Streszczenie pracy doktorskiej:
Combinatorial optimization problems in Internet applications
Problemy optymalizacji kombinatorycznej w aplikacjach internetowych

Jakub Marszałkowski

W pracy zaproponowano i rozwiązano trzy nowatorskie zagadnienia badawcze, których głównymi wspólnymi cechami było pochodzenie z rzeczywistych zastosowań aplikacji internetowych oraz użycie optymalizacji kombinatorycznej, a w szczególności algorytmów pakowania dwuwymiarowego, dla poprawy jakości działania. Elementów wspólnych jest jednak znacznie więcej: wszystkie rozpatrywane problemy są obliczeniowo trudne, należą do klasy problemów NP-trudnych i jako takie w praktyce muszą być rozwiązywane algorytmami heurystycznymi. Wspomniane trzy zagadnienia badawcze opisano poniżej.

1 Optymalizacja układu szerokości kolumn strony internetowej w celu umieszczania reklam

Jednostka reklamowa to format reklamy uzgodniony między reklamodawcą, wydawcą a siecią reklamową. Jest to prostokąt o zadanych wymiarach. Wydawca musi przygotować strony internetowe tak, żeby mieć możliwie wiele dogodnych możliwości umieszczania jednostek reklamowych, ale też by zachować estetyczny wygląd całości. Witryny mają układ kolumn z przewijaniem pionowym, w którym szerokości są zadane na stałe, zaś wysokość jest właściwie nieograniczona. Obecnie stosowana jest metoda dobierania szerokości kolumn ad-hoc. Podział całkowitej szerokości strony na kolumny sformułowano jako problem optymali-

zacyjny. Reklamy mogą być umieszczane w kolumnach w grupach, które przez ograniczenia HTML podobne są do problemu cięcia gilotynowego. Szerokości kolumn, które w modelu matematycznym są zmiennymi decyzyjnymi, będą dostosowywane do możliwie wszechstronnego mieszczenia takich grup reklam.

W optymalizacji użyto trzech funkcji celu, które odzwierciedlają kolejno: 1) Elastyczność układu kolumn, tj. jak wiele różnych kombinacji reklam jest on w stanie zmieścić. 2) Zdolność mieszczenia najmniej wygodnej, tzn. najszerszej jednostki reklamowej. 3) Minimalizację marnowanego miejsca, przy pesymistycznym założeniu umieszczania tylko jednej reklamy. Aby właściwie odzwierciedlić wielokryterialność problemu, jako wyniki podawano zarówno wszystkie rozwiązania niezdominowane w sensie Pareto, jak i jedno rozwiązanie najlepsze, uzyskanie przez nadanie funkcjom celu wag. Dla wyznaczenia wag przeprowadzono ankietę wśród 21 ekspertów, profesjonalistów zajmujących się stronami internetowymi.

Zaproponowany model ma aż 13 ograniczeń, począwszy od tak oczywistych jak nie przekraczanie ograniczeń szerokości strony, poprzez ograniczające redundancję rozwiązań, skończywszy na takich które mają odzwierciedlać wymagania estetyczne w łączeniu reklam w grupy, czy pozostawiania pustej przestrzeni. Model poprzez wartości wejściowe parametrów używanych w ograniczeniach pozwala na znaczne dostosowanie wyników do potrzeb konkretnego web-developera. Zaproponowany algorytm działa w czterech etapach:

1. Jednostki reklamowe są łączone, tworząc wszystkie możliwe kombinacje.
2. Tworzona jest lista dopuszczalnych szerokości kombinacji i częściowe wartości dla późniejszego wyliczenia funkcji celu.
3. Generowane są wszystkie dopuszczalne układy szerokości kolumn i ostateczne wyniki funkcji celu.
4. Z tej listy wybierane są najlepsze rozwiązanie i listy rozwiązań Pareto-optymalnych.

