On two problems related to communication in wireless sensor networks

Gabriel Semanišin

Ústav informatiky
Prírodovedecká fakulta, Univerzita P.J. Šafárika
Košice, Slovensko
e-mail: gabriel.semanisin@upjs.sk

CEX CaKS

23. 06. 2010

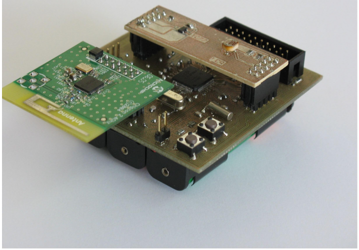
Európska únia
Európsky fond regionálneho rozvoja

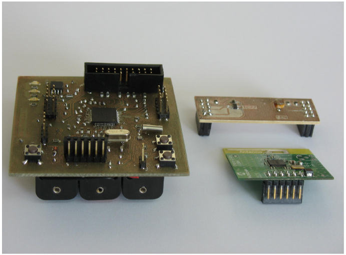
Operačný program VÝSKUM a VÝVOJ

Agentúra Ministerstva školstva SR pre štrukturálne fondy EÚ

Podporujeme výskumné aktivity na Slovensku/
Projekt je spolufinancovaný zo zdrojov EÚ

The beginning of the story



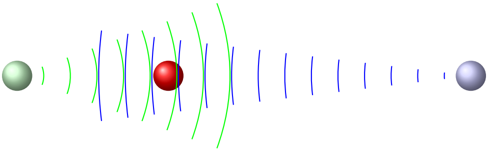


Our research is motivated by real construction of wireless sensor networks based on **CDMA sensors.**

Wireless sensor networks (WSN)

WSN is a special type of ad-hoc wireless networks such that its nodes are devices with embedded

- microcontroller,
- sensor,
- FM radio and
- power source.



Requirements for WSN

From practical point of view, there is a need for WSN with

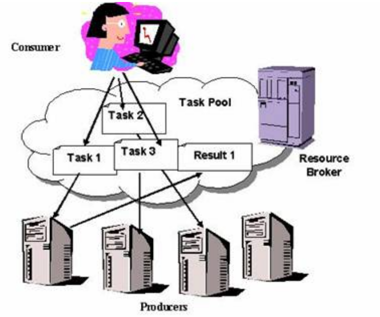
- low message latency,
- real time reaction,
- high message delivery reliability,
- high robustness,
- high security.

Security aspects of WSN

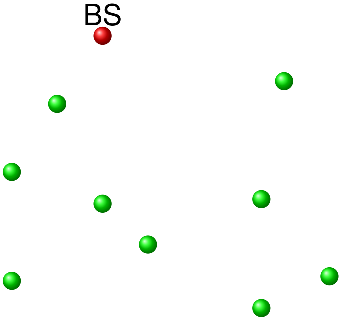
In general, a standard **sensor device** is not considered as tamper-resistant and due to increasing costs it is not supposed to make all devices of a sensor network tamper-proof.

Hence, the design of new protocols has become a challenge for security research, as **it is necessary to combine the properties of cryptographic primitives and the network topologies**.

