RÓWNOLEGŁY ALGORYTM POSZUKIWANIA Z ZABRONIENIAMI UKŁADANIA PLANU ZAJĘĆ

Wojciech BOŻEJKO, Łukasz GNIEWKOWSKI

Streszczenie: Praca dotyczy zastosowania równoległego algorytmu poszukiwania z zabronieniami (*tabu search*) do rozwiązywania ogólnego problemu układania planu zajęć (*timetabling*). Problem układania planu zajęć wyrażono jako problem kolorowania grafów, a następnie uzyskano dobre rozwiązania przybliżone stosując współbieżny algorytm metaheurystyczny dla GPU (*Graphics Processing Unit*).

Słowa kluczowe: poszukiwanie z zabronieniami, plan zajęć, algorytm równoległy, kolorowanie grafu

1. Wprowadzenie

Plan zajęć jest przyporządkowaniem zbioru zadań (lekcji) do odpowiednich terminów, tak aby spełnić ograniczenia postawione przed realizacją układania planu. Na podstawie takiego przystosowania można wyznaczyć plan lekcji dla klas i nauczycieli. Zagadnienie *układania planu zajęć* (pracy) jest problemem optymalizacji kombinatorycznej silnie inspirowanym przez praktykę. Opierając się na rozmowach z przedstawicielami polskich szkół podstawowych gimnazjalnych, średnich i technicznych można stwierdzić, że problem ten jest rozwiązywany jednokrotnie lub dwukrotnie każdego roku – co każdy nowy semestr lub rok szkolny, zależnie od rodzaju szkoły. W większości szkół problem ten wciąż jest analizowany i rozwiązywany metodami tradycyjnymi co oznacza, że całą pracę wykonuje jedna lub kilka osób w ciągu kilku dni do nawet kilku tygodni przy użyciu kartki papieru i/lub tablicy. Procedura taka jest zarówno czaso- jak i praco- chłonna. Chociaż istnieją narzędzia programowe (pakiety, aplikacje) wspomagające ten proces, to zazwyczaj i tak większość pracy wykonuje człowiek, co więcej właśnie do jego kompetencji należy podjęcie ostatecznej decyzji. Wynika to z powolnego działania aplikacji lub z faktu, że otrzymane rozwiązanie nie spełnia stawianych wymagań.

W niniejszej pracy rozważamy wymagania występujące w procesie tworzenia planu lekcji, które zostały wyznaczone na podstawie informacji zawartych w literaturze, a także rozmów z osobami zajmującymi się tym zagadnieniem w placówkach oświaty.

2. Sformułowanie problemu

Na potrzeby publikacji zostały przyjęte dwa typy ograniczeń: *ograniczenia twarde* (które muszą być spełnione) i *ograniczenia miękkie* (nie muszą być spełnione aby plan był akceptowalny, ale zaleca się ich spełnienie). Do *ograniczeń twardych* należą:

- Żadna klasa nie może mieć dwóch zajęć w jednym terminie.
- Żaden nauczyciel nie może prowadzić dwóch zajęć równolegle.
- Liczba zajęć wymagających danego typu sali, odbywających się w tym samym terminie nie może być większa od liczby dostępnych sal tego typu.

- Żadna klasa nie może posiadać okienek (okresów bez zajęć pomiędzy zajęciami w danym dniu) w żadnym dniu.
- Klasa jednego dnia musi mieć odpowiednią liczbę zajęć, minimalnie 4 lekcje, maksymalnie 8.
- Dla klasy nie mogą odbywać się dwie lekcje z tego samego przedmiotu w jednym dniu.

Jako *ograniczenia miękkie* zostały obrane następujące:

- Nauczyciel nie powinien posiadać w dniu jednej godziny zajęć.
- Nauczyciel nie powinien posiadać więcej niż dwóch okienek w trakcie dnia.

Praca składa się z 8 rozdziałów. Po krótkim wprowadzeniu w tematykę (Rozdział 1) sformułowano cele pracy (Rozdział 2). Rozdział 3 zawiera ogólny zarys stosowanych wcześniej podejść do rozwiązania problemu *harmonogramowania*. Kolejno, w Rozdziale 4, zamieszczono model matematyczny zagadnienia wraz z opisem *użytego algorytmu* (Rozdział 5). Rozdział 6 zawiera opis zrównoleglenia algorytmu. Przeprowadzone badania nad przyspieszeniami algorytmu dzięki zrównolegleniu wraz z wynikami zamieszczono w Rozdziale 7, natomiast Rozdział 8 zawiera płynące z nich wnioski.

