*Studia i Materiały Informatyki Stosowanej, Tom 2, Nr 2, 2010*
*str. 59-66*

# ZASTOSOWANIE ALGORYTMÓW POSZUKIWANIA Z TABU DO OPTYMALIZACJI UKŁADANIA PLANU ZAJĘĆ

**Eliza Witczak**
*Uniwersytet Kazimierza Wielkiego*
*Instytut Techniki*
*ul. Chodkiewicza 30, 85-064 Bydgoszcz*
*e-mail: elizawitczak@go2.pl*

**Streszczenie:** *TS jest mataheurystyką szukającą rozwiązania problemu poprzez nadzorowanie innych procedur heurystycznych lokalnego przeszukiwania, w celu eksploracji przestrzeni rozwiązań poza lokalne optimum. Proces przeszukiwania przestrzeni rozwiązań jest koordynowany za pomocą strategii opartych na mechanizmach pamięci, będących cechą charakterystyczną algorytmu TS. Niniejszy artykuł opisuje zastosowanie metody TS w procesie optymalizacji rozkładu zajęć.*

**Słowa kluczowe:** *Tabu serach, generowanie rozkładu zajęć*

## Tabu search algorithms for timetabling optimization

**Abstarct:** *Tabu Search is a metaheuristic that guides a local heuristic search procedures to explore the solution space beyond local optimum. The process of searching the solutions space is coordinated by strategies based on the memory mechanisms, which are characters of the TS algorithm. This paper describes the method of TS in the timetable optimization process.*

**Keywords:** *Tabu search, timetabling*

## 1. WSTĘP

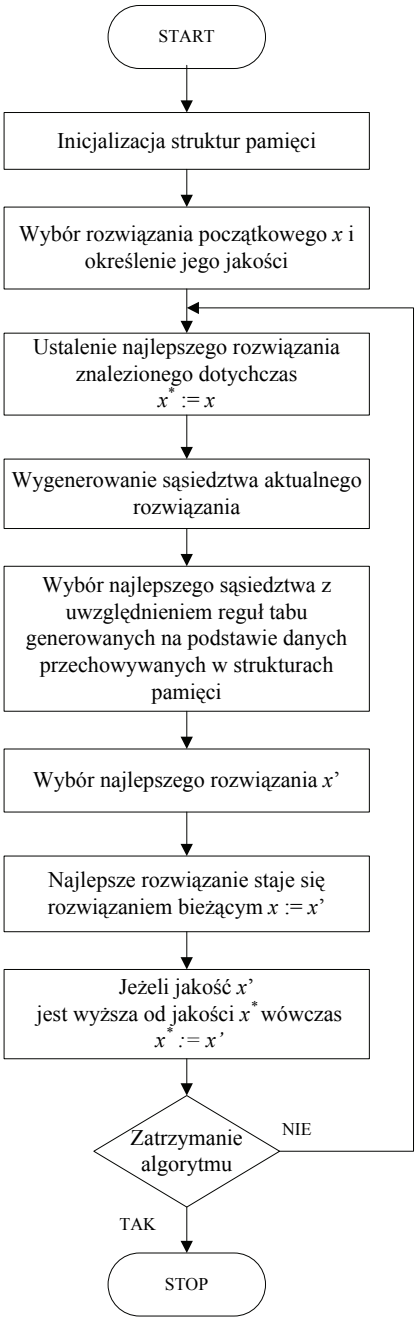
Tworzenie rozkładu zajęć polega na wyznaczeniu kombinacji wykładowca-grupa dla określonych okresów czasu, pedagogów i grup uczniów. Celem tego procesu jest zmniejszenie liczby konfliktów, które pojawiają się gdy kilku wykładowców musi prowadzić zajęcia z jedną grupą studentów w tym samym ograniczonym przedziale czasu lub gdy różne lekcje wymagają wspólnych sal. Główna trudność związana jest ze skalą problemu. Z jednej strony należy uwzględnić potrzeby nauczycieli, oczekujących zwartego planu zajęć i minimalnej liczby okienek, z drugiej strony natomiast higienę nauki uczniów. Często oba te cele nawzajem się wykluczają i jedynym rozwiązaniem jest znalezienie jak najlepszego kompromisu.

