Yags, un programa para teoría de gráficas

Rafael Villarroel Flores, UAEH

1 de octubre de 2015

Contenido

Cómputo en investigación

- En la investigación en combinatoria, y en particular en teoría de gráficas, la computadora nos ayuda a verificar hipótesis y plantear conjeturas.

Cómputo en investigación

- En la investigación en combinatoria, y en particular en teoría de gráficas, la computadora nos ayuda a verificar hipótesis y plantear conjeturas.
- En la plática se pretende presentar el programa yags (Yet Another Graph System), el cual es un paquete basado en el programa GAP (Groups, Algorithms, Programming).

El programa GAP

GAP 4

Open source system
for discrete computational algebra

El programa GAP

- GAP se inició en RWTH-Aachen, bajo la dirección de Joachim Neubüser en 1985.

El programa GAP

Open source system
for discrete computational algebra

- GAP se inició en RWTH-Aachen, bajo la dirección de Joachim Neubüser en 1985.
- La página de GAP es `http://gap-system.org/`.

Componentes de GAP

GAP se compone de:

- Un núcleo, escrito en C, que proporciona un intérprete para el lenguaje GAP, y funciones y estructuras básicas.

Componentes de GAP

GAP se compone de:

- Un núcleo, escrito en C, que proporciona un intérprete para el lenguaje GAP, y funciones y estructuras básicas.
- Una librería de funciones, escritas en el lenguaje GAP.

Componentes de GAP

GAP se compone de:

- Un núcleo, escrito en C, que proporciona un intérprete para el lenguaje GAP, y funciones y estructuras básicas.
- Una librería de funciones, escritas en el lenguaje GAP.
- Una librería de datos algebraicos, como la librería de grupos pequeños.

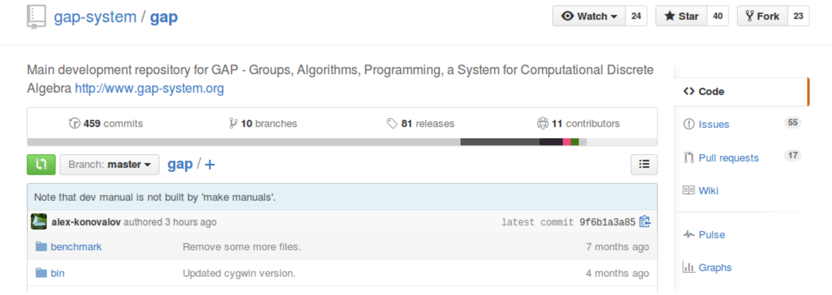
Componentes de GAP

GAP se compone de:

- Un núcleo, escrito en C, que proporciona un intérprete para el lenguaje GAP, y funciones y estructuras básicas.
- Una librería de funciones, escritas en el lenguaje GAP.
- Una librería de datos algebraicos, como la librería de grupos pequeños.
- Manuales (tutorial y referencia).

Código de acceso libre

El desarrollo de GAP se lleva a cabo en
`https://github.com/gap-system/gap`



Ejemplo de sesión interactiva

Cálculos simples

```
gap> 2+2;
4
gap> 34^3;
39304
gap> 34^30;
8797666833317830254826668219153233583932440576
```

Cálculos con permutaciones

```
gap> (1,2,3)^2;
(1,3,2)
gap> (1,2)*(1,2,3);
(1,3)
gap> (1,2,3)*(1,2);
(2,3)
```

Ejemplo de código

```
# Encontrar el primer grupo no soluble

solvable := true;
i := 0;

while solvable do
    i := i+1;
    j := 0;
    while solvable and j < NrSmallGroups(i) do
        j := j+1;
        g := SmallGroup(i,j);
        Print("Checking SmallGroup(",i,",",j,")\n");
        solvable := IsSolvable(g);
    od;
od;

Print("First non solvable group is SmallGroup(",i,",",j,")\n");
```

Paquetes de GAP

- GAP es *extensible*, lo cual significa que es relativamente sencillo añadirle nuevas funciones y capacidades, por medio de paquetes.

Paquetes de `GAP`

- `GAP` es *extensible*, lo cual significa que es relativamente sencillo añadirle nuevas funciones y capacidades, por medio de paquetes.
- Muchos paquetes ya vienen con `GAP`.