3. Przegląd literatury

Układanie planu zajęć (ang. *timetabling*) należy do zagadnień wielokryterialnej optymalizacji kombinatorycznej. Problem ten występuje względnie często w literaturze, jest stale przedmiotem badań, choć niektóre jego warianty zostały dotychczas stosunkowo dobrze przeanalizowane (najczęstszym wariantem jest układanie planu zajęć dla szkoły wyższej bądź rozkład egzaminów w sesji dla uczelni wyższych). W literaturze można znaleźć wiele sposobów modelowania rozwiązania zagadnienia. Do najczęściej stosowanych można zaliczyć trzy modele:

- Macierz 3-wymiarowa [3,9] – rozwiązanie przedstawione jest w postaci macierzy zero-jedynkowej x, gdzie x_{ijk} przyjmuje wartość 1 gdy i-ta lekcja odbywa się w j-tym terminie w k-tej sali, wartość 0 przypisana jest w przeciwnym przypadku.
- Macierz 2-wymiarowa [7] – rozwiązanie przedstawione jest w dwu wymiarowej macierzy, w której na pozycji x_{ij} znajduje się numer lekcji odbywającej się w i-tym terminie w j-tej sali.
- Model grafowy [2,4] – model ten wymaga przeprowadzenia przepisania problemu zagadnienia *układaniu planu zajęć na ograniczony problem kolorowania grafu*, w którym krawędzie łączą wierzchołki które nie mogą występować w tym samym czasie (ze względu na wspólnych uczestników bądź prowadzących).

Problem układania planu zajęć należy do grupy zagadnień silnie NP.-trudnych. W wyniku czego, praktycznych instancji tego problemu nie można rozwiązać w sposób dokładny (wyznaczając rozwiązanie globalnie optymalne) z powodu znacznego czasu obliczeń odpowiedniego algorytmu optymalizacyjnego. Dlatego też dominującymi są metody wyznaczające rozwiązania przybliżone. Faktycznie, w praktyce nie wymagamy, aby znalezione rozwiązanie było rozwiązaniem globalnie najlepszym. Wystarczy że otrzymane rozwiązanie będzie spełniało w pełni stawiane przez nas tzw. *wymagania twarde* i dostatecznie dobrze tzw. *wymagania miękkie*. Istnieje cały wachlarz technik obliczeniowych pozwalający na odnajdywanie rozwiązań suboptymalnych. Największą ich zaletą jest możliwość otrzymania dobrego jakościowo rozwiązania w satysfakcjonującym czasie dla problemów nierozwiązywalnych w sposób ścisły. Do najczęściej stosowanych metod można zaliczyć :

- Hiperheurystyki [4] – podejście polega na posiadaniu kilku heurystyk niskiego poziomu, z których w danym kroku wybierana jest jedna do rozwiązania problemu. Sposoby wyboru heurystyki mogą być różne, np. poprzez inną heurystykę.
- Poszukiwanie z zabronieniami (ang. *tabu search*) [3,7,9] – bardzo popularna metoda rozwiązania problemu układania planu zajęć. Jednak najczęściej autorzy rozwiązań dodają dodatkowe człony do algorytmu w celu otrzymania lepszych wyników – np. memetyczny algorytm poszukiwania z zabronieniami [7].
- Algorytmy populacyjne [2,8] – pod tym pojęciem znajduje się bardzo szeroka gama, od algorytmów ewolucyjnych, po algorytmy mrówkowe, rojowe itp. Najczęściej te algorytmy łączone są z grafowym modelem zagadnienia.

Wyraźnie widać, że do zagadnienia można podchodzić na wiele sposobów, dlatego też problem *harmonogramowania* wciąż jest w centrum zainteresowań wielu okręgów badawczych.

4. Model grafowy

W niniejszym rozdziale zostanie przedstawiony model matematyczny problemu. Zostaną zdefiniowane dane wejściowe problemu, rozwiązanie a także postać funkcji celu.

