Algorytm Browna Łomnickiego,
szeregowanie zadań, środowisko Java,
harmonogramy, oprogramowanie

Maciej NOWAK[1]
Paweł NOWAK[1]

ALGORYTM BROWNA ŁOMNICKIEGO – PROPOZYCJA OPROGRAMOWANIA KOMPUTEROWEGO

W referacie przedstawiono problem szeregowania zadań i optymalizacji harmonogramów w celu zminimalizowania zbędnych przerw w pracy stosowanych maszyn i urządzeń. Metoda podziału i ograniczeń, na której opiera się opracowany w połowie dwudziestego wieku algorytm Browna Łomnickiego, może zostać zastosowana do tego celu. Autorzy prezentują próbą oprogramowania algorytmu, a także pewną koncepcję zminimalizowania niezbędnych iteracji prowadzących do otrzymania wyniku optymalnego. W referacie przedstawiono oprogramowanie, sposób jego działania oraz przykład praktyczny.

BROWN-ŁOMNICKI ALGORITHM – COMPUTER SOFTWARE PROPOSAL

This paper considers a machines and equipment work scheduling problem of minimizing the unnecessary pauses in their work. This important scheduling problem need to be solved in modern construction systems, and is well known to be intractable (i.e., NP-hard). Branch-and-bound algorithms were developed by Łomnicki and Brown in the middle of twentieth century. In order to improve the use of branch-and-bound algorithms authors present a new and simply software by which the calculations could be reduced to the minimum. Brown-Łomnicki Algorithm is presented and its adjustment for computer platform. Authors present also some screen shots, chosen information about the software and practical example.

1. WSTĘP

Programowanie całkowitoliczbowe, zwane inaczej optymalizacją dyskretną, pomaga w rozwiązywaniu zagadnień, w których wszystkie lub część zmiennych może przyjmować tylko wartości całkowite. W praktyce budowlanej takie sytuacje występują bardzo często, chociażby w przypadku optymalizacji wykorzystania niepodzielnych środków produkcji. Do takich środków można zaliczyć np. liczbę stanowisk załadowczych w magazynie, liczbę

[1] *Politechnika Warszawska, Wydział Inżynierii Lądowej, Zakład Inżynierii Produkcji i Zarządzania w Budownictwie, 00-637 Warszawa, Armii Ludowej 16, tel.: +48 22 2346515, fax.: +48 22 8257415, e-mail: p.nowak@il.pw.edu.pl, maciej.novak@gmail.com*

osób, które mają być skierowane do określonej pracy, liczbę maszyn budowlanych przeznaczonych do wykonania pewnego procesu budowlanego, itp.

Innym źródłem dyskretności modelu decyzyjnego może być jego kombinatoryczny charakter. W takich przypadkach rozwiązania problemu reprezentowane są przez permutację lub kombinację pewnego zbioru obiektów. Przykładem takiego zagadnienia w budownictwie jest m.in. optymalizacja harmonogramów, czyli wyznaczanie najkorzystniejszej kolejności wykonywania robót w ramach danej inwestycji czy np. tzw. problem komiwojażera, który polega na wyznaczeniu najkrótszej trasy o obwodzie zamkniętym, przebiegającej przez dane punkty.

2. ALGORYTM ŁOMNICKIEGO

Algorytm Łomnickiego opiera się na metodzie podziału i ograniczeń. Zaprojektowany został z myślą o jego wykorzystaniu przy ustalaniu kolejności obróbki wyrobów na maszynach produkcyjnych, jednak zasada jego działania pozwala zastosować go z powodzeniem w dziedzinie optymalizacji harmonogramów budowlanych.

