POLSKA AKADEMIA NAUK
Instytut Badań Systemowych

ANALIZA SYSTEMÓW PRZESTRZENNYCH

WYBRANE ZAGADNIENIA

Redakcja

Jan W. Owsiński

Warszawa 2010

ANALIZA SYSTEMÓW PRZESTRZENNYCH. WYBRANE ZAGADNIENIA Jan W. Owsiński (Red.)

ANALIZA SYSTEMÓW PRZESTRZENNYCH

WYBRANE ZAGADNIENIA

Polska Akademia Nauk • Instytut Badań Systemowych

Seria: BADANIA SYSTEMOWE
tom 67

Redaktor naukowy:
Prof. dr hab. inż. Jakub Gutenbaum

Warszawa 2010

INSTYTUT BADAŃ SYSTEMOWYCH
POLSKIEJ AKADEMII NAUK

ANALIZA SYSTEMÓW PRZESTRZENNYCH

WYBRANE ZAGADNIENIA

Redakcja
Jan W. Owsiński

Warszawa 2010

Autorzy:
Jan W. Owsiński, redaktor
Instytut Badań Systemowych PAN
Pracownia Zastosowań Metod Badań Systemowych
Tel. (48 22) 3810 213
e-mail: Jan.Owsinski@ibspan.waw.pl

Jan Gadomski
Jerzy W. Hołubiec
Barbara Mażbic-Kulma
Michał Milczewski
Jan W. Owsiński
Grażyna Petriczek
Aneta M. Pielak
Henryk Potrzebowski
Krzysztof Sęp
Eugeniusz Sobczak
Jarosław Stańczak

Recenzenci:

Prof. dr hab. inż. Jacek Mercik
Prof. dr hab. Tadeusz Trzaskalik

ISBN 9788389475251
ISSN 0208-8029

Redakcja i opracowanie techniczne: Jan W. Owsiński i Aneta M. Pielak

VI. Heurystyczne i ewolucyjne metody znajdowania pokrycia grafu, korzystające z pojęcia α -kliki i innych ograniczeń

Henryk Potrzebowski, Krzysztof Sęp, Jarosław Stańczak

VI.1. Wstęp

Klastering, jako problem formułowany w odniesieniu do grafu nieskierowanego, polega na rozbiciu skończonego zbioru jego wierzchołków na rozłączne podzbiory – klastry. Podział taki może uwzględniać różne kryteria przy istniejących powiązaniach wierzchołków w klastrach. W rozpatrywanym w tej pracy przypadku, kryterium podziału definiowane jest, najogólniej, w taki sposób, że wierzchołki wchodzące w skład klastrów są relatywnie silniej ze sobą powiązane niż wierzchołki znajdujące się w różnych klastrach (por. zadanie analizy skupień, czyli „klasteryzacji", sformułowane w Rozdziale III). Powiązanie to może dotyczyć zarówno liczby połączeń między wierzchołkami, jak i wag połączeń oraz złożenia tych kryteriów.

Omawiany problem posiada wiele zastosowań w szeregowaniu zadań, wyszukiwaniu informacji, projektowaniu wyrobów, sieci teleinformatycznych i komunikacyjnych oraz w innych dziedzinach. Wiele z tych zastosowań ma charakter jawnie przestrzenny (transport, komunikacja, logistyka, a także organizacja, uwzględniająca wymiar przestrzenny). Jest to zagadnienie trudne obliczeniowo, a jego formułowanie i rozwiązywanie może nastręczyć wiele problemów.

W pracy prezentowane są dwa podejścia do klasteryzacji grafu. Pierwsze z nich oparte jest na ideach α-kliki i β-kliki jako pojęć rozszerzających klasyczne pojęcie kliki. Zostały one wykorzystane do skonstruowania efektywnych metod klasteryzacji grafu. Osiągnięto to dzięki zastosowaniu prostych algorytmów typu „greedy" (zachłannych) o wielomianowej złożoności obliczeniowej oraz algorytmów ewolucyjnych (AE). Niewątpliwie pozytywną cechą AE jest to, że są w stanie uwzględnić wiele uwarunkowań logicznych, z reguły trudnych do spełnienia w metodach programowania całkowitoliczbowego, oraz to, że mogą poprawiać rozwiązania otrzymywane przy pomocy prostych algorytmów zachłannych, choć oczywiście najczęściej kosztem zwiększonego czasu obliczeń.

Druga z prezentowanych metod polega na potraktowaniu macierzy sąsiedztwa lub wag połączeń w grafie jako macierzy DSM (Decision Structure Matrix) i zastosowaniu do jej przetworzenia algorytmów grupujących silnie powiązane węzły, w omawianym przypadku również wykorzystujących metody ewolucyjne, a następnie prostym wydzieleniu klastrów.

O ile metody oparte na pojęciach pochodnych kliki można wykorzystywać jedynie w przypadkach typowo grafowych, w których macierz opisująca problem jest kwadratowa i można założyć, że każdy węzeł takiego grafu zawsze jest połączony sam ze sobą (tworzy klikę), to metodę opartą na tablicach DSM można również stosować do rozwiązywania zadań, w których macierz opisująca związki między atrybutami grupowanych obiektów nie jest kwadratowa. Nie należy tego stwierdzenia rozumieć jako podkreślenia wyższości jednej z metod, a jedynie jako stwierdzenia, że jedna z nich może mieć nieco szerszy zakres stosowalności.

VI.2. Pojęcia podstawowe

VI.2.1. Ogólne zdefiniowanie problemu klasteryzacji w grafie

Mamy zbiór obiektów $V=\{v_1, \ldots, v_n\}$ oraz funkcję E: $V \times V \rightarrow W$, gdzie W jest skończonym, niepustym podzbiorem N, do którego dodajemy zbiór jednoelementowy $\{0\}$.

Naszym zadaniem jest podać pokrycie zbioru V zbiorami $V_1, .., V_m$ takimi, by

1. $V_1 \cup V_2 \cup \ldots \cup V_m = V$ oraz dla dowolnego $i \neq j$ $i,j=1, \ldots, m$, $V_i \cap V_j = \varnothing$
2. dla każdego elementu zbioru V_i liczba przypadków, gdy funkcja E dla argumentów ze zbioru $V_i \times V_i$ przyjmuje wartość większą od zera, była jak największa.

Zbiór V możemy interpretować jako zbiór wierzchołków grafu, a funkcję E jako zbiór krawędzi grafu.

Wiele praktycznych problemów klasteringu przedstawianych jest nie jako problem dla grafu, ale jako problem dla tablicy, której wiersze najczęściej odpowiadają obiektom, a kolumny ich atrybutom. Tablica taka znana jest pod nazwą DSM (*Decision Structure Matrix*, Browning, 2001). W tym przypadku istotne jest to, że większość problemów DSM można równoważnie

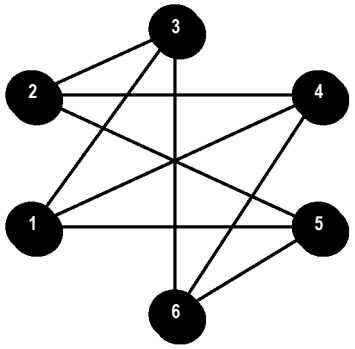
przekształcić w problemy dla grafu pod warunkiem, że dla par obiektów użyta będzie stosowna miara odległości. W praktyce, DSM można przedstawić jako specyficzny problem klasteryzacji grafu, w którym stosuje się metodę preprocesingu tablicy danych maksymalizującą kryterium jakości (VI.5), a następnie wydzielenie powiązanych grup – klastrów.

VI.2.2. Podstawowe pojęcia związane z grafami

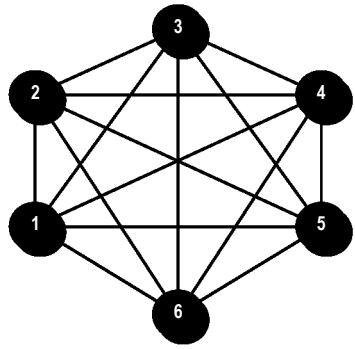
Graf jest parą $G = (V, E)$, gdzie V jest a niepustym zbiorem *wierzchołków,* a E jest zbiorem *krawędzi*. Każda krawędź jest parą wierzchołków (v_1, v_2) takich, że $v_1 \neq v_2$.

Dwa wierzchołki grafu $G=(V, E)$ są ***incydentne*** jeżeli, gdy $v_1, v_2 \in V$ to $\{v_1, v_2\} \in E$.

Podgraf grafu $G = (V, E)$ jest grafem $G' = (V', E')$, gdzie $V' \subseteq V$ i $E' \subseteq E$ takim, że dla każdego $e \in E$ i $e=\{v_1, v_2\}$, jeżeli $v_1, v_2 \in V'$, to $e \in E'$.



Rys.VI.1. Przykładowy graf



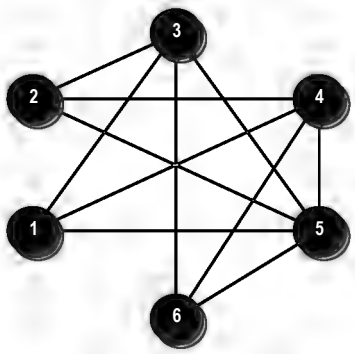
Rys. VI.2 Graf pełny

Źródło: opracowanie własne

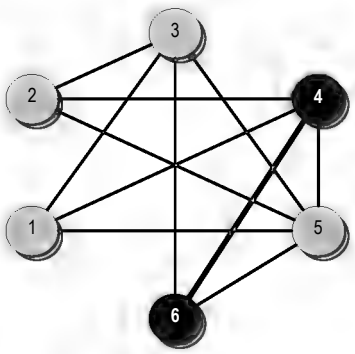
Drogą (ścieżką) w grafie $G=(V, E)$ z wierzchołka s do wierzchołka t nazywamy ciąg wierzchołków $\{v_1, \ldots, v_n\}$ taki, że: $\{s, v_1\} \in E$, $\{v_i, v_{i+1}\} \in E$ dla $n=1, 2, \ldots, n-1$, $\{v_n, t\} \in E$.