Etapy 2. i 4. to w gruncie rzeczy dość trywialne przeglądanie listy wartości dostarczonych przez poprzednie etapy. Dla etapu 1. użyto zmodyfikowanego algorytmu Wanga dla dwuwymiarowego rozkroju z ograniczeniami. Modyfikacje polegały na dostosowaniu go do potrzeb rozważanego problemu, oraz na poprawkach wydajnościowych, w szczególności ograniczeniu generowania duplikatów i sprawniejszego ich wyszukiwania i usuwania. Dla etapu 3. skonstruowano algorytm przeglądający układy kolumn w możliwie sprawny sposób, przez wykorzystanie ograniczeń zbioru możliwych szerokości. Algorytm ma złożoność wykładniczą, jednak wobec ograniczonych rozmiarów szerokości strony, jest to rozwiązaniem akceptowalnym.

Dla potrzeb eksperymentów obliczeniowych, zaproponowano jako benchmarki zestawy jednostek reklamowych trzech popularnych sieci reklamowych oraz zestaw rekomendacji jednostek reklamowych Internet Advertising Bureau. Wartości wejściowe parametrów zostały dobrane tak, by w miarę możliwości jak najlepiej odzwierciedlały rzeczywiste warunki. Algorytm na przeciętnym komputerze biurowym generował wyniki w czasach od kilku milisekund o 160 sekund, co jest rezultatem bardzo dobrym dla czynności, która wykonywana jest tylko raz przy tworzeniu strony internetowej. Czas działania algorytmu może się zmieniać w zależności od parametrów wejściowych, co zostało przebadane.

Prezentowane wyniki składają się z najlepszego układu kolumn i zestawu układów Pareto-optymalnych dla kolejnych benchmarków i różnych szerokości stron. Zakresy wartości przyjmowanych przez trzy składowe funkcje celu świadczą o ich czułości. Z wyników można też wyciągnąć dodatkowe wnioski. Na przykład, zestaw reklam Adbrite zawierający tylko pięć jednostek jest zbyt mały i jego użycie jest utrudnione. Z kolei dla wielu innych zestawów wyniki wykazują, że zmienienie układu kolumn o kilka pixeli może znacznie poprawić elastyczność umieszczania reklam.

Wnioski końcowe wskazują dodatkowe przyszłe kierunki badań. Możliwa jest próba zastosowania danych o częstości stosowania jednostek reklamowych, np. w danym kraju, w celu znalezienia układów kolumn mających pewną wszechstron-

ną stosowalność. Z kolei odwrócenie problemu tj. badanie zestawów jednostek reklamowych, pozwoliłoby stwierdzić które z nich mają braki, jakie jednostki należałoby dodać dla najlepszego efektu, nawet pozwoliłyby na pokuszenie się na stworzenie od podstaw kompletnego zestawu. Badanie można też uogólniać do problemu podziału dwuwymiarowej przestrzeni w jednym wymiarze, czyli na kolumny. Takie zagadnienie może znaleźć zastosowanie w planowaniu struktury portu, czy cięciu szerokich bel papieru produkowanych przez fabryki na węższe, dające się transportować i składować.

2 Budowanie chmur tagów dla zastosowań internetowych

Tag to inaczej fraza reprezentująca tekstowo jakiś obiekt. Tagi mają przypisaną wartość istotności w relacji do innych tagów. Chmura tagów to ułożenie tagów na płaszczyźnie z wizualizacją graficzną ich istotności zazwyczaj przez większy rozmiar. Celem badania było rozwiązanie problemu konstrukcji estetycznych i czytelnych chmur tagów.

Na potrzeby przeglądu literaturowego zaproponowana została taksonomia klasyfikacji chmur tagów w oparciu o 5 parametrów i zakresy ich wartości:

1. Sortowanie tagów. Dostępne opcje to: alfabetycznie, według znaczenia, kontekstowo, losowo, bądź kolejność układana przez algorytm pakowania.
2. Kształt całej chmury. Możliwe opcje: prostokątny, inny kształt regularny (np. okrągły), nieregularny, zadany (np. zadany wielobok, granice mapy).
3. Kształt samych tagów. Opcje: prostokąt, lub kształt znaków.
4. Obracanie znaczników. Opcje: brak, swobodne, dozwolone z ograniczeniami.
5. Wyrównanie w pionie. Opcje: użycie typograficznych linii podstawowych, ograniczone przez właściwości algorytmu (np. grupowanie tagów), swobodne.

