Separating Codes and Traffic Monitoring

Thomas Bellitto

AAIM 2016
July 18

LaBRI
CPU
Le monde numérique au service
de la certification et de la
sécurisation des systèmes
université de BORDEAUX

1. Preliminaries
   - Language theory
   - Separating codes
   - Separators and solving methods
2. Traffic monitoring
   - The problem
   - Separation on a language
3. Resolution
   - Finite case
   - An infinite case : total identification
   - Total separation on restricted-walk graphs (RWG)

Alphabet, word, language

Basic definitions

- Alphabet : non-empty finite set.
- Word : finite sequence of elements of an alphabet (letters).
- Language : set of words on a given alphabet.

Regular expression

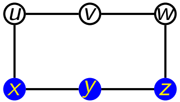
Rational language

- $\varnothing$ is rational.
- For all word a, $\{a\}$ is rational.
- For all rational language L_1 and L_2,
  $L_1 + L_2 = \{u \in A^* \mid u \in L_1 \vee u \in L_2\}$ is rational.
- For all rational language L_1 and L_2,
  $L_1 L_2 = \{w \in A^* \mid \exists u \in L_1, \exists v \in L_2, w = uv\}$ is rational.
- For all rational language L, $L^* = \sum_{k \in \mathbb{N}} L^k$ is rational.

On a graph

Separating code on a graph : set of vertices $\mathcal{C} \subset V$ such that each vertex is characterised by its neighbours (including him) in the code.

$$\forall v, v' \in V, \quad \underbrace{N[v] \cap \mathcal{C}}_{\text{signature of the vertex } v} = N[v'] \cap \mathcal{C} \Rightarrow v = v'$$



$$N[u] \cap \mathcal{C} = \{x\} \qquad N[x] \cap \mathcal{C} = \{x, y\}$$
$$N[v] \cap \mathcal{C} = \{\} \qquad N[y] \cap \mathcal{C} = \{x, y, z\}$$
$$N[w] \cap \mathcal{C} = \{z\} \qquad N[z] \cap \mathcal{C} = \{y, z\}$$

Problem : find a separating code as small as possible.

Test cover

Test cover problem

Set of individuals $\mathcal{I}$, set of attributes $\mathcal{A}$.
Looking for the smallest set $\mathcal{C} \subset \mathcal{A}$ such that each individual of $\mathcal{I}$ is characterised by the attributes of $\mathcal{C}$ it possesses.

Generalisation of the previous problem.

Very wide range of applications : pattern detection, routing or fault detection in networks, bio-informatics (molecular analysis), medicine (bacteria identification)...

Separating sets

Separating sets

The separating set $\text{Sep}(i, i')$ of two individuals i et i' is the set of attributes that distinguish them (symmetrical difference of their attributes).

A code $\mathcal{C}$ separate i and i' iff $\exists x \in \text{Sep}(i, i'), x \in \mathcal{C}$.

$$\begin{cases} \forall i \neq i' \in \mathcal{I}, \displaystyle\sum_{a \in \text{Sep}(i,i')} x_a \geqslant 1 \\ \text{minimise} \displaystyle\sum_{a \in \mathcal{A}} x_a \quad (\times c_a) \end{cases}$$

1. Preliminaries
2. Traffic monitoring
   - The problem
   - Separation on a language
3. Resolution

Presentation of the problem

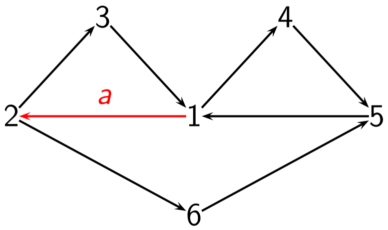
Network modelled by an directed graph. We can put censors on the arcs.
Signature of a walk : ordered sequence of the activated censors.

Traffic monitoring

An object walks in the graph and pick an route in a given set of possible walks.
Problem : find the smallest set of arcs to monitor all the possible walk have different signatures.

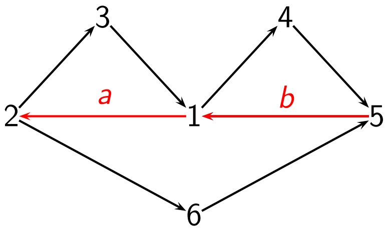
Main difficulties

- The set of activated sensors is not sufficient to distinguish two routes.
  Ex : $(1, 2, 3, 1)$ et $(1, 2, 3, 1, 2, 3, 1)$.



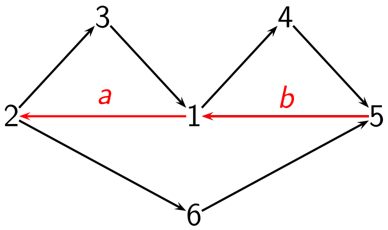
Main difficulties

- The set of activated sensors is not sufficient to distinguish two routes.
  Ex : $(1, 2, 3, 1)$ et $(1, 2, 3, 1, 2, 3, 1)$.
- Neither is the number of times each got activated.
  Ex : $(1, 2, 3, 1, 4, 5, 1)$ and $(1, 4, 5, 1, 2, 3, 1)$.