Podstawowe założenia algorytmu to [9]:

1. Na każdej działce roboczej (j = 1, 2,..., n) pracują kolejno maszyny/brygady robocze (i = 1, 2,..., m, przy czym m = 3).
2. Każda maszyna w danej chwili może pracować tylko na jednej działce roboczej i na każdej działce roboczej w danej chwili może pracować tylko jedna maszyna.
3. Każda maszyna pracuje na działkach roboczych ustawionych w tej samej kolejności.
4. Na działkę roboczą j może być wprowadzona maszyna *i*, jeżeli maszyna ta ukończyła pracę na działce *j - 1* (dla j ≥ 2) oraz jeżeli na działce *j* ukończyła pracę maszyna *i - 1* (dla i ≥ 2).

Istotą algorytmu jest znalezienie takiej kolejności (permutacji) działek roboczych ($W = \{w_1, w_2,...,w_n\}$), która zapewni optimum funkcji celu.

Funkcję celu przedstawia następująca zależność [9]:

$$min: T = T_{mn}^{k}, \tag{1}$$

gdzie

T_{mn}^{k} - oznacza najwcześniejszy termin zakończenia prac na działce n przez maszynę m, czyli termin zakończenia wszystkich planowanych robót.

Ponieważ algorytm wykorzystuje metodę podziału i ograniczeń, jego najważniejszymi elementami są funkcja ograniczająca oraz określenie sposobu podziału zbioru rozwiązań.

Zbiór wszystkich rozwiązań – G to n! permutacji elementów zbioru działek roboczych. Algorytm przeszukuje go dzieląc na podzbiory, gdzie określone jest ustawienie *r* pierwszych działek (2):

$$G(W_k), \tag{2}$$

gdzie:

$W_k = \{w_1, w_2,...,w_r\}$,

$\{w_1, w_2,...,w_r\} \subset j = \{1, 2,..., n\}$.

Podzbiór taki zawiera (n – r)! rozwiązań, a wyznaczona dla niego wartość funkcji ograniczającej (3) określa czy będzie on podlegał dalszemu podziałowi.

$$\xi(G(W_k)) = max\{\xi_1, \xi_2, \xi_3\}, \tag{3}$$

gdzie:

$$\xi_1 = \sum_{j=1}^{n} t_{1j} + \min_{j \notin W_i}\{t_{2j} + t_{3j}\}, \quad \xi_2 = T_{2r}^k + \sum_{j \notin W_i} t_{2j} + \min_{j \notin W_i}\{t_{3j}\}, \quad \xi_3 = T_{3r}^k + \sum_{j \notin W_i} t_{3j},$$

T_{2r}^k, T_{3r}^k - wyznaczone na podstawie wzorów rekurencyjnych,

r = |Wk|.

Algorytm dzieli podzbiory o najmniejszej wartości funkcji ograniczającej do momentu znalezienia jednego lub więcej rozwiązań dopuszczalnych, dla których spełniony jest warunek:

$$\xi(G(W_k)) \le \xi_{min}, \tag{4}$$

gdzie: |Wk| = n – 1.

3. ALGORYTM BROWNA ŁOMNICKIEGO

Zastosowanie algorytmu Łomnickiego jest bardzo ograniczone ze względu na jedno z podstawowych jego założeń, które pozwala wykorzystać tę metodę do optymalizacji pracy jedynie trzech maszyn na dowolnej liczbie działek roboczych.

Udoskonaloną metodą jest algorytm Browna-Łomnickiego, który może być używany we wszystkich przypadkach gdzie liczba maszyn m spełnia warunek m ≥ 3. Sposób przeszukiwania zbioru rozwiązań pozostał niezmieniony, nową postać przyjęła natomiast funkcja ograniczająca

$$\xi(G(W_k)) = max\{\xi_1, \xi_2, \ldots, \xi_m\}, \tag{5}$$

gdzie:

$$\xi_i = T_{ir}^k + \sum_{j \notin W_i} t_{ij} + \min_{j \notin W_i} \sum_{l=i+1}^{m} t_{lj}, \tag{6}$$