Przeanalizowano 14 chmur tagów z literatury, zarówno w zakresie tych parametrów jaki użytych algorytmów oraz zastosowań chmury. Przeanalizowano również literaturę dotyczącą badań użyteczności chmur tagów.

Następnie przeprowadzono analizę wymagań i zaleceń dla chmur tagów, które mają być używane na stronach internetowych. Wymagania i zalecenia te wynikają zarówno z ograniczeń zasad konstrukcji i technologii (np. HTML, CSS) samych stron internetowych, fragmentacji rynku klientów jak i tego, że strony mają być czytelne zarówno dla ludzi, jak i dla robotów indeksujących. Wynikające z analizy rekomendacje i decyzje dotyczące chmur tagów dla WWW są następujące: 1) chmura jest prostokątna, 2) tagi są traktowane jak prostokąty, 3) algorytm pakowania ustala kolejność tagów, 4) obracanie tagów nie jest dopuszczalne, 5) znaczniki umieszczane są na linii bazowej (na półkach), 6) realizowana winna być minimalizacja marnowanej przestrzeni w prostokącie chmury. Chociaż może się wydawać, że w większości przypadków dokonano wyborów najprostszych, nadal wyjściowy problem optymalizacyjny jest NP-trudny, jako że jest szczególną wersją problemów bin- lub strip-packing.

Dla uzyskania estetycznego wyglądu chmury zastosowano regułę typograficznej równomiernego koloru typograficznego, tj. tekst tak ułożony powinien wizualizować się np. jako możliwie jednorodna masa szarości. Osobnym wymaganiem jest uruchamianie algorytmu konstruowania chmur po stronie użytkownika. Wykazało to badanie rozmiarów tagów przeprowadzone na 4201 użytkownikach rzeczywistej strony internetowej, w którym zidentyfikowano 112 kombinacji rozmiarów tagów wynikających z różnych czcionek dostępnych na urządzeniach i różnić w ich renderowaniu. Dodatkowo chmury powinny za każdym razem być generowane w tym samych wyglądzie, aby nie powodować konsternacji dla użytkownika, a więc przez algorytm deterministyczny, oraz ze względu na wymagania szybkości wyświetlania stron internetowych w czasie rzędu dziesiątych części sekundy.

Przeprowadzona została analiza problemu pakowania rozwiązywanego przy budowaniu chmur tagów. Mamy tu do czynienia z problemem typu strip-packing,

w którym szerokość prostokąta jest ustalona, zaś wysokość może być zmieniana w miarę potrzeb przez przesuwanie elementów strony www znajdujących się pod chmurą. Następnie przedstawione zostało matematyczne sformułowanie problemu. Dla każdego tagu możliwe jest wyliczenie jego zaczernienia, a z tagów także dla każdej półki. Minimalizowana winna być potęga k różnicy między zaczernieniem półki a maksymalnym możliwym, sumowana po wszystkich półkach w chmurze. Wykładnik potęgi k został wyznaczony eksperymentalnie. Drugim elementem do dobranym eksperymentalnie był sposób reprezentowania zaczernienia półek, który może być masą, czyli sumą czarności pixeli w tagach lub gęstością, czyli masą dzieloną przez powierzchnię.

Eksperymenty obliczeniowe przeprowadzono za pomocą specjalnie zaprojektowanego algorytmu typu Branch and Bound, pozwalającego rozwiązać problem do optymalności. Ponieważ jednak algorytm ten jest wykładniczy, może rozwiązywać w akceptowalnym czasie ograniczone rozmiary instancji, wstępne testy ograniczono do 16 tagów. Wyniki zostałuy wykorzystane dla zmierzenia dystansu od rozwiązań optymalnych oraz dostrojenia funkcji celu. Strojenie przeprowadzono przez wygenerowanie 55 testowych chmur tagów w 6 kombinacjach parametrów i poddanie ich ocenie pięciu ekspertów. Na tej podstawie oceniono, że gęstość jest bardziej czułym parametrem niż masa, zaś dla gęstości najlepiej ocenianą wratością k jest 0,5. W dalszych pracach użyto więc tych parametrów funkcji celu.