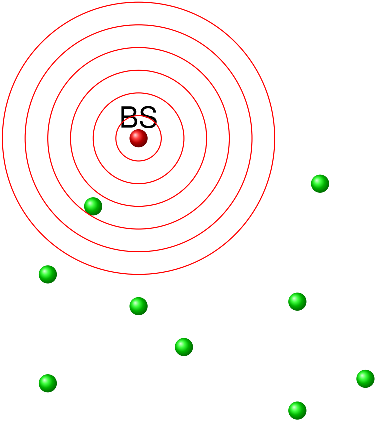
1st Problem: **Semi-matching**



Building WSN

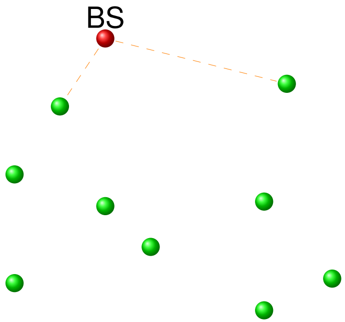


Building WSN



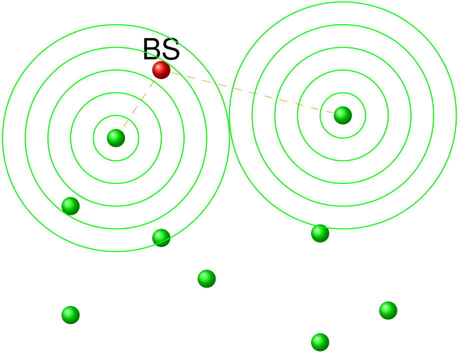
1. Building reachability graph

Building WSN



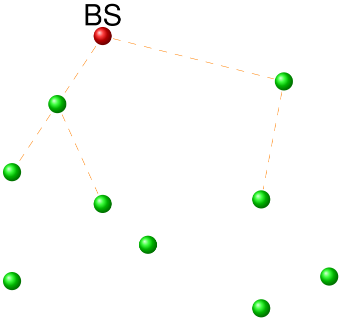
1. Building reachability graph

Building WSN



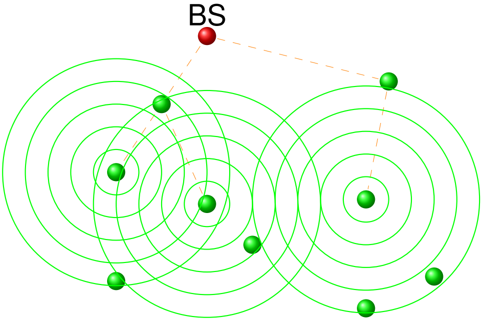
1. Building reachability graph

Building WSN



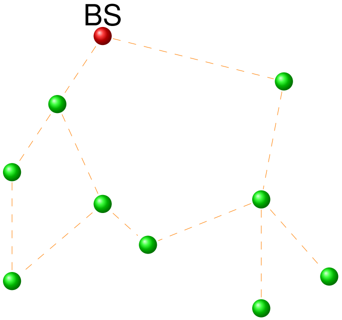
1. Building reachability graph

Building WSN



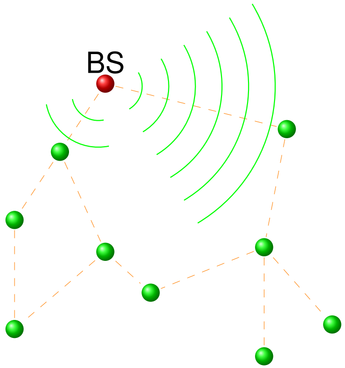
1. Building reachability graph

Building WSN



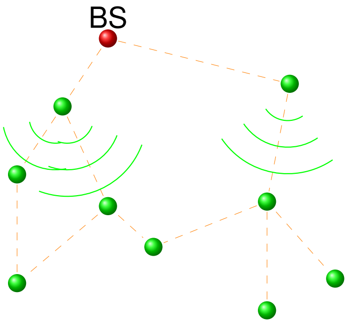
1. Building reachability graph

Building WSN



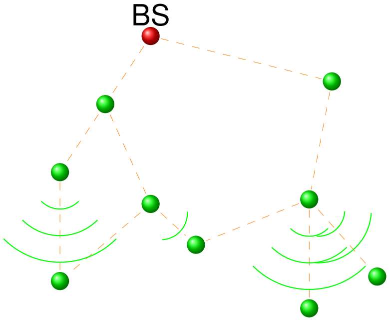
1. Building reachability graph
2. Building BFS tree, establishing connections

Building WSN



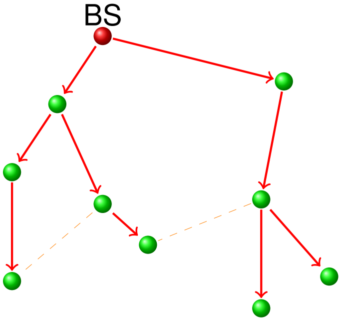
1. Building reachability graph
2. Building BFS tree, establishing connections

Building WSN



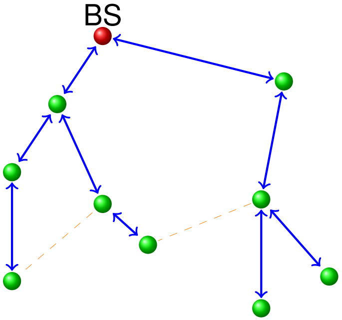
1. Building reachability graph
2. Building BFS tree, establishing connections

Building WSN



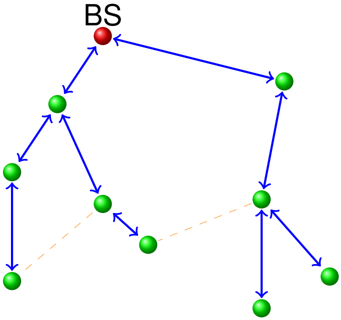
1. Building reachability graph
2. Building BFS tree, establishing connections

Building WSN



1. Building reachability graph
2. Building BFS tree, establishing connections
3. Establishing secure connections

Building WSN



1. Building reachability graph
2. Building BFS tree, establishing connections
3. Establishing secure connections
4. Secure communication

WSN based on CDMA technology

WSN based on CDMA technology

- consists of node that are able to communicate each other with respect to their physical limitations and mutual distance,
- the sink of the network (base station) has relatively large computational capabilities and energy sources,
- the number of hops from a given node to sink must be as small as possible,
- the number of communication channels available at the sensors is limited (say 16).

Efficiency of the algorithms for network building

From practical point of view it is not important **how efficient** are the algorithms used for establishing communication channels (providing that the algorithms are "good" enough) because

- the time of creating WSN very small with respect to time of operation,
- the operations performed at the nodes take more time than the transmission.

Efficiency of the algorithms for network building

From practical point of view it is not important **how efficient** are the algorithms used for establishing communication channels (providing that the algorithms are “good” enough) because

- the time of creating WSN very small with respect to time of operation,
- the operations performed at the nodes take more time than the transmission.