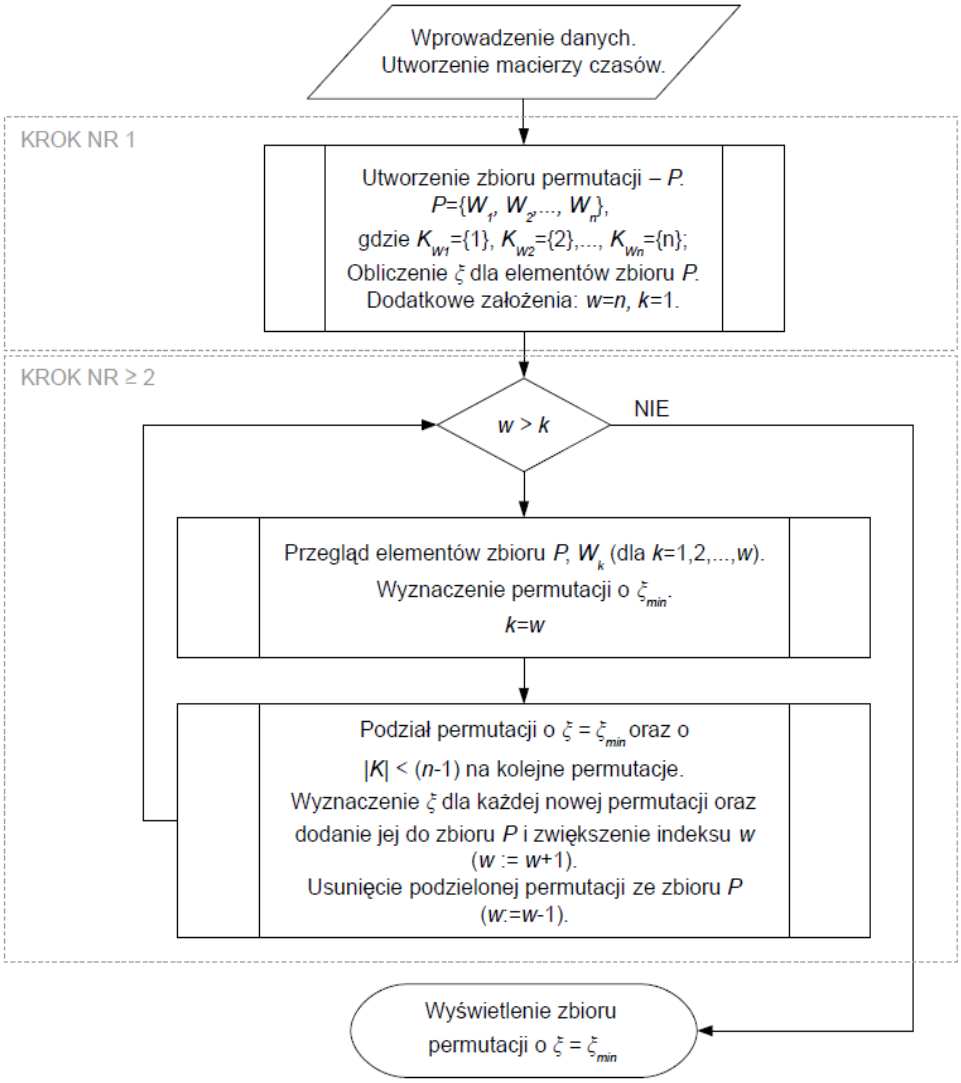
dla j = 1, 2,..., n; i = 1, 2,...,m-1,

oraz

$$\xi_i = T_{ir}^k + \sum_{j \notin W_i} t_{ij}, \tag{7}$$

dla j = 1, 2,..., n; i = m.

Algorytm Browna-Łomnickiego, aby mógł być wykorzystany w programie komputerowym, musiał zostać przedstawiony w formie schematu blokowego. Jego podstawowa postać została pokazana na rys. 1.



