Presentation at the IMNS –2024

Jerez de la Frontera, 8 July 2024 [1]

Manuel Delgado

(https://cmup.fc.up.pt/cmup/mdelgado/)

Faculdade de Ciências da Universidade do Porto

Work partially supported by CMUP, a member of LASI, which is financed by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., under the projects with reference UIDB/00144/2020 and UIDP/00144/2020.

Special thanks to the Proyecto de Excelencia de la Junta de Andalucía (ProyExcel 00868).

Examples, counter-examples and the absence of examples.

Many people influenced my work in numerical semigroups, a fascinating area I discovered several years after completing my PhD degree.

Regarding the work I present here, I would like to mention Pedro García-Sánchez, Shalom Eliahou, and Neeraj Kumar. A big thank you to them and to all the regular participants in the IMNS meetings.

The **aim of my talk** is to call the audience's attention to the usefulness of doing experiments.

Being more precise...

In short:

I want to advertise the numericalsgps GAP package, a freely available tool for computing with numerical, affine and good semigroups.

In **not so short:**

- Well-conducted experiments can produce examples pointing out good research problems.
- Experiments may also decisively contribute to discarding time-wasting or uninteresting research problems by producing counter-examples or for other reasons.
- The absence of examples in well-conducted experiments can also point out good research problems.

Being more specific:

[1] This document was prepared from the jupyter notebook used for the talk delivered.

- I plan to talk about some reasonable search spaces where one may look for examples/counter-examples (of semigroups with a particular property) or check that there are no examples/counter-examples in there.
- At the end of my talk, I will refer to the inexistence of a counter-example to a famous conjecture in a vast search space. (This is joint work with S. Eliahou and J. Fromentin.)

Using the *numericalsgps* package

(some examples)

The package must be loaded **in a running *GAP* session**

```
[1]: LoadPackage("numericalsgps");
```

```
[1]: true
```

For visualization one can use the package **intpic**

```
[2]: LoadPackage("intpic");
```

```
[2]: true
```

Examples of search spaces

- Set of numerical semigroups with a given genus

```
[3]: g3 := NumericalSemigroupsWithGenus(3);
```

```
[3]: [ <Numerical semigroup with 4 generators>,  <Numerical semigroup with 3
generators>,  <Numerical semigroup with 3 generators>,  <Numerical semigroup
with 2 generators> ]
```

Warning. Do not try `NumericalSemigroupsWithGenus(100);`

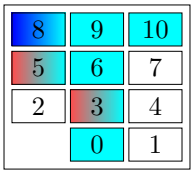
Theoretically, one would obtain a list of length over 10^{20}, an estimated lower bound for the number of numerical semigroups of genus 100, but the current computational power does not permit to reach this number.

```
[4]: List(g3,s->Gaps(s));
```

```
[4]: [ [ 1, 2, 3 ], [ 1, 2, 4 ], [ 1, 2, 5 ], [ 1, 3, 5 ] ]
```

```
[5]: DrawNumericalSemigroup(g3[2]);;
```

(The following picture popped up)



- Set of numerical semigroups with a given Frobenius number

[6]:
```
N5 := NumericalSemigroupsWithFrobeniusNumber(5);
```

[6]:
```
[ <Numerical semigroup with 2 generators>,   <Numerical semigroup with 2
generators>,   <Numerical semigroup with 3 generators>, <Numerical semigroup>,
<Numerical semigroup> ]
```

[7]:
```
List(N5,s->SmallElements(s));
```

[7]:
```
[ [ 0, 2, 4, 6 ], [ 0, 3, 4, 6 ], [ 0, 3, 6 ], [ 0, 4, 6 ], [ 0, 6 ] ]
```

- Set of numerical semigroups with a given maximum primitive
  (See the following talk by Neeraj Kumar.)

[8]:
```
A5 := NumericalSemigroupsWithMaxPrimitive(5);
```

[8]:
```
[ <Numerical semigroup with 2 generators>,   <Numerical semigroup with 2
generators>,   <Numerical semigroup with 3 generators>,   <Numerical semigroup
with 2 generators> ]
```

[9]:
```
List(A5,s->MinimalGenerators(s));
```

[9]:
```
[ [ 2, 5 ], [ 3, 5 ], [ 3 .. 5 ], [ 4, 5 ] ]
```

pictorial views using *intpic*

(Pictures popped up...)

[10]:
```
DrawNumericalSemigroup(A5[2]);;
```

[11]:
```
for n in [1..Length(A5)] do
    DrawNumericalSemigroup(A5[n]);;
od;
```

The introductory part came to an end.

Some recent developments in the *numericalsgps* package

In what follows, my presentation will include a few technical details, but I hope to find a good balance. In case of success, I will keep the audience awake during this "siesta" time.

Computing a search space from scratch is time-consuming, and better strategies for finding examples or performing tests may exist.

I am currently working on including search spaces (databases) previously computed in the numericalsgps package.

One needs to make some compromises between efficiency, space occupied, etc.

Space:

One needs to remember that the package is distributed by GAP (thus, it is widely distributed), and therefore, it needs to be **reasonable in size**.

Once the framework is available, larger databases may be made available for those interested. Efficiency: the database should consist of numerical semigroups...

Databases (enumerating numerical semigroups)

- What is storing a numerical semigroup?
- What should we store having relatively fast computations in mind?
- One should use a compromise between the amount of data stored and the **space occupied after compression**.

A numerical semigroup in GAP is just a **"bag"** containing some information defining the semigroup (e.g. a set of generators, the set of gaps, an Apery set, etc.)

As computations proceed, other information is added to the "bag" (the computational system increases the knowledge of the semigroup).