Graf $G=(V, E)$ jest ***grafem spójnym***, jeżeli dla każdych dwóch różnych wierzchołków istnieje droga łącząca te wierzchołki.

Kliką (*podgrafem pełnym*) $Q=(V_q, E_q)$ w grafie $G=(V, E)$ jest graf taki, że $V_q \subseteq V$ i $E_q \subseteq E$ oraz każda para wierzchołków $v_1, v_2 \in V_q$ spełnia warunek $\{v_1, v_2\} \in E_q$.



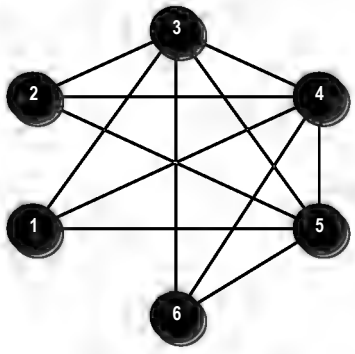
Rys.VI.3. Przykładowy graf



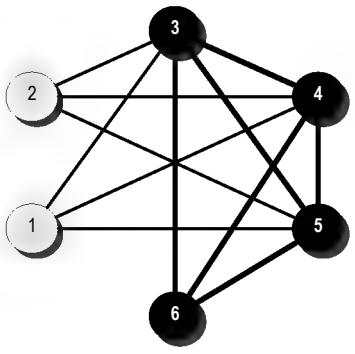
Rys. VI.4. Klika w grafie z rys. VI.3

Źródło: opracowanie własne

Maksymalną kliką nazywamy klikę $Q_M=(V_q, E_q)$ w grafie $G=(V, E)$ taką, że nie istnieje wierzchołek $v \in V$ i $v \notin V_q$ taki, że $Q'=(V', E')$ jest kliką, gdzie $V'=V \cup \{v\}$ i $E' \subseteq E$, gdzie każda para $v_1, v_2 \in V'$ wierzchołków spełnia warunek $\{v_1, v_2\} \in E'$.



Rys. VI.5. Przykładowy graf



Rys. VI.6. Podgraf pełny grafu z rys. VI.5

Źródło: opracowanie własne

Stopniem wierzchołka w *grafie prostym* (*nieskierowanym*) nazywamy liczbę krawędzi do których ten wierzchołek należy.

Na przykład, wierzchołek 2 w grafie z Rys. VI.5 ma stopień 3, a wierzchołek 4 ma stopień 5.

VI.2.3. α-klika

Niech $A=(V_\alpha, E_\alpha)$ będzie podgrafem grafu $G=(V, E)$, gdzie $\alpha \in (0,1]$, $V_\alpha \subseteq V$, $E_\alpha \subseteq E$, $k=Card(V_\alpha)$, k_i jest liczbą wierzchołków $v_j \in V_\alpha$ takich, że $\{v_i, v_j\} \in E_\alpha$.

Graf A jest ***α-kliką*** w grafie G jeżeli dla każdego $v_i \in V_\alpha$ zachodzi warunek:

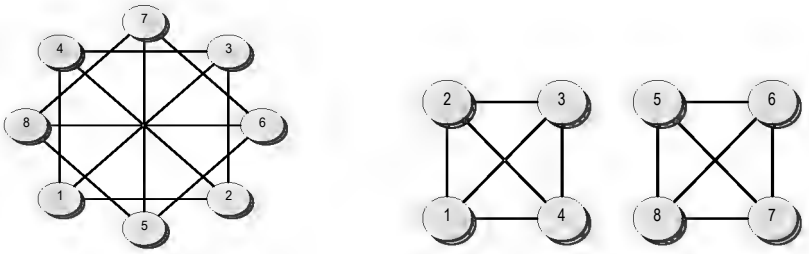
$$\alpha \le \frac{k_i}{k} \tag{VI.1}$$

Wykażemy, że dla $\alpha < \frac{1}{2}$ α-klika jest grafem spójnym, czyli grafem, w którym dla każdych dwóch różnych wierzchołków istnieje ciąg krawędzi należących do grafu, łączących te wierzchołki.

Dowód: Weźmy dowolny wierzchołek v_i, należący do α-kliki $A=(V_\alpha , E_\alpha)$. Niech $k=Card(V_\alpha)$, k_i jest liczbą wierzchołków $v_j \in V_\alpha$ takich, że $\{v_i,v_j\} \in E_\alpha$.

Dla $\alpha > \frac{1}{2}$ mamy $\frac{k_i}{k} \ge \alpha > \frac{1}{2}$ czyli $\frac{k_i}{k} > \frac{1}{2}$, zatem $k_i > \frac{1}{2}k$.

Z rachunku zbiorów wynika, że dla każdych dwóch wierzchołków zbiory wierzchołków z nimi incydentnych mają część wspólną, zatem graf jest spójny. Dla $\alpha = \frac{1}{2}$ graf nie musi spełniać warunku spójności. Na przykład graf z Rys. VI.7 nie jest spójny.



Rys VI.7. Przykład grafu niespójnego dla $\alpha = ½$
Źródło: opracowanie własne

Algorytm zachłanny

Maksymalna klika

Niech:

$G(V, E)$ – graf, w którym V jest niepustym zbiorem wierzchołków, a E jest zbiorem krawędzi.

Q – podzbiór zbioru wierzchołków V (klika)

$deg(v)$ – stopień wierzchołka

$maxdeg(G)$ – wierzchołek o największym stopniu w grafie G

remove(G, v) – usuwa z grafu G wierzchołek v oraz wszystkie krawędzie do których ten wierzchołek należy

$Y(Q)$ – podzbiór zbioru wierzchołków V takich, że każdy wierzchołek z Y jest incydentny ze wszystkimi wierzchołkami z Q

Z – graf w którym Y jest zbiorem wierzchołków, a zbiorem krawędzi jest podzbiór zbioru V takich, że każda krawędź jest parą wierzchołków (v_1, v_2) takich, że $v_1 \neq v_2$ oraz $v_1, v_2 \in Y$

Cl – zbiór grafów $\{G_1(V_1, E_1), G_2(V_2, E_2), ..., G_m(V_m, E_m)\}$ takich, że $V_i \cap V_j = \varnothing$ dla każdego $i \neq j$ oraz $i,j=1, 2, ... , m$, $V_1 \cup V_2 \cup ... \cup V_m = V$.

Maksymalna klika

```
procedure max_clique(G)
input
V;
E;
output
Q;
begin
```

```
Q:=∅;
Q= Q ∪ maxdeg(G);
        while Y(Q)≠∅ do
                begin
                    Q= Q ∪ maxdeg(Z);
                    remove(G, maxdeg(Z));
                end;
end;
```

Dla problemu maksymalnej α-kliki wystarczy, by zbiór $Y(Q)$ był takim podzbiorem zbioru wierzchołków V, aby po dodaniu do zbioru Q dowolnego wierzchołka ze zbioru $Y(Q)$, zbiór Q wraz z odpowiednimi krawędziami był α-kliką.

Aby uzyskać podział grafu $G(V, E)$ proponujemy następujący algorytm:

Klastering

```
procedure cluster(G)
input
V;
E;
output
Cl;
begin
Cl:=∅;
        while V≠∅ do
                begin
                    Cl= max_clique(G); //Cl{Vi, Ei}
                    V=V\Vi;
                    E=E\Ei;
                end;
end;
```

Powyższy algorytm podaje jedno rozwiązanie, którego dalej nie modyfikuje. Algorytm ten jest algorytmem zachłannym, który podaje tylko aproksymację rozwiązania optymalnego.

VI.2.4. β-klika

Niech $B=(V_\beta, E_\beta)$ będzie podgrafem grafu $G=(V, E)$, gdzie $\beta \in (0, 1]$, $V_\beta \subseteq V$, $E_\beta \subseteq E$, $k=\text{Card}(V_\beta)$, l jest liczbą łuków $e \in E_\beta$.

Graf *B* jest **β-kliką** w grafie *G*, jeżeli spełniony będzie warunek

$$\beta \le \frac{l}{k(k-1)} \qquad \text{(VI.2)}.$$

Zauważmy, że dla $\beta \ge 1 - 1/k$ β-klika będzie podgrafem spójnym.

VI.3. Złożoność problemu

Klasę P-time, w skrócie *P*, definiujemy jako zbiór wszystkich języków akceptowalnych przez deterministyczną maszynę Turinga w wielomianowej liczbie kroków względem swojego rozmiaru. W praktyce znamy algorytmy dokładne o wielomianowej złożoności obliczeniowej, rozwiązujące takie problemy.

Klasę NP-time, w skrócie *NP*, definiujemy jako zbiór wszystkich języków akceptowalnych przez niedeterministyczną maszynę Turinga w wielomianowej liczbie kroków. W praktyce znamy algorytmy dokładne o wykładniczej złożoności obliczeniowej, rozwiązujące takie problemy. Algorytmy dokładne o wielomianowej złożoności obliczeniowej, rozwiązujące te problemy nie są znane, jak również nie jest rozwiązany problem istnienia takich algorytmów.