## 2. ALGORYTM TABU SEARCH

TS jest mataheurystyką szukającą rozwiązania problemu poprzez nadzorowanie innych procedur heurystycznych lokalnego przeszukiwania, w celu eksploracji przestrzeni rozwiązań poza lokalne optimum. Proces przeszukiwania przestrzeni rozwiązań jest koordynowany za pomocą strategii opartych na mechanizmach pamięci, będących cechą charakterystyczną dla algorytmu TS.

Algorytm TS dokonuje pełnego przeszukiwania sąsiedztwa aktualnego rozwiązania. Rozwiązanie bieżące zastępowane jest przez najlepsze rozwiązanie w sąsiedztwie, nawet jeśli powoduje to pogorszenie jakości. W procesie przeszukiwania wykorzystywany jest system ograniczeń nałożonych na zbiór należący do sąsiedztwa. Ma to na celu przeciwdziałanie możliwości zapętlenia się algorytmu i powracania do tych samych wąskich

obszarów rozwiązań. Ze zdefiniowanego sąsiedztwa usuwa się rozwiązania, które wcześniej były zaakceptowane jako rozwiązania bieżące tworząc tzw. zbiór tabu. Ograniczenia stosowane w TS te mają formę absolutnego zakazu lub pewnych restrykcji.




**Rys. 1.** Schemat działania algorytmu Tabu Search

Podstawowym i najbardziej charakterystycznym elementem mechanizmu tabu jest pamięć. W trakcie przeszukiwania gromadzone są w niej informacje o przeszukiwanej przestrzeni. Lokalne wybory są uzależnione od informacji zebranych podczas całego przeszukiwania. Na podstawie zapisanych w pamięci informacji tworzone są ograniczenia, które chronią algorytm przed powracaniem do przeszukanych już obszarom przestrzeni. Ograniczenia te zależą od:

- częstotliwości wykonywania zapisu określonych danych,
- aktualności zapisanych w pamięci danych,
- wpływu określonych danych na jakość uzyskiwanych wyników.

W kolejnych iteracjach algorytm przeszukuje sąsiedztwo aktualnie znalezionego rozwiązania w celu określenia nowego położenia rozwiązania bieżącego. W związku z tym dla zadanej przestrzeni poszukiwań musi zostać zdefiniowana relacja sąsiedztwa dla wszystkich jej elementów.

Struktura pamięci przechowuje przede wszystkim informacje o zrealizowanych przejściach. Może także zawierać inne informacje, np. o częstotliwości tych przejść lub czasie, który upłynął od wykonania każdego z przejść. W algorytmach TS mamy do czynienia z dwoma rodzajami pamięci: krótkoterminową i długoterminową. Każda z nich jest wykorzystywana przez charakterystyczne dla siebie strategie. Efekt działania ich obu jest widoczny jako modyfikacja sąsiedztwa $N(x)$ aktualnego rozwiązania $x$. Zmodyfikowane sąsiedztwo $N^*(x)$ jest rezultatem przechowywania informacji dotyczących dotychczasowego procesu przeszukiwania.

Pamięć krótkoterminowa jest wykorzystywana w każdej iteracji i służy do zapamiętywania ostatnio odwiedzonych rozwiązań i nałożonych na nie kar. Jej głównym zadaniem jest uniknięcie wyboru operatora ruchu, który może doprowadzić do zapętlenia się algorytmu w pewnym wąskim obszarze przestrzeni poszukiwań.

Pamięć długoterminowa przechowuje informacje o przebiegu procesu poszukiwania Pozwala na zapamiętanie najlepszych rozwiązań z odwiedzonych już obszarów przestrzeni poszukiwań a nie wyłącznie najlepszego rozwiązania z bieżącego sąsiedztwa.

W celu określenia relacji sąsiedztwa należy określić operator zmiany bieżącego rozwiązania, który zdefiniuje sąsiedztwo punktu: każdy punkt, który da się uzyskać z punktu bieżącego, wykorzystując operator zmiany, jest sąsiadem tego punktu bieżącego, a pozostałe punkty - nie.