Paquetes de GAP

- GAP es *extensible*, lo cual significa que es relativamente sencillo añadirle nuevas funciones y capacidades, por medio de paquetes.
- Muchos paquetes ya vienen con GAP.
- Es posible enviar paquetes a los autores de GAP, los cuales pasan por un proceso de arbitraje análogo al de los artículos de investigación.

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Alnuth Algebraic number theory and an interface to KANT

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Alnuth Algebraic number theory and an interface to KANT

ANUPQ ANU p-Quotient

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Alnuth Algebraic number theory and an interface to KANT

ANUPQ ANU p-Quotient

Automata A package on automata

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Alnuth Algebraic number theory and an interface to KANT

ANUPQ ANU p-Quotient

Automata A package on automata

AutPGrp Computing the Automorphism Group of a p-Group

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Alnuth Algebraic number theory and an interface to KANT

ANUPQ ANU p-Quotient

Automata A package on automata

AutPGrp Computing the Automorphism Group of a p-Group

Carat Interface to CARAT, a crystallographic groups package

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Alnuth Algebraic number theory and an interface to KANT

ANUPQ ANU p-Quotient

Automata A package on automata

AutPGrp Computing the Automorphism Group of a p-Group

Carat Interface to CARAT, a crystallographic groups package

Circle Adjoint groups of finite rings

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Alnuth Algebraic number theory and an interface to KANT

ANUPQ ANU p-Quotient

Automata A package on automata

AutPGrp Computing the Automorphism Group of a p-Group

Carat Interface to CARAT, a crystallographic groups package

Circle Adjoint groups of finite rings

cohomolo Cohomology groups of finite groups on finite modules

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Alnuth Algebraic number theory and an interface to KANT

