.connect`, która umożliwia połączenie z bazą. Następnie, po nawiązaniu połączenia, tworzymy obiekt kursora za pomocą metody `cursor()` - służy on do wykonywania poleceń SQL oraz do odbierania wyników.

Użytkownik jest następnie proszony o podanie zakresu dat, z którego chce pobierać zlecenia produkcyjne. Po uzyskaniu odpowiednich dat, formułowane jest zapytanie SQL, które filtruje zlecenia produkcyjne na podstawie podanego zakresu dat.

Zapytanie SQL jest następnie wykonane w bazie danych przy użyciu metody `execute` obiektu kursora. Po wykonaniu zapytania, wyniki są pobierane za pomocą metody `fetchall()`, która zwraca wszystkie wiersze wynikowe jako listę krotek.⁵

⁵ Firebirdsql [dostęp 15.09.2023],
https://firebirdsql.org/file/documentation/drivers_documentation/python/fdb/getting-started.html

```
cur.execute(f"select zamowienia_spec.zamowienia_spec_id as numer_zamowienia,  
wg.artikul_kod as MODEL_KOD, wg.nazwa1 as model,  
zamowienia_spec.zebrana_nazwa as tkanina, limit_spec.ilosc_plan as ilosc  
from przegzlecprod_spec  
inner join limit_spec  
on przegzlecprod_spec.limit_spec_id = limit_spec.limit_spec_id  
inner join przegzlecprod  
on przegzlecprod.pregzlecprod_id = przegzlecprod_spec.pregzlecprod_id  
inner join zamowienia_spec  
on limit_spec.zamowienia_spec_id = zamowienia_spec.zamowienia_spec_id  
inner join zamowienia  
on zamowienia_spec.zamowienia_id = zamowienia.zamowienia_id  
inner join artykul wg  
on zamowienia_spec.artikul_id = wg.artikul_id  
inner join klienci  
on zamowienia.klienci_id = klienci.klienci_id  
inner join limit  
on limit_spec.limit_id = limit.limit_id  
inner join lc_wydzialy  
on przegzlecprod.lc_wydzialy_id = lc_wydzialy.lc_wydzialy_id  
inner join jednostki  
on jednostki.jednostki_id = wg.jednostki_id  
where ({sql_condition}) and lc_wydzialy.lc_wydzialy_opis = 'KROJOWNIA'")
```

Rysunek 3 Zapytanie SQL wyciągające informacje o zleceniach ze zmienną {sql_condition} pobieraną od użytkownika.

3. Przygotowanie danych z normami i zleceniami

Przygotowanie danych to kluczowy etap każdego procesu analitycznego czy projektowego, a w przypadku połączenia norm i zleceń jest to szczególnie istotne. W takich zestawach danych mogą pojawiać się różnego rodzaju nieścisłości, które trzeba uwzględnić.

Założmy, że masz zestaw danych z normami i zleceniami, w których różne warianty tej samej nazwy zostały zapisane w różny sposób - np. z dodatkowymi spacjami, znakami specjalnymi czy literówkami. Aby zapewnić spójność pierwszym krokiem jest standaryzacja tych nazw. Można to zrobić, usuwając niepotrzebne znaki, przekształcając wszystkie litery na małe lub duże, a także poprzez wykorzystanie funkcji `replace` pokazaną na rysunku 4, która usuwa zbędne spacje z tekstu.

Następnie, po przetworzeniu nazw, warto przeszukać dane w poszukiwaniu duplikatów. W zestawach danych, zwłaszcza tych pochodzących z różnych źródeł, często pojawiają się powtarzające się rekordy. Jeśli chcemy zachować tylko rekord z największą wartością trzeba odpowiednio posortować dane i usunąć zbędne duplikaty.

Kolejnym krokiem jest identyfikacja i decyzja co zrobić z błędnymi wierszami. Błędy mogą pojawić się w różnych formach: niekompletne dane, błędne wartości czy brakujące rekordy. W zależności od charakterystyki błędu i jego wpływu na algorytm, możemy zdecydować się na korektę błędów, usunięcie błędnych wierszy lub ich zastąpienie wartościami domyślnymi.

Podsumowując, staranne przygotowanie danych to proces, który wymaga uwagi do detalu i gruntownej wiedzy na temat struktury i charakterystyki przetwarzanych informacji. Właściwe przygotowanie gwarantuje jednak większą dokładność i wiarygodność wyników.

```
normy[0] = normy[0].str.replace(' ', '')
zlecenia[1] = zlecenia[1].str.replace(' ', '')
#usuwanie spacji - przygotowywanie danych

normy = normy.sort_values(by=[0, 2], ascending=[True, False])
normy = normy.drop_duplicates(subset=[0], keep='first')
#usuwanie duplikatów, które wynikają z różnych kombinacji poduszek -
#- informacja nie do zweryfikowania w uwagach bez wpływu na algorytm optymalizacji

zlecenia.loc[zlecenia[2].str.contains('APR'), 1] = \
    zlecenia.loc[zlecenia[2].str.contains('APR'), 1].str.replace('A$', '', regex=True)
#pomijanie apretowania
filtered_df = zlecenia[~zlecenia[1].str.startswith('SERW')]
#pomijamy pliki SERW
```

Rysunek 4 Przedstawiający przygotowanie danych.