Przy każdym kroku algorytmu musi zostać określony zbiór przejść do nowych położeń należących do sąsiedztwa rozwiązania aktualnego. Z nich zostanie wybrane jedno, które stanie się nowym bieżącym rozwiązaniem. Dla każdego z wygenerowanych przejść muszą zostać określone i sprawdzone wszystkie jago atrybuty. Jeżeli wartość takiego atrybutu jest określona choć jednym ograniczeniem tabu, atrybut ten jest tabu-aktywny. W przeciwnym wypadku atrybut jest tabu-nieaktywny. Po określeniu statusu wszystkich atrybutów każdego nowego przejścia generowane są reguły określające czy dane przejście jest tabu-aktywne czy nie. Status tabu-aktywny jednego atrybutu przejścia nie musi oznaczać, że status całego przejścia jest również tabu-aktywny Zależy to od reguł wykorzystujących status atrybutów. Jeśli przykładowo reguła mówi, że status przejścia do nowego położenia jest sumą logiczną dwóch określonych atrybutów, wówczas tabu-nieaktywność obu rozważanych atrybutów powoduje tabu-nieaktywność całego przejścia. Ustawienie statusu tabu-aktywnego dla określonego atrybutu wymaga zawsze określenia liczby iteracji, przez które status ten będzie utrzymywany. Po wykonaniu tej liczby iteracji status atrybutu ustawiany jest na tabu-nieaktywny. Jeżeli jednak wcześniej wykonany zostanie ruch tabu, ponownie uaktywniający ten atrybut, wówczas czas aktywności dla niego jest liczony od początku. Określenie długości kadencji tabu nie musi być jednakowa dla wszystkich atrybutów, powinna nawet być określona indywidualnie dla każdego z nich. Długość ta może być stała lub zmienna w czasie. W takim przypadku może być ustalona na sztywno (np. może być funkcją numeru aktualnie wykonywanej iteracji) lub zmieniać się losowo. Może być również funkcją współczynników określających postępy czynione przez algorytm. Każde przejście może zostać określone różnymi atrybutami. Może to być np. wartość kary, jaka jest mu przypisana, liczba bądź częstotliwość wykonywania tego przejścia w ostatnim czasie, okres tabu, tj. pozostałą liczbę iteracji, przez które będzie jeszcze obowiązywał zakaz jego wykonania. Wszystkie te informacje będą zapisywane do struktur pamięci w trakcie realizacji algorytmu. Te dwa typy zdarzeń różnią się między sobą: wysoka częstotliwość występowania pewnej wartości na określonej pozycji oznacza, że ta właśnie wartość atrybutu jest szczególnie pożądana. Natomiast wysoka zmienność wartości na określonej pozycji określa wysoką nieprzewidywalność tego atrybutu i oznacza, że jego rola jest prawdopodobnie większa przy końcowym dostrajaniu się do optimum, niż we wstępnej części procesu optymalizacji, tj. przy początkowej eksploracji w poszukiwaniu najbardziej obiecującego regionu dziedziny.

Poszukiwanie nowej wartości rozwiązania bieżącego wygląda następująco: operator przejścia do następnego położenia wybiera pewien podzbiór rozwiązań z sąsiedztwa aktualnego położenia i szuka wśród nich najlepszego, uwzględniając jednak przy określaniu wartości i dostępności rozwiązań reguły tabu. Po wygenerowaniu zbioru sąsiadów jest on analizowany i po uwzględnieniu wszystkich reguł tabu najlepszy znaleziony w tym zbiorze osobnik staje się nowym bieżącym rozwiązaniem algorytmu. Informacja o tym przejściu zapisywana jest do struktur pamięci.

Reguły tabu, którymi posługuje się operator przejścia, mają za zadanie niedopuszczenie do utknięcia procesu przeszukiwania w miejscowych optimach. Umożliwiają także wycofanie się z już odwiedzonych rejonów przestrzeni rozwiązań, dzięki czemu algorytm może dokonać przeszukiwania na większej przestrzeni. Reguły tabu mogą być różnorodne. Mogą one wprowadzać zakazy całkowite lub częściowe. Reguły te mogą także uwzględniać czas, np. zakaz przejścia między punktami może zostać anulowany po upływie ściśle określonego czasu (czyli liczby iteracji) od ostatniego takiego przejścia. Po upływie pewnego czasu mogą również maleć kary by w końcu zostać anulowane. Jednak po powtórnym wykonaniu takiego przejścia są one przywracane.