!!! BUT !!!

The time spent during building WSN is crucial for establishing secure WSN.

Graph theoretical approach

The limited number of sensor channels leads to the problem of finding BSF-tree with bounded degree.

Graph theoretical approach

The limited number of sensor channels leads to the problem of finding BSF-tree with bounded degree.

Bad news

The problem of finding spanning tree with bounded degree is NP-complete.

Graph theoretical approach

The limited number of sensor channels leads to the problem of finding BSF-tree with bounded degree.

Bad news

The problem of finding spanning tree with bounded degree is NP-complete.

Good news

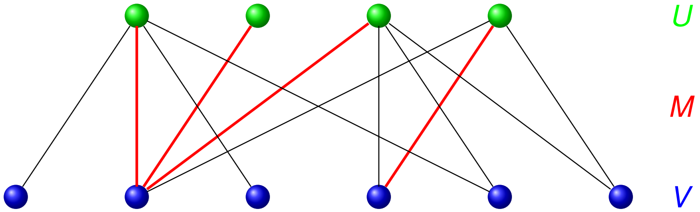
We can restrict our consideration to a bipartite graph that is formed by two layers in BSF-tree.

Semi-matching

Given a positive integer k and a bipartite graph $G = (U \cup V, E)$ with n vertices and m edges.

Definition

A $k-$*cover* of U in G is a set $M \subseteq E$ such that **each vertex from** U **has at least** k **incident edges from** M.
If the vertices from U are incident exactly to k**-edges** from M then we say that U is a k-semi-matching. If $k = 1$ then we simply say semi-matching.



Load balancing problem

Load balancing problem

If U is a set of tasks and V is a set of machines, one might want to assign every task to a machine.

To satisfy this condition, more than one task might be assigned to a single machine, i.e. more than one vertex in U is assigned to some vertex in V. Thus, the set of edges corresponding to the assignment in this case is a semi-matching.

Load balancing problem

N. J. A. Harvey, R. E. Ladner, L. Lovász, and T. Tamir consider several goals to optimize:

- to minimize the makespan (the maximal number of tasks assigned to any given machine) of the schedule,
- to minimize the average completion time of the tasks (flow time),
- to maximize the fairness of the assignment from the machines point of view, i.e. to minimize the variance of the loads on the machine.

Load balancing problem (continued)

Harvey et al. showed that the optimal semi-matching minimizes simultaneously

- the maximal number of tasks assigned to a machine,
- the flow time,
- the variance of loads.

Algorithms

- The first proposed algorithm has complexity $O(mn)$.
- The second algorithm has upper bound for the complexity $O(\min(n^{3/2}, mn)m)$.

Lexicographically minimum semi-matching

Let $G = (U, V, E)$ be a bipartite graph, $F \subseteq E$ and $X \subseteq V$. Let $d_F(X)$ be the sequence $d_1 \geq d_2 \geq \cdots \geq d_{|X|}$ denoting the degrees of the vertices of V in $G\langle F\rangle$ respectively.

Theorem (D. Bokal, B. Brešar, J. Jerebic, 2009)

There exist an algorithm with running time $O(mn)$ that finds a semi-matching M of U in a bipartite graph $G(U, V, E)$ with lexicographically minimal sequence $d_M(V)$.

The algorithm

- generalises Hungarian method,
- can work on-line as well.

Bounded degree semi-matching

Given a positive integer d and a bipartite graph $G = (U \cup V, E)$ with n vertices and m edges.

Definition

A bounded-degree semi-matching on G is a set of edges $M \subseteq E$ such that

- **each vertex in** U is incident to **exactly one edge** in M and
- **every vertex from** V is incident to **at most** d **edges** of G.

Bounded-degree semi-matching algorithm

Algorithm 1: degree_semimatching(G, k)

Input: $G = (A, B, E)$ a bipartite graph and a positive integer k. ;
Output: 1-semimatching M, $M \subseteq E$, $d_M(v) \leq k$ for every vertex $v \in A$;
forall the $u \in A$ **do**
  $capacity(u) = k$;
forall the $u \in B$ **do**
  $capacity(u) = 1$;
$M = \emptyset$;
loop $2\sqrt{N}$ times **do**
  forall the $u \in B, capacity(u) > 0$ **do**
    Add u to L_0;
    $indegree(u) = 1$;
  `Layer_Construction(0)`
return M;

Bounded-degree semi-matching algorithm

Procedure Layer Construction(i)

repeat
- **forall the** $u \in L_i$ **do**
  - **forall the** v adjacent to u using an unmatched edge **do**
    - **if** v is not from an earlier layer **then**
      - add v to L_{i+1}; $indegree(v) = indegree(v) + 1$;
- **if** any of the vertex in L_{i+1} is a free vertex **then**
  - Delete all matched vertices from L_{i+1}; $t = i + 1$;
  - `AugmentPath();`
- **else**
  - **forall the** $u \in L_{i+1}$ **do**
    - **for** v adjacent to u using a matched edge **do**
      - **if** v is not from an earlier layer **then**
        - add v to L_{i+2}; $indegree(v) = indegree(v) + 1$;
- $i = i + 2$;

until all vertices have been classified or `AugmentPath()` was called;