Rys. 1. Schemat blokowy algorytmu Browna-Łomnickiego

Koncepcja programu uzależniona jest w znacznej mierze od wybranego języka programowania. JAVA należy do grupy języków programowania obiektowego, dlatego koncepcja programu również musiała opierać się na obiektach. Analiza przebiegu obliczeń w algorytmie Browna-Łomnickiego doprowadziła do wniosku, że ich podstawą są kolejne podzbiory zbioru wszystkich możliwych rozwiązań (permutacji), dla których wyliczana jest wartość funkcji ograniczającej. Tworzą one kolejne gałęzie drzewa metody podziału i ograniczeń. Każda gałąź określona jest poprzez założoną kolejność działek roboczych oraz

wartość ξ. Te dwa parametry składają się również na główny obiekt programu nazwany „permutacją".

$$W = [K, \xi_W] \tag{8}$$

W pierwszym kroku obliczeń, zgodnie z Algorytmem Browna-Łomnickiego, zbiór wszystkich możliwych rozwiązań (kolejności działek roboczych) zostaje podzielony na n podzbiorów, dla których zostaje wyznaczona wartość funkcji ograniczającej - ξ. Elementy zbioru P – permutacje otrzymują następującą postać:

$$W_1 = [\{1\},\xi_{W1}],\ W_2 = [\{2\},\xi_{W2}],...,\ W_n = [\{n\},\xi_{Wn}]. \tag{9}$$

Następne kroki przebiegają wg schematu zawartego w kolejnym bloku obliczeniowym. Zbiór P przeszukiwany jest w celu znalezienia permutacji o ξ_{min}, które o ile to możliwe podlegają dalszemu podziałowi. Warunkiem podziału permutacji jest spełnienie zależności:

$$|K| < n-1, \tag{10}$$

gdzie:

|K| – liczba elementów zbioru określającego założoną początkową kolejność działek roboczych.

Permutacja o założonej kolejności *n-1* działek roboczych jednoznacznie określa kolejność wszystkich działek roboczych, więc nie podlega podziałowi.

Powyższe czynności są powtarzane dopóki spełniona jest zależność:

$$w > k, \tag{11}$$

gdzie:

w = |P| – liczba permutacji w zbiorze P,

k – liczba permutacji w zbiorze *P* dla poprzedniego kroku.

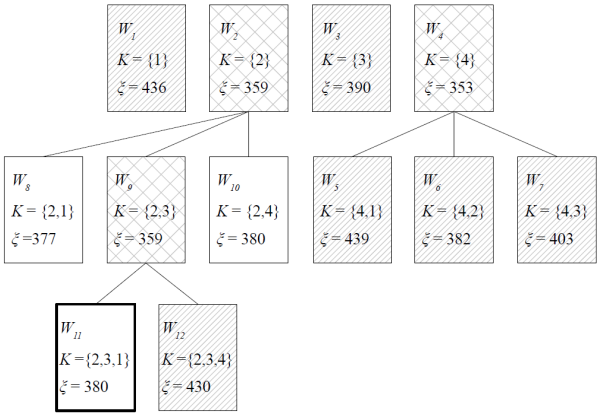
Warunek (10) przestaje być spełniony, kiedy w ramach poprzedniego kroku obliczeń do zbioru permutacji P nie zostały dodane kolejne elementy. Oznacza to, że wszystkie permutacje o wartości funkcji ograniczającej równej ξ_{min} są niepodzielne i tworzą ostateczny zbiór rozwiązań. W takiej sytuacji algorytm przechodzi do ostatniego zadania – prezentacji rozwiązania.