4.1. Dane wejściowe

Dane wejściowe, które odpowiednio definiują problem wraz z uwzględnieniem wszystkich ograniczeń opisanych w punkcie 2, są następujące:

- k – liczba klas,
- n – liczba nauczycieli,
- l – liczba lekcji,
- $V = \{V_1, V_2, \dots, V_l\}$ – zbiór lekcji,
- d – liczba dni,
- r – liczba przedziałów w dniu,
- s – liczba rodzajów sali,
- $S = \{S_1, S_2, \dots, S_s\}$ – liczba sali poszczególnego typu.

Jedna lekcja zawiera w sobie informacje, dla której klasy jest prowadzona, przez jakiego nauczyciela, z jakiego przedmiotu, ile czasu trwa a także rodzaj wymaganej sali.

4.2. Zmienne pomocnicze

Opisane tutaj zmienne są tworzone z danych wejściowych w celu szybszego i łatwiejszego sprawdzenia spełnienia ograniczeń.

- $E = \{E_1, E_2, \dots, E_e\}$ - zbiór krawędzi,
- $v = \{v_1, v_2, \dots, v_l\}$ - zbiór wierzchołków,
- $K = \{K_1, K_2, \dots, K_k\}$ – listy indeksów lekcji i-tej klasy,
- $N = \{N_1, N_2, \dots, N_n\}$ - listy indeksów lekcji i-tego nauczyciela,
- $I = \{I_1, I_2, \dots, I_s\}$ - listy indeksów lekcji wymagających i-tego rodzaju sali,
- t – liczba terminów,
- $T = \{T_1, T_2, \dots T_t\}$ –listy indeksów lekcji, które nie mogą wystąpić tego samego dnia.

Dla każdego wierzchołka zdefiniowane są indeksy klasy, nauczyciela, przedmiotu,

wymaganego rodzaju sali, zbioru lekcji, które nie mogą odbyć się jednego dnia oraz czas trwania. Odwołania do tych pól są następujące

- $v_i.k$ – indeks klasy, dla której prowadzone są i-te zajęcia,
- $v_i.n$ – indeks nauczyciela, który prowadzi i-te zajęcia,
- $v_i.p -$ indeks przedmiotu, z którego prowadzone są i-te zajęcia,
- $v_i.t$ – czas trwania i-tych zajęć,
- $v_i.s$ – wymagany rodzaj sali dla i-tych zajęć,
- $v_i.z -$ indeks zbioru lekcji, które nie mogą wystąpić jednego dnia.

Zbiór krawędzi tworzony jest pomiędzy lekcjami, które odbywają się dla tej samej klasy, bądź posiadają identycznego prowadzącego. Zbiór wierzchołków reprezentuje poszczególne lekcje. Listy K,N,I tworzone są w trakcie wczytywania danych wejściowych. Każda lekcja posiada informacje o klasie, nauczycielu i rodzaju sali, a więc po odczycie indeks lekcji dodawany jest do odpowiednich zbiorów.

Liczba terminów obliczana jest formułą:

$$t = d \cdot r. \tag{1}$$

Liczba terminów definiuje liczbę kolorów dostępnych w rozwiązaniu.

W skład jednego zbioru T wchodzą lekcje odbywające się dla tej samej klasy z identycznego przedmiotu.

4.3. Model rozwiązania

Rozwiązanie zagadnienia przedstawione jest za pomocą 4 wektorów:

- $P = \{P_1, P_2, \dots P_n\}$ –kolor wierzchołka, dla wierzchołków wielokolorowych jest to pierwszy kolor,
- $Z = \{Z_1, Z_2, \dots, Z_n\}$ – ostatni kolor dla wierzchołków wielokolorowych, dla wierzchołków jednokolorowych Z=P,
- $G = \{G_1, G_2, \dots, G_n\}$ – numer dnia, w którym odbywają się dane lekcje,
- $R = \{R_1, R_2, \dots, R_n\}$ – indeks odpowiedniego rodzaju sali, w której odbywają się zajęcia. W tej pracy nie uwzględnia się żadnych kryteriów dotyczących doboru sali, dlatego też wektor ten jest tworzony po zakończeniu działania algorytmu.

