Algorytmy Równoległe i Rozproszone
Część X - Algorytmy samostabilizujące.

Łukasz Kuszner
pokój 209, WETI
http://www.sphere.pl/∼kuszner/
kuszner@sphere.pl

Oficjalna strona wykładu
http://www.sphere.pl/∼kuszner/ARiR/
Wykład 15 godzin, Projekt 15 godzin

2006/07

1. Algorytmy samostabilizujące

Algorytmy samostabilizujące się rozważamy w systemach rozproszonych. Różnią się one od innych algorytmów rozproszonych w swej filozofii zapewniania poprawności obliczeń. Podczas gdy tradycyjnie stosujmy sumy kontrolne, czasem skomplikowane schematy potwierdzania i retransmisji, to tutaj konstruujemy algorytm tak, by z jakiegokolwiek nieprawidłowego stanu po awarii system powrócił w skończonym czasie do stanu poprawnego. Nie zakładamy więc nic o stanach początkowych, czy inicjalnych wartościach zmiennych.

Model systemu rozproszonego

System rozproszony modelujemy grafem $G = (V, E)$, gdzie wierzchołki ze zbioru V identyfikujemy z jednostkami obliczeniowymi, a krawędzie ze zbioru E odpowiadają połączeniom komunikacyjnym pomiędzy nimi.

Zmienne lokalne każdej jednostki determinują jej *stan*. Sumę mnogościową stanów lokalnych poszczególnych jednostek nazywamy *stanem globalnym* systemu. Spośród wszystkich możliwych stanów globalnych wyróżniamy podzbiór *stanów legalnych*.

Model . . . (2)

Wykonanie algorytmu polega na krokowych zmianach stanów poszczególnych wierzchołków. Zmianę stanu jednego wierzchołka, czyli inaczej mówiąc zmianę wartości jego zmiennych lokalnych, będziemy nazywać *ruchem*. Pożądaną cechą algorytmów samostabilizujących się jest doprowadzenie systemu do stanu legalnego przy jak najmniejszej liczbie ruchów.

Sterowanie systemem:

- Sterowanie zewnętrzne (demon centralny lub lokalny).
- Brak zewnętrznego sterowania (synchronicznie, asynchronicznie)

Rozróżnianie procesorów.

- Każdy z procesorów posiada unikalny identyfikator.
- Procesory są nie rozróżnialne.

Miara wydajności algorytmów:

- Zliczanie ruchów.
- Zliczanie rund.

Formułowanie algorytmów

Niech p będzie warunkiem, a M akcją. Algorytm dla każdego wierzchołka u jest dany za pomocą reguł postaci:

R: **if** $p(u)$ **then** M.

Warunek p może dotyczyć stanu wierzchołka u oraz stanów wierzchołków z sąsiedztwa wierzchołka u. Jeśli warunek p dla wierzchołka u jest spełniony, to mówimy, że wierzchołek jest *aktywny*. Przyjmujemy, że żadne dwa ruchy nie są wykonywane w tym samym czasie

Wierzchołkowe kolorowanie grafów

Dany jest graf nieskierowany bez pętli (prosty) $G = (V, E)$. Szukamy funkcji $c : V \longrightarrow N$ (pokolorowania) takiej, że dla każdego $u, v \in V$, jeśli $\{u, v\} \in E$, to $c(v) \neq c(u)$. Funkcję c spełniającą taki warunek nazywamy *pokolorowaniem legalnym*. Żądamy, by liczność zbioru przydzielonych kolorów $c(V)$ była najmniejsza z możliwych, taką liczbę oznaczamy $\chi(G)$ i nazywamy *liczbą chromatyczną* G.

Algorytm 1**:** Fast coloring

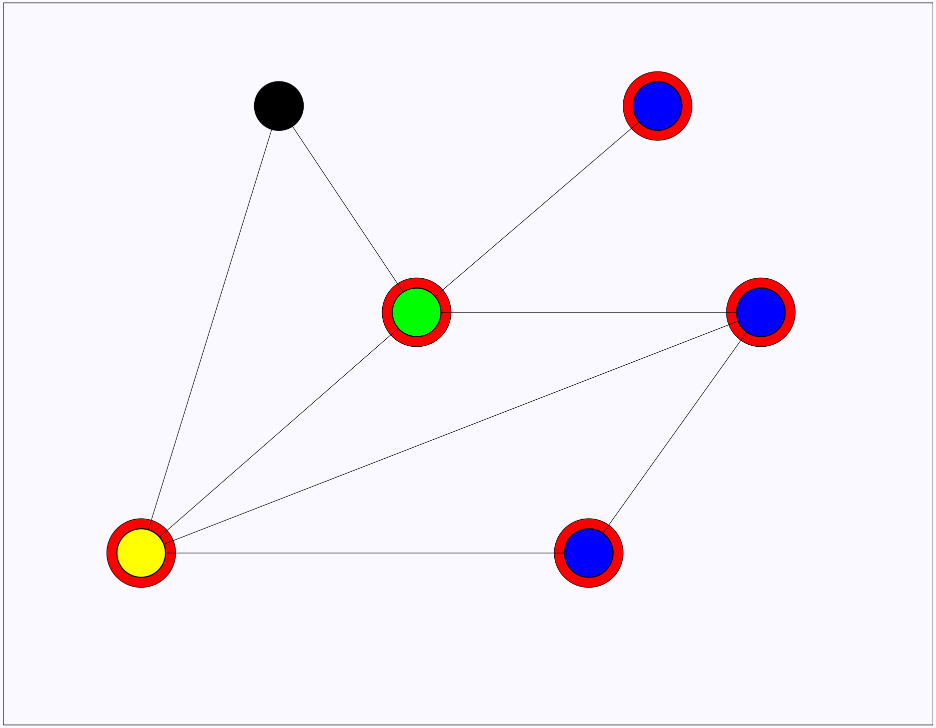
R: **if** $c(u) \geqslant \deg(u) + 1 \vee c(u) \in \{c(v) \mid v \in N(u)\}$
then $c(u) = \min\{d \geqslant 1 \mid (\forall v \in N(u))(c(v) \neq d)\}$