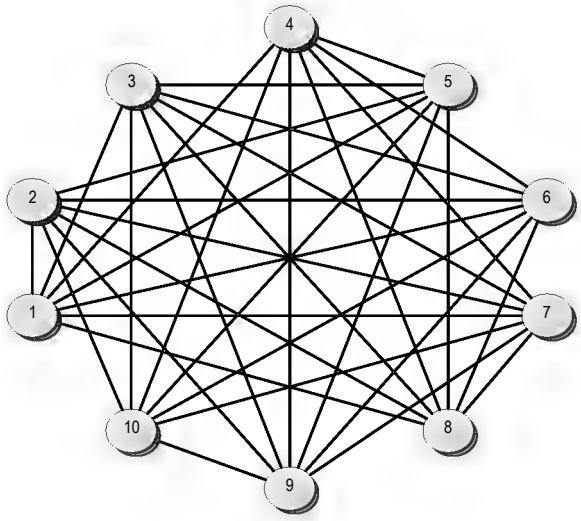
Nazwa *NP* pochodzi od słów *nondeterministic polynomial time*. Do dziś nie został rozstrzygnięty problem czy *P=NP*. Intuicyjnie, klasa *P* składa się z tych problemów, które można szybko rozwiązać, a klasa *NP* składa się z problemów, których rozwiązania można zweryfikować w czasie wielomianowym.

Problem maksymalnej kliki jest *NP*-trudny to znaczy, że ogólny algorytm rozwiązania tego problemu dla Niedeterministycznej Maszyny Turinga będzie miał wielomianową złożoność obliczeniową. Oznacza to również, że nie znamy algorytmów dokładnych o wielomianowej złożoności obliczeniowej rozwiązujących te problemy.

Problem maksymalnej kliki jest szczególnym przypadkiem problemu maksymalnej α-kliki oraz maksymalnej β-kliki zatem problemy maksymalnej α-kliki oraz maksymalnej β-kliki są *NP*-trudne. Problemy te można rozwiązać dokładnie generując wszystkie niepuste podzbiory zbioru wierzchołków przy

pomocy drzewa backtrackingowego. W tym przypadku złożoność obliczeniowa będzie $O(2^n)$.

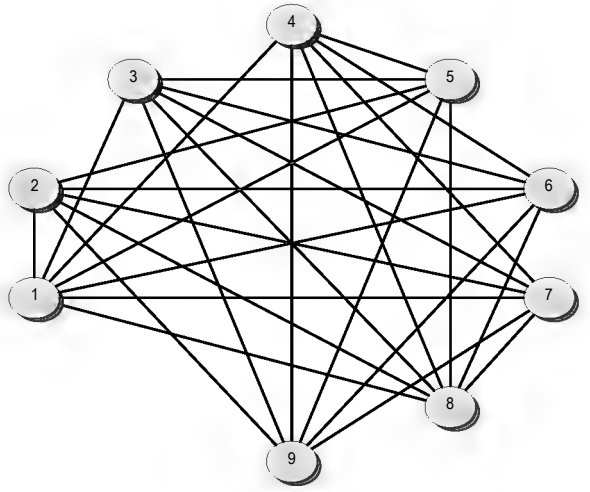
W przypadku problemu kliki w grafie, jeżeli zbiór wierzchołków jest kliką to każdy niepusty podzbiór tego zbioru również jest kliką. W przypadku problemu α-kliki dla α<1 jeżeli zbiór wierzchołków jest α-kliką to nie znaczy, że jego dowolny niepusty podzbiór jest również α-kliką. Powoduje to dodatkowe problemy przy konstrukcji algorytmu aproksymacyjnego zachłannego.



Rys. VI.8. Przykład α-kliki dla α =0,8
Źródło: opracowanie własne

Złożoność obliczeniowa sprawdzenia, czy graf jest α-kliką dla zadanego α jest w ogólnym przypadku wielomianowa (gdy dysponujemy tablicą wierzchołków wraz z ich stopniami złożoność jest liniowa). Aby sprawdzić czy graf jest α-kliką wystarczy znaleźć wierzchołek o najniższym stopniu. Jeżeli spełnia on kryterium α-kliki, to cały graf jest α-kliką, w przeciwnym wypadku, graf nie jest α-kliką.

W przypadku, gdy mamy macierz incydencji grafu z której wydzielono podgraf, którego α należy sprawdzić, to podstawowym problemem jest wyliczenie stopni wierzchołków dla takiego podgrafu. W najgorszym przypadku koszt będzie wynosił $(k^2-k)/2$, gdzie k jest liczbą wierzchołków w podgrafie.



Rys. VI.9. Podgraf grafu z Rys. VI.8 który dla α =0,8 nie jest α-kliką
Źródło: opracowanie własne

Dla problemów o wykładniczej złożoności obliczeniowej, w praktyce nie stosuje się algorytmów dokładnych, dlatego celowe jest stosowanie algorytmów aproksymacyjnych.

VI.4. Szybki algorytm kojarzenia skupień

Niech P będzie pokryciem zbioru V wierzchołków grafu G, przez m rozłącznych podzbiorów, tj. rozbiciem

$$P = \{V_i, i = 1, \ldots, m\}, \qquad \bigcup_i V_i = V, \;\; V_i \cap V_j = \Phi \;\; \forall i \neq j \,. \tag{VI.3}$$

Zakładając, że G jest grafem ważonym, tj. istnieją wagi $c_{ij} \geq 0$ dla każdego $(i, j) \in E$, dla rozbicia P definiujemy funkcje reprezentujące odpowiednio siłę wiązań wewnętrznych i siłę wiązań zewnętrznych jako

$$f(P) = \sum_{i=1}^{m} \sum_{k,l \in V_i, k \neq l} c_{kl} \quad \text{i} \quad g(P) = \sum_{i=1}^{m} \sum_{j \neq i}^{m} \sum_{k \in V_i} \sum_{l \in V_j} c_{kl} \,. \tag{VI.4}$$

Zauważmy, że dla rozpatrywanego grafu suma *f(P)+g(P)* jest stała i niezależna od sposobu powiązania wierzchołków w klastry, czyli od *P*. Stąd problem w postaci następującej: wyznaczyć rozbicie *P* i odpowiadające mu *m*, dla których pomiędzy *m* i *f(P)* zachodzi określony kompromis, co zapiszemy w sposób następujący:

$$(f(P), m) = \text{ minimum.} \tag{VI.5}$$

Takie sformułowanie sugeruje szukanie rozwiązań Pareto-optymalnych, chyba, że wartość *m* ustalimy lub wybierali możliwie najmniejszą.

Algorytm

Niech $P(k,l)$ będzie pokryciem otrzymanym z *P* w wyniku połączenia w jeden klaster jego dwóch składowych V_k i V_l. W wyniku tej operacji *f* wzrośnie o wielkość

$$\delta(k,l) = f(P(k,l)) - f(P). \tag{VI.6}$$

Szukając optymalnego pokrycia najczęściej startujemy z pokrycia, którego klastry zawierają pojedyncze wierzchołki, czyli z $P = \{V_i = \{i\}, i = 1,\ldots,n\}$ oraz $m=n$. Szukając rozwiązania postępujemy w sposób następujący:

1. Znajdź parę klastrów (*k*, *l*) zgodnie z regułą maksymalnego przyrostu powiązań, tj.

   $$(k,l) = \arg\max\{\delta(i,j), 0 \le i,j \le m\} \tag{VI.7}$$

   przy warunku, że V_k złączone z V_l spełniają określone warunki, w szczególności warunki α-kliki lub β-kliki.

2. Jeżeli nie znaleziono (*k, l*) to KONIEC (ostatnio znalezione rozbicia *P* są rozwiązaniami). W przeciwnym przypadku idź do kroku 3.

3. Aktualizuj P łącząc V_k z V_l, *m* zmniejsz do $m - 1$, idź do kroku 1.

Algorytm zaimplementowano w języku C++. Testy z różnymi przykładami pokazały, że algorytm można zmodyfikować, np. dodać ograniczenie na maksymalną liczbę wierzchołków w klastrze, wprowadzając współczynnik γ

($0< \gamma \leq 1$) i określając maksymalny rozmiar klastra jako $\gamma \bullet n$. Kryterium maksymalizacji przyrostu $\delta(k,l)$ można zaś zastąpić kryterium maksymalizacji uśrednionego przyrostu na jeden element tworzonego klastera. Tak powstały kolejne metody, B i C, klasteryzacji grafów, przy czym:
- metoda B – szukanie pokryć β-klikami,
- metoda C – (modyfikacja metody B) - szukanie pokryć zbiorami, których rozmiar nie przewyższa $\gamma \bullet n$ wierzchołków, $0< \gamma \leq 1$.

W kolejnych punktach zostaną opisane metody:
- metoda A – szukanie pokryć α-klikami,
- metoda D – szukanie pokryć algorytmem opartym na kryterium energetycznym, stosowanym przy przetwarzaniu macierzy DSM.

VI.5. Metoda energetyczna oparta na tablicy DSM

Celem jest maksymalizacja kryterium jakości (VI.8) lub, w przybliżeniu, powiększenie niezerowych obszarów w pobliżu przekątnej macierzy danych, permutując odpowiednio klasteryzowane dane. Rozwiązując ten problem możemy pomóc znaleźć lepszy model połączeń pomiędzy obiektami przestrzennymi i rozwinąć go na większe obszary. Najlepsze znalezione rozwiązanie problemu przy pomocy rozłącznego klasteringu jest przedstawione poniżej.

Została zastosowana metoda kryterium energetycznego (Lenstra, 1997), w której maksymalizuje się wartość następującej funkcji:

$$Q = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} a[i,j](a[i,j-1] + a[i,j+1] + a[i-1,j] + a[i+1,j]) \qquad \text{(VI.8)}$$

lub równoważnej:

$$Q = \sum_{i=1}^{n}\sum_{j=1}^{m} a[i,j](a[i,j+1] + a[i+1,j]) \qquad \text{(VI.9)}$$

gdzie: n, m- numery wierszy i kolumn, dla symetrycznego przypadku $n=m$,
$a[i,j]$ – element tablicy danych. Zakładamy, że elementy $a[0,j]$, $a[i,0]$, $a[n+1,j]$, $a[i,m+1]$ są równe 0.