Dla rozwiązania problemu zaproponowano też specjalny algorytm zachłanny. W oparciu o jego bazową wersję z wykorzystaniem 8 różnych reguł sortowania tagów (po masie, gęstości, wysokości i szerokości) oraz 4 różnych reguł wybierania półek (best fit, worst fit, najmniejsza oraz największa masa tonalna) oraz dwóch dodatkowych modyfikacji możliwe było 8*4*2*2 wersji algorytmu zachłannego. Całość postanowiono wykorzystać jako algorytm Super Fit, którego główną zaletą jest mniejsze prawdopodobieństwo wpadnięcia w pułapkę przypadków pesymistycznych. Jako ostatnią metodę rozwiązania problemu algorytmu przygotowano algorytm Tabu Search. Algorytm startuje z najlepszego

rozwiązania z Super Fit i następnie przeszukuje przestrzeń lokalnie wykorzystując tablicę ruchów tabu, w celu ominięcia już odwiedzanych rozwiązań.

Do eksperymentów obliczeniowych wykorzystano rzeczywiste chmury tagów pobrane z działających stron internetowych. Algorytm Super Fit rozwiązuje instancje do 142 tagów w czasie do 57ms (średnio 17ms). Parametry algorytmów zachłannych zastosowanych w Super Fit powodowały różną jego efektywność, 52 z 128 nigdy nie wyprodukowało rozwiązania, które byłoby lepsze od rozwiązań innych algorytmów i mogłyby zostać usunięte z pakietu, gdyby była taka potrzeba. Algorytm Tabu Search po strojeniu, wykonując 300 iteracji, działał średnio w czasie 170ms. Różnice numeryczne w wartości funkcji celu są niewielkie, jeżeli algorytm uzyska minimalną możliwą liczbę półek.

3 Pakowanie CSS-sprite

CSS-sprite to technika umieszczenia wielu grafik stanowiących elementy strony WWW na jednym obrazku (nazywanym właśnie CSS-sprtite) w celu zmniejszenia liczby zapytań do serwera. Fragmenty tego obrazka są wyświetlane za pomocą reguł CSS w miejscu oryginalnych grafik.

Rozwiązanie problemu rozpoczęto od analizy wyzwań związanych z pakowaniem CSS-sprite. Wyzwania natury geometrycznej związane są z układaniem grafik na CSS-sprite czyli z problemem pakowania. Występujący tu problem pakowania jest nietypowej natury ponieważ przestrzeń, do której będą pakowane elementy nie ma zadanych żadnych wymiarów. Zamiast tego poszukiwana jest najmniejsza powierzchnia w której upakowane mogą zostać elementy. Techniki kompresji grafiki stwarzają kolejne wyzwania, wpływając na rozmiar plików docelowych w sposób nie możliwy do przewidzenia. Pliki mają różne głębie kolorów, odzwierciedlane za pomocą różnej liczby bitów na pixel. Dodatkowo kompresja PNG będzie osiągać lepsze rezultaty, jeżeli w obrazku będą dłuższe ciągi punktów o jednakowym kolorze. A to może zależeć od wzajemnego położenia grafik. Z kolei kompresja JPEG jest stratna, ale też pixele z sąsiadujących grafik na jednym obrazku mogą na siebie wzajemnie wpływać. Oba formaty nadają

się też lepiej dla różnych typów grafik i mają wiele innych parametrów decydujących o rozmiarach obrazków i inych własnościach. Dalsze wyzwania są natury obliczeniowej. W pracy przedstawiony został dowód, że zarówno problem wyboru zbioru grafik dla wspólnej palety kolorów o ograniczonym rozmiarze jak i problem umiejscowienia obok siebie grafik tak, by maksymalizować sąsiadowanie takich samych kolorów są NP-trudne. Wreszcie wydajność komunikacji, przesyłu CSS-sprite między serwerem a przeglądarką nie jest znana. Wpływa na nią wiele czynników począwszy od parametrów serwera, łącza i klienta, poprzez parametry przesyłanych plików, a skończywszy na użytym algorytmie szeregowania pakietów. Jako przybliżenie tego procesu zaproponowana została zależność wykorzystująca algorytm McNaughtona. Zaproponowana metoda wyliczenia czasu komunikacji ma jednocześnie niski koszt akceptowalny dla zastosowania w praktyce, jak i akceptowalną dokładność.