4. Pogrupowanie danych w sposób dopasowany do używanego algorytmu i użytkowanych maszyn

Dane zostały wyfiltrowane w poprzednim punkcie, a teraz muszą zostać pogrupowane według pewnej wspólnej kategorii. Każda grupa jest sortowana według wartości w piątej kolumnie, a następnie przeszukiwana w celu zsumowania wartości z tej kolumny. Istnieje pewien próg, który wynosi 20. Gdy suma wartości przekracza ten próg, dane są dzielone w taki sposób, aby żadna suma nie przekraczała 20. To dzielenie odbywa się w iteracyjny sposób, aż do momentu, gdy cała ilość zostanie prawidłowo przydzielona.

Dla każdego wiersza w grupie, tworzone są też szczegółowe informacje, które łączą kilka kolumn w jeden ciąg tekstu, zawierając informacje takie jak numer systemowy, ilość, model i rodzaj tkaniny. Gdy suma osiąga wartość 20 lub gdy przetwarzane są ostatnie wiersze grupy, te szczegóły są zapisywane razem z sumą i kodem modelu.

Po przetworzeniu wszystkich wierszy w grupie, wynik jest dodawany do listy wyników. Na koniec, wyniki są konwertowane z powrotem na ramkę danych, tworząc w ten sposób przetworzony zestaw danych zawierający podsumowanie dla każdej grupy.


```
for index, group in grouped:
    group_sorted = group.sort_values(by=4, ascending=False)

    current_rows = []
    current_sum = 0

    for _, row in group_sorted.iterrows():

        if current_sum + row[4] > 20:
            remaining = row[4]

            while remaining > 0:
                amount_to_take = min(20 - current_sum, remaining)
                current_sum += amount_to_take
                remaining -= amount_to_take

                details = f"Nr.sys:{row[0]} Ilość={amount_to_take} Model:{row[2]} Tkanina:{row[3]}"
                current_rows.append(details)

            if current_sum == 20 or _ == group_sorted.index[-1]:
                results.append({
                    'KodModelu': index,
                    'SumaWarstw': current_sum,
                    'Details': "; ".join(current_rows)
                })

                current_rows = []
                current_sum = 0

        else:
            current_sum += row[4]
            details = f"Nr.sys:{row[0]} Ilość={row[4]} Model:{row[2]} Tkanina:{row[3]}"
            current_rows.append(details)

    if current_rows:
        results.append({
            'KodModelu': index,
            'SumaWarstw': current_sum,
            'Details': "; ".join(current_rows)
        })

results_df = pd.DataFrame(results)
```

Rysunek 5 Przedstawiający grupowanie danych.

5. Stworzenie połączenia pliku ze zleceniami i normami

Chcąc zoptymalizować proces produkcji w przedsiębiorstwie, kluczowe jest połączenie informacji o zleceniach produkcyjnych z dostępnymi normami tkaninowymi dotyczącymi poszczególnych modeli produktów. W tym celu warto trzeba połączyć dwa zbiory danych w oparciu o wspólny klucz, którym jest "KodModelu", który trzeba było wcześniej ujednolicić tak jak jest to opisane w pkt. 3. Rozpoczynając, mamy dwie ramki danych: jedna zawierająca

informacje o zleceniach, a druga dotycząca norm produkcji dla poszczególnych modeli. Naszym celem jest dodanie do ramki zleceniowej odpowiednich norm z metrami, które będą odpowiadały konkretnemu kodowi modelu.⁶

```
# 1. Utwórz słownik z normy_df
normy_dict = pd.Series(normy[2].values, index=normy[0]).to_dict()

# 2. Dla każdego wiersza w 'results_df' przypisz odpowiednią wartość z normy_dict do nowej kolumny
results_df['MetryTkaniny'] = results_df['KodModelu'].map(normy_dict)

results_df = results_df.dropna(subset=['MetryTkaniny'])
# pomijanie wzorów
```

Rysunek 6 Przedstawiający dołączenie do pliku ze zleceniami informacji o zapotrzebowaniu na tkaninę

Po dodaniu informacji o metrażu do ramki danych, wzory nieposiadające ustalonych norm zostały pominięte i nie były przydzielane do maszyn.

6. Przygotowanie wzoru na podstawie współczynników wydajności do wyliczenia mocy przerobowych poszczególnych maszyn

Przygotowanie wzoru, na podstawie współczynników wydajności, do wyliczenia mocy przerobowych poszczególnych maszyn, polega na wykorzystaniu danych historycznych i obserwacji dotyczących działania maszyn w danym procesie produkcyjnym. Przygotowanie współczynników zostało opisane w pkt. 1 i wynosi ono dla maszyn typu Cutter 0,185 mb/min, a dla urządzenia typu TopSpin 0,3 mb/min. Warto jednak wspomnieć, że o ile czas krojenia kolejnych warstw tego samego modelu rośnie liniowo mnożąc czas początkowy przez kolejne liczby naturalne odpowiadające ilości warstw o tyle na urządzeniach typu Cutter czas ten rośnie o 5% z każdą kolejną warstwą aż do granicznej wartości 20 warstw, która została umownie przyjęta jako granica możliwości dla tej maszyny.