Ograniczenia tabu mogą zakazywać atrakcyjnych ruchów, nawet kiedy nie ma żadnego niebezpieczeństwa utknięcia w lokalnym optimum lub mogą prowadzić do stagnacji procesu przeszukiwania. Jest więc konieczne użycie narzędzi, które pozwolą odwołać tabu. Nazywane są one kryteriami aspiracji. Złagodzenie ograniczeń może zostać wdrożone poprzez kasowanie wybranego ograniczenia i dodawaniu do funkcji oceny pewnej kary za naruszanie zakazu. Istnieje kilka sposobów określania wartości kar. Jednym z nich jest zastosowanie tzw. samoregulującej kary – jej wartość zmienia się dynamicznie na podstawie historii procesu przeszukiwania

## 3. IMPLEMENTACJA SYSTEMU UKŁADANIA PLANU WSPOMAGANEGO TS

Proces układania planu lekcji został zaimplementowany w środowisku Matlab i polega na wygenerowaniu planu dla każdego z wykładowców i każdej z grup. Tworzony jest również rozkład wykorzystania poszczególnych sal. Zbiór wszystkich cząstkowych rozkładów zajęć tworzy plan wynikowy. Celem działania programu jest zmniejszenie liczby konfliktów, które pojawiają się gdy kilku wykładowców musi prowadzić zajęcia z jedną grupą studentów w tym samym ograniczonym przedziale czasu lub gdy różne lekcje wymagają wspólnych sal. Główna trudność związana jest ze skalą problemu. Oprócz dużej ilości grup, wykładowców, zajęć i sal musi zostać uwzględnionych wiele dodatkowych warunków i celów.

W tworzeniu rozkładu zajęć ograniczenia podzielone są na dwa rodzaje. Ograniczenia ciężkie (H) muszą zostać bezwzględnie spełnione. Warunki, które są uważane za pomocne ale nie konieczne w dobrym rozkładzie zajęć są tzw. ograniczeniami lekkimi (S). W im większym stopniu zostaną one spełnione, tym lepsza jakość rozkładu zajęć zostanie osiągnięta. Z tego powodu mają wpływ na wartość funkcji oceny.
Plan zajęć składa się ze zbioru m wykładowców $t_1,\ldots t_m$ prowadzących zajęcia z n grupami studentów $c_1,\ldots,c_n$ w ciągu okresów $p_1,\ldots,p_j$. Przydział nauczycieli do poszczególnych grup jest z góry ustalony a obciążenie pracą zapisane jest w matrycy wymagań. Rozwiązanie problemu polega na minimalizacji funkcji oceny f(x) uwzględniającej stopień spełnienia ograniczeń przez wygenerowane rozwiązanie.

Funkcja oceny wyraża się wzorem:

$$f(x) = \sum_r w_r f_r(x)$$

gdzie r jest kolejnym numerem ograniczenia.

Wartości wag $w_r$ są zdefiniowane przez programistę i określają jaki znaczenie w całym procesie ma dane ograniczenie. Funkcje $f_r(x)$ są sumą wartości wszystkich rozwiązań $x_{ita}$ dla parametrów określonych przez dane ograniczenie. Sumy cząstkowe związane z ograniczeniami ciężkimi określają poziom wykonalności natomiast sumy powiązane z ograniczeniami lekkimi są miarą zadowolenia.

Ograniczenia ciężkie, przyjęte przy tworzeniu programu, przedstawiają się następująco:

(Hl) Każda lekcja musi być przypisana do okresu albo pewnej liczby okresów, zależących o długości lekcji.
(H2) Żaden nauczyciel nie może prowadzić dwóch różnych zajęć w tym samych czasie.
(H3) Żadna grupa nie może uczestniczyć w dwóch różnych zajęciach w tym samym czasie.
(H4) Dwie równoczesne lekcje nie mogą odbywać się w jednej sali.
(H5) Sala wyznaczona do danej lekcji musi być wymaganego typu.
(H6) Okresy dostępności/niedostępności nauczycieli muszą być przestrzegane.
(H7) Lekcje poszczególnych grup jednego rocznika nie mogą kolidować z zajęciami wspólnymi dla tego rocznika.