Algorytm 2**:** Grundy coloring

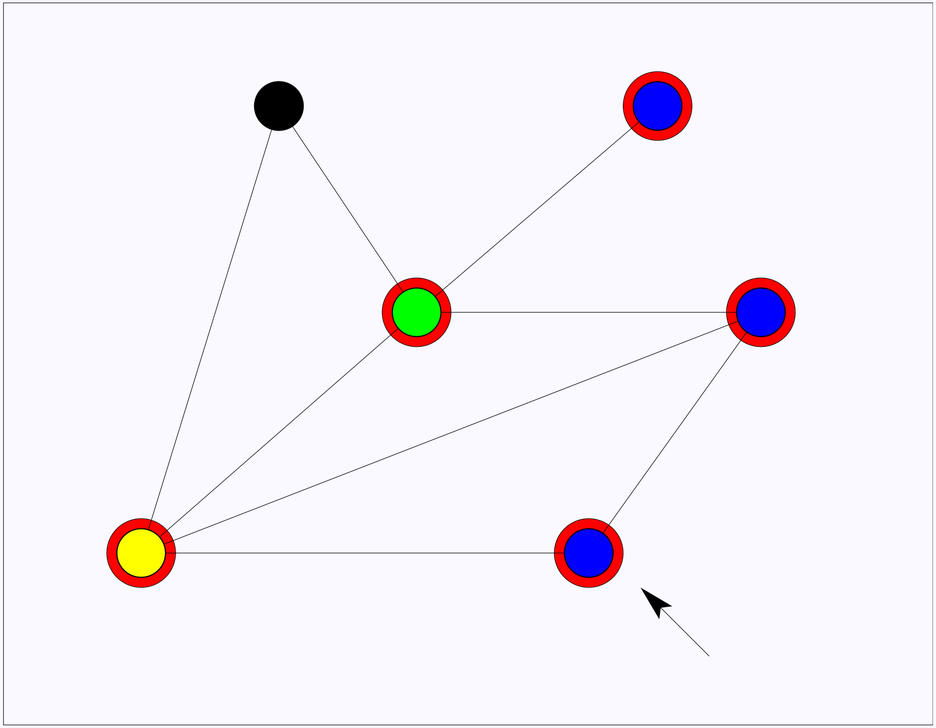
R: **if** $c(u) \neq \min\{d \geqslant 1 \mid (\forall v \in N(u))(c(v) \neq d)\}$
then $c(u) = \min\{d \geqslant 1 \mid (\forall v \in N(u))(c(v) \neq d)\}$

Algorytm 3: Kolorowanie według reguły silniejszy najpierw.

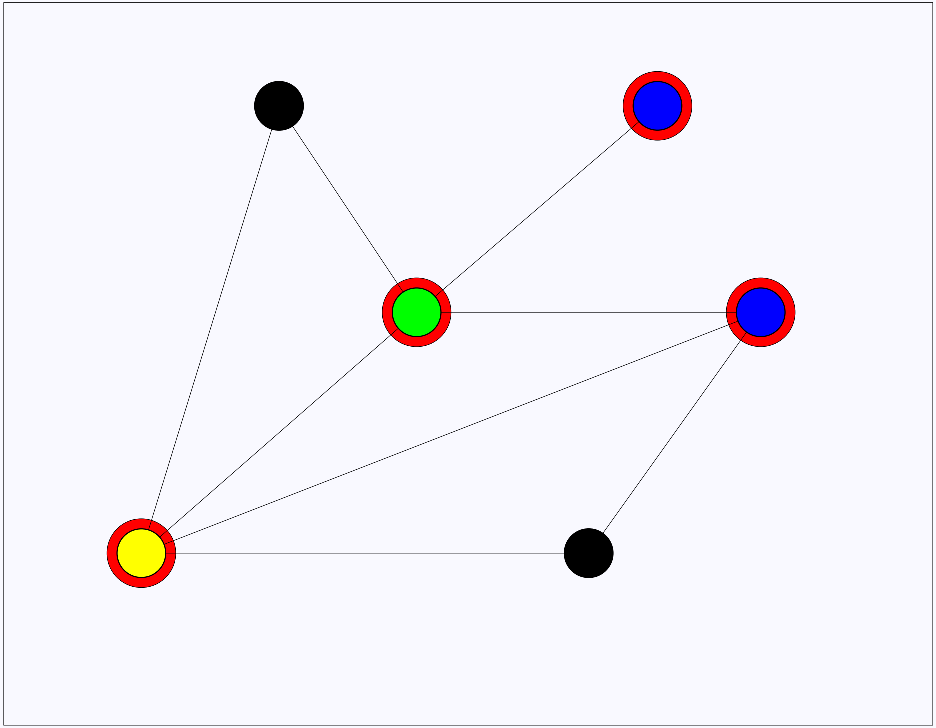
R: **if** $c(u) \neq \min\{d \geqslant 1 \mid (\forall v \in N_{\geqslant}(u))(c(v) \neq d)\}$
then $c(u) = \min\{d \geqslant 1 \mid (\forall v \in N_{\geqslant}(u))(c(v) \neq d)\}$