Do ramki danych, która zawiera informacje o pogrupowanych zleceniach, możemy łatwo dodać nową kolumnę reprezentującą czas potrzebny na produkcję dla poszczególnych

⁶ Python [dostęp 16.09.2023], <https://pl.python.org/docs/lib/built-in-funcs.html>

typów maszyn. Kolumna ta będzie wykorzystywać wzór na liczenie wydajności maszyny dzięki, któremu otrzymujemy zapotrzebowanie czasowe na wykonanie poszczególnych zadań.

Podsumowując, dodanie kolumny z czasami dla maszyn na podstawie współczynnika wydajności pozwala nam dokładnie wskazać, ile czasu każda maszyna potrzebuje na produkcję określonej ilości produktu, co jest kluczowe dla rozwiązania problemu optymalizacji procesu produkcyjnych.

```
def compute_time_for_topspin(sumawarstw, metrytkanin):  
    return (metrytkanin * sumawarstw) / 0.3  
  
def compute_time_for_cutter(sumawarstw, metrytkanin):  
    # obliczanie mnoznika  
    mnoznik = 1 + 0.05 * (sumawarstw - 1)  
    return (metrytkanin * mnoznik) / 0.185  
  
# Obliczanie czasów dla każdego wiersza w dataframe  
results_df['CzasTopSpin'] = results_df.apply\  
    (lambda row: compute_time_for_topspin(row['SumaWarstw'], row['MettryTkaniny']), axis=1)  
results_df['CzasCutter'] = results_df.apply\  
    (lambda row: compute_time_for_cutter(row['SumaWarstw'], row['MettryTkaniny']), axis=1)
```

Rysunek 7 Przedstawiający stworzenie kolumn z zapotrzebowaniem czasu dla maszyn.

7. Pobranie od użytkownika informacji o dostępności poszczególnych maszyn

W środowisku produkcyjnym, kluczowe jest odpowiednie zarządzanie dostępnymi zasobami, w tym maszynami. Dlatego stworzona została funkcja, która umożliwia bardziej precyzyjne planowanie pracy na podstawie rzeczywistego dostępu do maszyn.

Użytkownik jest proszony o podanie dokładnej ilości dostępnych maszyn dla dwóch typów: TopSpin i Cutter. Te informacje są kluczowe, ponieważ różne modele maszyn mogą spełniać różne funkcje w procesie produkcyjnym i mogą mieć odmienne wymagania co do obsługi oraz konserwacji.

Jednak samo podanie ilości maszyn to często niewystarczająca informacja. Maszyny, podobnie jak wszystkie urządzenia przemysłowe, wymagają regularnych przeglądów, konserwacji oraz mogą być czasowo niedostępne z powodu awarii lub innych prac serwisowych. Dlatego funkcja prosi również użytkownika o określenie dokładnego czasu

dostępności dla każdej z maszyn. Dzięki temu, planując produkcję, należy wziąć pod uwagę nie tylko ilość dostępnych maszyn, ale także czas, w którym są one rzeczywiście dostępne do pracy. Czas ten został ograniczony do 168 godzin, ponieważ w rozpatrywanym przez nas przypadku planowanie nigdy nie odbywa się na dłuższy okres, wymusza to tygodniowy system rozliczania materiałów (kontynuacja we wnioskach i problemach technicznych).

Takie podejście pozwala na uniknięcie wielu problemów, takich jak opóźnienia w produkcji czy niewłaściwe przydziały zasobów. Umożliwia to również bardziej precyzyjne prognozowanie i optymalizację procesów produkcyjnych, co w dłuższej perspektywie przekłada się na zwiększenie efektywności i rentowności produkcji.

```

def get_machine_hours(prompt):
    while True:
        try:
            time_input = input(prompt)

            # Sprawdzanie, czy użytkownik wprowadził dane w formacie godzina:minuta
            if ':' in time_input:
                hours, minutes = map(int, time_input.split(":"))
                if 0 <= hours <= 167 and 0 <= minutes <= 59:
                    total_minutes = hours * 60 + minutes
                    return total_minutes
                else:
                    raise ValueError
            # Sprawdzanie, czy użytkownik wprowadził dane tylko w formie godzin
            else:
                hours = int(time_input)
                if 0 <= hours <= 168:
                    return hours * 60
                else:
                    raise ValueError

        except ValueError:
            print("Niepoprawny format. Wprowadź ilość godzin (np. 5 lub 5:30).")

#pierwsze topspiny drugie cuttery

# Dla maszyn typu topspin
num_machines_t = int(input("Ile maszyn typu topspin jest dostępnych? "))
machines_data_t = []

for i in range(1, num_machines_t + 1):
    available_minutes = get_machine_hours(
        f"Dla maszyny typu topspin numer {i}, podaj dostępne godziny pracy (np. 5 lub 5:30): ")
    machines_data_t.append({
        "machine_number": i,
        "available_minutes": available_minutes
    })

# Dla maszyn typu cutter
num_machines_k = int(input("Ile maszyn typu cutter jest dostępnych? "))
machines_data_k = []

for i in range(1, num_machines_k + 1):
    available_minutes = get_machine_hours(
        f"Dla maszyny typu cutter numer {i}, podaj dostępne godziny pracy (np. 5 lub 5:30): ")
    machines_data_k.append({
        "machine_number": i,
        "available_minutes": available_minutes
    })

```

Rysunek 8 Przedstawiający pobranie od użytkownika informacji o dostępności maszyn.