Można zauważyć, że na podstawie wektora P lub Z można wyznaczyć pozostałe wektory. Wyznaczenie wektora Z z wektora P przedstawia formuła (2), natomiast wektora G opisuje wzór (3).

$$Z = P + C - 1 \tag{2}$$

$$G = \lfloor P/d \rfloor \tag{3}$$

Korzystając z wyrażeń (2) i (3) w użytym algorytmie modyfikowano wektor P w celu uzyskania lepszych rozwiązań.

4.4. Funkcja celu

Dla rozważanego zadania ograniczonego problemu kolorowania grafu minimalizowana funkcja celu sformułowana jest następująco:

$$f(P) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_j f_j(P,i) \tag{4}$$

gdzie : $w_1, w_2, \dots, w_m$- odpowiednie wagi ograniczeń,

$f_1(P,i)$, $f_2(P,i), \dots, f_m(P,i)$ – funkcje wyznaczające kary za niespełnienie odpowiednich ograniczeń, definicje znajdują się poniżej. W zaimplementowanym algorytmie *m* wynosi 7.

$f_1(P)$ – funkcja sprawdzająca ograniczenie, mówiące o tym, że żadna klasa i żaden nauczyciel nie może mieć dwóch zajęć w jednym czasie. Wartość funkcji jest proporcjonalna do liczby wspólnych terminów tych lekcji (dla lekcji trwających dłużej niż jeden okres). Jest to funkcja opisująca poprawność rozwiązania w klasycznym *problemie kolorowania grafu*.

Wzór definiujący funkcję $f_1(P)$.

$$f_1(P,i) = \sum_{\{v_i, v_j\} \in E} (v_j.t * w(v_i, v_j)) \tag{5}$$

gdzie: $w(v_i, v_j) = |[P_i, Z_i] \cap [P_j, Z_j]|$ - zwraca liczbę wspólnych kolorów

$f_2(P)$ – funkcja odpowiada za sprawdzenie, czy liczba lekcji wymagających tego samego typu sali, odbywających się w danym terminie nie jest większa od liczby tych sal. Warunek ten jest sprawdzany dla wszystkich terminów, w których odbywa się *i*-ta lekcja. Funkcja ta jest obliczana za pomocą zależności (6).

$$f_2(P,i) = \sum_{j=P_i}^{Z_i} d\Big(\sum_{v_m \in I_{v_i.s}} |[P_m, Z_m] \cap j| - S_{v_i.s} \Big) \tag{6}$$

gdzie funkcja $d(x) = \begin{cases} 0 \text{ gdy } x < 0 \\ x \text{ gdy } x \geq 0 \end{cases}$ (7)

$f_3(P)$ – funkcja sprawdzająca czy lekcje, które nie mogą występować jednego dnia (odbywają się dla jednej klasy z tego samego przedmiotu), nie są realizowane jednego dnia.

Wzór funkcji jest następujący (8) :

$$f_3(P,i) = \sum_{v_j \in T_{\mathrm{v_i}.\mathrm{z}}} \mathrm{v_j}.\mathrm{t} * |\{G_i\} \cap \{G_j\}| \tag{8}$$

$f_4(P)$ – funkcja odpowiadająca za sprawdzanie liczby okienek w dniu zajęć klasy, dla której odbywa się *i*-ta lekcja. W związku z brakiem wyrażenia matematycznego opisującego takowe sprawdzenie, działanie funkcji opisane jest pseudokodem zamieszczonym poniżej (Algorytm 1.).

```
int f4(P,i){
        poczatekZbioru=(D_i - 1)·r;
        kolor=poczatekZbioru;
        stan=0; okna=0; mozliweOkna=0;
        while(poczatekZbioru + r > kolor) {
                if (stan == 0){
                        if( ∑_{v_m∈K_{v_i.k}} |[P_m, Z_m]∩kolor|==1)
                                stan=1;
                }
                else{
                        if( ∑_{v_m∈K_{v_i.k}} |[P_m, Z_m]∩kolor|==0)
                                mozliweOkna+=1;
                        else{
                                Okna+=mozliweOkna;
                                mozliweOkna=0;
                        }
                }
        }
        return okna;
}
```