Augmenting path procedure

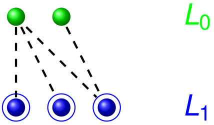
Procedure AugmentPath

while there is a vertex u in L_t, $capacity(u) > 0$ **do**
  Trace backwards from u to a free vertex in L_0 to obtain an augmenting path P;
  If no such $\mathcal{P}$ was found, erase u from L_t and continue with a next vertex;
  forall the $v \in P$ **do**
    $indegree(v) = indegree(v) - 1$
  Place all vertices of $\mathcal{P}$ with $indegree = 0$ into deletion queue Q;
  Decrease *capacity* for the first and last vertex of $\mathcal{P}$;
  while Q is non-empty **do**
    remove a vertex $v \in Q$ from the queue Q;
    foreach edge (v, w), such that w is in the layer after u **do**
      $indegree(w) = indegree(w) - 1$;
      if $indegree(w) = 0$ **then**
        Place w into Q;

How the algorithm works

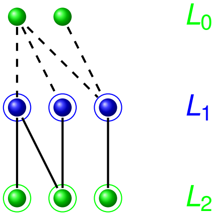
1. Put all unmatched vertices from V to L_0

How the algorithm works



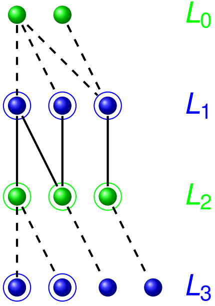
1. Put all unmatched vertices from V to L_0
2. Scan unmatched edges from L_0 and create L_1

How the algorithm works



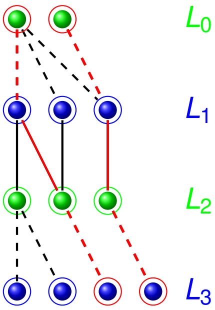
1. Put all unmatched vertices from V to L_0
2. Scan unmatched edges from L_0 and create L_1
3. Scan matched edges from L_1 and create L_2.

How the algorithm works



1. Put all unmatched vertices from V to L_0
2. Scan unmatched edges from L_0 and create L_1
3. Scan matched edges from L_1 and create L_2.
4. Scan unmatched edges from L_2 and create L_3.
5. ...

How the algorithm works



1. Put all unmatched vertices from V to L_0
2. Scan unmatched edges from L_0 and create L_1
3. Scan matched edges from L_1 and create L_2.
4. Scan unmatched edges from L_2 and create L_3.
5. ...
6. Find all vertex-disjoint augmenting paths and improve semi-matching .

The complexity of the algorithm

Lemma

The length of the shortest augmenting path increases in each phase.

The complexity of the algorithm

Lemma

The length of the shortest augmenting path increases in each phase.

Lemma

Let M be a semi-matching that is not maximum. Let M^* be a maximum semi-matching. Let $|M^*| - |M| = k$. Then there are k vertex-disjoint augmenting paths in $M^* \oplus M$.

The complexity of the algorithm

Lemma

The length of the shortest augmenting path increases in each phase.

Lemma

Let M be a semi-matching that is not maximum. Let M^* be a maximum semi-matching. Let $|M^*| - |M| = k$. Then there are k vertex-disjoint augmenting paths in $M^* \oplus M$.

Theorem (F. Galčík, J. Katrenič, G.S., 2009)

There exists a deterministic algorithm that find bounded-degree semi-matching in a bipartite graph G with n vertices and m edges in $O(\sqrt{n}m)$ running time.

The comparison with the previous results

Theorem (F. Galčík, J. Katrenič, G.S., 2009)

There exists a deterministic algorithm that find bounded-degree semi-matching in a bipartite graph G with n vertices and m edges in $O(\sqrt{n}m)$ running time.

This provides an improvement of the efficiency of the algorithms designed by Bokal et al. and Harvey et al. - both they need running time $O(mn)$.

The algorithm is based on the idea presented by Hopcroft and Karp.

A new challenge

Theorem (M. Mucha and P. Sankowski, 2004)

There an algorithm that solves the Matching Problem in bipartite graphs with running time $O(n^{\omega})$, where ω is the exponent of the best known matrix multiplication algorithm.

Currently $\omega < 2.38$.

We believe that the Gaussian Elimination approach can be utilized for the Bounded-degree semi-matching Problem as well.

Not sufficiently answered questions

Modification of BSF-tree

- How to modify the communication tree if we cannot find any solution of bounded-degree semi-matching problem?

Not sufficiently answered questions

Modification of BSF-tree

- How to modify the communication tree if we cannot find any solution of bounded-degree semi-matching problem?

Rearranging of the communication topology

- How we have to change the communication tree in a case of sensor failure?

Not sufficiently answered questions

Modification of BSF-tree

- How to modify the communication tree if we cannot find any solution of bounded-degree semi-matching problem?