ANUPQ ANU p-Quotient

Automata A package on automata

AutPGrp Computing the Automorphism Group of a p-Group

Carat Interface to CARAT, a crystallographic groups package

Circle Adjoint groups of finite rings

cohomolo Cohomology groups of finite groups on finite modules

Crime A GAP Package to Calculate Group Cohomology and Massey Products

Algunos paquetes que vienen con GAP

ACE Advanced Coset Enumerator

Alnuth Algebraic number theory and an interface to KANT

ANUPQ ANU p-Quotient

Automata A package on automata

AutPGrp Computing the Automorphism Group of a p-Group

Carat Interface to CARAT, a crystallographic groups package

Circle Adjoint groups of finite rings

cohomolo Cohomology groups of finite groups on finite modules

Crime A GAP Package to Calculate Group Cohomology and Massey Products

CRISP Computing with Radicals, Injectors, Schunck classes and Projectors

Paquetes combinatorios

DESIGN The Design Package for GAP

Paquetes combinatorios

DESIGN The Design Package for GAP

GRAPE GRaph Algorithms using PErmutation groups

Paquetes combinatorios

DESIGN The Design Package for GAP

GRAPE GRaph Algorithms using PErmutation groups

simpcomp A GAP toolbox for simplicial complexes

Otros paquetes no incluidos (todavía) en GAP

Simplicial Homology `http://www.eecis.udel.edu/~dumas/Homology/Homology`

Otros paquetes no incluidos (todavía) en GAP

Simplicial Homology `http://www.eecis.udel.edu/~dumas/Homology/Homology`

Digraphs `http://www-groups.mcs.st-andrews.ac.uk/~jamesm/digraphs.php`

Otros paquetes no incluidos (todavía) en GAP

Simplicial Homology `http://www.eecis.udel.edu/~dumas/Homology/Homology`

Digraphs `http://www-groups.mcs.st-andrews.ac.uk/~jamesm/digraphs.php`

FinIng Finite Incidence Geometry `http://cage.ugent.be/fining/`

Otros paquetes no incluidos (todavía) en GAP

Simplicial Homology `http://www.eecis.udel.edu/~dumas/Homology/Homology`

Digraphs `http://www-groups.mcs.st-andrews.ac.uk/~jamesm/digraphs.php`

FinIng Finite Incidence Geometry `http://cage.ugent.be/fining/`

YAGS Yet Another Graph System

Gráficas en GRAPE

```
gap> LoadPackage("grape");
true
gap> P:=Graph(SymmetricGroup(5),[[1,2]],OnSets,function(x,y)
    return Intersection(x,y)=[]; end);
rec( adjacencies := [ [ 3, 5, 8 ] ], group := Group([
    (1,2,3,5,7)(4,6,8,9,10), (2,4)(6,9)(7,10) ]),
  isGraph := true, names := [ [ 1, 2 ], [ 2, 3 ], [ 3, 4 ], [ 1,
      3 ], [ 4, 5 ], [ 2, 4 ], [ 1, 5 ],
     [ 3, 5 ], [ 1, 4 ], [ 2, 5 ] ], order := 10,
         representatives := [ 1 ],
  schreierVector := [ -1, 1, 1, 2, 1, 1, 1, 1, 2, 2 ] )
gap> Diameter(P);
2
gap> Girth(P);
5
```

Algunas características de GRAPE

- Si un grupo Γ actúa en la gráfica G, GRAPE utiliza tal información para optimizar cálculos en la gráfica.

Algunas características de GRAPE

- Si un grupo Γ actúa en la gráfica G, `GRAPE` utiliza tal información para optimizar cálculos en la gráfica.
- Por ejemplo, la gráfica completa con 6 vértices se da como `CompleteGraph(SymmetricGroup(6))`.

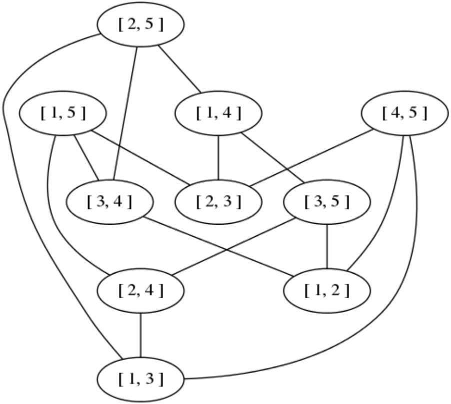
Algunas características de GRAPE

- Si un grupo Γ actúa en la gráfica G, `GRAPE` utiliza tal información para optimizar cálculos en la gráfica.
- Por ejemplo, la gráfica completa con 6 vértices se da como `CompleteGraph(SymmetricGroup(6))`.
- `GRAPE` no incluye una herramienta para dibujar gráficas.

Un dibujo de la gráfica usando Graphviz

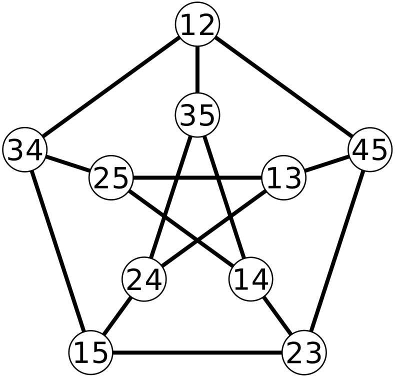
Graphviz es un programa que sirve para dibujar gráficas (`http://www.graphviz.org/`)

```
graph G {
"[ 3, 4 ]" -- "[ 1, 2 ]";
"[ 4, 5 ]" -- "[ 1, 2 ]";
"[ 4, 5 ]" -- "[ 2, 3 ]";
"[ 4, 5 ]" -- "[ 1, 3 ]";
"[ 2, 4 ]" -- "[ 1, 3 ]";
"[ 1, 5 ]" -- "[ 2, 3 ]";
"[ 1, 5 ]" -- "[ 3, 4 ]";
"[ 1, 5 ]" -- "[ 2, 4 ]";
"[ 3, 5 ]" -- "[ 1, 2 ]";
"[ 3, 5 ]" -- "[ 2, 4 ]";
"[ 1, 4 ]" -- "[ 2, 3 ]";
"[ 1, 4 ]" -- "[ 3, 5 ]";
"[ 2, 5 ]" -- "[ 3, 4 ]";
"[ 2, 5 ]" -- "[ 1, 3 ]";
"[ 2, 5 ]" -- "[ 1, 4 ]";
}
```



La gráfica de Petersen

Un mejor dibujo de la gráfica anterior:



Gráfica de Petersen

El programa yags

- Yags es un paquete para GAP creado por M. Pizaña y R. MacKinney en 2003 en la UAM-Iztapalapa.

El programa yags

- Yags es un paquete para GAP creado por M. Pizaña y R. MacKinney en 2003 en la UAM-Iztapalapa.
- De momento es incompatible con GRAPE.