4. OPTYMALIZACJA PRACY PROGRAMU

W dobie szybkiego rozwoju technologii i komputerów wydawać by się mogło, że rozwiązywanie problemów optymalizacyjnych takim algorytmem jak Browna-Łomnickiego, nie powinno napotkać żadnych przeszkód wydajnościowych. Jednak problemy optymalizacyjne są zagadnieniami bardzo wymagającymi, jeżeli chodzi o liczbę wykonywanych obliczeń i ilość przechowywanych danych. Jak się okazało podczas prac nad programem, w przypadku wybranej metody podziału i ograniczeń, zwłaszcza ten drugi czynnik ma znaczący wpływ na możliwości aplikacji. Ścisłe rozwiązanie zagadnienia wymaga n! wyliczeń ξ, komputer musi również być w stanie przechować te wyniki wraz z

przypisaniem im odpowiednich kolejności działek roboczych. Dla 4 działek należy przeprowadzić i przechować 24 obliczenia ale dla 14 - liczba ta wynosi: 87 178 291 200.

Algorytm Browna-Łomnickiego w znakomitej większości przypadków znacznie zmniejsza liczbę kalkulacji, prowadząc również do rozwiązania ścisłego. Mimo to, w przypadku liczby działek roboczych większej niż 10, program zaprojektowany zgodnie z podstawową wersją schematu blokowego, często nie był w stanie dotrzeć do ostatniego kroku obliczeń. Wprowadzone zostało szereg mniejszych poprawek oraz dwie dość istotne zmiany. Pierwszą z nich było uruchomienie mechanizmu pozbywającego się części przechowywanych danych, która nie miała już znaczenia w dalszym procesie obliczeniowym. Algorytm dzieląc kolejne zbiory w pewnym momencie odkrywa pierwsze potencjalne rozwiązanie – permutację o |K| = n – 1. Wartość ξ wyliczona dla tej permutacji stanowi górne ograniczenie wartości funkcji ograniczającej dla pozostałych permutacji - ξ_{Pmax}. Wszystkie permutacje spełniające warunek: $\xi > \xi_{Pmax}$, mogą zostać usunięte ze zbioru P, ponieważ nie mogą prowadzić do rozwiązania optymalnego. Czas trwania robót dla podzbiorów tych permutacji byłby dłuższy niż w przypadku już znalezionych potencjalnych rozwiązań. Zależność tą obrazuje poniższy schemat (rys. 2).



Rys. 2. Optymalizacja - usuwanie zbędnych permutacji

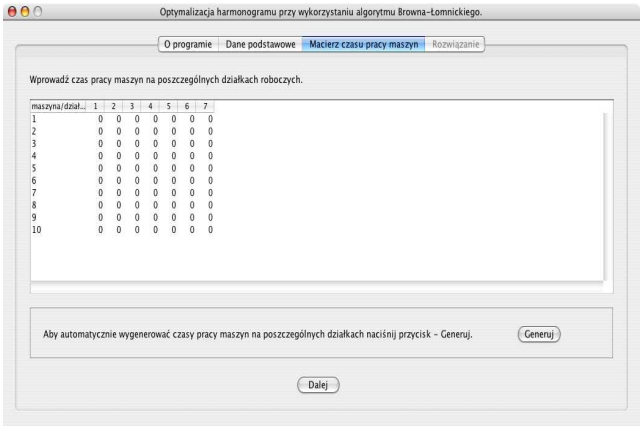
Permutacje W_4, W_2, W_9, są kolejno usuwane w konsekwencji podziału na podzbiory. Podział W_9 prowadzi do znalezienia dwóch pierwszych zbiorów o |K| = n - 1. Permutacja W_{11} staje się pierwszym potencjalnym rozwiązaniem, a jej wartość ξ staje się nowym ξ_{Pmax}.