Algorytm 1. Funkcja obliczająca liczbę okienek dla klasy

$f_5(P,i)$ – funkcja odpowiadająca za sprawdzenie liczby zajęć danej klasy w dniu. Funkcja przyjmuje wartości różne od zera w przypadku liczby zajęć mniejszej od *Min* lub większej od *Max*. Wzór opisujący funkcje znajduje się poniżej (9) :

$$f_5(P,i) = h\Big(\sum_{\mathrm{v_j} \in \mathrm{K}_{\mathrm{v_i}.k}} \big|\{G_i\} \cap \{G_j\}\big|\Big) \tag{9}$$

gdzie funkcja

$$h(x) = \begin{cases} Min_k - x & \text{gdy} \quad \mathrm{x} < Min_{\mathrm{k}} \\ 0 & \text{gdy} \quad \mathrm{Min_k} \le \mathrm{x} \le \mathrm{Max_k} \\ x - Max_k & \text{gdy} \quad \mathrm{x} > Max_{\mathrm{k}} \end{cases} \tag{10}$$

Parametry Min_k i Max_k zostały arbitralnie ustawione na wartości 4 i 8.

$f_6(P,i)$ – funkcja odpowiadająca za sprawdzanie liczby okienek w dniu zajęć nauczyciela, który prowadzi i-te zajęcia. Funkcja jest bardzo podobna do definicji funkcji czwartej (Algorytm 1.), z tym wyjątkiem że wyrażenie $\mathrm{v_m} \in \mathrm{K}_{\mathrm{v_i}.k}$ zostało zamienione na wyrażenie $\mathrm{v_m} \in \mathrm{N}_{\mathrm{v_i}.n}$. Tzn. brane pod uwagę są zajęcia które prowadzi ten sam nauczyciel, nie zaś zajęcia w których uczestniczy ta sama klasa.

$f_7(P,i)$ – funkcja odpowiadająca za sprawdzenie liczby zajęć danego nauczyciela w dniu.

Funkcja zdefiniowana jest następująco:

$$f_7(P,i) = h\Big(\sum_{v_j \in N_{v_i \cdot n}} \big|\{G_i\} \cap \{G_j\}\big|\Big) \tag{11}$$

gdzie funkcja $h(x) = \begin{cases} Min_n - x & gdy \quad x < Min_n \\ 0 & gdy \quad Min_n \le x \le Max_n \\ x - Max_n & gdy \quad x > Max_n \end{cases}$ (12)

$Min_n = 2; Max_n = 8$

Wagi dla poszczególnych funkcji zostały dobrane tak, aby zależały do wierzchołka, dla którego obliczana jest funkcja kary, tak więc $w_{1-7} = w_{1-7}(i)$. Zauważono również, że najważniejszym z kryteriów jest kryterium klasycznego *problemu kolorowania grafu*, dlatego też w_1 posiada największą wagę (równanie (13)). Wagi dla pozostałych *ograniczeń twardych* i *ograniczeń miękkich* opisane są równaniami (14) i (15).

$$w_1(i) = \text{INF} * \sum_{\{v_i, v_j\} \in E} C_j \tag{13}$$

$$w_2(i) = w_3(i) = w_4(i) = w_5(i) = \text{INF} * C_i \tag{14}$$

$$w_6(i) = w_7(i) = C_i \tag{15}$$

Ograniczenia od pierwszego do piątego muszą być spełnione aby plan był akceptowalny. Dlatego też te wagi otrzymały mnożnik INF (tzn. bardzo dużą liczbę), aby ich spełnienie było priorytetem dla algorytmów.

5. Metoda rozwiązania

Do rozwiązania zagadnienia układania planu zajęć użyto metody poszukiwania z zabronieniami (ang. *tabu search*) z listą nawrotów (ang. *backtrack-jump list*) . Metoda tabu jest modyfikacją metody lokalnych poszukiwań. Dopuszcza się możliwość zwiększania wartości funkcji celu, aby w ten sposób opuścić minimum lokalne i zwiększyć szanse na osiągnięcie minimum globalnego. Jednak ruchy te należy kontrolować, w przeciwnym przypadku nastąpiłyby szybkie powroty do otrzymanego wcześniej minimum lokalnego. W celu wyeliminowania tego zjawiska, a także skierowania poszukiwań w obiecujące regiony przestrzeni wprowadza się tzw. mechanizm zabronień. Wykonując ruch zapamiętuje się rozwiązania (lub częściej atrybuty rozwiązań) na tzw. liście tabu. Generując otoczenie rozwiązania znajdujące się na liście nie są uwzględniane chyba, że spełniają kryterium aspiracji. Podstawowymi elementami metody poszukiwania z zabronieniami są :

- Operator zmiany (ruch) – działanie nałożone na rozwiązanie bieżące w celu wygenerowania innego rozwiązania.
- Sąsiedztwo (otoczenie) – każde rozwiązanie, które da się uzyskać z rozwiązania

bieżącego wykorzystując operator zmiany.