Rearranging of the communication topology

- How we have to change the communication tree in a case of sensor failure?

Improving the reliability of the network

- How we can improve the reliability of the communication network by a duplication of established communication channels?

Extensions

Definition

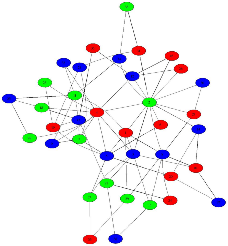
Let $G = (U, V, E)$ be a bipartite graph. Let $f : U \to \mathbf{N}$ and $g : V \to \mathbf{N}$ be mappings. We say that M is a (f, g)-cover of G if

- $\deg_M(u) \leq f(u)$ for each vertex $u \in U$ and
- $\deg_M(v) \geq g(v)$ for each vertex $v \in V$.

The function f provides a limitation for degree of the vertices in the first layer, while g is the need for the vertices in the second layer.

Obviously, the (f, g)-cover is a generalization of bounded-degree semi-matching (just take constant functions $f(x) = d$ and $g(x) = 1$.

2nd Problem: **Path Security Number**



A communication protocol for WSN

One of the protocol for **ensuring data integrity communication in WSN** is called Canvas scheme. M. Novotný proposed a protocol for protecting data integrity of the message **witnessed by** k **previous hops in the routing.**

Main idea of the protocol

The protocol is based on the fact that each node v shares keys with nodes that have distance at most k from v. Moreover, the node v knows information about the local topology of the network up to the distance k.

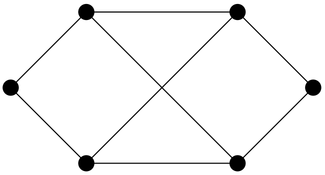
In such a way the node v can check the **validity of a path of length** k, ending in v. The k-generalized Canvas scheme guarantees data integrity under **the assumption** that each path of length $k-1$ contains a node that is not captured. This can be ensured by using a set of so-called protected sensors that are resistant against the attacks of a potential intruder.

Path Security Number of a Graph

Definition

Let G be a graph and let $k \geq 2$ be an integer. A subset of vertices $S \subseteq V(G)$ is called a *k-path security set* if **every path of order** k in G **contains at least one vertex** from S.
The *k-path security number* $\psi_k(G)$ is defined as follows:

$$\psi_k(G) = \min\{|S| : \text{S is a } k\text{-path security set}\}.$$



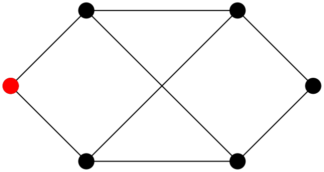
$\psi_4(G) = ?$

Path Security Number of a Graph

Definition

Let G be a graph and let $k \geq 2$ be an integer. A subset of vertices $S \subseteq V(G)$ is called a k-path security set if **every path of order k in G contains at least one vertex** from S.
The k-path security number $\psi_k(G)$ is defined as follows:

$$\psi_k(G) = \min \{|S| : \text{S is a } k\text{-path security set}\} .$$



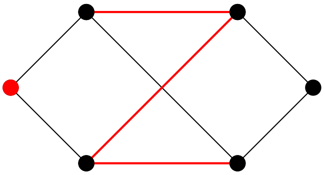
$\psi_4(G) = ?$

Path Security Number of a Graph

Definition

Let G be a graph and let $k \geq 2$ be an integer. A subset of vertices $S \subseteq V(G)$ is called a *k*-path security set if **every path of order** k in G **contains at least one vertex** from S.
The *k*-path security number $\psi_k(G)$ is defined as follows:

$$\psi_k(G) = \min \{|S| : S \text{ is a } k\text{-path security set}\} .$$



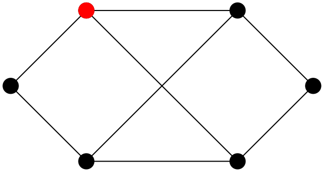
$\psi_4(G) = ?$

Path Security Number of a Graph

Definition

Let G be a graph and let $k \geq 2$ be an integer. A subset of vertices $S \subseteq V(G)$ is called a *k-path security set* if **every path of order k in G contains at least one vertex** from S.
The *k-path security number* $\psi_k(G)$ is defined as follows:

$$\psi_k(G) = \min\{|S| : S \text{ is a } k\text{-path security set}\}.$$



$\psi_4(G) = ?$

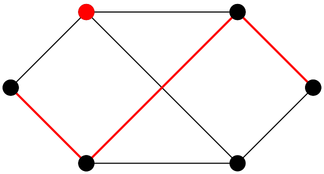
Path Security Number of a Graph

Definition

Let G be a graph and let $k \geq 2$ be an integer. A subset of vertices $S \subseteq V(G)$ is called a k-path security set if **every path of order k in G contains at least one vertex** from S.