Main difficulties

- The set of activated sensors is not sufficient to distinguish two routes.
  Ex : $(1, 2, 3, 1)$ et $(1, 2, 3, 1, 2, 3, 1)$.
- Neither is the number of times each got activated.
  Ex : $(1, 2, 3, 1, 4, 5, 1)$ and $(1, 4, 5, 1, 2, 3, 1)$.
- The set of possible walks can be infinite.



Separation on a language

Projection of a word

Projection of a word $u \in A^*$ on a subalphabet $A' \subset A$: longer subword of $u \subset A'^*$.

Ex : $p_{\{a,b\}}(abacacb) = abaab$.

Separation on a language

We are looking for the smallest subalphabet $A' \subset A$ such that the projection of the given language L on A' is injective.

Ex : $L = \{aabcc, acabc, baacb, cbaac\}$.

Separation on a language

Projection of a word

Projection of a word $u \in A^*$ on a subalphabet $A' \subset A$: longer subword of $u \subset A'^*$.

Ex : $p_{\{a,b\}}(abacacb) = abaab$.

Separation on a language

We are looking for the smallest subalphabet $A' \subset A$ such that the projection of the given language L on A' is injective.

Ex : $L = \{aabcc, acabc, baacb, cbaac\}$.

$A' = \{ac\}$ is the only optimal solution.

Relation with traffic monitoring

Walk $\leftrightarrow$ word on $\underbrace{\text{the set of arcs of the graph}}_{\text{alphabet}}$.

Signature of a walk $\leftrightarrow$
projection of the word on $\underbrace{\text{the set of monitored arcs}}_{\text{subalphabet}}$.

1. Preliminaries
2. Traffic monitoring
3. Resolution
   - Finite case
   - An infinite case : total identification
   - Total separation on restricted-walk graphs (RWG)

Separating set

Separating set of two words

A separator of two words u and $v \in A^*$ is a minimal set of letter that separates them. Hence, $\mathrm{Sep}(u,v) \subset \mathcal{P}(A)$ is defined such that a subalphabet $\mathcal{C}$ separate u and v iff
$\exists x \in \mathrm{Sep}(u,v), x \subset \mathcal{C}$.

Theorem (B. 2016)

The separating set of two words contain only set of letters of cardinal at most 2.

Reduction to integer linear programming : we want to contain a separator of each pair of words.

Presentation of the problem

- Directed graph $G = (V, A)$
- Non-empty set $V_I \subset V$ of potential starting points
- Non-empty set $V_F \subset V$ of potential destination

Problem : separate all the walks leading from a vertex of V_I to a vertex of V_F.

Reachable language

A language $L \subset A^*$ is said reachable iff there exists a graph $G = (V, A)$, $V_I \subset V$ et $V_F \subset V$ such that L is the set of all walks leading from a vertex of V_I to a vertex of V_F.

Reduction theorem

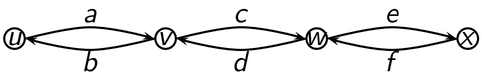
Restriction of a rational language

- $\overline{\varnothing} = \varnothing$.
- $\forall a \in A^*, \overline{\{a\}} = \{a\}$.
- For all rational languages L_1 and L_2, $\overline{L_1 + L_2} = \overline{L_1} + \overline{L_2}$.
- For all rational languages L_1 and L_2, $\overline{L_1 L_2} = \overline{L_1}\ \overline{L_2}$.
- For all rational languages L, $\overline{L^*} = \varepsilon + \overline{L} + \overline{L}^2$.

Reduction theorem (B. 2016)

For all language L reachable on an alphabet A, $\mathcal{C} \subset A$ separate L iff it separates $\overline{L}$.

Limits of the previous model



Set of possible walks from u to x :

$$(a(c(ef)^*d)^*b)^*(ac(ef)^*d)^*c(ef)^*e$$

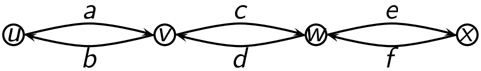
- Unrealistic behaviour,
- increases the computation time, decreases the size of the instance we can solve,
- lower the quality of the solutions.

Presentation of the problem

In addition to the graph, we are provided a set $\mathcal{F}$ of pair of arcs that denote forbidden turn.

New problem : we want to separate all the permitted walks leading from a vertex of V_I to a vertex of V_F.

Ex : forbidding half-turns : $\mathcal{F} = \{(a, b), (c, d), (e, f)\}$.



Set of permitted walks from u to x :

ace

Results

RWG-reachable languages

A language $L \subset A^*$ is said RWG-reachable iff there exists a RWG graph $G = (V, A, F)$, $V_I \subset V$ and $V_F \subset V$ such that L is the set of permitted walks leading from a vertex V_I to a vertex of V_F.

RWG-reachable languages are rational too. We can define their reduction !

Reduction theorem (B. 2016)

For all RWG-reachable language L on an alphabet A, $\mathcal{C} \subset A$ separate L iff it separates $\overline{L}$.

Conclusion

Contribution

- Reformulation of the problem of traffic monitoring as a separation problem, introduction of a new stronger model based on languages.
- Development of new tools to solve this problem on several kind of instances of practical interest, including infinite instances.

Perspectives

Optimisation of the algorithm (divide and conquer, deeper study of the ILP...).

Thank you !