Ograniczenia lekkie mają postać:

(S1) Rozkłady zajęć dla poszczególnych grup powinny mieć jak najmniej „okienek" by wyeliminować czasy bezczynności studentów.
(S2) Rozkłady zajęć dla poszczególnych nauczycieli powinny mieć jak najmniej „okienek" by wyeliminować czasy bezczynności nauczycieli.
(S3) Studenci nie powinni mieć więcej niż określoną liczbę godzin poszczególnych bloków zajęć (przedmiot/wykładowca/rodzaj zajęć).

Maksymalna liczba godzin ograniczenia (S3) określana jest przez użytkownika. Istnieje także możliwość zmiany maksymalnej ilości iteracji, ilości zjazdów dla wprowadzonej siatki godzin oraz rodzaju studiów. W przypadku studiów zaocznych zajęcia odbywają się od piątku do niedzieli, przy czym użytkownik ma możliwość wyboru godziny rozpoczęcia lekcji w pierwszy dzień zjazdu. Zajęcia studiów dziennych odbywają się w dni powszednie.

Stałymi parametrami algorytmu są:

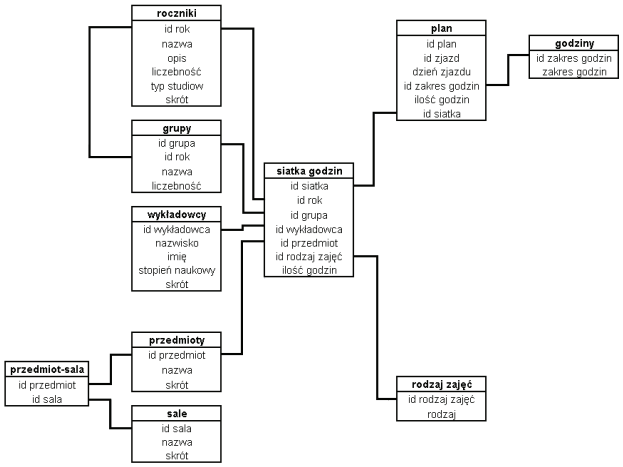
- czas tabu,
- wagi $w_r$ funkcji oceny,
- ilość iteracji, po której następuje intensyfikacja/dywersyfikacja.

Generowanie rozkładu zajęć musi zostać poprzedzone wprowadzeniem podstawowych informacji. Opisują one:

- wykładowców,
- roczniki,
- grupy,
- przedmioty,
- sale,
- rodzaj zajęć.

Dane te zostają wykorzystane do wprowadzenia siatki godzin, tj. listy wszystkich zajęć odbywających się w określonym semestrze/roku. Na podstawie zawartych w niej informacji tworzony jest plan zajęć.

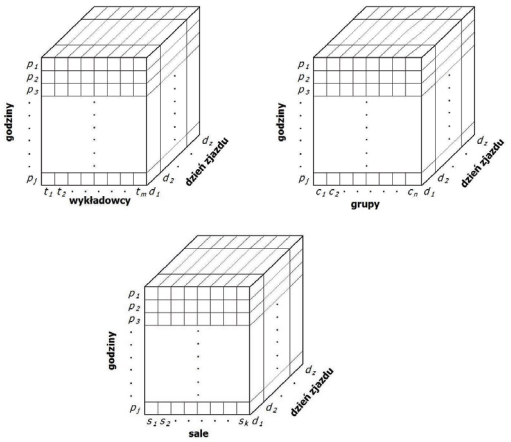
Wymienione powyżej dane przechowywane są w postaci tablic powiązanych ze sobą określonymi zależnościami, przedstawionymi na rys. 2.




**Rys. 2.** Struktura danych i powiązania między nimi

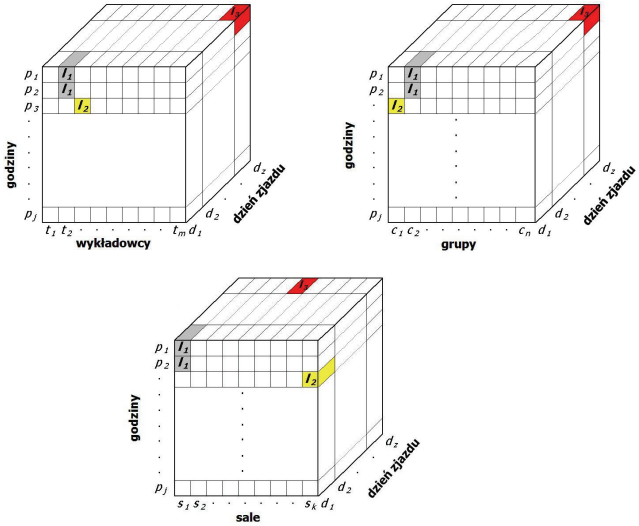
Przyjęta jednostka czasowa jest równa 15 minutom.