El programa yags

- Yags es un paquete para GAP creado por M. Pizaña y R. MacKinney en 2003 en la UAM-Iztapalapa.
- De momento es incompatible con GRAPE.
- Yags define varias familias de gráficas y permite calcular parámetros sobre gráficas, sin referencia a un grupo actuando en la gráfica.

Primer sesión con yags

```
gap> g:=RandomGraph(20,1/5);
Graph( Category := SimpleGraphs, Order := 20, Size :=
42, Adjacencies := [ [ 4, 5, 9, 20 ], [ 3, 9, 11, 12, 15, 19 ],
  [ 2, 17 ], [ 1, 5, 16 ], [ 1, 4, 6, 11, 16 ], [ 5, 7, 16, 18
      ],
  [ 6, 8, 10, 11, 14, 15, 16 ], [ 7, 10, 14 ],
  [ 1, 2, 12, 14, 20 ], [ 7, 8, 13, 16 ], [ 2, 5, 7, 13, 16, 20
      ],
  [ 2, 9, 13, 18, 19 ], [ 10, 11, 12, 15 ], [ 7, 8, 9 ],
  [ 2, 7, 13, 16, 19 ], [ 4, 5, 6, 7, 10, 11, 15 ], [ 3, 19 ],
  [ 6, 12 ], [ 2, 12, 15, 17 ], [ 1, 9, 11 ] ] )
gap> Diameter(g);
4
```

Dibujos con yags

```
gap> g:=WheelGraph(7);
Graph( Category := SimpleGraphs, Order := 8, Size := 14,
    Adjacencies :=
[ [ 2, 3, 4, 5, 6, 7, 8 ], [ 1, 3, 8 ], [ 1, 2, 4 ], [ 1, 3, 5
    ], [ 1, 4, 6 ],
  [ 1, 5, 7 ], [ 1, 6, 8 ], [ 1, 2, 7 ] ] )
gap> Draw(g);
```

```
gap> g:=WheelGraph(15,4);;
gap> Draw(g);
```

Problema de Leo

- En una plática en la UAEH en mayo de este año, Leonardo Martínez planteó la siguiente pregunta:

Problema de Leo

- En una plática en la UAEH en mayo de este año, Leonardo Martínez planteó la siguiente pregunta:
- ¿Cuál es el máximo de aristas que una gráfica de 8 vértices puede tener, entre las gráficas con número de clan a lo más 3 y número de independencia a lo más 2?

Problema de Leo

- En una plática en la UAEH en mayo de este año, Leonardo Martínez planteó la siguiente pregunta:
- ¿Cuál es el máximo de aristas que una gráfica de 8 vértices puede tener, entre las gráficas con número de clan a lo más 3 y número de independencia a lo más 2?
- (El número de clan de una gráfica G es el mayor n tal que K_n es subgráfica de G. Se denota con $\omega(G)$).

Problema de Leo

- En una plática en la UAEH en mayo de este año, Leonardo Martínez planteó la siguiente pregunta:
- ¿Cuál es el máximo de aristas que una gráfica de 8 vértices puede tener, entre las gráficas con número de clan a lo más 3 y número de independencia a lo más 2?
- (El número de clan de una gráfica G es el mayor n tal que K_n es subgráfica de G. Se denota con $\omega(G)$).
- (El número de independencia de G es $\omega(\overline{G})$).

Problema de Leo

- En una plática en la UAEH en mayo de este año, Leonardo Martínez planteó la siguiente pregunta:
- ¿Cuál es el máximo de aristas que una gráfica de 8 vértices puede tener, entre las gráficas con número de clan a lo más 3 y número de independencia a lo más 2?
- (El número de clan de una gráfica G es el mayor n tal que K_n es subgráfica de G. Se denota con $\omega(G)$).
- (El número de independencia de G es $\omega(\overline{G})$).
- Como el *número de Ramsey* $R(4,3) = 9$, toda gráfica con 9 vértices tiene $\omega(G) \geq 4$ o $\omega(\overline{G}) \geq 3$.