The speed of a particular computation may depend on the knowledge the system has of the semigroup.

Example. Compute, using numericalsgps, the **conductor** of a numerical semigroup S. - if the "bag" S contains the conductor, the computation is **immediate**; - if the "bag" S contains the (possibly ordered) list G of gaps, the computation is **fast** - it reduces to calculate max(G) + 1; - if the "bag" S contains just a (possibly minimal) set of generators, the computation may be much **slow**er.

Wilf number

For a numerical semigroup S, denote - $c(S)$ the conductor; - $L(S) = \{n \in \mathbb{Z}_{\geq 0} : n < c(S)\}$ (left elements) - $P(S)$ the minimal set of generators (set of primitives)

The **Wilf number** of S is:

$$W(S) = |L(S)| * |P(S)| - c(S).$$

[12]:
```
# my function to compute the Wilf number of a numerical semigroup
MyFunction := function(S)
    return(
        (Length(SmallElements(S))-1) * Length(MinimalGenerators(S)) -␣
 ↪Conductor(S)
    );
end;
```

[12]:
```
function( S ) ... end
```

`MyFunction`is implemeted in the *numericalsgps* package under the name `WilfNumber`

[14]:
```
F10 := NumericalSemigroupsWithFrobeniusNumber(10);;
ForAll(F10,s -> MyFunction(s) = WilfNumber(s));
```

[14]: true

A question posed by Wilf: Does every numerical semigroup have a positive Wilf number?

I say that a numerical semigroup is **Wilf** if its Wilf number is nonnegative.

Wilf's conjecture. Every numerical semigroup is Wilf.

```
[16]: A10 := NumericalSemigroupsWithMaxPrimitive(10);;
      Filtered(A10,s->WilfNumber(s)<0); # numerical semigroups with maximum primitive␣
       ↪10 are Wilf
```

[16]: [ ]

Question. Is there any numerical semigroups with Frobenius number 10 and Wilf Number 0?

```
[17]: s := First(F10, s -> WilfNumber(s)=0);
```

[17]: <Numerical semigroup with 11 generators>

```
[18]: SmallElements(s);
```

[18]: [0, 11]

Exercise. Determine all numerical semigroups with Frobenius number 10 and Wilf Number 0.

```
[19]: Filtered(F10, s -> WilfNumber(s)=0);
```

[19]: [<Numerical semigroup with 11 generators>]

Some databases

The current development version of the *numericalsgps* package contains some pre-computed data, namely the small elements and the minimal generators of the numerical semigroups with genus up to 22, of those with Frobenius number up to 32 and those with maximum primitive up to 32.

These sets of semigroups ("bags" containing the small elements and the minimal generators) can be obtained using the NC versions of the functions used above. **The names are temporarily used un the current development version of *numericalsgps*.**

```
[20]: NumericalSemigroupsWithGenus(5)[2]=NumericalSemigroupsWithGenusNC(5)[2];
```

[20]: true

```
[21]: KnownAttributesOfObject(NumericalSemigroupsWithMaxPrimitive(5)[2]);
```

[21]: ["Generators", "ModularConditionNS", "FrobeniusNumber", "MinimalGenerators"]

```
[22]: KnownAttributesOfObject(NumericalSemigroupsWithMaxPrimitiveNC(5)[2]);
```

```
[22]: [ "Generators", "ModularConditionNS", "SmallElements", "FrobeniusNumber",
      "MinimalGenerators" ]
```

```
[24]: F30 := NumericalSemigroupsWithFrobeniusNumber(30);;Length(F30); #slow
```

```
[24]: 31822
```

```
[26]: FNC30 := NumericalSemigroupsWithFrobeniusNumberNC(30);;Length(FNC30);
```

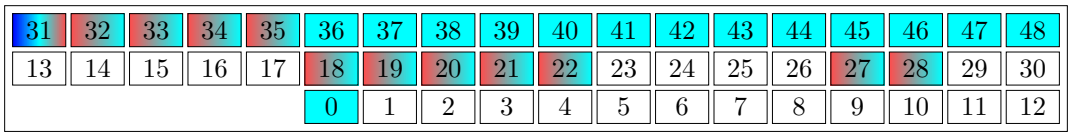
```
[26]: 31822
```

```
[27]: s := RandomList(FNC30); # a pseudo random numerical semigroup with Frobenius␣
      ↪number 30
```

```
[27]: <Numerical semigroup with 12 generators>
```

```
[28]: DrawNumericalSemigroup(s);;
```

(A picture popped up...)




```
[30]: gnNC:=NumericalSemigroupsWithFrobeniusNumberNC(32);;
      Length(gnNC);
```

```
[30]: 68681
```

```
[31]: # Numerical semigroups with Frobenius number up to 32 verify Wilf's conjecture
      Filtered(gnNC,s->WilfNumber(s)<0);
```

```
[31]: [  ]
```

```
[32]: w0 := Filtered(gnNC,s->WilfNumber(s)=0);
```

```
[32]: [ <Numerical semigroup with 3 generators>,   <Numerical semigroup with 11
      generators>,   <Numerical semigroup with 33 generators> ]
```

```
[33]: List(w0,s->MinimalGenerators(s));
```

```
[33]: [ [ 3, 34, 35 ], [ 11, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43 ], [ 33 .. 65 ]  ]
```

A few words on the theory behind the experimental approach used in joint work with Eliahou and Fromentin to show the absence of counter-examples to Wilf's conjecture among the more than 42×10^{20} (estimated value) numerical semigroups of genus up to 100.

For more, see the preprint on ArXiv and attend Eliahou's talk.

See also a post I wrote for Red de Matemática Discreta y Algorítmica.

Muchas Gracias