Funkcję Q maksymalizuje się przez permutowanie wierszy i kolumn macierzy A. Aby uzyskać optymalne ustawienie danych, zastosowaliśmy algorytm

ewolucyjny. Po uzyskaniu odpowiedniego zestawienia elementów macierzy *A*, maksymalizującego kryterium jakości *Q*, dokonano podziału danych na klastry w miejscach, w których występują minima lokalne następującej funkcji:

$$G[j] = \sum_{i=1}^{n} a[i,j](a[i,j-1] + a[i,j+1] + a[i-1,j] + a[i+1,j]) \qquad \text{(VI.10)}$$

gdzie:*j* – numer wiersza, *j* = 1,2, ,*m*, *i* – numer kolumny, *i* = 1,2, ,*n*, *a*[*i*,*j*] – element macierzy.

Minima te odpowiadają miejscom, w których sąsiednie grupy danych są ze sobą słabiej powiązane, a więc stanowią one granice silniej powiązanych obszarów – klastrów i mogą służyć za kryterium, dzięki któremu można oddzielać od siebie silniej powiązane jednostki.

VI.6. Ewolucyjne metody rozwiązywania zadań klasteryzacji w grafach

Algorytmy ewolucyjne są wzorowane na darwinowskiej teorii ewolucji. W naturze genotyp organizmów żywych zmienia się nieznacznie z pokolenia na pokolenie na skutek rekombinacji cech dziedziczonych po rodzicach oraz pewnych przypadkowych mutacji powodowanych przez czynniki zewnętrzne (np. promieniowanie UV, gamma, pewne substancje chemiczne itp.). Modyfikacje te w pewnych sytuacjach prowadzą do powstania nowych korzystnych cech, dzięki czemu osobniki obdarzone nimi mają znacznie większe szanse przeżycia i wydania potomstwa niż osobniki ich pozbawione.

Mechanizm ewolucyjny zastosowano w informatyce. W ten sposób powstał cały szereg metod ewolucyjnych:

- algorytmy genetyczne,
- strategie ewolucyjne,
- programowanie genetyczne i ewolucyjne
- oraz inne zbliżone np. przeszukiwanie rozproszone (ang. scatter search).

Umożliwiają one dość szybkie i precyzyjne rozwiązywanie skomplikowanych zadań. Dodatkowo, metody te charakteryzują się dużą uniwersalnością i po pewnych modyfikacjach można je zastosować do rozwiązania szerokie-

go wachlarza problemów. Do rozwiązania konkretnego zagadnienia algorytm ewolucyjny (AE) trzeba odpowiednio dostosować. Bez zmian pozostaje jedynie ogólna koncepcja działania metody: dzięki niewielkim zmianom genotypu, zachodzącym w kolejnych pokoleniach (iteracjach) i selekcji promującej najlepsze osobniki, następuje rozwój populacji w kierunku coraz lepszego spełniania funkcji celu (lub opartej na niej funkcji dopasowania) postawionego zagadnienia. Dostosowanie to polega na odpowiednim zakodowaniu problemu oraz doborze operatorów ewolucyjnych, umożliwiających modyfikację populacji zakodowanych osobników.

Początkowo w AE stosowano wyłącznie kodowanie binarne oraz standardowe operatory krzyżowania i mutacji. Obecnie z reguły stosuje się dużo bardziej skomplikowane metody kodowania i modyfikacji rozwiązań, ściśle dopasowane do specyfiki rozwiązywanego problemu. Kryterium stopu dla AE stanowi zazwyczaj liczba iteracji, osiągnięcie pewnych zadowalających rezultatów lub też brak pozytywnych zmian w ciągu ustalonej liczby pokoleń. Możliwe są również bardziej wyrafinowane metody oparte na miarach zunifikowania populacji lub na kombinacjach kilku kryteriów z wyżej wymienionych.

Standardowy algorytm ewolucyjny działa według schematu pokazanego na Rys. VI.10.

1. Losowa inicjacja populacji rozwiązań.
2. Reprodukcja i modyfikacja rozwiązań, przy użyciu operatorów genetycznych.
3. Ocena otrzymanych rozwiązań.
4. Selekcja osobników do następnej populacji.
5. Stop, jeśli warunek stopu został osiągnięty, w przeciwnym razie powrót do punktu 2.

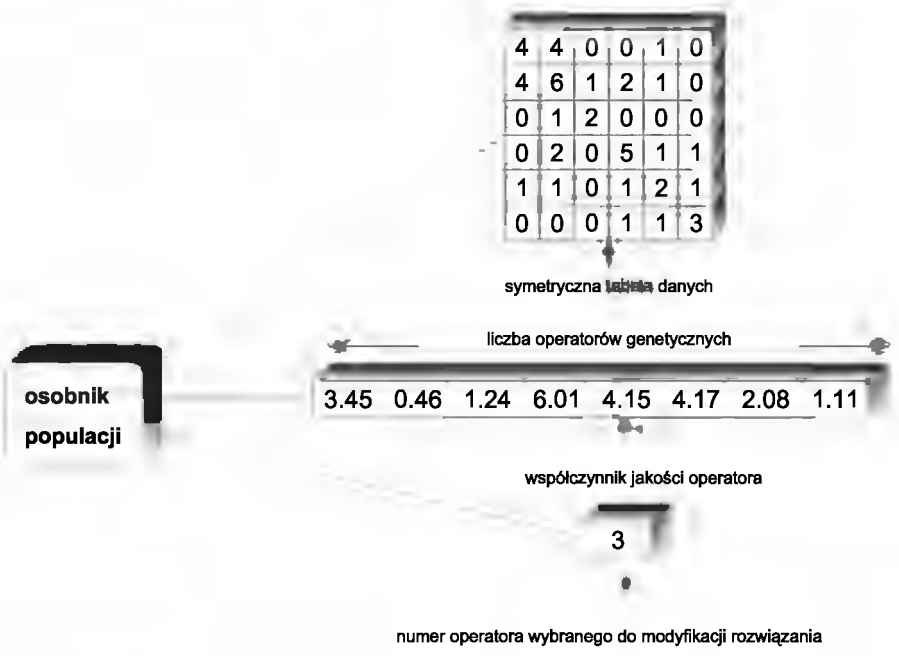
Rys. VI.10. Schemat działania algorytmu ewolucyjnego.

VI.6.1 Zakodowanie problemu

Problem DSM

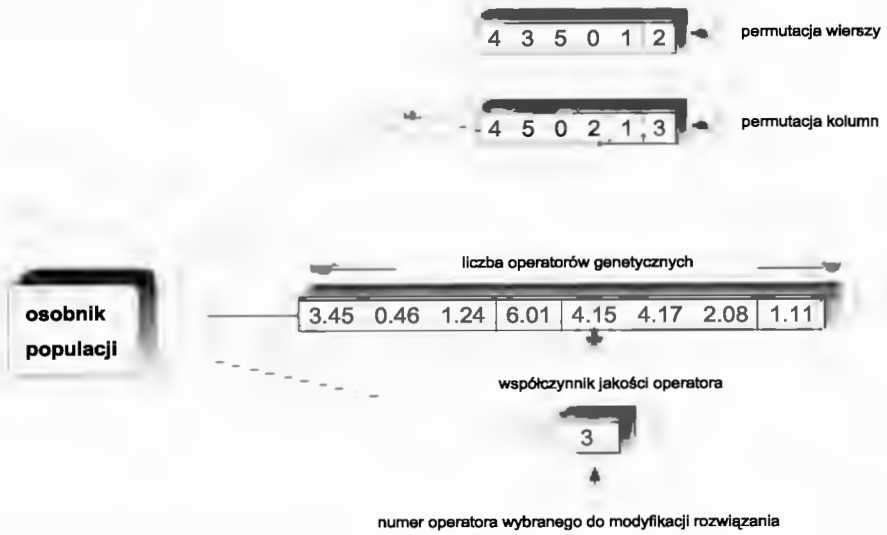
Zakodowanie problemu jako macierzy DSM polega na zapamiętaniu aktualnej tablicy parametrów ($C = [c_{ij}]$) w strukturze danych osobnika (Rys.

VI.11). Różne rozwiązania posiadają tę samą macierz danych, tylko z „przetasowanymi" kolumnami i wierszami. Dlatego też przy większych rozmiarach macierzy *C* korzystniejsze wydaje się zastosowanie metody kodowania, w której pamiętane jest w bieżącym rozwiązaniu jedynie uszeregowanie wierszy i kolumn względem wejściowej macierzy danych i zapamiętanie w programie tylko jednej kopii macierzy *C*. Daje to oszczędność miejsca w pamięci komputera i przyspiesza działanie obliczeń ewolucyjnych (m.in. przy modyfikacjach rozwiązań nie trzeba przesyłać w pamięci komputera zawartości odpowiednich wierszy i kolumn macierzy *C*). Dodatkowo, w przypadku zadania symetrycznego można pamiętać jedynie położenie kolumn lub wierszy. Ilustruje to Rys. VI.12.



Rys. VI.11. Struktura zakodowanego osobnika z macierzą danych w każdym rozwiązaniu.