Problema de Leo

- En una plática en la UAEH en mayo de este año, Leonardo Martínez planteó la siguiente pregunta:
- ¿Cuál es el máximo de aristas que una gráfica de 8 vértices puede tener, entre las gráficas con número de clan a lo más 3 y número de independencia a lo más 2?
- (El número de clan de una gráfica G es el mayor n tal que K_n es subgráfica de G. Se denota con $\omega(G)$).
- (El número de independencia de G es $\omega(\overline{G})$).
- Como el *número de Ramsey* $R(4,3) = 9$, toda gráfica con 9 vértices tiene $\omega(G) \geq 4$ o $\omega(\overline{G}) \geq 3$.
- Por lo tanto, es interesante considerar las gráficas de 8 vértices con $\omega(G) \leq 3$ y $\omega(\overline{G}) \leq 2$.

Problema de Leo

- En una plática en la UAEH en mayo de este año, Leonardo Martínez planteó la siguiente pregunta:
- ¿Cuál es el máximo de aristas que una gráfica de 8 vértices puede tener, entre las gráficas con número de clan a lo más 3 y número de independencia a lo más 2?
- (El número de clan de una gráfica G es el mayor n tal que K_n es subgráfica de G. Se denota con $\omega(G)$).
- (El número de independencia de G es $\omega(\overline{G})$).
- Como el *número de Ramsey* $R(4,3) = 9$, toda gráfica con 9 vértices tiene $\omega(G) \geq 4$ o $\omega(\overline{G}) \geq 3$.
- Por lo tanto, es interesante considerar las gráficas de 8 vértices con $\omega(G) \leq 3$ y $\omega(\overline{G}) \leq 2$.
- Por el *teorema de Turán*, tales gráficas tienen entre 12 y 21 aristas.

Solución

- Definimos una función para checar la condición deseada, y la guardamos en el archivo `leo.gap`.

```
CondicionLeo := function (g)
    return CliqueNumber(g)<=3 and
           CliqueNumber(ComplementGraph(g))<=2;
end;
```

Solución (continuación)

- En una sesión interactiva con yags, obtenemos:

```
gap> Read("leo.gap");
gap> g8:=ConnectedGraphsOfGivenOrder(8);;
gap> f:=Filtered(g8,CondicionLeo);;
gap> List(f,x->Size(x));
[ 16, 17, 18 ]
```

- Por lo que la respuesta a la pregunta de Leonardo es 18.

Operador de clanes

- En mi investigación me interesa el *operador de clanes*. Dada una gráfica G, su gráfica de clanes $K(G)$ es la gráfica de intersección de los clanes de G.

Operador de clanes

- En mi investigación me interesa el *operador de clanes*. Dada una gráfica G, su gráfica de clanes $K(G)$ es la gráfica de intersección de los clanes de G.
- (Un clan de G es una subgráfica completa maximal).

Operador de clanes

- En mi investigación me interesa el *operador de clanes*. Dada una gráfica G, su gráfica de clanes $K(G)$ es la gráfica de intersección de los clanes de G.
- (Un clan de G es una subgráfica completa maximal).
- Definimos $K^n(G)$ como $K(K^{n-1}(G))$ si $n \geq 2$, $K^1(G) = K(G)$.

Operador de clanes

- En mi investigación me interesa el *operador de clanes*. Dada una gráfica G, su gráfica de clanes $K(G)$ es la gráfica de intersección de los clanes de G.
- (Un clan de G es una subgráfica completa maximal).
- Definimos $K^n(G)$ como $K(K^{n-1}(G))$ si $n \geq 2$, $K^1(G) = K(G)$.
- Hay algunas gráficas para las que la sucesión de órdenes de las gráficas $\{|K^n(G)|\}$ tiende a infinito. Tales gráficas se llaman divergentes, las otras se llaman convergentes.

Operador de clanes

- En mi investigación me interesa el *operador de clanes*. Dada una gráfica G, su gráfica de clanes $K(G)$ es la gráfica de intersección de los clanes de G.
- (Un clan de G es una subgráfica completa maximal).
- Definimos $K^n(G)$ como $K(K^{n-1}(G))$ si $n \geq 2$, $K^1(G) = K(G)$.
- Hay algunas gráficas para las que la sucesión de órdenes de las gráficas $\{|K^n(G)|\}$ tiende a infinito. Tales gráficas se llaman divergentes, las otras se llaman convergentes.
- No se conoce un algoritmo general para determinar el comportamiento de una gráfica.