- Lista tabu – lista atrybutów rozwiązań już odwiedzonych. Używana jest do chronienia algorytmu przed powrotem do rozwiązań już odwiedzonych.
- Kryterium aspiracji – warunki, przy spełnieniu których można pominąć ograniczenia wprowadzone przez listę tabu.
- Warunek zakończenia – warunek bądź warunki po spełnieniu których algorytm kończy swoje działanie.

Wprowadzona modyfikacja do podstawowej metody poszukiwania z zabronieniami, tj. mechanizm listy nawrotów pozwala na powrót do najlepszego rozwiązania otrzymanego do tej pory. Po powrocie algorytm rozpoczyna poszukiwania lepszego rozwiązania w innym kierunku niż do tej pory. Mechanizm ten szczególnie przydatny jest gdy generowane sąsiedztwo składa się z wielu elementów. Powrót do najlepszego rozwiązania następuje po spełnieniu warunku powrotu. Najczęściej warunkiem powrotu jest przekroczenie określonej liczby iteracji bez poprawy najlepszego rozwiązania.

Niech $x \in X$ będzie dowolnym rozwiązaniem dopuszczalnym, LT listą tabu, a x^* najlepszym do tej pory znalezionym rozwiązaniem (w pierwszym kroku przyjmujemy za x^* rozwiązanie początkowe x).

```
do
      Wyznaczyć otoczenie N_x rozwiązania x;
      Usunąć z N_x rozwiązania znajdujące się na LT,
            przy uwzględnieniu kryterium aspiracji
      Znaleźć rozwiązanie x' ∈ N_x takie że:
            F(x') = min{F(a): a ∈ N_x};
      if F(x') < F(x*) then x* := x';
      Umieść rozwiązanie x' na liście LT
      if Warunek_Powrotu then x := x*
            else x := x'
while Warunek_Końca
```

Algorytm 2. Algorytm poszukiwania z zabronieniami z listą nawrotów

Złożoność obliczeniowa algorytmu opartego na metodzie poszukiwania z zabronieniami zależy od wyboru poszczególnych jego elementów, tj. metody wyznaczania sąsiedztwa, długości a także od rodzaju elementów przechowywanych na liście tabu, sposobu wyliczenia wartości funkcji celu oraz warunków powrotu i zakończenia.

W powyższej pracy użyto algorytm poszukiwania z zabronieniami z następująco określonymi elementami:

- Operator zmiany – zmiana koloru jednego z wierzchołków, jest to równoznaczne ze zmianą terminu odbywania się lekcji.
- Lista tabu – na listę tabu zapisywany był parametr ruchu – tzn. zapisywany był numer i kolor wierzchołka na którym wykonywany był operator zmiany. Długość listy tabu wynosiła 30.
- Kryterium aspiracji – spełnione gdy otrzymane rozwiązanie jest lepsze od aktualnie najlepszego.
- Warunek powrotu – liczba iteracji bez poprawy, w trakcie badań wynosiła ona 10.
- Warunek zakończenia – liczba iteracji algorytmu (1000) lub liczba powrotów do rozwiązania najlepszego – trzy powroty.

6. Implementacja dla GPU

W celu skrócenia czasu działania algorytmu zaproponowano zrównoleglenie algorytmu na poziomie obliczenia funkcji celu. Wiązało się to z zrównolegleniem obliczenia każdej z poszczególnych funkcji celu opisanych w poprzednim rozdziale. Z powodu postaci poszczególnych funkcji celu zrównoleglenie wiązało się głównie z zmianą obliczenia sumy sekwencyjnej na obliczenia sumy równolegle (funkcje o indeksach 1-3, 5 i 7). W przypadku funkcji obliczającej liczbę okienek klasy bądź nauczyciela (funkcje o indeksach 4 i 6) zrównoleglenie pozwoliło na równoległe obliczenie liczby okienek dla poszczególnych terminów. W związku z faktem, że dla każdej poszczególnej funkcji celu należało obliczyć sumę pewnych wartości złożoność obliczeniowa została zmniejszona z $O(n)$ do $O(log\ n)$ gdzie n oznacza

- f_1 - liczbę sąsiednich wierzchołków,
- f_2 - liczbę lekcji tej samej klasy z jednego przedmiotu,
- f_3 - liczbę lekcji posiadających zajęcia w tej samej klasie,
- f_4 i f_6– liczbę dni,
- f_5i f_7- liczbę terminów.