Źródło: opracowanie własne



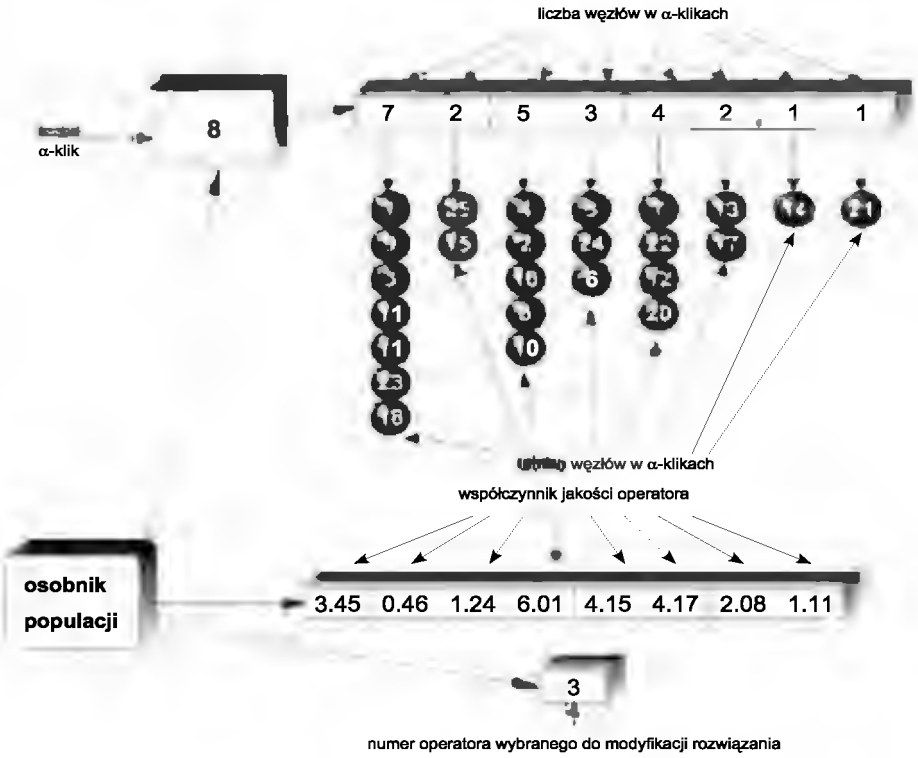
Rys. VI.12. Struktura zakodowanego osobnika z wektorami odpowiadającymi permutacjom wierszy i kolumn (wersja niesymetryczna)
Źródło: opracowanie własne

Problem α-klik

Informacja o rozwiązywanym zadaniu jest zawarta w macierzy $C = [c_{ij}]$ powiązań dla rozpatrywanego grafu, gdzie 0 oznacza brak połączenia pomiędzy wierzchołkami, natomiast inne możliwe wartości traktowane są jako połączenia o odpowiedniej wadze (sile). W szczególności C, może być macierzą incydencji, jeśli zawiera tylko elementy 0 i 1.

Każdy członek populacji zawiera swoje własne rozwiązanie zadania w postaci dynamicznej tablicy znalezionych α-klik (ich liczba może się zmieniać). Każda α-klika jest niepustą listą wierzchołków grafu (wierzchołek może wystąpić tylko w jednej α-klice). Pojedyncze wierzchołki tworzą również najmniejsze możliwe α-kliki, jednakże algorytm ewolucyjny dąży do utworzenia jak największych i jak najsilniej powiązanych α-klik.

Oprócz tego osobnik populacji zawiera też pewne dodatkowe dane, wymagane przez użytą wersję algorytmu ewolucyjnego: wektor wskaźników jakości operatorów genetycznych i numer aktualnego operatora genetycznego, wylosowanego do modyfikacji rozwiązania.



Rys. VI.13. Struktura pojedynczego członka populacji
Źródło: opracowanie własne

VI.6.2 Funkcja dopasowania

Problem DSM

Funkcja dopasowanie osobnika jest ściśle oparta na funkcji celu zagadnienia (VI.11), jej jedyna modyfikacja polega na przeskalowaniu otrzymywanych wartości do przedziału (0,1).

Problem α-klik

Funkcja celu rozwiązywanego zadania jest ściśle powiązana z funkcją dopasowania, która służy do oceny otrzymanych rozwiązań. W rozpatrywanym

przypadku funkcja celu ma sprzyjać tworzeniu jak największych α-klik. Jednakże można to zrobić na wiele sposobów, stąd podane poniżej wzory (VI.11) i (VI.12). Zastosowanie funkcji (VI.11) powoduje tworzenie pokryć α-klikami o zbliżonym rozmiarze, podczas gdy funkcja (VI.12) prowadzi do generowania pokryć bardziej zróżnicowanych.

Można również wytworzyć α-kliki o bardzo wyrównanych rozmiarach. Wynik taki można otrzymać stosując funkcję (VI.13). W funkcji tej nie maksymalizuje się już liczby połączeń w α-klikach lecz, uznając, że osiągnięcie zadanego poziomu α daje już wystarczający poziom powiązań w klastrze, minimalizuje się różnice pomiędzy wyliczonym średnim rozmiarem klastra a rzeczywistym.

Funkcja dopasowania nie posiada kary za przekroczenie warunku bycia α-kliką, gdyż przyjęte operatory genetyczne i procedura losująca populację startową nie pozwalają na powstawanie rozwiązań niedopuszczalnych.

$$\max Q = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{l_i}\sum_{k=1}^{j} C[t_{ik}, t_{ij}] \tag{VI.11}$$

$$\max Q = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{l_i}\sum_{k=1}^{j} C[t_{ik}, t_{ij}] \tag{VI.12}$$

$$\max Q = \frac{1}{m}\sum_{i=1}^{m}\left(l_i - \left|\frac{n}{m} - l_i\right|\right) \tag{VI.13}$$

gdzie: m – liczba α-klik w ocenianym rozwiązaniu, n – liczba węzłów w grafie, l_i – liczba węzłów w i-tej α-klice, C – macierz powiązań (w szczególności macierz incydencji), t_{ij}, t_{ik} – węzły i-tej α-kliki.

VI.6.3. Specjalizowane operatory genetyczne

Problem DSM

Specjalizowane operatory genetyczne są w znacznym stopniu modyfikacją tradycyjnego krzyżowania i mutacji, przystosowaną do specyfiki rozwiązy-

wanego zadania – możliwe do zastosowania są tylko operatory dokonujące permutacji wierszy i/lub kolumn[VI.1]:

Problem α-klik

Opisana wcześniej struktura danych wymaga użycia odpowiednich operatorów genetycznych do jej modyfikacji. Każdy operator zaprojektowany jest w taki sposób, aby zachowywać właściwość bycia odpowiednią α-kliką modyfikowanych fragmentów rozwiązań. Gdyby działanie operatora genetycznego doprowadziło do powstania rozwiązania niedopuszczalnego, to operacja taka będzie anulowana i rozwiązanie pozostanie bez zmian.

Zaprojektowano następujące operatory:

- mutacja – zamiana losowo wybranych wierzchołków w α-klikach
- przeniesienie losowo wybranego wierzchołków do innej α-kliki
- „inteligentne" przeniesienie – operacja podobna do poprzedniej, wykonywana tylko wtedy, gdy zmiana daje lepsze rozwiązanie niż rodzicielskie
- konkatenacja – próbuje łączyć (głównie małe) α-kliki
- zastosowano także wersje wielokrotne opisywanych operatorów.

VI.6.4 Algorytm ewolucyjny z oceną operatorów

Użycie dużej liczby specjalizowanych operatorów ewolucyjnych wymagało zastosowania niestandardowej metody wyboru operatorów genetycznych modyfikujących rozwiązania w kolejnych epokach działania AE. W tym przypadku użyto metody z oceną jakości operatorów. Metoda ta zakłada, że operator, który generuje dobre wyniki, powinien być używany częściej niż ten, który uzyskuje słabsze. W związku z tym każde rozwiązanie w populacji prowadzi własny ranking operatorów na podstawie swoich współczynników jakości, które są ściśle powiązane z generowanymi przez operatory rozwiązaniami. Operatory do modyfikacji rozwiązań są losowane na podstawie tych współczynników jakości – im lepszy operator, tym większą ma szansę na ponowne wystąpienie. Przyspiesza to w dużym stopniu obliczenia. Jednocześnie słabsze operatory nie są eliminowane, gdyż ich działanie, choć rzadsze, jest korzystne dla rozwoju populacji.

[VI.1] W przypadku symetrycznym, permutacje wierszy i kolumn muszą być takie same.

Ogólna zasada działania metody jest następująca: im bardziej dany operator polepsza funkcję celu danego osobnika, tym większa staje się przypisana mu wartość współczynnika jakości, a co za tym idzie, większe jest prawdopodobieństwo jego występowania, co ilustruje wzór (VI.14):

$$p_{ij}(t) = \frac{q_{ij}(t)}{\sum_{i=1}^{i=L(t)} q_{ij}(t)} \tag{VI.14}$$

gdzie: $q_{ij}(t)$ jest współczynnikiem jakości i-tej operacji w chwili t dla j-tego osobnika populacji, $p_{ij}(t)$ jest prawdopodobieństwem wystąpienia i-tej operacji w chwili t dla j-tego osobnika, a $L(t)$ jest liczbą operacji genetycznych w chwili t.

Każdy osobnik populacji posiada, oprócz zakodowanego rozwiązania, swój własny ranking operatorów (Rys. VI.12 i VI.13), na podstawie którego losuje operator, który będzie go modyfikował w bieżącej iteracji. Metoda ta umożliwia adaptacyjne strojenie prawdopodobieństw występowania operatorów ewolucyjnych w zależności od potrzeb członków populacji rozwiązań i zgodnie z ich preferencjami. Jest bardzo prawdopodobne, że w zależności od fazy działania optymalizacji ewolucyjnej, jak i położenia danego osobnika w przestrzeni rozwiązań lepiej działają różne operatory.