5. PREZENTACJA OPROGRAMOWANIA Z PRZYKŁADEM LICZBOWYM

5.1. Oprogramowanie

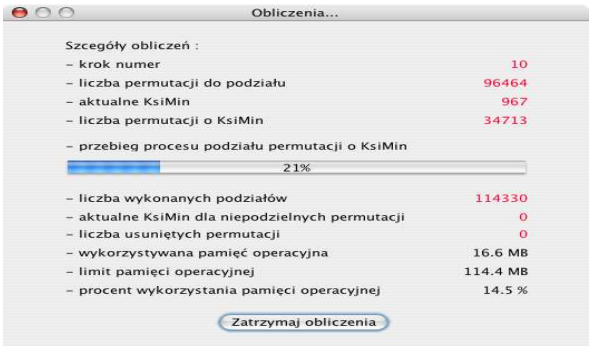
Program powstał w technologii JAVA i dzięki wyborze tej platformy z aplikacji mogą skorzystać użytkownicy większości komputerów i systemów operacyjnych. Jedynym koniecznym do spełnienia warunkiem przed uruchomieniem programu jest posiadanie zainstalowanego w systemie środowiska JRE (Java Runtime Environment). Pierwszym krokiem po rozpoczęciu pracy z programem jest oczywiście wprowadzenie danych.

Zadanie to zostało podzielone na dwa etapy. Na początku użytkownik pytany jest o dane podstawowe – liczbę maszyn oraz działek roboczych. Wpisywane wartości muszą należeć do zbioru liczb całkowitych oraz być większe od zera. Jeżeli powyższe warunki nie są spełnione, przy próbie przejścia do kolejnego okna zostanie wyświetlony komunikat o błędzie.

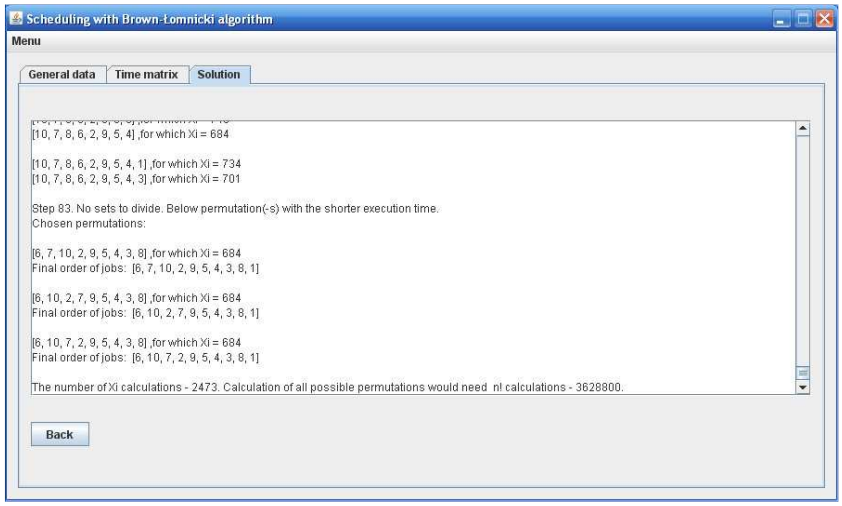


Rys. 3. Macierz czasu pracy maszyn

W tym widoku należy zdefiniować czasy pracy maszyn na poszczególnych działkach roboczych. Dla potrzeb dydaktycznych została również udostępniona opcja „Generuj", która uzupełnia macierz liczbami pseudolosowymi, wygenerowanymi przez komputer. Wprowadzane wartości muszą należeć do zbioru liczb całkowitych oraz zawierać się w przedziale od 0 do 100. W przypadku podania niewłaściwej wartości pojawi się komunikat o błędzie.