Następnie problem został sformułowany jako model matematyczny. Dany zestaw grafik ma być umieszczonych na CSS-sprite, a umiejscowienie grafik, jak i liczba CSS-sprite są zmiennymi decyzyjnymi. Funkcja celu zakłada minimalizowanie czasu przesyłu CSS-sprite dla zadanych parametrów łącza, w tym przyspieszenia wynikającego z zrównoleglenia przesyłania.

Przed przystąpieniem do dalszych prac przeprowadzono szereg dodatkowych eksperymentów. W pierwszym z nich testowano wpływ kształtu CSS-sprite i wzajemnego umiejscowienia grafik na rozmiar plików. Przygotowane grafiki układano na CSS-sprite o wszystkich możliwych kształtach, począwszy od bardzo długich ale niskich, przez zbliżone do kwadratu, a skończywszy na bardzo wysokich ale wąskich. Jednocześnie testowano 200 permutacji wzajemnego ułożenia grafik. Z 36 testowych zestawów grafik 17 silnie preferowało dla układ długi, 14 układów wysoki, a 5 nie wykazało preferencji. Przez dobór właściwego układu rozmiar CSS-sprite może być zmniejszony o 2% do 35%. Nie znaleziono czynników pozwalających określać preferencję inaczej niż eksperymentalnie. Jednakże po znalezieniu właściwego układu permutacje kolejności grafik pozwalały na zysk mniejszy niż 1,5%. Uznano, że dla grafik typu PNG istotne

będzie testowanie obu układów, zaś kolejność ułożenia grafik można pominąć. Nie stwierdzono podobnych zależności w plikach JPEG.

Drugi eksperyment miał na celu znalezienie przykładowych parametrów wydajności komunikacji i sprawdzenie przyspieszenia wynikającego z równoległego pobierania danych. W tym celu skonstruowano skrypt do pomiarów, który umieszczono na działającej stronie internetowej z rzeczywistym ruchem i zebrano pomiary z 17460 unikatowych adresów IP. Wykazane zostało, że przeglądarki są zdolne do równoległego pobierania. Co najmniej dwa kanały stwierdzono w 100% z nich, zaś przeszło połowa miała ich więcej niż 7. Z kolei przyspieszenie z pobierania równoległego wynosiło średnio od 36% dla trzech kanałów do 77% dla 9 kanałów.

Następnie przystąpiono do analizy dostępnych rozwiązań generujących CSS Sprite. Zidentyfikowano przeszło 30 gotowych programów. Część z nich nie mogła zostać włączona do dalszych eksperymentów, ponieważ były zamkniętymi rozwiązaniami dostępnymi tylko dla konkretnej technologii, np. serwera, bądź nie dało się ich uruchomić, np. przez niedziałające strony internetowe. W pozostałej grupie można było jeszcze wyróżnić rozwiązania nie stosujące żadnych algorytmów pakowania, układających grafiki po prostu jedna obok drugiej. Wreszcie ostatnia grupa to narzędzia do tworzenia CSS-sprite, używające algorytmów pakowania, czasem dość zaawansowanych, dla minimalizowania wymiarów CSS-sprite. Wszystkie znalezione rozwiązania generują dokładnie jednego CSS-sprite, nie biorą pod uwagę odkrytych zależności formatów kompresji, nie optymalizują rozmiaru pliku, nie optymalizują czasu pobierania, które to cechy są głównymi nowościami proponowanego rozwiązania. Niektóre ze znalezionych programów używają postprocessingu algorytmów pakowania dla możliwego poprawienia kompresji i zmniejszenia rozmiaru wyjściowego pliku.