Jak można zauważyć, dla pierwszych funkcji zysk asymptotycznie powinien być wysokiego rzędu, natomiast dla ostatnich funkcji zysk jest niewielki.

7. Eksperymenty obliczeniowe

Zaproponowana metoda zrównoleglenia obliczenia funkcji celu została zastosowana w opisanym wcześniej algorytmie poszukiwania z zabronieniami dla problemu układania planu zajęć dla szkół podstawowych, gimnazjalnych i średnich. Działanie algorytmu testowane było na instancjach generowanych na podstawie rzeczywistych danych (zebranych z placówek oświaty w Polsce) i uruchomionym na komputerze wyposażonym w 6-rdzeniowy procesor Intel Core i7 CPU X980 (3.33GHz) wyposażonym w nVidia Tesla S2050 GPU (1792 rdzeni) pracującym na 64-bitowym systemie operacyjnym Linux Ubuntu 10.04.4 LTS. Algorytmy równoległe zostały napisane przy użyciu technologii *CUDA*. Przeprowadzono dwa typy badań – czas działania całego algorytmu w zależności od rozmiaru instancji (liczby lekcji), a także zaprezentowano czas obliczania poszczególnych składowych funkcji celu. Wyniki przedstawiono w Tabelach 1. i 2.

Tab. 1. Czas działania algorytmu w zależności od liczby lekcji

Liczba lekcji	*t*[s]			Przyspieszenie	
	TS-CS	*TS-GS*	*TS-GR*	Absolutne	Względne
156	5,1	18,3	8,4	0,60	2,18
500	23,0	108,1	30,0	0,77	3,60
1000	69,2	367,5	70,4	0,98	5,22
1500	143,6	798,0	119,1	1,21	6,70
2000	241,0	1613,0	182,0	1,32	8,86

Poszczególne kolumny Tabeli 1. oznaczają:

- *t* – czas działania algorytmu w danej konfiguracji,
- *TS-CS* – algorytm poszukiwania z zabronieniami z sekwencyjnie obliczaną funkcją celu na *CPU* (*Central Processing Unit*),

- *TS-GS* – algorytm poszukiwania z zabronieniami z sekwencyjnie obliczaną funkcją celu na *GPU* (*Graphics Processing Unit*),
- *TS-GR* - algorytm poszukiwania z zabronieniami z równolegle obliczaną funkcją celu na *GPU*.
- Przyspieszenie absolutne – stosunek czasów działania algorytmu równoległego na *GPU* do sekwencyjnego na *CPU*.
- Przyspieszenie względne – stosunek czasów działania algorytmu równoległego na *GPU* do sekwencyjnego na *GPU*.

Otrzymane rezultaty pokazują, że użycie równoległego algorytmu poszukiwania z zabronieniami poprawia czas działania algorytmu. Wynika to z porównania wyników działania algorytmów obliczanych na tym samym urządzeniu (w tym przypadku na *GPU*). Przyspieszenie algorytmu rośnie wraz z rozmiarem instancji co wskazuje na poprawny zapis zrównoleglenia. Z drugiej strony widać, że dla małych instancji algorytm wykonuje się szybciej sekwencyjnie na *CPU* niż równolegle na *GPU*. Powodami takiej sytuacji jest przede wszystkim większy czas komunikacji i mniejsza częstotliwość taktowania na urządzeniu graficznym.