8. Stworzenie algorytmu zachłannego, który będzie przydzielał zadania na podstawie zleceń, norm, dostępnych maszyn oraz ich wydajności

Wybór algorytmu zachłannego dla danego zadania jest motywowany kilkoma kluczowymi czynnikami. Przede wszystkim, jego główną zaletą jest szybkość działania, co w wielu przypadkach jest kluczowe dla efektywności rozwiązania problemu. W porównaniu z metodą simpleksową czy technikami metaheurystycznymi, algorytm ten jest znacznie szybszy, co pozwala na uzyskanie wyników w krótszym czasie. Ta cecha jest niezwykle cenna, zwłaszcza w sytuacjach, gdy czas jest ograniczony lub gdy mamy do czynienia z dużą ilością danych do przetworzenia.

Jednym z kluczowych elementów poprawy efektywności przydziału jest przemyślane sortowanie danych przed podjęciem właściwych decyzji. Dzięki odpowiedniemu uporządkowaniu danych, algorytm zachłanny może działać jeszcze bardziej efektywnie, dokonując wyborów opartych na najbardziej wartościowych kryteriach, a w połączeniu z odpowiednim sortowaniem, jest w stanie podejmować decyzje, które są nie tylko szybkie, ale też trafne.

Dodatkowo, dla konkretnych problemów, decyzyjność wspomnianego algorytmu okazuje się być bardzo trafna. Mimo że nie zawsze gwarantuje on znalezienie rozwiązania optymalnego, w wielu przypadkach dostarcza odpowiedzi dostatecznie bliskie, a czasami nawet optymalne. Dlatego, biorąc pod uwagę wymagania szybkości oraz jakości uzyskanych rozwiązań, algorytm zachłanny, wspomagany przemyślanym sortowaniem, jest najbardziej dopasowanym wyborem dla rozwiązania tego problemu.

```

def assign_tasks_to_machine(machine_data, tasks, machine_type):
    if machine_data['available_minutes'] <= 0:
        return tasks

    tasks_for_machine = []
    machine_time = machine_data['available_minutes']
    to_remove = []

    for index, task in tasks.iterrows():
        task_time = task['CzasCutter'] if machine_type == 'Cutter' else task['CzasTopSpin']

        if machine_time >= task_time:
            tasks_for_machine.append(task)
            machine_time -= task_time
            to_remove.append(index)

    machine_data['available_minutes'] = machine_time
    # Dodajemy nowe zadania do istniejącej listy zadań maszyny
    if 'assigned_tasks' in machine_data:
        machine_data['assigned_tasks'].extend(tasks_for_machine)
    else:
        machine_data['assigned_tasks'] = tasks_for_machine

    tasks.drop(index=to_remove, inplace=True)

    # Jeśli przypisujesz zadania do maszyn typu cutter i nie ma już zadań w tasks_k
    # , zacznij brać zadania z tasks_t
    if machine_type == 'Cutter' and tasks.empty:
        return tasks_t

    # Jeśli przypisujesz zadania do maszyn typu topspin i nie ma już zadań w tasks_t,
    # zacznij brać zadania z tasks_k
    # elif machine_type == 't' and tasks.empty:
    #     return tasks_k

    return tasks

# Sortowanie zadań
tasks_t = results_df[results_df['SumaWarstw'] == 1].copy()
tasks_k = results_df[results_df['SumaWarstw'] > 1].sort_values(by='SumaWarstw', ascending=False).copy()

# print(f"Liczba zadań w tasks_k po przydzieleniu dla maszyn typu cutter: {len(tasks_k)}")
# print(f"Liczba zadań w tasks_t po przydzieleniu dla maszyn typu topspin: {len(tasks_t)}")

# Przydzielanie dla maszyn typu topspin (tylko z sumawarstw = 1)
for machine in machines_data_t:
    tasks_t = assign_tasks_to_machine(machine, tasks_t, 'TopSpin')

# Przydzielanie dla maszyn typu cutter
for machine in machines_data_k:
    tasks_k = assign_tasks_to_machine(machine, tasks_k, 'Cutter')

if tasks_t.empty:
    tasks_k = tasks_k.sort_values(by='SumaWarstw')
    for machine in machines_data_t:
        tasks_k = assign_tasks_to_machine(machine, tasks_k, 'Topspin')

```

Rysunek 9 Przedstawiający sortowanie i zastosowany algorytm.