Sposób obliczania wskaźników jakości operatorów przedstawia wzór (VI.15). Opisywana metoda zakłada, że w danej iteracji tylko jeden, wybrany operator modyfikuje rozwiązanie, dzięki czemu można mu przypisać nagrodę bądź karę za uzyskany efekt. Współczynniki jakości niewybranych osobników pozostają bez zmian (dolna część wzoru (VI.15) dla $i \neq l$), natomiast efekty zadziałania wybranego operatora są oceniane na podstawie górnej części wzoru (VI.15) (dla $i=l$).

$$q_{ij}(t+1) = \begin{cases} q_{0ij}(t) + \dfrac{x_{ij}(t+1)}{\bar{x}(t)} + \alpha_{ij}(t) \cdot q_{ij}(t) \text{ dla } i = l \\ q_{ij}(t) \text{ dla } i \neq l \end{cases} \tag{VI.15}$$

gdzie $q_{ij}(t)$ jest wartością współczynnika jakości operacji i osobnika j w iteracjach t, $q_{0ij}(t)$ jest niewielką wartością stałą (kredytem), uniemożliwiającą całkowite wyeliminowanie jakiegoś operatora, $x_{ij}(t+1)$ jest poprawą funkcji celu, osiągniętą przez operację i dla osobnika j w iteracji t, w przypadku braku polepszenia równą zero, $\bar{x}(t)$ to średnia war-

tość poprawy funkcji celu uzyskana dotychczas, $a_{ij}(t)$ to współczynnik zapominania, uwydatniający rolę najświeższych danych, przy osłabianiu dawniejszych, l zaś jest indeksem operatora wybranego w danej iteracji do modyfikacji rozwiązania.

Pierwszy człon górnej części wzoru (VI.15) zapewnia utrzymywanie pewnej minimalnej wartości współczynnika jakości dla operatorów nieprzynoszących bezpośrednich korzyści. Mogą one jednak wprowadzać do populacji odpowiednie modyfikacje, które w kolejnych iteracjach zaowocują korzystnymi zmianami, dlatego też nie powinny być eliminowane przez wyzerowanie prawdopodobieństwa ich wystąpienia. Drugi człon wzoru (VI.15) stanowi poprawę funkcji celu rozwiązywanego zadania. W przypadku braku poprawy jest on równy zero. Trzeci człon wzoru (VI.15) przechowuje informację o dawniejszych osiągnięciach operatora, przemnożonych przez współczynnik zapominania $\alpha(t)$. Współczynnik zapominania jest odpowiedzialny za właściwe wyważenie wpływu nowszych i starszych osiągnięć na wartość współczynnika jakości operatora.

Wykorzystana do rozwiązania tego problemu metoda selekcji sterowanej składa się z dwóch metod składowych o znacząco różnych właściwościach: selekcji histogramowej (mającej właściwość znaczącego zwiększania zróżnicowania populacji) i deterministycznej ruletki (z silnym naciskiem na promowanie najlepszych osobników). Są one losowo wybierane i wykonywane w trakcie działania obliczeń ewolucyjnych. Prawdopodobieństwo wyboru i wykonania każdej z metod jest określane metodą ze wzoru (VI.16).

$$\begin{aligned} p_{his}(t+1) &= \begin{cases} p_{his}(t)*(1-a) \text{ for } & R(t) > 3*\sigma(F(t)) \\ p_{his}(t)*(1-a)+0.5*a \text{ for } & R(t) \geq 0.5*\sigma(F(t)) \wedge R(t) \leq 3*\sigma(F(t)) \\ p_{his}(t)*(1-a)+a \text{ for } & R(t) < 0.5*\sigma(F(t)) \end{cases} \\ R(t) &= \max(F_{av}(t)-F_{\min}(t), F_{\max}(t)-F_{av}(t)) \end{aligned} \quad \text{(VI.16)}$$

gdzie: p_{his} - prawdopodobieństwo wystąpienia selekcji histogramowej, p_{det} - prawdopodobieństwo wystąpienia selekcji metodą deterministycznej ruletki, $F_{mean}(t)$, $F_{min}(t)$, $F_{max}(t)$ - średnia, minimalna i maksymalna wartość funkcji dopasowania w populacji.

Jeżeli populacja osobników ma zbyt małe odchylenie standardowe ($\sigma(F(t))$) w stosunku do rozpiętości wartości funkcji dopasowania ($R(t)$), wtedy następuje zwiększenie prawdopodobieństwa wystąpienia selekcji histogramowej. W przeciwnym wypadku zwiększane jest prawdopodobieństwo wystąpienia

selekcji metodą deterministycznej ruletki. Jeśli natomiast parametry populacji zawarte są w przedziale uznanym za korzystny, to prawdopodobieństwa wystąpienia obu metod selekcji są prawie równe. Należy zaznaczyć także fakt, że zawsze $p_{his}+p_{det}=1$, czyli któraś z metod zawsze musi wystąpić.

VI.7. Przykładowe wyniki obliczeń

VI.7.1. Zależność liczby α-klik od wartości α oraz czas obliczeń

Przede wszystkim należało sprawdzić jak podział grafu na α-kliki zależy od wartości parametru. W tym celu posłużyliśmy się danymi testowymi - BHOSLIB:

`http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm`.

Dla zastosowanych przez nas zadań testowych znane są rozmiary maksymalnej kliki, mogące służyć do porównania otrzymanych wyników, wynoszą one 100.

Pierwszym z przykładów był graf o 4000 wierzchołków, 7 425 226 krawędzi i maksymalnej klice o 100 wierzchołkach (test frb100-40.clq.gz). Drugim z przykładów był graf o 4000 wierzchołków, 572 774 krawędzi i maksymalnej klice o rozmiarze 100 wierzchołków (test frb100-40mis.gz).

W Tabelach VI.1 i 2 pokazaliśmy uzyskane wyniki w zależności od wartości parametru α. Oba zadania testowe dotyczą bardzo dużych grafów i są to wartości porównywalne z lub większe od typowych zadań rozwiązywanych w technice. Użycie tak dużych zadań testowych umożliwiło nam sprawdzenie praktycznej stosowalności proponowanych metod.

Graf z pierwszego przykładu (test frb100-40.clq.gz) jest grafem bliskim grafowi pełnemu. Graf pełny o 4000 wierzchołków miałby 7 998 000 krawędzi, czyli w tym przykładowym prawie wszystkie wierzchołki są ze sobą połączone. Dla $\alpha \leq 0{,}8$ cały graf jest α-kliką, zatem testy przeprowadzaliśmy dla $\alpha \geq 0{,}8$, przy czym zadaniem nie jest znalezienie maksymalnej α-kliki, ale znalezienie rozłącznego pokrycia grafu jak najmniejszą liczbą α-klik.

Graf z drugiego przykładu ma tylko 572 774 krawędzi, więc jego wierzchołki są dość słabo połączone, zatem testy przeprowadzono dla $\alpha \geq 0{,}51$, gdyż

nie występowało tu zjawisko takie, jak w poprzednim przypadku, gdzie cały graf stawał się α-kliką i przeprowadzanie obliczeń dla wartości α mniejszych od 0,8 nie miało sensu.

Analizując dane zebrane w Tab. VI.1 i VI.2 można zauważyć, że zmieniając wartość parametru α można dość płynnie regulować wielkość otrzymanych α-klik (oczywiście w granicach, w jakich umożliwia to charakter rozpatrywanych danych) i sposób podziału grafu na klastry, będące w tym przypadku α-klikami. Umożliwia to dość łatwe manipulowanie sposobem podziału grafu na klastry w zależności od wartości jednego parametru α, co może mieć decydujące znaczenie w wielu rozwiązywanych zadaniach.

α	Min α-klika	Max α-klika	Liczba α-klik	Średni rozmiar α-kliki
0,80	4000	4000	1	4000
0,90	41	607	15	267
0,95	1	277	39	103
0,97	8	131	56	71
0,99	5	74	78	51
1,00	10	73	76	53

Tabela VI.1. Uzyskane rezultaty dla kilku wartości α i grafu o 4000 wierzchołków i 7 425 226 krawędziach.
Źródło: opracowanie własne

α	Min α-klika	Max α-klika	Liczba α-klik	Średni rozmiar α-kliki
0,51	40	119	50	80
0,60	6	111	67	60
0,70	2	80	94	43
0,80	40	40	100	40
0,90	40	40	100	40
0,95	40	40	100	40
0,97	40	40	100	40
0,99	40	40	100	40
1,00	40	40	100	40

Tabela VI.2. Uzyskane rezultaty dla kilku wartości α i grafu o 4000 wierzchołków i 572 774 krawędziach.
Źródło: opracowanie własne

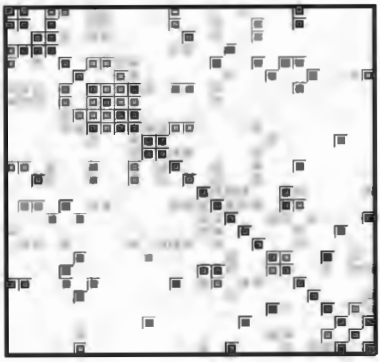
Testy przeprowadzono na komputerze PENTIUM© III 800 MHz pod kontrolą systemu operacyjnego Linux. Dla pierwszego przykładu obliczenia zajęły od 8 godzin do 10 dni w zależności od wartości parametru α. Dla drugiego przykładu obliczenia zajęły od 3 do 24 godzin w zależności od wartości parametru α.