W omawianej implementacji do przechowywania poszczególnych planów cząstkowych wykorzystane zostały trzy macierze trójwymiarowe T, C i S. Pierwsza z nich zawiera informacje o planach wykładowców t, druga –grup c, trzecia - sal s.

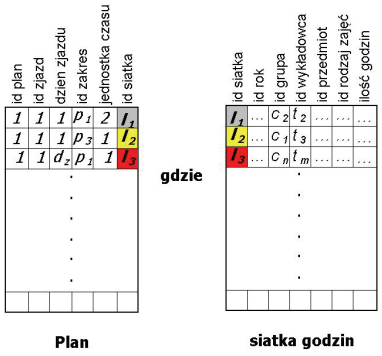



**Rys. 3.** Macierze przechowujące plany cząstkowe:
a) wykładowców, b) grup, c) sal

Tabele te są odrębnymi strukturami, łączy je jednak powiązanie z tabelą siatka godzin. Umieszczenie wylosowanej pozycji z macierzy siatka godzin (id siatka) w jednej z macierzy T, C lub S powoduje jednocześnie umieszczenie tego elementu również w analogicznych miejscach pozostałych dwóch macierzy. Z uwagi na fakt, że każdy element id siatka jednoznacznie definiuje wykładowcę, grupę i salę, których dotyczy, w poszczególnych komórkach macierzy T, C, S zapisywana jest tylko jedna wartość. W innym wypadku konieczne byłoby umieszczanie w komórkach macierzy większej ilości informacji. Rys. 4. i 5 ilustrują sposób zapisu informacji w macierzach T, C i S oraz tabeli Plan.




**Rys. 4.** Sposób zapisu informacji w macierzach T, C, S. Element $l_i$ oznacza i-tą wartość id siatka




**Rys. 5.** Sposób zapisu poszczególnych pozycji generowanego planu zajęć na podstawie danych z macierzy T, C, S z rys. 4

Celem działania programu jest wygenerowanie planu lekcji o jak najmniejszej wartości funkcji oceny. Przy jej ustalaniu analizowany jest stopień spełnienia przez wygenerowane rozwiązanie ograniczeń określonych w punkcie 3.2. Zaimplementowany algorytm uniemożliwia stworzenie rozkładu zajęć łamiącego którekolwiek z ograniczeń ciężkich (H1)-(H7). Do wyznaczenia wartości funkcji oceny uwzględniane są więc tylko ograniczenia lekkie (S1)-(S3). Za każde złamanie ograniczeń przyznawany jest punkt karny. Wartość funkcji oceny wygenerowanego planu jest równa sumie wszystkich punktów karnych poszczególnych planów cząstkowych.

Proces tworzenia rozkładu zajęć rozpoczyna się od rozwiązania początkowego generowanego losowo. W trakcie każdej iteracji przeszukiwana jest przestrzeń rozwiązań X będąca zestawem rozwiązań spełniających ograniczenia (H1)-(H7).

Zbadanie całego sąsiedztwa rozwiązania x jest kosztowne obliczeniowo. Z tego powodu w programie zastosowana została strategia listy potencjalnych obiektów. Każda określona kombinacja zajęcia-wykładowca-grupa-sala (id siatka) jest wybierana przypadkowo jednak przypadkowość jest ograniczona. Rozwiązanie x′ musi spełniać ograniczenia ciężkie wobec czego nie jest konieczne przeszukiwanie całego sąsiedztwa N(x).

W pamięci programu przechowywana jest lista tabu oraz atrybuty zaakceptowanych ruchów. Zapisywana jest także liczba powtórzeń, dla której ruch ma pozostać zakazany. Przy każdej kolejnej iteracji liczba ta jest zmniejszana, a gdy przyjmie wartość równą 0 ruch zostaje usunięty z listy. Do wykonania wybrany zostaje natomiast ruch generujący rozwiązanie x' z najlepszą wartością funkcji oceny lepszą niż wartość funkcji oceny najlepszego otrzymanego dotychczas rozwiązania.