Rys. 4. Podgląd obliczeń



Rys. 5. Zestawienie wyników

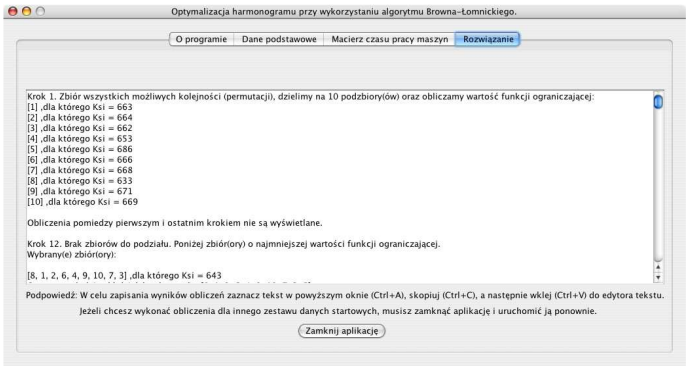
5.1. Przykład liczbowy

W celu przedstawienia praktycznego wykorzystania programu autorzy zaproponowali optymalizacje harmonogramu dla projektu budowy drogi lokalnej. Zaproponowano podział drogi na 10 sekcji (różniących się długością w dopasowaniu do elementów koniecznych do wybudowania, np. zjazdów do lokalnych zabudowań). Tabela 1 zawiera zestawienie czasów pracy poszczególnych ekip na 10 budowanych sekcjach (działkach roboczych). Konieczne do przeprowadzenia prace są następujące:

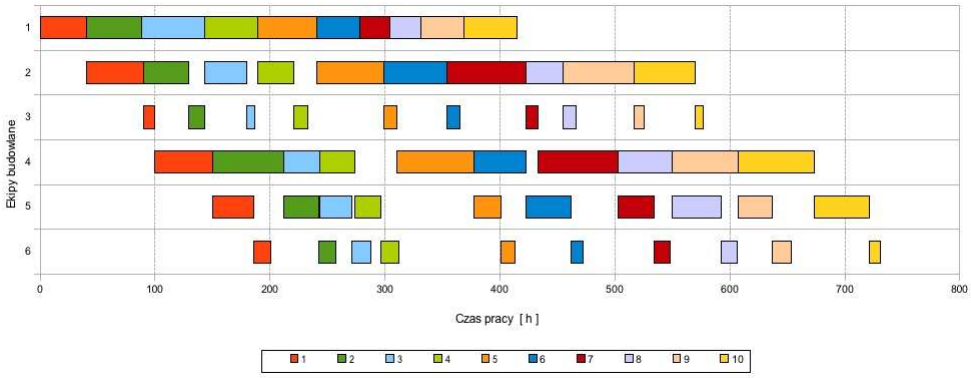
1. roboty przygotowawcze i roboty ziemne,
2. wykonanie podbudowy jezdni z gruntu stabilizowanego cementem,
3. wykonanie krawężników na ławie betonowej i wykonanie poboczy,
4. wykonanie nawierzchni jezdni,
5. wykonanie podbudowy chodnika i ścieżki rowerowej – wykonanie nawierzchni chodnika i ścieżki rowerowej,
6. roboty związane z organizacją ruchu (oznakowanie pionowe, oznakowanie poziome, elementy bezpieczeństwa ruchu) i roboty wykończeniowe.

Tab. 1. Zestawienie czasów realizacji poszczególnych sekcji

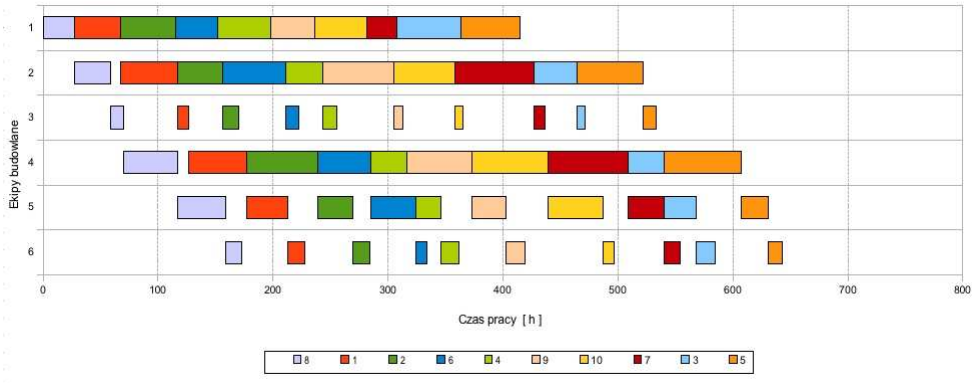
Brygady robocze	Sekcja drogi (1-10) / dni robocze									
	1	2	3	4	5	6	7	8	9	10
1	40	48	55	46	52	37	26	27	38	46
2	50	39	37	32	58	55	69	32	62	53
3	10	14	7	12	11	11	10	11	8	7
4	50	62	31	31	67	46	70	47	57	66
5	36	30	28	22	24	39	31	42	30	48
6	15	15	17	16	12	10	14	14	16	10