The k-path security number $\psi_k(G)$ is defined as follows:

$$\psi_k(G) = \min \{|S| : \text{S is a } k\text{-path security set}\}.$$



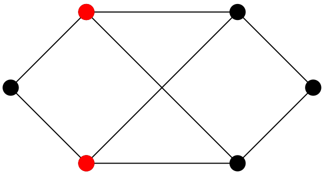
$\psi_4(G) = ?$

Path Security Number of a Graph

Definition

Let G be a graph and let $k \geq 2$ be an integer. A subset of vertices $S \subseteq V(G)$ is called a *k*-path security set if **every path of order** k in G **contains at least one vertex** from S.
The *k*-path security number $\psi_k(G)$ is defined as follows:

$$\psi_k(G) = \min \{|S| : \text{S is a } k\text{-path security set}\}.$$



$\psi_4(G) = ?$

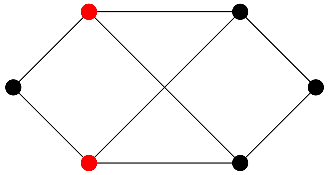
Path Security Number of a Graph

Definition

Let G be a graph and let $k \geq 2$ be an integer. A subset of vertices $S \subseteq V(G)$ is called a *k-path security set* if **every path of order** k in G **contains at least one vertex** from S.

The *k-path security number* $\psi_k(G)$ is defined as follows:

$$\psi_k(G) = \min\{|S| : \text{S is a } k\text{-path security set}\}.$$



$\psi_4(G) = 2$

PSN - another formulation

Definition

Given a graph G and a path k. What is the minimum number of vertices that **we have to colour** so that **the non-coloured vertices form a P_k-free set**?

Character of the problem

- related to **generalised domination**
- related to **generalised colouring** (partition according to prescribed rules)
- related to **vertex cover** (set cover)
- ...

(Partially) related invariant

Definition (F. Bullock, M. Frick, . . .)

Given a graph G and a constant k. The k-path chromatic number is the minimum number of colours that are necessary for colouring the vertices of G in such a way that each colour class forms a P_k-free set.

Proposition

If the k**-path chromatic number** *of a graph* **is two** *then*

$$\psi_k(G) \leq |V(G)|/2.$$

Basic properties of PSN

Proposition

Let G be a graph.

- *If $2 \leq k_1 \leq k_2$ then $\psi_{k_2}(G) \leq \psi_{k_1}(G)$.*
- *(Heredity) If $2 \leq k$ and H is a subgraph of G then $\psi_k(H) \leq \psi_k(G)$.*
- *(Additivity) If $2 \leq k$ then*
  $\psi_k(G) = \max\{\psi_k(G^*) : G^*$ *is a component of* $G\}$.
- *(Minimum Vertex Cover) $\psi_2(G) = \beta(G)$.*

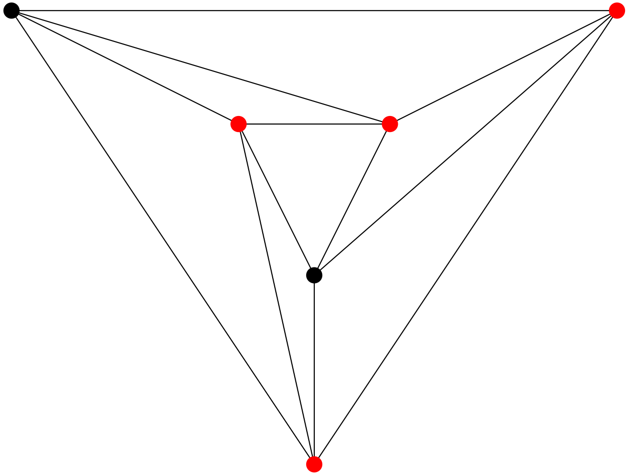
PSN for certain classes of graphs

Proposition

Let $k \geq 2$ and $n \geq k$ be positive integers. Then

- $\psi_k(P_n) = \left\lfloor \frac{n}{k} \right\rfloor$;
- $\psi_k(C_n) = \left\lceil \frac{n}{k} \right\rceil$;
- $\psi_k(W_n) = 1 + \left\lceil \frac{n}{k} \right\rceil$;
- $\psi_k(K_n) = n - k + 1$;
- $\psi_k(H_{2n+1}) = \frac{n}{k-2} + 1$;
- $\psi_k(F_n) = 1 + \left\lfloor \frac{n}{k} \right\rfloor$.

Special classes of graphs - ψ_3 for planar graphs is at least $\frac{2}{3}n$



Computational complexity

k-Path Security Problem (k-PSP)

INSTANCE: Graph G and a positive integer t.
QUESTION: Is there a k-path security set S for G of size at most t?

It is not very surprising that

Theorem

For any fixed integer $k \geq 2$ the k-Path Security Problem is NPC.

Approximation of k-PSN

Proposition

It is NP-hard to approximate k-PSP, for $k > 2$, within a factor of 1.3606, unless $P \neq NP$.

On the other hand:
For k-PCP we can construct an **approximation algorithm** in the following way: Find a path on k vertices, put its k vertices into the constructed set and remove them from the graph.
Since at least one of these k vertices belongs to an optimal solution, this provides a k-approximation algorithm for k-PSP.