Operador de clanes

- En mi investigación me interesa el *operador de clanes*. Dada una gráfica G, su gráfica de clanes $K(G)$ es la gráfica de intersección de los clanes de G.
- (Un clan de G es una subgráfica completa maximal).
- Definimos $K^n(G)$ como $K(K^{n-1}(G))$ si $n \geq 2$, $K^1(G) = K(G)$.
- Hay algunas gráficas para las que la sucesión de órdenes de las gráficas $\{|K^n(G)|\}$ tiende a infinito. Tales gráficas se llaman divergentes, las otras se llaman convergentes.
- No se conoce un algoritmo general para determinar el comportamiento de una gráfica.
- Usaremos las listas de gráficas para encontrar las que sean divergentes más pequeñas.

Operador de clanes (continuación)

- Hay una condición (*propiedad de Helly*) que es fácilmente verificable y que implica convergencia. Para que una gráfica no tenga la propiedad de Helly necesita tener al menos 6 vértices.

Operador de clanes (continuación)

- Hay una condición (*propiedad de Helly*) que es fácilmente verificable y que implica convergencia. Para que una gráfica no tenga la propiedad de Helly necesita tener al menos 6 vértices.
- Si G tiene un *vértice dominado* v, entonces G y $G-v$ tienen el mismo comportamiento.

Operador de clanes (continuación)

- Hay una condición (*propiedad de Helly*) que es fácilmente verificable y que implica convergencia. Para que una gráfica no tenga la propiedad de Helly necesita tener al menos 6 vértices.
- Si G tiene un *vértice dominado* v, entonces G y $G - v$ tienen el mismo comportamiento.
- Por lo tanto, si G tiene 6 vértices y uno de ellos es dominado, entonces G es convergente.

Operador de clanes (continuación)

- Hay una condición (*propiedad de Helly*) que es fácilmente verificable y que implica convergencia. Para que una gráfica no tenga la propiedad de Helly necesita tener al menos 6 vértices.
- Si G tiene un *vértice dominado* v, entonces G y $G-v$ tienen el mismo comportamiento.
- Por lo tanto, si G tiene 6 vértices y uno de ellos es dominado, entonces G es convergente.
- Puede ser que G no sea Helly, pero para alguna n se tenga que $K^n(G)$ sea Helly. Por supuesto que en ese caso G es convergente.

Código de clanes

```
HasNoDominatedVertex := function (g)
    return IsEmpty(DominatedVertices(g));
end;

IsNotCliqueHelly := function (g)
    return not(IsCliqueHelly(g));
end;

IsNotEventuallyHelly := function (g)
    local kcurrent, isit;
    kcurrent := g;
    isit := not(IsCliqueHelly(kcurrent));
    while isit do
        kcurrent := CliqueGraph(kcurrent,100);
        if kcurrent = fail then
            return true;
        else
            kcurrent := CompletelyParedGraph(kcurrent);
            isit := not(IsCliqueHelly(kcurrent));
        fi;
    od;
    return isit;
end;
```

Sesión interactiva. Gráficas de 6 vértices

```
gap> graphs:=ConnectedGraphsOfGivenOrder(6);;
gap> Length(graphs);
112
gap> graphs:=Filtered(graphs,HasNoDominatedVertex);;
gap> Length(graphs);
9
gap> graphs:=Filtered(graphs,IsCliqueHelly);;
gap> Length(graphs);
8
gap> graphs:=Filtered(graphs,IsNotCliqueHelly);;
gap> Length(graphs);
1
gap> Draw(graphs[1]);
```

La única gráfica de 6 vértices que obtenemos es la gráfica del octaedro, y es de hecho divergente (Neumann-Lara, 1975).

Gráficas de 7 vértices

```
gap> graphs:=ConnectedGraphsOfGivenOrder(7);;
gap> Length(graphs);
853
gap> graphs:=Filtered(graphs,HasNoDominatedVertex);;
gap> Length(graphs);
46
gap> graphs:=Filtered(graphs,IsNotCliqueHelly);;
gap> Length(graphs);
6
gap> graphs:=Filtered(graphs,IsNotEventuallyHelly);;
gap> Length(graphs);
3
```

Las dos primeras gráficas tiene una *retracción* al octaedro, la tercera es la *suspensión* del ciclo C_5. Las tres son divergentes por teoremas de Neumann-Lara.

Página de yags

Yags se puede obtener de la página:

`https://github.com/yags/`