9. Zapisanie wyników do pliku

Stworzenie raportu w formie arkusza kalkulacyjnego jest ważnym krokiem. W tym konkretnym przypadku zostaje stworzony raport na podstawie przypisanych zadań maszynom i zapisuje je w dedykowanych arkuszach w jednym pliku Excel. Każdy arkusz jest przypisany dla konkretnego urządzenia, a jego nazwa odzwierciedla jej typ oraz numer. W raporcie dla każdego przyrządu uwzględnione są informacje takie jak kod modelu, suma warstw, szczegóły zadania, metry tkaniny oraz przewidywany czas realizacji. Co więcej, każdy arkusz jest dopełniony sumarycznym wierszem, podający łączną ilość metrów tkaniny oraz całkowity czas realizacji dla poszczególnych maszyn.

Ostatni krok skryptu odnosi się do zleceń, które nie zostały przypisane do żadnego urządzenia. Te zadania są zapisywane w osobnym arkuszu o nazwie "Nieprzypisane_zlecenia", aby ułatwić analizę i potencjalne przyszłe przypisanie. Taki sposób organizacji danych w raporcie ułatwia szybkie zrozumienie stanu zadań i ich dystrybucji między maszynami, a także identyfikację zleceń, które jeszcze oczekują na przypisanie.⁷

⁷ F. Zumstein, Python i Excel. Helion 2022, ISBN: 978-83-283-8287-9


```
# Zapisywanie do plików Excel
with pd.ExcelWriter("C:/Desktop/raport.xlsx") as writer:
    for machine in machines_data_k + machines_data_t:
        machine_number = machine['machine_number']
        machine_type = 'Cutter' if machine in machines_data_k else 'TopSpin'
        task_time_column = 'CzasCutter' if machine_type == 'Cutter' else 'CzasTopSpin'

        machine_tasks = machine.get('assigned_tasks', []) # Pobieramy przypisane zadania dla maszyny
        if machine_tasks:
            machine_tasks_df = pd.DataFrame(machine_tasks)
            machine_tasks_df = machine_tasks_df[
                ['KodModelu', 'SumaWarstw', 'Details', 'MetryTkaniny', task_time_column]]

            # Mnożymy metrykę tkaniny przez sumę warstw
            machine_tasks_df['MetryTkaniny'] *= machine_tasks_df['SumaWarstw']

            # Dodajemy wiersz sumujący dla kolumn 'metrytkaniny' i 'Czas realizacji'
            sum_row = pd.DataFrame({
                'KodModelu': ['Suma'],
                'MetryTkaniny': [machine_tasks_df['MetryTkaniny'].sum()],
                task_time_column: [machine_tasks_df[task_time_column].sum()]
            })
            machine_tasks_df = pd.concat([machine_tasks_df, sum_row], ignore_index=True)

            machine_tasks_df.rename(columns={task_time_column: 'CzasRealizacji(min)'}, inplace=True)
            machine_tasks_df.to_excel(writer, sheet_name=f"Maszyna-{machine_type}-{machine_number}",
                                     index=False)

# Zapis nieprzypisanych zleceń
unassigned_tasks = pd.concat([tasks_k, tasks_t])
# print(f"Liczba nieprzypisanych zadań przed zapisaniem do Excela: {len(unassigned_tasks)}")
if not unassigned_tasks.empty:
    unassigned_tasks[['KodModelu', 'SumaWarstw', 'Details', 'MetryTkaniny']].to_excel(
        writer, sheet_name="Nieprzypisane-zlecenia", index=False)
```

Rysunek 10 Przedstawiający zapisanie wyników do pliku Excel.

10. Wyniki

Poniżej wyniki opisanego programu, który przydzielił zadania do dwóch typów maszyn: TopSpin oraz Cutter. Najpierw zlecenia o wartości 'SumaWarstw' równej 1 zostały przypisane do urządzeń rodzaju TopSpin. Następnie pozostałe zadania, zaczynając od tych z najwyższą wartością 'SumaWarstw', zostały przydzielone do przyrządów typu Cutter. W trakcie tego procesu dostępność maszyn Cutter została wyczerpana, a urządzenia rodzaju TopSpin miały jeszcze dostępne miejsce, toteż zadania zostały dalej przypisywane do tego typu, ale w odwrotnej kolejności - od najmniejszej wartości 'SumaWarstw'. Program zatrzymał przydzielanie zadań, gdy wszystkie maszyny zostały pełne (alternatywnie kończy proces, gdy wszystkie zadania zostały już przypisane).

KodModelu	SumaWarstw	Details	MetryTkaniny	CzasRealizacji(min)
1	20	Nr.sys:6	255,4	134,6027027
2	19	Nr.sys:6	141,17	76,30810811
3	18	Nr.sys:6	34,56	19,2
4	18	Nr.sys:6	20,34	11,3
5	18	Nr.sys:6	128,34	71,3
6	18	Nr.sys:6	21,24	11,8
7	17	Nr.sys:6	81,26	46,50810811
8	17	Nr.sys:6	40,8	23,35135135
9	17	Nr.sys:6	27,2	15,56756757
10	16	Nr.sys:6	40,8	24,12162162
11	16	Nr.sys:6	141,12	83,43243243
12	14	Nr.sys:6	5,18	3,3
13	13	Nr.sys:6	4,42	2,940540541
14	13	Nr.sys:6	13,78	9,167567568
15	12	Nr.sys:6	4,08	2,848648649
16	12	Nr.sys:6	4,08	2,848648649
Suma			963,77	538,5972973

Rysunek 11 Przedstawiający wyniki z zadaniami przypisanymi do jednej z maszyn.