Jako alternatywę zaproponowano algorytm SpritePack działa w czterech etapach:

1. klasyfikacja grafik – w którym testowane są ich parametry takie jak głębia kolorów i podatność na kompresję,

2. pakowanie geometryczne - w którym obrazki są wstępnie grupowane w zadaną liczbę k grup na podstawie pasowania do siebie wymiarami w pakowaniu geometrycznym i zbieżnopści parametrów grafik poznanych w klasyfikacji,

3. pakowanie z kompresją obrazu – grupy z poprzedniego kroku są łączone dalej, pakowane 2-wymiarowo i testowo kompresowane, łączenie w grupy ma charakter algorytmu zachłannego, a funkcją celu jest oszacowanie czasu pobierania,

4. postprocessing – wykonywane jest dodatkowe ulepszenie kompresji.

W procesie pakowania 2D testowane były algorytmy: First-Fit Decreasing Height (oraz w wersji z Fit2), Best-Fit Decreasing Height (oraz w wersji z Fit2), Bottom-Left, Modified Bottom Left oraz Variable Height Left Top. W praktyce dwa ostatnie radziły sobie najlepiej w 99% przypadków, choć oczywiście wiązało się to z pewnym kosztem czasowym, zwłaszcza w porównaniu do algorytmów zachłannych otwierających tą listę. Docelowo do tej dwójki postanowiono dołączyć jeszcze First-Fit Decreasing Height Two-Fit, który radził sobie najlepiej w większości instancji niezdominowanych przez tamte dwa.

Na potrzeby eksperymentów obliczeniowych przygotowano 32 instancje testowe będące zestawami grafik ze skórek dla popularnych aplikacji webowych w otwartym kodzie. Wstępne testy posłużyły to strojenia parametrów pracy SpritePacka, w szczególności parametr k, który wydatnie wpływa na czas wykonania najkosztowniejszego trzeciego etapu. Eksperymentalnie ustalono $k = 10$. Następnie przeprowadzono na pięciu zestawach testowych porównanie z dostępnymi rozwiązaniami zmuszając SpritePack (dla uzyskania zgodności formy rezultatów) do generowania dokładnie jednego CSS-sprite. Tylko dla jednego z zestawów testowych tylko jedno z konkurencyjnych rozwiązań dało lepszy wynik. Do dalszych testów wybrano cztery najlepsze rozwiązania, które były gorsze od SpritePack śrefunkcji celudnio o 14-33%. Tym razem porównywano funkcję celu, a więc czas pobierania CSS-sprite, ale także rozmiary plików CSS-sprite na

wszystkich 32 instancjach testowaych. Dla rozmiarów plików, optymalizowanych przez SpritePack pośrednio, był on lepszy średnio o 38-43%, jednak zdarzały się przypadki, w których konkurencyjne rozwiązania dawały rozwiązania lepsze o do 18%. W funkcji celu SpritePack był średnio lepszy od każdego z konkurentów o przynajmniej 31% i nigdy nie wystąpił przypadek, żeby konkurencja była lepsza od SpritePack. Na koniec przeprowadzono jeszcze eksperyment z wygenerowanymi CSS-sprite na rzeczywistym serwerze porównując CSS-sprite generowane przez SpritePack z zestawem obrazków przesyłanych bez użycia CSS-sprite i z rozwiązaniem generowanym przez najlepszego z konkurentów. Na czterech testowanych zestawach grafik Spritepack zmniejszał czasy pobierania o 350ms do 2.4s w porównaniu z brakiem CCC-sprite, podczas gdy najlepsza z rozwiązań konkurencyjnych zaledwie o 140-800ms. Eksperyment ten potwierdził też, że użyty model, w szczególności prognozowanie czasów pobierania za pomocą funkcji McNaughtona oraz zmierzone przyspieszenie pobierania równoległego, są poprawne. Współczynnik korelacji między medianami zmierzonego czasu pobierania a prognozowanymi przez funkcję celu wynosił 0,952, a jego wartość p była poniżej 2E-06.