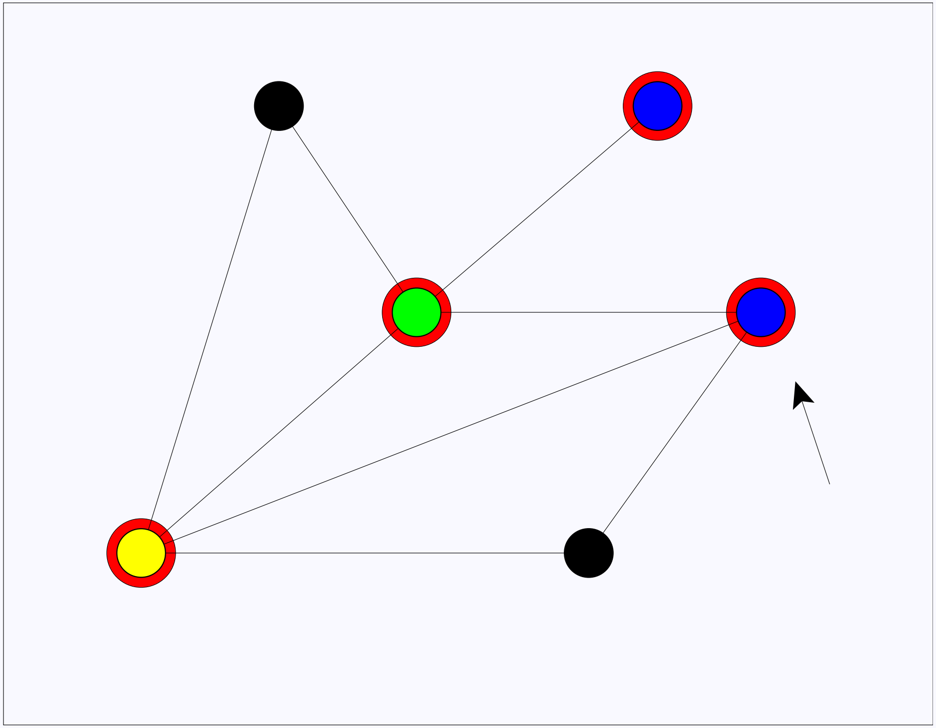
Sekwencja kolorów: czarny, niebieski, zielony, żółty,...



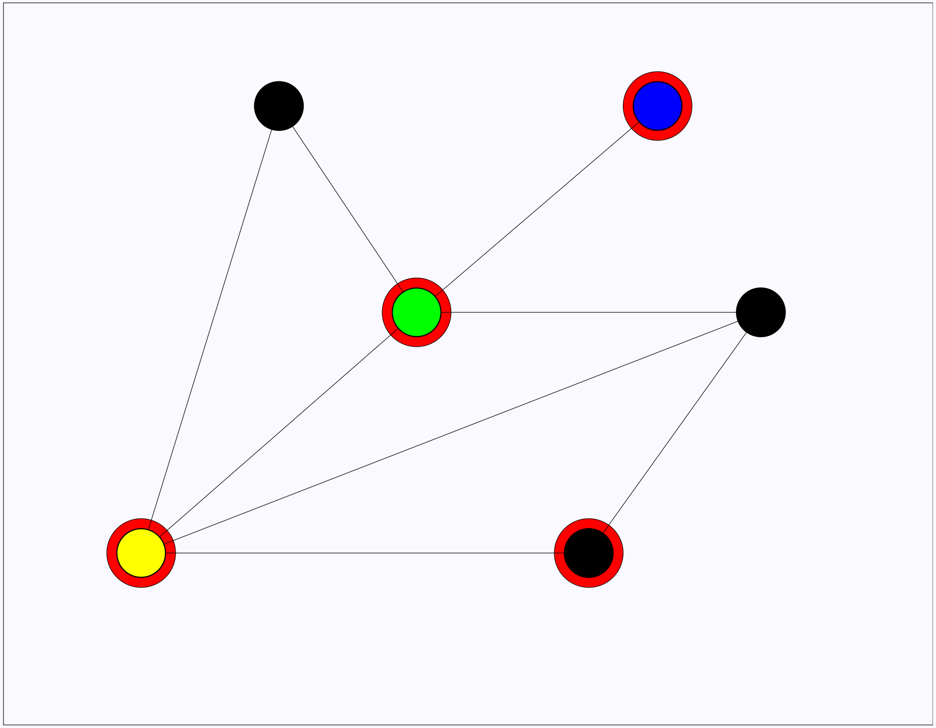
Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .



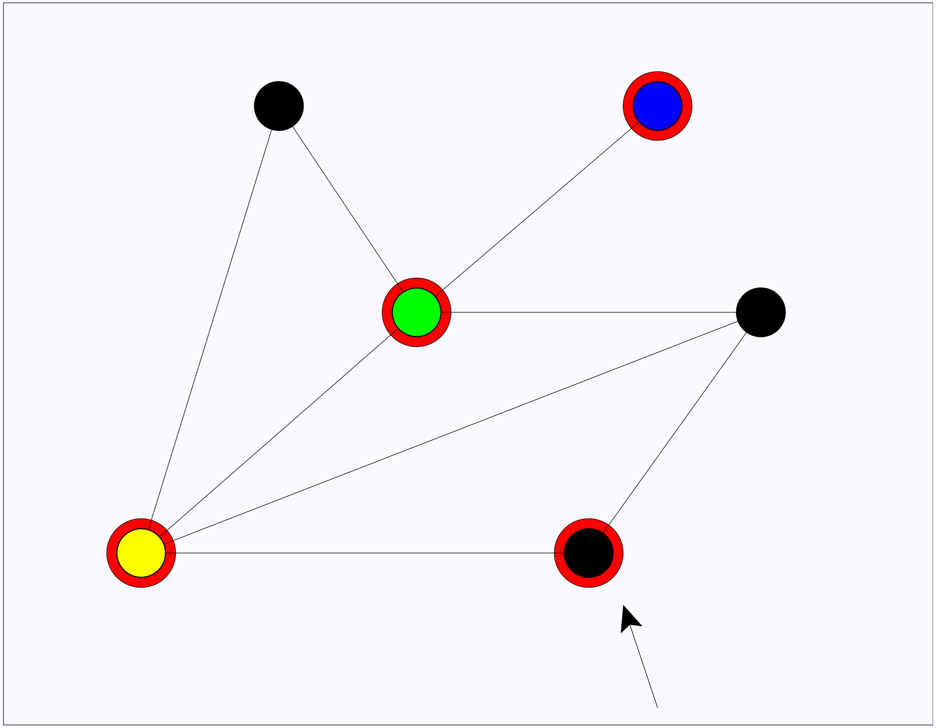
Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .



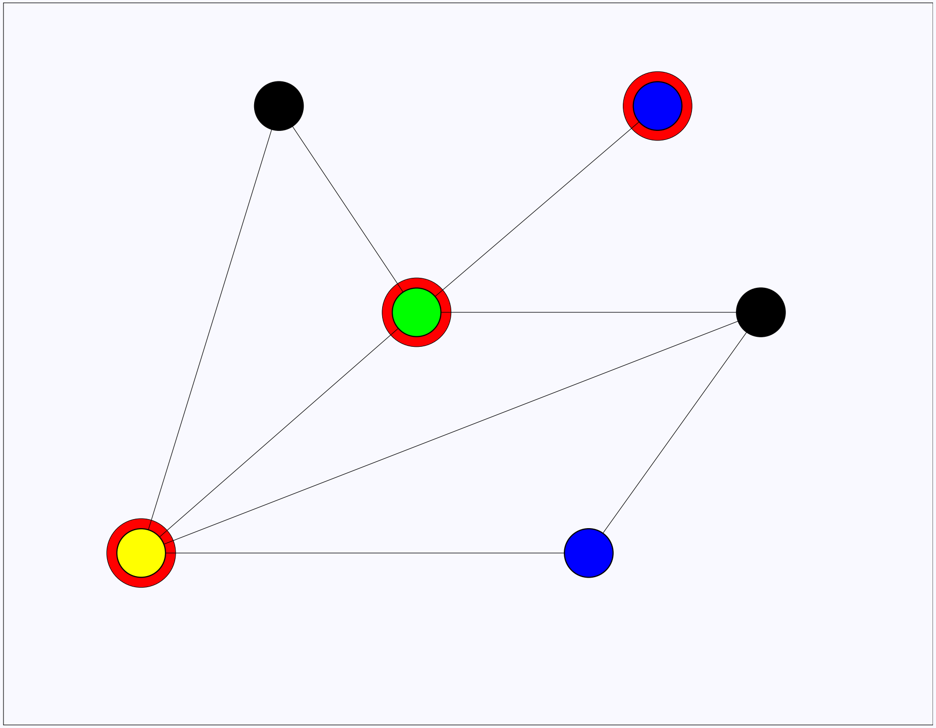
Sekwencja kolorów: czarny, niebieski, zielony, żółty,...



Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .

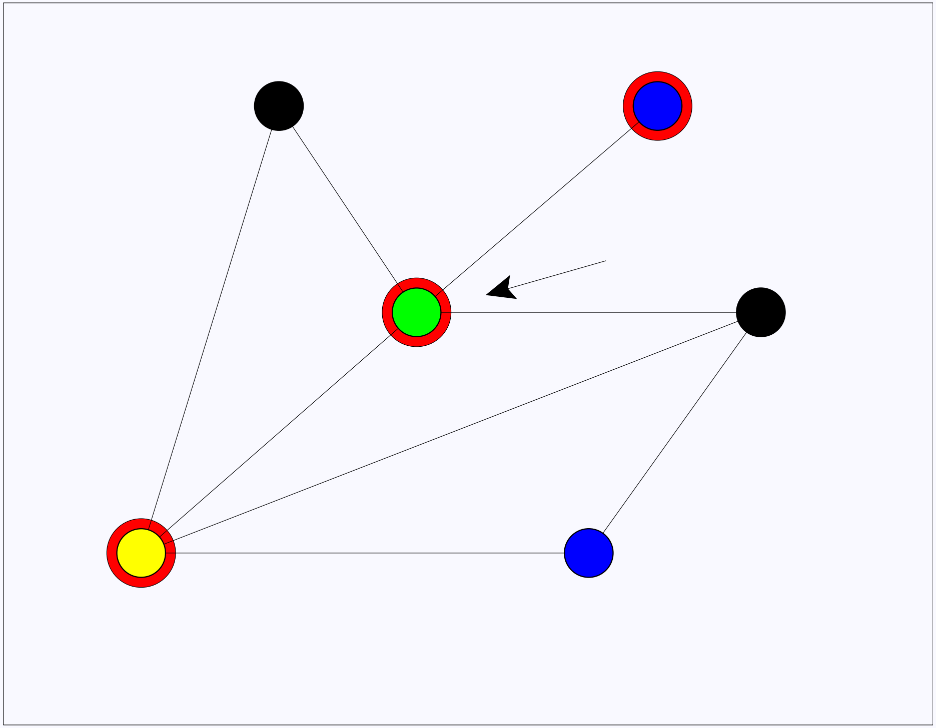


Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .

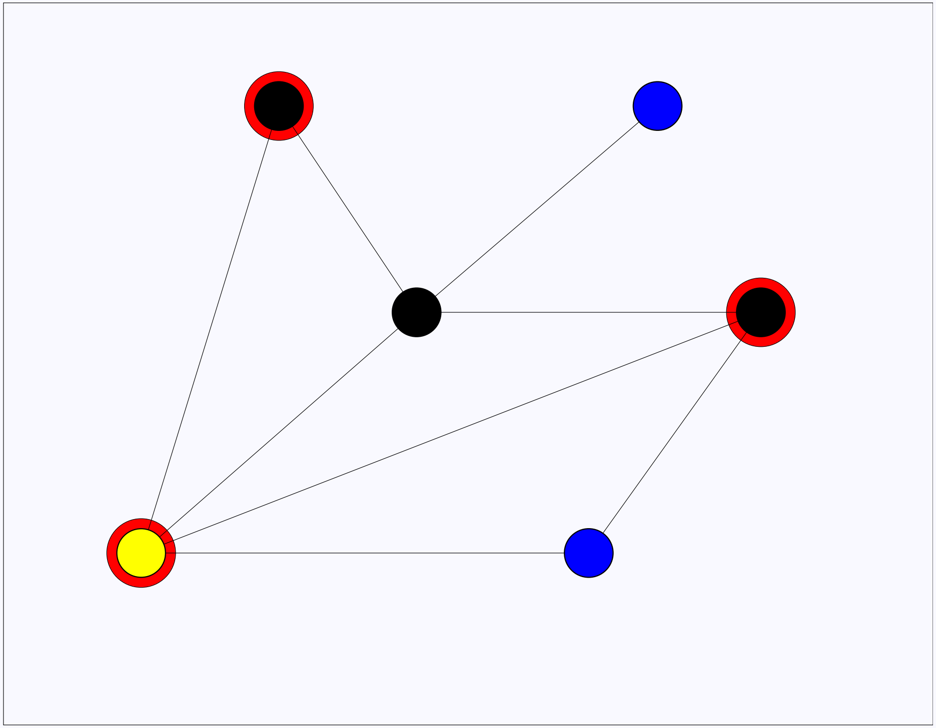


Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .

stan: 4

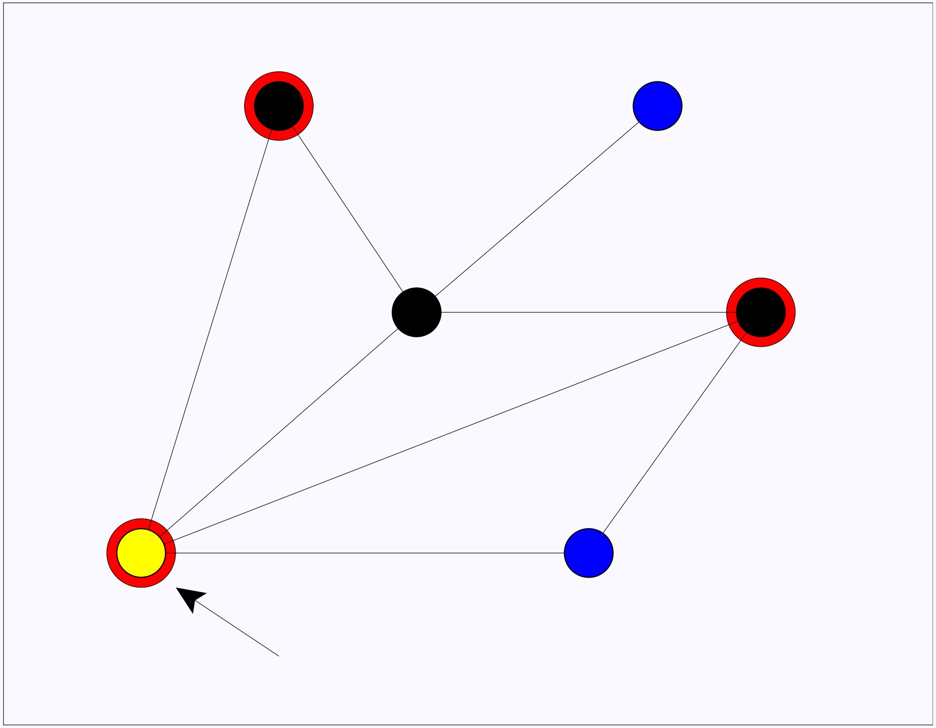


Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .

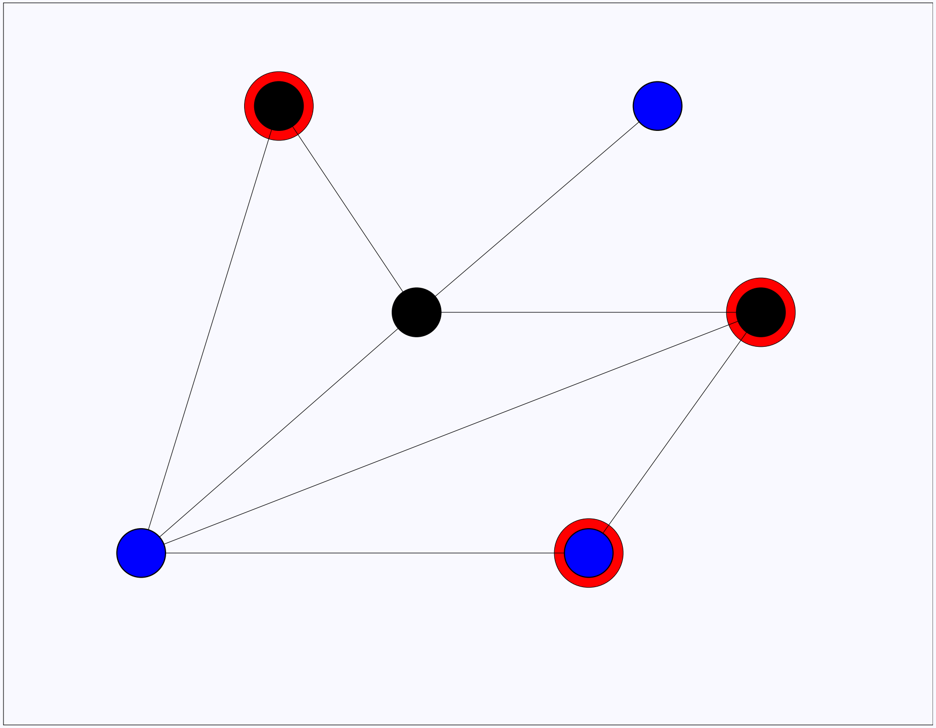


Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .

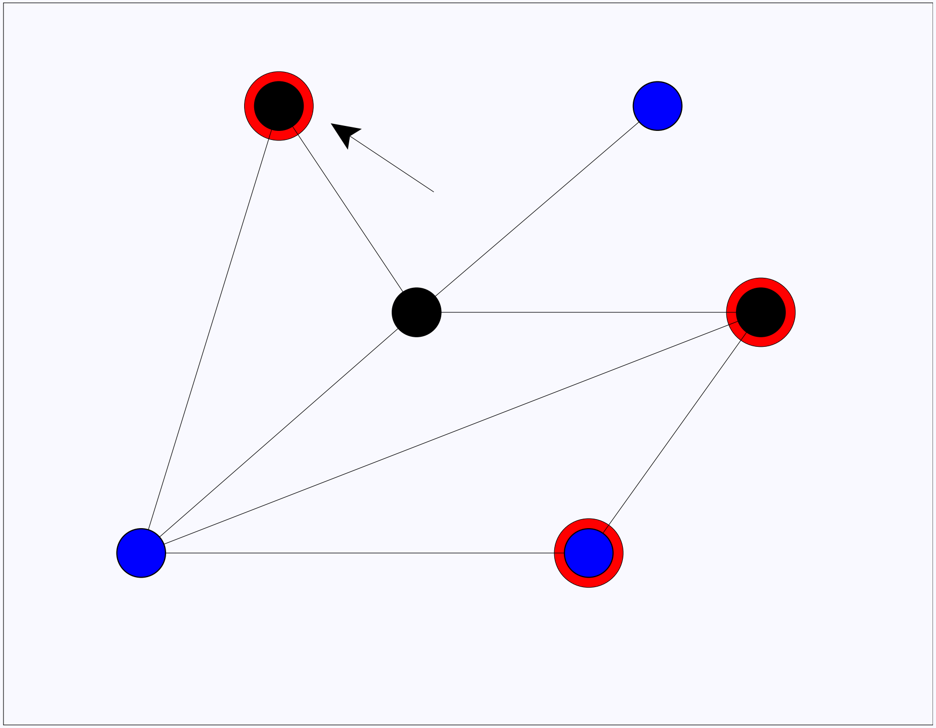
stan:



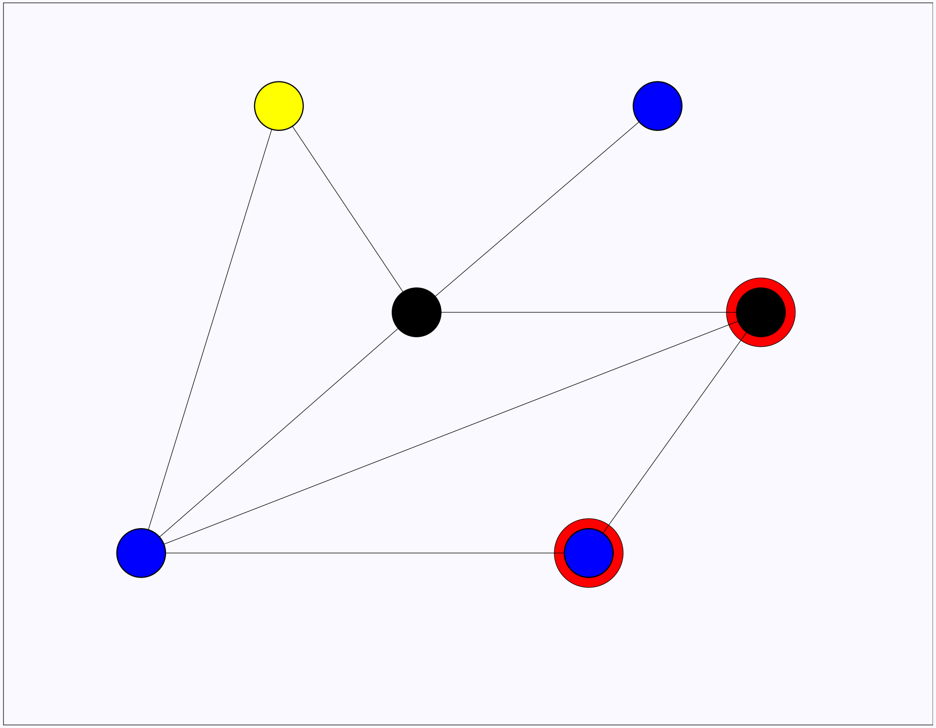
Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .



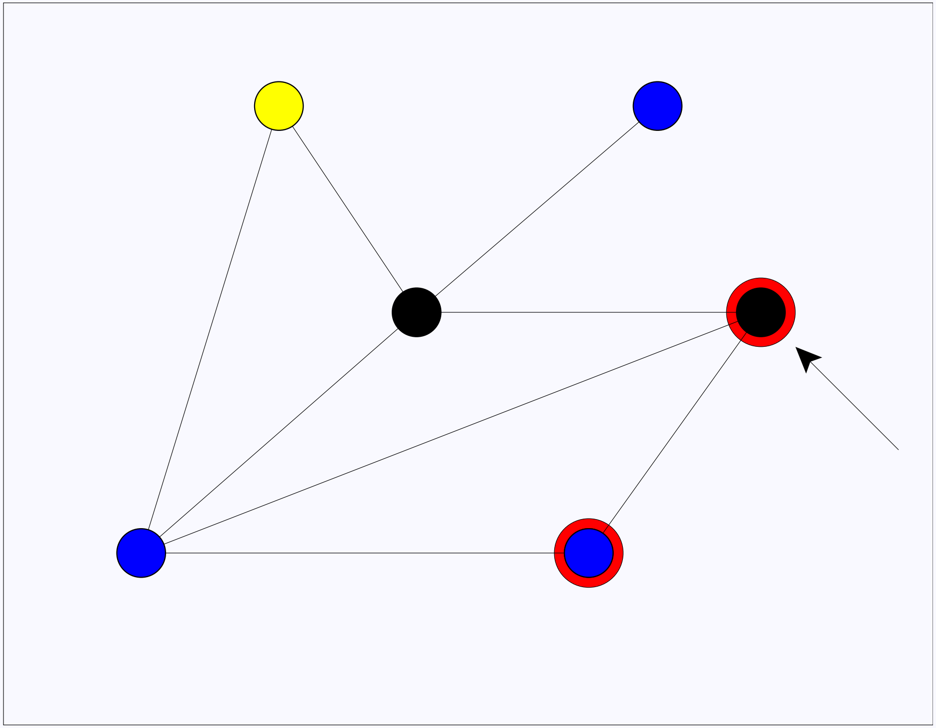
Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .



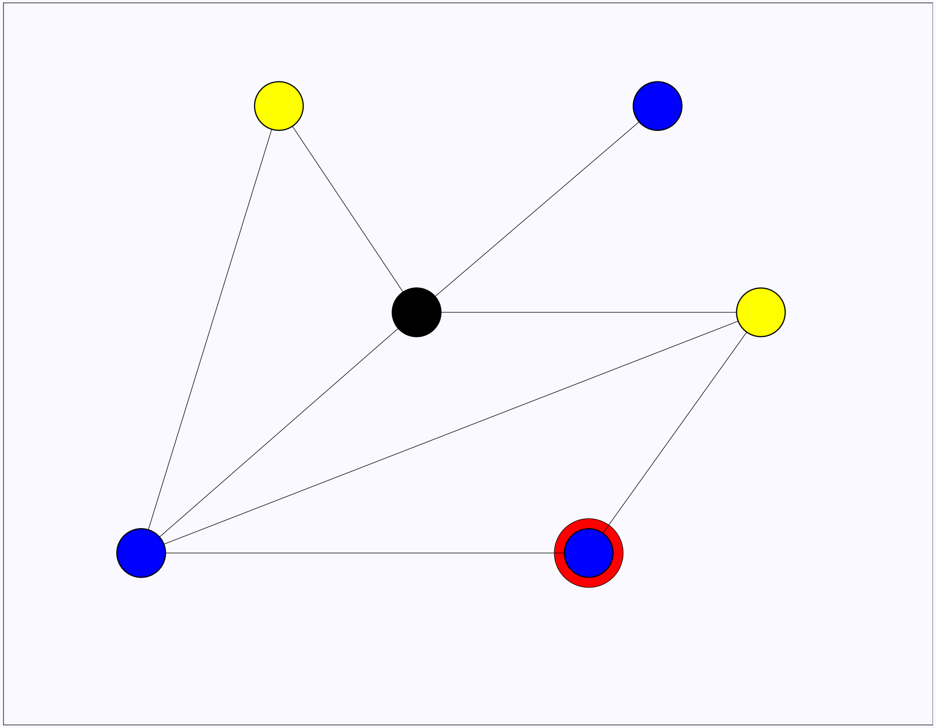
Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .



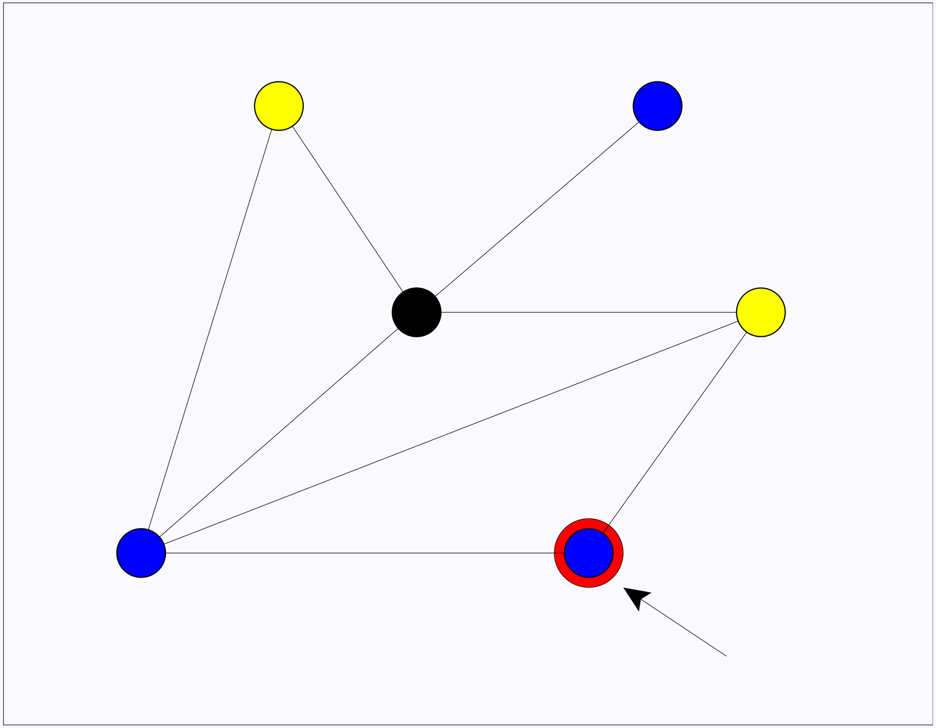
Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .



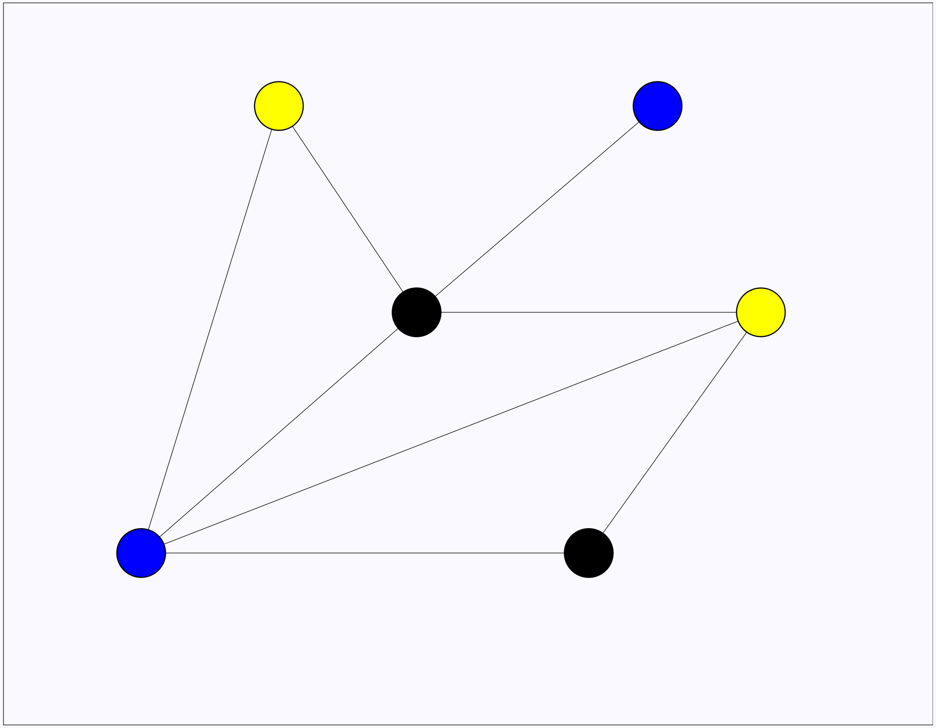
Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .



Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .



Sekwencja kolorów: czarny, niebieski, zielony, żółty,...



Sekwencja kolorów: czarny, niebieski, zielony, żółty,. . .

Maksymalne skojarzenie

Algorytm 4: Hsu-Huang

R1 : **if** $(i \rightarrow null) \wedge (\exists j \in N(i))(j \rightarrow i)$
then $\mathrm{set}(i \rightarrow j)$

R2: **if** $(i \rightarrow null) \wedge$
$(\forall k \in N(i))(\neg(k \rightarrow i) \wedge (\exists j \in N(i))(j \rightarrow \mathrm{null}))$
then $\mathrm{set}(i \rightarrow j)$

R3: **if** $(i \rightarrow j) \wedge (j \rightarrow k) \wedge (k \neq i)$
then $\mathrm{set}(i \rightarrow \mathrm{null})$

Wierzchołki i,j są skojarzone jeśli $(j \to i) \wedge (i \to j)$

Fakt 1 *Jeśli* i *jest skojarzony z* j*, to żaden z nich nie zmieni stanu.*

Fakt 2 *Po zmianie stanu* (i, j, R_2) *na parze* i,j *może zajść tylko zmiana stanu* (i, j, R_3)

Fakt 3 *Po zmianie stanu* (i, j, R_2) *musi zajść dokładnie jedna zmiana na* i,j *a mianowicie* (i, j, R_1) *lub* (i, j, R_3)*.*

Fakt 4 *Po zmianie stanu* (i, j, R_3) *mogą być co najwyżej dwie zmiany stanu na* i,j*.*

Fakt 5 *Liczba zmian stanu na każdej krawędzi może być co najwyżej* 3*. I jest równa* 3 *na co najwyżej* n *parach.*

Fakt 6 *Dla dowolnego grafu algorytm Hsu i Huang'a ustabilizuje się po co najwyżej* $2m + n$ *zmianach stanu.*