IV. PROBLEMY TECHNICZNE

1. Wyznaczenie wydajności maszyny

Problem z określeniem współczynnika wydajności był złożony ze względu na wiele czynników, które mogą wpłynąć na jego dokładność. Jednym z głównych problemów jest duża ilość zmiennych, które muszą zostać uwzględnione podczas analizy. W tym konkretnym przypadku trzeba było określić ilość wykroi, które były wykonywane na urządzeniach typu Cutter, a w związku z tym, że niektóre układy są przypisane do modeli o różnych nazwach oraz że nie ma spisane zbioru na ten temat koniecznym była weryfikacja każdego układu osobno. Dodatkowo nie wszystkie czynności przez pomyłki pracowników nie są zaraportowane lub nie odbywa się to w czasie rzeczywistym, co też znacznie spowalnia proces analizy. Cały proces liczenia trzeba było przeprowadzić od podstaw, ponieważ dokumentacja technologiczna nie dostarczała wszystkich niezbędnych elementów do programu. Część z nich można było zidentyfikować jedynie na drodze doświadczenia empirycznego, co wymagało poświęcenia dużej ilości czasu na hali produkcyjnej i rozmowy z operatorami maszyn.

2. Tworzenie zapytań do bazy

Pozyskiwanie danych z rozbudowanej bazy danych, składającej się z 4000 tabel i 1300 procedur, przy użyciu skryptów SQL stanowiło do pokonania szereg wyzwań technicznych.

Po pierwsze, sama skala bazy danych czyni jej analizę i nawigację wyjątkowo złożoną. Znalezienie odpowiednich tabel lub procedur w tak ogromnej strukturze było czasochłonne, a błędy w zapytaniach prowadziły do niekompletnych lub błędnych wyników, które na bieżąco należało sprawdzać i poprawiać.

Po drugie, wydajność zapytań SQL w tak dużej bazie danych była problematyczna, toteż korzystanie z niektórych procedur było nieefektywne. Złożone zapytania, zwłaszcza te, które łączą wiele tabel znacząco obciążają komputer i prowadzą do opóźnień w odpowiedziach. Optymalizacja takich zapytań, by działały efektywnie w tak dużym środowisku, jest kluczem do skutecznego korzystania z bazy.

Po trzecie, wysoka liczba procedur powodowała, że nie wszystkie z nich są odpowiednio udokumentowane. Bez odpowiedniej dokumentacji było trudno zrozumieć, jakie dane są zwracane przez poszczególne procedury i jak je należy interpretować.

Kolejnym wyzwaniem było zapewnienie spójności danych. W tak dużej bazie istniało ryzyko, że niektóre tabele mogą zawierać powtarzające się, sprzeczne lub przestarzałe informacje, toteż wszystkie dane zostały porównane z końcowymi wynikami w naszym systemie w celu uzyskania wiarygodnych wyników z zapytań SQL.

Podsumowując, pozyskiwanie danych z tak rozbudowanej bazy danych za pomocą skryptów SQL wiąże się z wieloma wyzwaniami technicznymi, od nawigacji po złożonej strukturze, poprzez optymalizację wydajności zapytań, aż po zarządzanie jakością danych. Wymaga to nie tylko umiejętności technicznych, ale także dogłębnego zrozumienia struktury bazy i danych, które zawiera.

3. Poprawa jakości danych

Unormowanie danych w taki sposób, aby wszystkie były w jednym wzorcu, dając w efekcie możliwość wykonywania na nich kolejnych operacji to następne wyzwanie techniczne i koncektualne wymagające dokładnego rozpoznania również na hali produkcyjnej.

Po pierwsze należało unormować kody modeli usuwając wszystkie spacje, ponieważ w zależności od tabeli oraz osoby wprowadzającej kod modelu zdarzały się sytuacje, gdzie na końcu wierszy zostały dodane niepotrzebne spacje nie zawsze w tej samej ilości. Usunięcie ich dało możliwość stworzenia połączenia między plikami w programie.

Po drugie zostały usunięte 10-te znaki w kodzie modelu, gdy ten miał dodatkową czynność do wykonania, a nie wpływa ona na układ cięcia, toteż żeby wykroje były bardziej zoptymalizowane należy łączyć te same modele z i bez tego rozszerzenia podczas jednego cięcia.

Po trzecie usunięcie wszystkich wzorów, reklamacji itd., które nie są znormowane, w efekcie czego nie można ustalić ilości materiałów potrzebnych na ich zrealizowanie, a co za tym idzie czasu potrzebnego na realizację.