Po wykonaniu ruchu, rozwiązanie z najlepszą wartością funkcji oceny w nowym regionie zostaje zapisane w pamięci. Jeżeli rozwiązanie lepsze od zapisanego nie zostanie otrzymane po ustalonej liczbie powtórzeń, zapisane rozwiązanie będzie badane ponownie (intensyfikacja). Dzięki temu poszukiwanie skupia się na sąsiadach dobrych rozwiązań. Jeżeli rozwiązanie lepsze niż najlepsze regionalne nie zostanie znalezione w ciągu określonej liczby iteracji, algorytm powróci do najlepszego regionalnego rozwiązania.

Wszystkie opisane powyżej kroki algorytmu są wykonywane do momentu spełnienia jednego z kryteriów końca: znalezione zostanie rozwiązanie z wartością funkcji oceny równą 0 lub ilość iteracji osiągnie zadaną wartość.

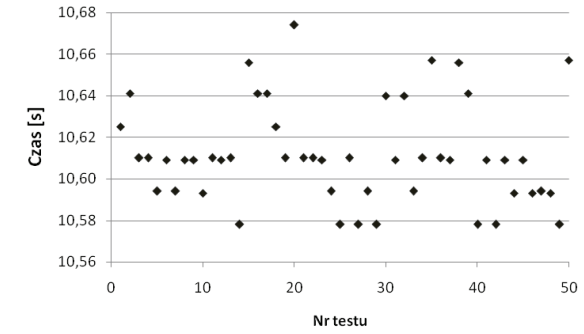
## 4. WYNIKI

Eksperyment został podzielony na dwie części. W pierwszej z nich dokonano pomiaru czasu niezbędnego do wygenerowania planu lekcji w sposób losowy bez użycia algorytmów optymalizacyjnych. Zbadano skuteczność metody oraz jakość otrzymanego rozkładu zajęć. Pomiar skuteczności przeprowadzony został poprzez uruchomienie aplikacji 100 razy i określeniu ilości poprawnie wygenerowanych planów zajęć. Czas generacji rozkładów oraz ich jakość określony został na podstawie 50 poprawnie wylosowanych planów. W drugiej części eksperymentu dokonano analogicznych pomiarów dla planów lekcji utworzonych przy zastosowaniu algorytmu TS. Przyjęty okres tabu był równy 5 bez możliwości warunkowego wykonania zakazanego ruchu. Porzucenie aktualnie przeszukiwanego sąsiedztwa na rzecz nowych obszarów następowało po upływie 10 iteracji. Wartości wag $w_r$ funkcji oceny miały wartość równą 0.01, zarówno dla cząstkowych wartości funkcji oceny związanych z planami poszczególnych wykładowców, jak i poszczególnych grup. Wprowadzenie jednakowych wartości wag powoduje, że każdy czynnik odgrywa taką samą rolę w ocenie jakości planu. Maksymalna ilość iteracji wynosiła 500. Plan generowany był dla studiów dziennych.

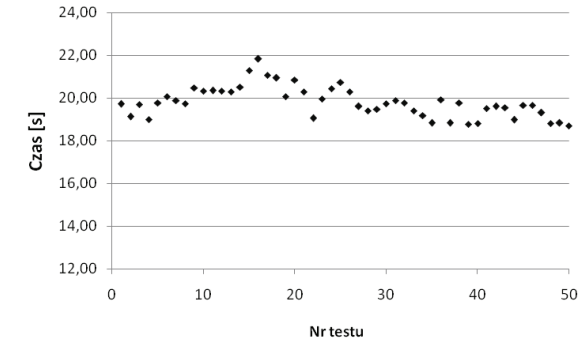
Wyniki eksperymentu omówionego w punkcie 3.6 zostały zilustrowane na rys. 6-8.