VI.7.2. Przykład 1: projektowanie lotniska

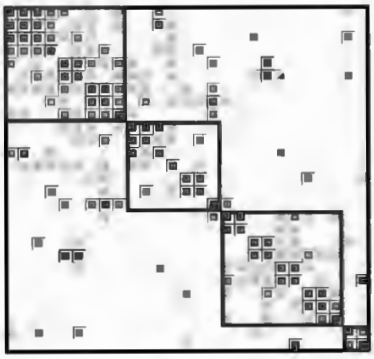
W zadaniu tym rozpatrywany jest problem projektowania lotniska (Lenstra, 1977), które składa się z 27 elementów, powiązanych między sobą różnego rodzaju połączeniami, których siłę/znaczenie można przedstawić w postaci wag. W związku z tym można utworzyć symetryczną macierz (DSM) $A=[a_{ij}]$ (prezentowane np. w Browning, 2001, i Lenstra, 1977) składającą się z parametrów reprezentujących wzajemne odniesienia zadań w skali 0-3. Rozwiązaniem tego zadania są klastry składające się z jednostek o zbliżonych profilach, najbardziej ze sobą powiązanych, które finalnie będą umieszczone dość blisko siebie lub też połączone magistralami o odpowiednio dużej przepustowości.

Na Rys. VI.14 pokazano wyniki klasteryzacji przeprowadzonej omawianymi w pracy metodami:

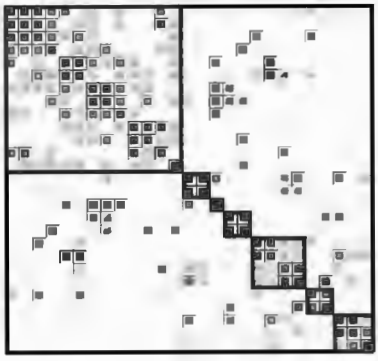
- metoda A – szukanie pokryć α-klikami,
- metoda B – szukanie pokryć β-klikami,
- metoda C – modyfikacja metody B w której poszukiwane jest pokrycie zbiorami, których rozmiar nie przewyższa γn wierzchołków, $0< \gamma \leq 1$,
- metoda D – szukanie pokryć algorytmem opartym na kryterium energetycznym, stosowanym przy przetwarzaniu macierzy DSM.



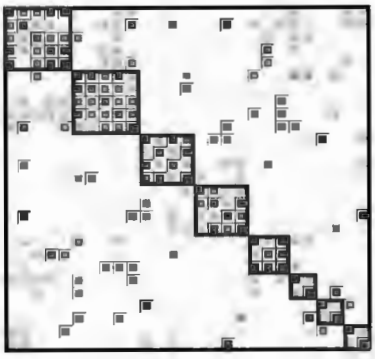
Dane źródłowe



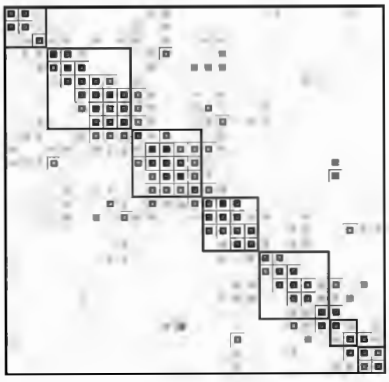
Rozwiązanie 1 – metoda C, γ=0,334



Rozwiązanie 2 – metoda B, $\beta = 0,7$



Rozwiązanie 3 – metoda A, $\alpha = 0,8$

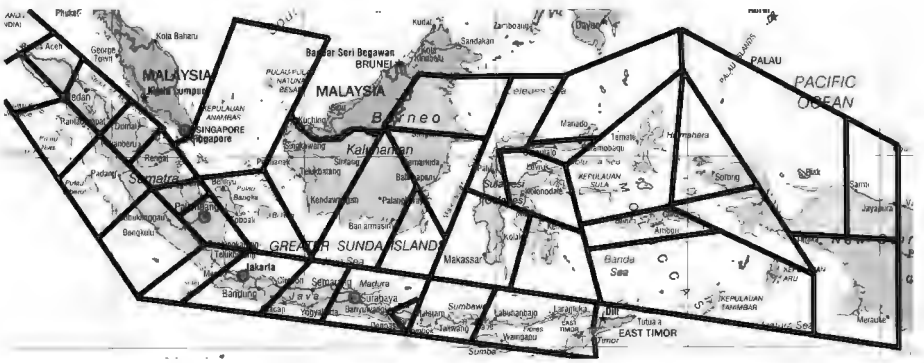


Rozwiązanie 4 –metoda D

Rys. VI.14. Wyniki symulacji dla problemu projektowania lotniska
Źródło: opracowanie własne

VI.7.3 Przykład 2: współpraca gospodarcza regionów

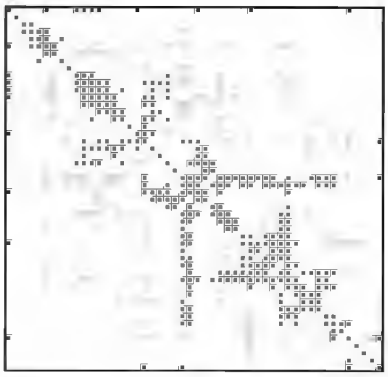
Dane testujące są przykładem problemu opisanego kwadratową, niesymetryczną macierzą 50x50 (Lenstra, 1977). Jednakże do obliczeń użyto macierzy z danymi w wersji nieco zmodyfikowanej, w której znaczenie ma sam fakt współpracy, a nie jej kierunek, wobec czego macierz wejściowa poddana została symetryzacji. Dane są opisane 50 atrybutami, reprezentującymi jedynie relacje zerojedynkowe o istnieniu lub braku bezpośrednich powiązań handlowych między regionami Indonezji (Rys.VI.15). Należało wyróżnić rodziny regionów o intensywnej wymianie handlowej. Zestawienie wybranych wyników obliczeniowych pokazano na Rys. VI.16.



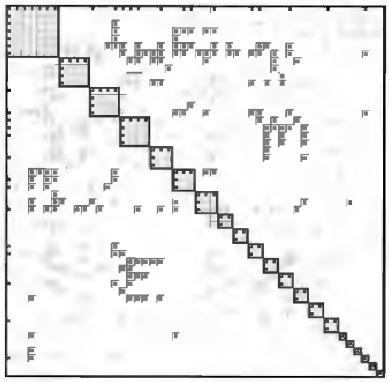
Rys. VI.15. Podział Indonezji na regiony
Źródło: opracowanie własne na podstawie Lenstra (1977)

Podobnie jak w przypadku poprzedniego przykładu, zadanie rozwiązano czterema metodami:

- metoda A – szukanie pokryć α-klikami,
- metoda B – szukanie pokryć β-klikami,
- metoda C – modyfikacja metody B w której poszukiwane jest pokrycie zbiorami, których rozmiar nie przewyższa γ wierzchołków, $0< \gamma \leq 1$,
- metoda D – szukanie pokryć algorytmem opartym na kryterium energetycznym, stosowanym przy przetwarzaniu macierzy DSM.

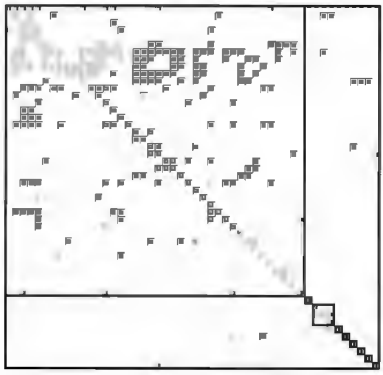


Dane źródłowe

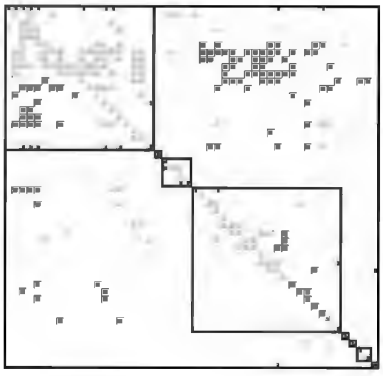


Rozwiązanie 1 – metoda A, α =1

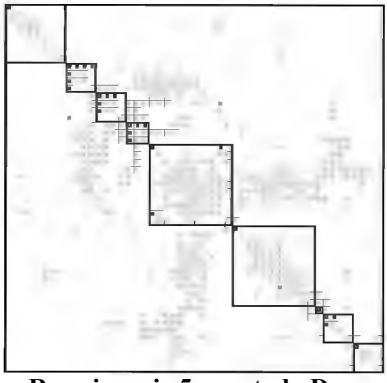
Rozwiązanie 2 – metoda A, α =0.51



Rozwiązanie 3 – metoda B, β = 0,7

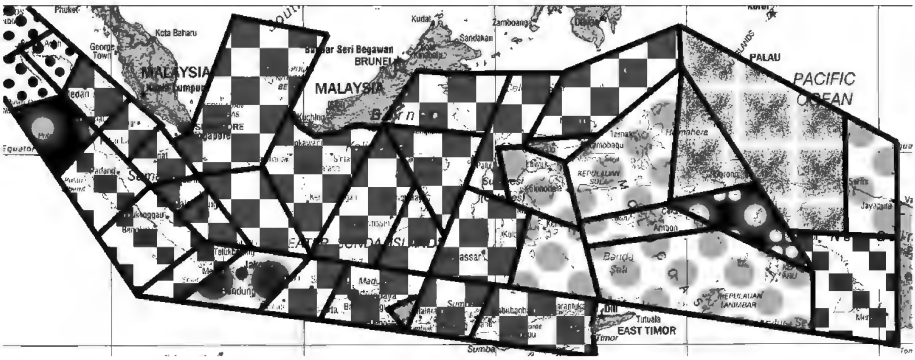


Rozwiązanie 4 – metoda C, γ = 0,33



Rozwiązanie 5 – metoda D

Rys.VI.16. Wyniki symulacji dla problemu współpracy regionów Indonezji
Źródło: opracowanie własne



Rys.VI.17. Podział Indonezji na grupy regionów uzyskany metodą A
Źródło: opracowanie własne na podstawie Lenstra (1977)

W przypadku zadania analizy powiązań gospodarczych regionów Indonezji, najbardziej równomierne wyniki daje metoda D (Rys. VI.16). Przykład ten jest jednak dla pozostałych metod dość skomplikowany, gdyż rozpatrywana macierz nie jest symetryczna. W metodach opartych na pochodnych pojęcia kliki dość trudno jest wobec tego wyróżnić zwarte obszary odpowiedzialne za handel w obu kierunkach równocześnie. Właściwie w tym przypadku należałoby rozwiązywać zadanie zmodyfikowane, w którym pod uwagę bierze się dowolne połączenia, bez rozważania ich kierunku. Sam fakt asymetrii stanowi o szczególnej trudności nie tylko zresztą wykorzystywanych do obliczeń algorytmów, ale także interpretacji danych i wyników, zwłaszcza, jeśli asymetria jest znacząca.