Rys. 6. Rozwiązanie problemu w oknie programu (Java)



Rys. 7. Harmonogram prac (przed optymalizacją – 731 dni)



Rys. 8. Harmonogram prac (po optymalizacji – 643 dni)

6. WNIOSKI

Na podstawie pracy programu można wyprowadzić kilka wniosków praktycznych:
- dodatkowa modernizacja programu mogłaby być związana z możliwością deterministycznego, bardziej praktycznego zakładania niektórych kolejności wykonania poszczególnych działek, prowadziłoby to do kolejnego ograniczenia liczby iteracji,
- program ma problemy przy mało zróżnicowanych czasach pracy maszyn (zbyt dużo rozwiązań optymalnych); dla identycznych n działek jest potrzebnych $\sum_{x=1}^{x=n-1} \frac{n!}{x!}$ obliczeń, czyli $\sum_{x=2}^{x=n-1} \frac{n!}{x!}$ więcej obliczeń niż w przypadku rozwiązania ścisłego
- w przypadku wyraźnych różnic w nakładach pracy dla poszczególnych maszyn cały harmonogram może determinować czas pracy jednej lub kilku maszyn/brygad roboczych (tych o najdłuższym czasie pracy).

7. BIBLIOGRAFIA

[1] Brown J.W., Churchill R.V.: Complex variables and applications 7th ed., Boston, McGraw-Hill, 2003.

[2] Clausen, J. L. Troa: *Do Inherently Sequential Branch-and-Bound Algorithms Exist*?, Parallel Processing Letters 4, 1-2, p. 3 -13, 1994.

[3] Dominguez-Marin P.: *Combinatorial Optimization*, Kluwer Academic Publications, 2003.

[4] Eckel B.: *Thinking in Java*. IV Edition. Wydawnictwo Helion, Gliwice, 2006.

[5] Graham I.: Metody obiektowe w teorii i w praktyce, WNT, Warszawa, 2004.

[6] Hall M., Brown L., Chaikin Y.: *Core Servlets and Javaserver Pages. Vol. 2, Advanced technologies*, Gliwice, Wydawnictwo Helion, 2009.

[7] Harris S., Ross J.: Algorytmy. Od podstaw. Wydawnictwo Helion, Gliwice, 2006.

[8] Horowitz E., S. Sahni, R. Sanguthevar: *Fundamentals of Computer Algorithms*, Delhi: Universities Press (India) Limited, 2008.

[9] Jaworski K. M.: *Methodology of building planning and realisation*, Wydawnictwo Naukowe PWN, Warsaw, Poland, 1999.

[10] Janicki A., Izydorczyk A.: *Komputerowe metody w modelowaniu stochastycznym. Modele w finansach, technice i biologii. Algorytmy numeryczne i statystyczne. Symulacja i wizualizacja zjawisk losowych*, WNT, Warszawa, 2001.

[11] Sahni S.: *Data Structures, Algorithms, and Applications in Java* (second Edition), Delhi: Universities Press, 2005.

[12] Schildt H.: Java. Kompendium programisty, Helion, Warszawa, 2005.

[13] Stawowy A., Mazur Z.: *Heurystyczne algorytmy szeregowania zadań produkcyjnych i grupowania wyrobów, Nowoczesne metody zarządzania produkcją*, pod red. Z.Martyniaka, Wydział Zarządzania AGH, Kraków, 1996.

[14] Tyagi S., McCammon K., Vorburger M., Bobzin H.: *Java Data Objects*, Wydawnictwo Helion, Warszawa, 2004.