Czas niezbędny do losowego wygenerowania planu zajęć jest krótszy dla procesu losowego (rys. 6).

a)



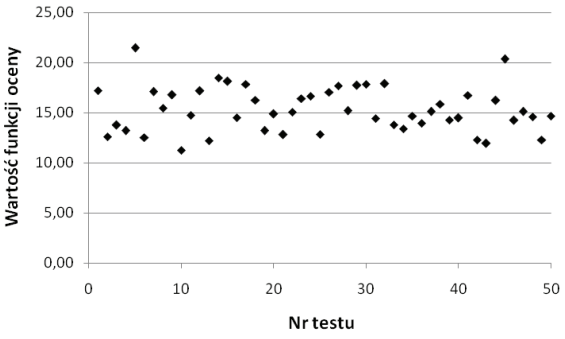

b)



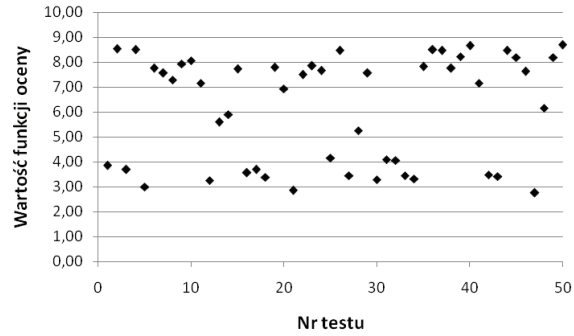

**Rys. 6.** Czas niezbędny do wygenerowania planu zajęć: a) generowanie losowe, b) z wykorzystaniem TS

Pierwsze wygenerowane rozwiązanie staje się rozwiązaniem końcowym, niezależnie od wartości funkcji oceny, co znajduje swoje potwierdzenie na rys. 7 a, b. Jakość planu wynikowego otrzymanego przy wykorzystaniu algorytmu TS jest średnio dwa razy lepsza niż w przypadku tworzenia go w sposób losowy.

a)



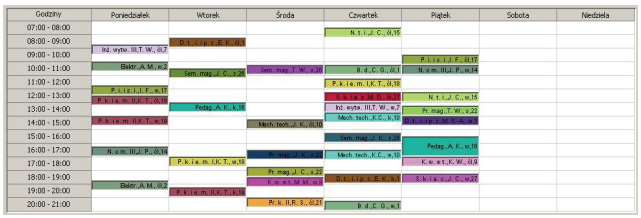

b)



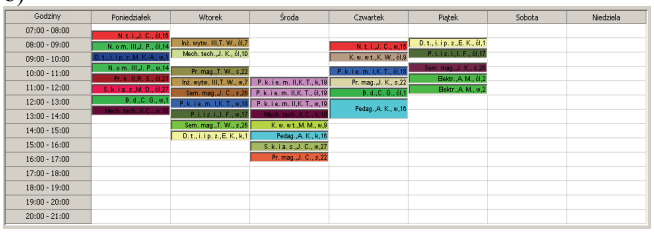

**Rys. 7.** Jakość wygenerowanego planu zajęć: a) generowanie losowe, b) z wykorzystaniem TS

Wartość funkcji oceny ma decydujący wpływ na ostateczną postać wynikowego planu zajęć. Różnice przedstawione w postaci wykresów na rys. 7 a i b wyraźnie widać na graficznej wersji harmonogramów.

a)




b)




**Rys. 8.** Przykładowe plany zajęć: a)generowanie losowe, b) z wykorzystaniem TS

## 5. PODSUMOWANIE

Przedstawione powyżej wyniki dowodzą, że zastosowanie algorytmu TS wpływa na znaczne zwiększenie jakości ostatecznego rozwiązania w stosunku do wyników generacji losowej. Dzięki zastosowaniu systemu ograniczeń możliwe jest sprawdzenie wszystkich dostępnych wariantów opracowywanego planu i określenie, który z nich w największym stopniu spełnia określone z góry założenia.

## Literatura

1. Aladağ Ç. H,. Hocaoğlu G., *A Tabu Search Algorithm to Solve a Course Timetabling Problem*, Hacettepe Journal of Mathematics and Statistics, vol. 36(1), 2007, s.53-64
2. Glover F., Laguna M., *Tabu Search*, Kluwer Academic Publishers, Norwell, MA, 1997, ISBN:079239965X, s. 53
3. Glover, F., *Tabu Search — Part I*, ORSA Journal on Computing, vol. 1,no 3, 1989, s. 190-206
4. Glover, F., *Tabu Search — Part II*, ORSA Journal on Computing, vol. 2,no 1, 1990, s. 4-32