Approximation of k-PSN (2)

By an application of Brach and Bounds approach we can obtain:

Proposition

The value of ψ_3 can be computed in time $O(1.749)^{|V(G)|}$.

The following provides an improvement of brute-force k-approximation algorithm:

Proposition

There is a randomized polynomial approximation algorithm for ψ_3 with expected approximation ratio 2.25.

Properly rooted subtree

Definition

A properly rooted subtree is a subtree rooted in a vertex v satisfying the following properties:

1. T_v does contain a path on k vertices;
2. $T_v \setminus v$ does not contain a path on k vertices.

The following result is important for the designed algorithm:

Lemma

If T_u is a tree containing a path on k vertices, then T_u contains at least one properly rooted subtree.

An algorithm for k-PSN for trees

Algorithm 2: $PSN\_Tree(T, k)$

Input: A tree T on n vertices and a positive integer k, $k \geq 2$;
Output: k-path security set S, $S \subseteq V(T)$;
Take an arbitrary vertex $u \in T$ and construct a rooted tree T_u from T;
$S := \emptyset$;
while T_u contains a properly rooted subtree T_v **do**
    $S := S \cup \{v\}$;
    $T_u := T_u \setminus T_v$;
return S;

Theorem

Let T be a tree of order n and k a positive integer. The algorithm $PSN\_Tree(T, k)$ returns an optimal solution for k-PSP.

Linear algorithm

Algorithm 3: *Linear_PSN_Tree*(T, k)

Take an arbitrary vertex $u \in T$ and construct a rooted tree T_u from T;
Create a sequence $v_1, v_2, \ldots, v_n = u$ of vertices of T in decreasing order with respect to their depth (distance from u in T_u);
$S := \emptyset$;
for $i := 1$ to n **do**
  $D :=$ children of v_i in T_u;
  $x := 1^{st} max_{d \in D} height(T_d)$;
  $y := 2^{nd} max_{d \in D} height(T_d)$;
  if $x + y + 1 \geq k$ **then**
    $S := S \cup v_i$;
    $height(T_{v_i}) := 0$;
  else
    $height(T_{v_i}) := x + 1$;
return S;

k-PSN for trees

Theorem

Let T be a tree and $k \geq 2$ be a positive integer. The algorithm PSN_Tree(T, k) returns a k-path security set of T of size at most $\frac{1}{k}n$.

Theorem

Let T be a tree and k be a positive integer. The algorithm PSN_Tree(T, k) returns an optimal solution in linear time and space.

Outerplanar graphs

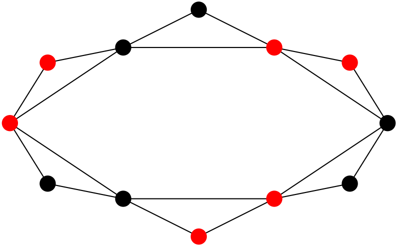
Theorem

Let G be an outerplanar graph of order n. Then $\psi_3(G) \leq \frac{n}{2}$.

Theorem

There is an algorithm to compute 3-path security number for any outerplanar graph G in polynomial time.

The bound $\frac{n}{2}$ for outerplanar graphs is the best possible.



Vertex decomposition according to $\Delta(G)$

Theorem (Lovász)

If s and t are non-negative integers, and if G is a graph with maximum degree $s + t + 1$, then the vertex set of G can be partitioned into two sets which induce subgraphs of maximum degree at most s and t, respectively.

The following is a straightforward generalisation of the previous result for k colours.

Theorem (Cowen and Jesurum)

For any graph G of maximum degree Δ, the vertex set of G can be partitioned into k sets which induce subgraphs of maximum degree at most $\frac{\Delta}{k}$. Moreover, this can be computed in running time $O(\Delta|E(G)|)$.

Graphs with bounded maximum degree

Lemma

Let G be a graph of maximum degree Δ. Then

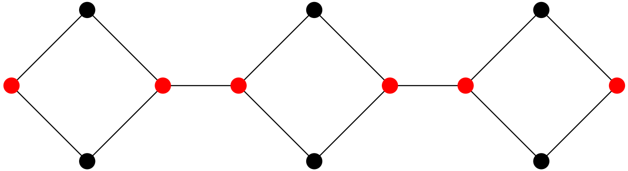
$$\psi_3(G) \leq \frac{\lceil \frac{\Delta-1}{2} \rceil}{\lceil \frac{\Delta+1}{2} \rceil} |V(G)|.$$

By another technique we can obtain the following:

Lemma

Let G be a graph. For each $k > 2$, $\psi_k(G) \leq \frac{2|E(G)|}{k+1}$.