Po czwarte modele mają różne kombinacje tkaninowe, których może być 5, a każda ma inaczej rozpisane normy materiałowe, toteż należało zastosować funkcję, która będzie odwoływać się tylko do największej wartości, z której tworzona jest norma czasowa. Różnice między kombinacjami zostały pominięte ze względu na brak szczegółowych informacji w systemie. Różnice te są na tyle niewielkie, że nie mają większego wpływu na algorytm.

4. Nadpisywanie pliku z przydzielonymi zadaniami

W niektórych sytuacjach algorytm, mimo że stworzył plik dla maszyny np. Cutter1 i przydzielił zadania wypełniając całkowicie tę maszynę zadaniami, to za nim została dodana linia kodu która sprawdza dostępny czas pracy (i rozwiązuje problem):

„if machine_data['available_minutes'] <= 0: return tasks”

usuwał już wcześniej przydzielone zadania i wypełniał maszynę nowymi zleceniami.

Innymi słowy, ten fragment kodu służy jako wczesna weryfikacja warunku, aby nie zostały przypisywane zadania maszynie, która nie ma dostępnego czasu, mimo że logicznie nie powinno to mieć miejsca. Ponadto ma też zaletę, bo uniemożliwia, żeby zadania byłyby niepotrzebnie przetwarzane, co jest marnotrawstwem zasobów. Właśnie dlatego wczesne wyjście z funkcji, gdy dostępny czas maszyny wynosi zero lub mniej, jest ważne dla zachowania integralności i prawidłowego działania algorytmu.

V. ZAKOŃCZENIE

1. Podsumowanie projektu

Podsumowując realizację projektu, udało nam się opracować algorytm odpowiadający bieżącym wymaganiom przedsiębiorstwa w zakresie wsparcia procesu przydzielania zadań. Naszym celem było:

- Minimalizacja presji na osoby odpowiedzialne za przydział zadań podczas jednoczesnego otrzymywania dużej liczby zleceń w formie papierowej (do paru tysięcy dziennie). Dzięki skutecznemu przydzielaniu zadań przez algorytm w pierwszych godzinach po otrzymaniu zleceń, tworzy się bufor czasowy dla koordynatorów, aby skierować swoją uwagę na trudniejsze zadania, które wynikają z niedostatecznych danych w systemie (jedno z potencjalnych kierunków dalszego rozwoju);
- Ograniczenie pomyłek spowodowanych czynnikiem ludzkim, jak na przykład zgubienie zleceń. Brygadzistka ma teraz możliwość weryfikacji ilości zleceń w systemie elektronicznym w stosunku do ich fizycznej liczby;
- Ze względu na właściwości algorytmu i przyjazne wizualnie zaprezentowanie czasu potrzebnego na wykonanie zleceń możliwym jest również wykluczenie części błędów optymalizacyjnych pracowników przydzielających zadania.

2. Kierunki rozwoju i dalszych badań

Program jest i będzie rozwijany w najbliższym czasie, ponieważ nie wszystkie elementy potrzebne do wykonania w pełni rozwiniętego programu zostały zrealizowane. Pomimo że proces analizy danych i procesów biznesowych w tak dużym przedsiębiorstwie to tak naprawdę praca bez końca, zostały już zaplanowane kolejne kroki do realizacji, a niektóre z nich są już realizowane.⁸

Zacznijmy zatem od rzeczy w trakcie realizacji:

- a) Rozpoczęte zostały prace nad stworzeniem realnych norm dla każdego typu maszyny z osobna, które uwzględniają zdecydowanie większą ilość zmiennych niż dotychczas w efekcie, czego wyniki działania programu będą jeszcze bardziej optymalne. W chwili obecnej normy bazują na zapotrzebowaniu materiałów, a nie

⁸ F. Provost, T. Fawcett, Analiza danych w biznesie. Helion 2019, ISBN: 978-83-283-5833-1

na rzeczywistej pracochłonności, która wynika z poziomu skomplikowania mebla. Do tego również nie został jeszcze wykonany podział na dwa typy maszyn, które charakteryzują się różną wydajnością oraz czasem potrzebnym na przygotowanie środowiska do wykonywania kolejnych zadań.

- b) Kolejną rzeczą, której została rozpoczęta normalizacja to nazwy układów. W chwili obecnej nie ma informacji w systemie, które modele mają ten sam układ, bo mimo różnych nazw mogą mieć ten sam, co uniemożliwia z poziomu programu agregowania układów do jednego zadania. W podsumowaniu projektu pisałem o „kłopotliwych zleceniach” i właśnie w tym punkcie to wyjaśniam. Sposób agregowania układów odbywa się przez brygadzystki, które dzięki swojemu wieloletniemu doświadczeniu mają wiedzę na ten temat, ale niestety nie została ona spisana ani wprowadzona do bazy.
- c) Ostatnim elementem jest priorytetowość wykroi tzn. w systemie nie ma konkretnych dat w których należy wykonać daną czynność i zdarza się tak, że zlecenia z datą późniejszą muszą zostać zrealizowane przed tymi z datą wcześniejszą. Drugim aspektem z tym powiązany jest brak informacji w systemie o statusie wykonania danego zadania. Rozwiązaniem tego problemu może być wykorzystanie wprowadzanego aktualnie modułu zawierającego informację o statusie realizacji danej czynności, a to w połączeniu z rzetelnym skanowaniem czynności przez operatorów i stworzenie „deadline” na realizację zadań może dać możliwość stworzenia funkcji wyboru zakresu dat z których zostałyby pobierane zlecenia z wykluczeniem już zrealizowanych z czego największy priorytet miały te z najkrótszym okresem do przedawnienia.