VI.8. Wnioski

W niniejszym rozdziale zaproponowano kilka różnych metod rozwiązywania zadań związanych ze znajdowaniem rozbicia grafu na grupy silnie powiązanych ze sobą wierzchołków oraz przedstawiono możliwe klasy zastosowań tych metod, wraz z dwoma przykładami, odnoszącymi się do zagadnień przestrzennych.

Zaprezentowane w pracy metody rozwiązywania takich trudnych obliczeniowo zadań, dowodzą, że można uzyskać zadawalające wyniki konstruując

proste algorytmy zachłanne. Czasy obliczeń za pomocą takich algorytmów są rzędu sekund i niżej, co uzasadnia ich stosowanie w różnych procedurach interaktywnych. Problemem może być dobór parametrów algorytmu, co oznacza, że w przypadku konkretnych zastosowań trudno będzie prowadzić obliczenia osobie nie będącej ekspertem.

Propozycja algorytmu genetycznego jest pełniejsza. Wykorzystuje ona kilka adekwatnych, komputerowo wydajnych operacji uniwersalnych, kierując się tak złożonością obliczeniową jak i pamięciową. W przypadku tego algorytmu w mniejszym stopniu potrzebny jest ekspert, a funkcja celu może być bardziej złożona. Niestety, zwykle płacimy za to zwiększonym nakładem obliczeń.

Prezentując różne wyniki obliczeniowe staraliśmy się pokazać jak wsparcie metod analitycznych w postaci środków graficznych pomaga prezentować rozwiązanie, a tym samym ocenić werbalnie ich wartość. Wizualizując rozwiązania na dostatecznie wczesnych etapach algorytmu można dzięki temu podpowiadać kolejne ulepszenia rozwiązania. Jest to szczególnie przydatne w zadaniach, dotyczących przestrzeni, w których doświadczenie i intuicja eksperta dziedzinowego, związane z położeniami obiektów i ich powiązaniami są szczególnie istotne.

VI.9. Literatura

Aho A.V., Hopcroft J.E., Ullman J.D. (1982) *Projektowanie i analiza algorytmów komputerowych*. PWN, Poznań.

Altus S. S, Kroo I. M., Gage P. J. (1996) A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems. *Journal of Mechanical Design*, **118**, 4, 486-489.

Aussiello G., Crescenzi P., Gambosi G., Kann V., Marchetti-Spaccamela A., Protasi M. (1999) *Complexity and Approximation*. Springer.

Bagirov A. M. (2006) A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *EJOR* **170**, 578-595.

Berger B., Rompel J. (1994) Efficient NC Algorithms for Set Cover with Applications to Learning and Geometry. *Journal of Computer and System Sciences* 49.

Błażewicz J. (1988) *Złożoność obliczeniowa problemów kombinatorycznych*. WNT.

Browning T. R. (2001) Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management*, **48**, 3.

Benson S.J., Ye Y. (2000) Approximating maximum stable set and minimum graph coloring problems with the positive semi-definite relaxation*, Computational complexity: The problem of approximation*. Kluwer, New York.

Cichosz P. (2000) *Systemy uczące się*. WNT, Warszawa.

Cormen T., Leiserson C., Rivest R. (1997) *Wprowadzenie do algorytmów*. WNT, Warszawa.

Hansen P., Mladenovic N., Urosevic D. (2004) Variable neighborhood search for the maximum clique. *The Fourth International Colloquium on Graphs and Optimisation (GO-IV)*, Lòeche-les-Bains, Suisse (20/08/2000) **145**, 1, 117-125.

Hassin R., Khuller S. (1986) z-Approximation. *Computational complexity: The problem of approximation*. Kluwer, New York.

Hochbaum D. S. (ed.) (1997) *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company.

Hromkovic J. (2001) *Algorithmics for Hard Problems.* Springer.

Jukna S. (2001) *Extremal Combinatorics.* Springer, Berlin-Heidelberg.

Kloks T., Kratsch D. (1998) Listing all minimal separators of a graph*. SIAM J. Comput*. **27**, 3, 605-613.

Korte B., Vygen J. (2000) *Combinatorial optimization, theory and algorithms*. Springer.

Kumlander D. (2007) An Approach for the Maximum Clique Finding Problem Test Tool Software Engineering. *Software Engineering*, **SE 2007**, Innsbruck, Austria.

Lenstra J.K. (1977) *Sequencing by Enumerative Methods*. Mathematisch Centrum, Amsterdam.

Lovasz L. (1975) On the ratio of optimal integral and fractional covers. *Discrete Mathematics* **13**.

McCormick W. T. , Schweitzer P. J. , White T. W. (1972) Problem decomposition and data reorganization by a clustering technique. *Operations Res*. **20**, 993-1009.

McCulley C., Bloebaum C. (1996) A Genetic Tool for Optimal Design Sequencing in Complex Engineering Systems. *Structural Optimization*, **12**, 2-3, 186-201.

Mulawka J., Stańczak J. (1999) Genetic Algorithms with Adaptive Probabilities of Operators Selection. *Proceedings of ICCIMA'99*, New Delhi, India, 464-468.

Owsiński J.W. (1990) On a new naturally indexed quick clustering method with a global objective function. *Applied Stochastic Models and Data Analysis*, 6.

Potrzebowski H., Stańczak J., Sęp K. (2004) Evolutionary method in grouping of units with argument reduction. *Proceedings of the 15th International Conference on Systems Science*, Z. Bubnicki, A. Grzech, eds., III, 29-36.

Potrzebowski H., Stańczak J., Sęp K. (2005) Evolutionary method in grouping of units. *Proceedings of the 4th International Conference on Recognition Systems CORES'05*, Springer-Verlag, Berlin-Heidelberg.

Potrzebowski H., Stańczak J., Sęp K. (2006a) Evolutionary Algorithm to Find Graph Covering Subsets Using α-Cliques. W: J. Arabas, red., *Evolutionary Computation and Global Optimization*, 351-358. *Prace naukowe Politechniki Warszawskiej*.

Potrzebowski H., Stańczak J., Sęp K. (2006b) Heurystyczne i ewolucyjne metody znajdowania pokrycia grafu, korzystające z pojęcia alfa-kliki i innych ograniczeń. *Badania operacyjne i systemowe 2006. Metody i techniki*. Akademicka Oficyna Wydawnicza EXIT, Warszawa.

Protasi M. (2001) Reactive local search for the maximum clique problem. *Algoritmica*, **29**, 4. 610-637.

Rogers J. L. (1997) Reducing Design Cycle Time and Cost Thorough Process Resequencing. *International Conference on Engineering Design ICED*, Tampere, Finland, 1997.

Stańczak J. (1999) Rozwój koncepcji i algorytmów dla samodoskonalących się systemów ewolucyjnych. Rozprawa doktorska, Politechnika Warszawska, 1999.

Stańczak J. (2003) Biologically inspired methods for control of evolutionary algorithms. *Control and Cybernetics*, **32**(2), 411-433.

Sysło M. M., Deo N., Kowalik J. S. (1983) *Algorithms of discrete optimization*. Prentice-Hall.

Williamson D. (1999) *Lecture Notes on Approximation Algorithms*. IBM Research Report RC 21409.

Wilson R.J. (1996) *Introduction to graph theory* Addison Wesley Longman.

Yu T.L., Goldberg D.E., Yassine A., Yassine C. (2003) A Genetic Algorithm Design Inspired by Organizational Theory. *Genetic and Evolutio-*

nary Computation Conference (GECCO) 2003, Chicago. Springer-Verlag, Heidelberg, *Lecture Notes in Computer Science*, **2724**, 1620-1621.

Książka poświęcona jest opisowi zastosowań metod sformalizowanych do wybranych zagadnień społeczno-gospodarczych i administracyjnych o charakterze przestrzennym. Rozpatrywane są zagadnienia regionalizacji i typologii przestrzennej, logistyki i organizacji transportu, zrównoważonego rozwoju, czy jakości stron internetowych samorządów w zestawieniu z położeniem odpowiednich jednostek.

ISSN 0208-8029
ISBN 9788389475251

Instytut Badań Systemowych PAN

W celu uzyskania bliższych informacji i zakupu dodatkowych egzemplarzy prosimy o kontakt z Instytutem Badań Systemowych PAN
ul. Newelska 6, 01-447 Warszawa
tel. (22) 3810 277; e-mail: bibliote@ibspan.waw.pl

ANALIZA SYSTEMÓW PRZESTRZENNYCH. WYBRANE ZAGADNIENIA Jan W. Owsiński (Red.)