The best possible bound for ψ_3 for graphs with maximum degree at most 3 is $\frac{n}{2}$



$$\psi_3(G) = 6 = \frac{|V(G)|}{2}$$

Sparse graphs

Algorithm 4: $SPARSE_3(G)$

Input: A graph G on n vertices;
Output: 3-path security set H, $H \subseteq V(G)$;
$H := \emptyset$;
while G contains any vertex v of degree at least 4 **do**
    $H := H \cup \{v\}$;
    Remove from G the vertex v and all edges incident with v;
$H := H \cup \mathrm{Solve\_Cubic}(G)$;
return H;

Theorem

Let G be a graph having n vertices and m edges. Then the $SPARSE_3(G)$ algorithm returns a solution H of size at most $\frac{2n+m}{6}$.

Ψ_3 for sparse graphs

As a corollary of the previous result we have:

Corollary

Let G be a graph on n vertices and m edges. Then

$$\psi_3(G) \leq \frac{2n+m}{6}.$$

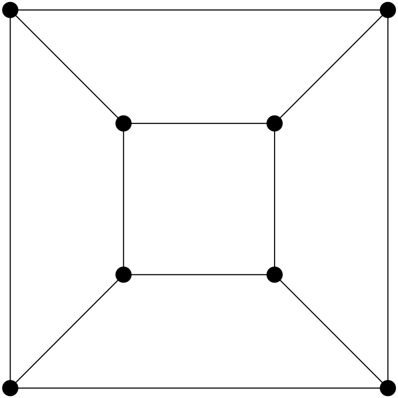
The result is sharp because:

Theorem

Let a, b are integers such that $b \leq a \leq 2b$. Then there is a graph on n vertices and m edges such that $\frac{m}{n} = \frac{a}{b}$ and

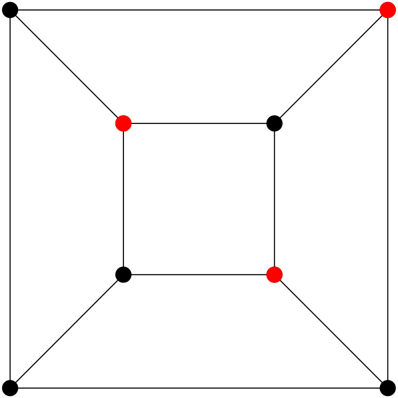
$$\psi_3(G) \geq \frac{2n+m}{6}.$$

Hypercubes - example



$\psi_4(Q_3) = ?$

Hypercubes - example



$\psi_4(Q_3) = 3$

Hypercubes of specific order

Theorem

Let G be a graph. Then

- $\psi_3(Q_2) = 2^{n-1}$.
- $\psi_4(Q_2) = 1$.
- $\psi_4(Q_3) = 3$.
- $\psi_4(Q_4) = 6$.
- $\psi_4(Q_5) = 14$.
- $\psi_4(Q_n) =$? *for* $n \geq 6$.

Regular graphs

Theorem (Erdős and Gallai)

If G is a graph with n vertices, and it does not contain a path of order k then it cannot have more than $\frac{n(k-2)}{2}$ edges. Moreover, the bound is achieved when the graph consists of disjoint complete graphs on $k-1$ vertices.

Theorem

Let $k \geq 2$ and $d \geq 1$ be positive integers. Then for any graph G with $\delta(G) \geq d$ the following holds:

$$\psi_k(G) \geq \frac{d-k+2}{2d-k+2}|V(G)|.$$

The special examples are ***d*-regular graphs.**

Direct consequences

Corollary

If G is a cubic graph then $\psi_k(G) \geq \frac{5-k}{8-k}$.

Corollary

Let $k \geq 2$ be a fixed positive integer. Then for any bipartite d-regular graph G

$$\lim_{n\to\infty} \psi_k(G) = \frac{1}{2}|V(G)|.$$

In particular, if Q_n **is a hypercube** then $\lim_{n\to\infty} \psi_k(Q_n) = 2^{n-1}$, and for **complete bipartite graphs** we have $\lim_{n\to\infty} \psi_k(K_{n,n}) = n$.

Another bounds for ψ_3

For a general k we can prove:

Theorem

For any graph G of order n the following holds:

$$\Psi_k(G) \leq n - \frac{k-1}{k} \sum_{v \in G} \frac{2}{1 + d_G(v)}$$

For a particular case we have an alternative result:

Theorem (Göring, Harant, Rautenbach, Schiermeyer, 2009)

Let G be a graph of order n. Then

$$\psi_3(G) \leq n - \sum_{u \in V(G)} \frac{1}{1 + d(u)} - \sum_{uv \in E(G)} \frac{2}{|N[u] \cup N[v]|(|N[u] \cup N[v]| - 1)}.$$

A general characterisation

For $k = 2$ Vizing observed already that

$$\alpha(G \square H) \leq \min\{\alpha(G)\,|V(H)|\,, \alpha(H)\,|V(G)|\}.$$

We are able to prove a generalisation of the following form:

Theorem

Let G, H be arbitrary graphs. Then

$$\psi_k(G \square H) = \max\{\psi_k(G) \cdot |V(H)|, |V(G)| \cdot \psi_k(H)\}.$$

if and only if $H \to \Psi_k(G)$ or $G \to \Psi_k(H)$.

Thank you for your attention

Gabriel Semanišin
gabriel.semanisin@upjs.sk