Tab. 2. Czas działania algorytmu w zależności od liczby lekcji

Składowa funkcja celu	*t*[s]		
	TS-CS	*TS-GS*	*TS-GR*
f_1	207,38	575,33	34,00
f_2	5,02	501,51*	31,72
f_3	6,08	403,54*	23,88
f_4	6,16	28,28	23,98
f_5	4,93	22,02	22,78
f_6	6,37	28,28	23,66
f_7	4,87	221,8	22,34

*znaczny wzrost czasu wynika z konieczności zastosowania innej metody wynikającej z ograniczeń technologii *CUDA*

Poszczególne kolumny Tabeli 2. oznaczają:

- *t* – czas działania algorytmu poszukiwania z zabronieniami w danej konfiguracji
- *TS-CS* – algorytm poszukiwania z zabronieniami z sekwencyjnie obliczaną funkcją celu na *CPU* (*Central Processing Unit*)
- *TS-GS* – algorytm poszukiwania z zabronieniami z sekwencyjnie obliczaną funkcją celu na *GPU* (*Graphics Processing Unit*)
- *TS-GR* - algorytm poszukiwania z zabronieniami z równolegle obliczaną funkcją celu na *GPU*

Na podstawie zamieszczonych wyników w Tabeli 2. można stwierdzić, że w przypadku sekwencyjnego algorytmu, głównym czynnikiem wpływającym na czas działanie jest obliczenie f_1. Sytuacja wygląda odmiennie dla równoległego algorytmu gdzie czas obliczenia poszczególnych składowych funkcji celu jest zbliżony. Można zauważyć, że dla funkcji o indeksach 4-7 czas działania algorytmu sekwencyjnego i równoległego jest zbliżony.

8. Wnioski

W pracy zaproponowano zrównoleglenie algorytmu poszukiwania z zabronieniami dla problemu układania planu zajęć wykorzystujące architekturę GPU. Zrównoleglono część algorytmu odpowiadającą za obliczanie funkcji celu. Rozwiązanie takie pozwala na poprawę czasu działania algorytmu, jednak nie wpływa na otrzymane rezultaty. Wyniki wskazują na rzeczywisty zysk ze zrównoleglenia algorytmu (porównując wykonywania obliczeń sekwencyjnie i równolegle na tym samym urządzeniu). Z drugiej strony dla małych instancji czasy wykonywania algorytmu równoległego na urządzeniu GPU były gorsze niż czasy wykonania algorytmu sekwencyjnego na procesorze CPU. Wskazuje to na dużą różnicę w działaniu tych dwóch typów urządzeń, aczkolwiek wraz z rozwojem technologii różnice te powinny być coraz mniejsze, co pozytywnie wpłynie na opłacalność użycia zrównoleglonej wersji algorytmu poszukiwania z zabronieniami.

Literatura

1. Adenso-Diaz B.: Restricted neighborhood in the tabu search for the flow shop problem. European Journal of Operational Research, 62, 1992, 27-37.
2. Ahandani M., Baghmisheh M., Zadeh M., Ghaemi S.: Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem. Swarm and Evolutionary Computation, 7, 2012, 21–34.
3. Alvarez-Valdes R., Crespo E., M. Tamarit J.: Design and implementation of a course scheduling system using Tabu Search. European Journal of Operational Research, 137, 2002, 512–523.
4. Burke E., McCollum B., Meisels A., Petrovic S., Qu R.: A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research, 176, 2007, 177–192.
5. Grabowski J., Wodecki M.: A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Computers & Operations Research, 31, 2004, 1891-1909.
6. Nowicki E., Smutnicki C.: A fast tabu search algorithm for the permutation flow-shop problem. European Journal of Operational Research, 91, 1996, 160-175.
7. Salwani A., Hamza T.: On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems. Information Sciences, 191, 2012, 146–168.
8. Socha K., Sampels M., Manfrin M.: Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. Proceedings of EvoCOP 2003, 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization, LNCS, 2611, 2003, 334–345.
9. Zhipeng L., Jin-Kao H.: Adaptive Tabu Search for course timetabling. European Journal of Operational Research, 200, 2010, 235–244.

Dr hab. Wojciech BOŻEJKO
Mgr inż. Łukasz GNIEWKOWSKI
Instytut Informatyki, Automatyki i Robotyki
Politechnika Wrocławska
50-370 Wrocław, Wyb. Wyspiańskiego 27

tel./fax: 71 320 27 45 / 71 321 26 77
e-mail: wojciech.bozejko@pwr.wroc.pl
lukasz.gniewkowski@pwr.wroc.pl