Są też elementy, które zostały rozpoznane, ale z uwagi na sukcesywne rozrastanie się projektu nie zostały jeszcze rozpoczęte:

- a) Po pierwsze magazyn tkanin musiałby mieć odświeżane stany możliwie jak najczęściej, a najlepiej być prowadzony w czasie rzeczywistym, a nie tak jak teraz raz w tygodniu. Pozwoliłoby to stworzyć połączenie między programem, a magazynem, w efekcie którego ewentualne zlecenia na które nie ma odpowiedniej tkaniny zostałyby przesuwane na koniec kolejności bądź trafiały do bufora, gdzie czekałyby na uzupełnienie stanów.
- b) Po drugie każda tkanina powinna mieć swój „współczynnik twardości”. Obecnie została przyjęta zasada, że na urządzeniach typu Cutter można ciąć maksymalnie 20

warstw, ale są sytuacje, gdy ta liczba się zwiększa lub zmniejsza w zależności od tkaniny, którą kroimy. W sytuacji, gdy będzie określony taki współczynnik sumowałyby się go aż do umownej granicy, której nie można by było przekroczyć. Taki współczynnik byłby też jasną informacją jakich tkanin nie można ciąć na maszynach typu TopSpin, ponieważ są to urządzenia o niższej zdolności, toteż graniczny punkt zostałby ustalony niżej niż dla maszyn typu Cutter i najtwardsze tkaniny nie byłyby przypisywane na to urządzenie tak jak ma to miejsce w rzeczywistości.

- c) Ostatnim oczywiście na ten moment analizy punktem jest określenie ścisłych zasad dla wyrobów specjalnych, które ze względu na swoją specyfikę w teorii mogą być przypisywane tylko do konkretnych typów maszyn. Zasady te muszą być jednoznaczne i przestrzegane, żeby algorytm znajdował coraz bardziej optymalne rozwiązania.

Jak widać w projekcie jest duży potencjał i mam nadzieję, że będzie cały czas rozwijany. Wszystkie kolejne kroki będą dokumentowane tak, żeby praca doczekała się kontynuacji w innych artykułach czy pracy magisterskiej. Możliwym jest, że wraz z rozwojem programu zostanie zmieniony algorytm, jednak na ten moment w tych okolicznościach wybór aktualnego uważam za najodpowiedniejszy.

VI. LITERATURA

1. H. Tysza, Excel Solver w praktyce. Helion 2021, ISBN: 978-83-283-8399-9
2. P. Wróblewski, Algorytmy w Pythonie. Helion 2022, ISBN: 978-83-8322-161-8
3. M. Gągolewski, M. Bartoszek, A. Cena, Przetwarzanie i analiza danych w języku Python. Wydanie I, PWN 2016, ISBN: 978-83-01-18940-2
4. Excel [dostęp 14.09.2023], <https://support.microsoft.com/excel>
5. Firebirdsql [dostęp 15.09.2023],
https://firebirdsql.org/file/documentation/drivers_documentation/python/fdb/getting-started.html
6. Python [dostęp 16.09.2023], <https://pl.python.org/docs/lib/built-in-funcs.html>
7. F. Zumstein, Python i Excel. Helion 2022, ISBN: 978-83-283-8287-9
8. F. Provost, T. Fawcett, Analiza danych w biznesie. Helion 2019, ISBN: 978-83-283-5833-1

VII. OŚWIADCZENIE AUTORA PRACY DYPLOMOWEJ

Breuszko Michał
nazwisko i imię
92548
nr albumu
Informatyka
kierunek studiów
Inżynierskie, niestacjonarne
typ studiów i forma kształcenia

OŚWIADCZENIE autora pracy dyplomowej *

Świadomy(a) odpowiedzialności prawnej oświadczam, że praca dyplomowa

Projekt i wykonanie programu optymalizującego linię produkcyjną

(tytuł pracy dyplomowej w języku polskim / języku pracy i języku polskim)

została wykonana samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że:

- 1) przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w uczelni;
- 2) drukowana wersja pracy dyplomowej jest identyczna z wprowadzoną do systemu APD wersją elektroniczną.

Michał Breuszko
(podpis studenta)

Bydgoszcz, dn. 25.09.2023

Wyrażam zgodę /nie wyrażam zgody** na udostępnienie przez Uniwersytet pracy dyplomowej dla potrzeb działalności badawczej i dydaktycznej.

Michał Breuszko
(podpis studenta)

Bydgoszcz, dn. 25.09.2023

*w przypadku zbiorowej pracy dyplomowej, dołącza się oświadczenia każdego ze współautorów pracy dyplomowej

** niepotrzebne skreślić