On the importance (in Operations Research) of the techniques that can make programming (with `cplex` or `gurobi`) easier

Daniel Porumbel[1]

CEDRIC-CNAM, 292 rue Saint Martin, Paris France

daniel.porumbel@cnam.fr

1 Technical aspects and source code

I present a software library of about 1000 lines of code C++ that may help one write cutting planes algorithms more easily. When I first wrote it, I did not think I will ever present such code in an official communication. But one day I realized that the research work (in Operations Research) can more easily stumble or fail because of implementation issues rather than because of other aspects more often discussed (like modeling the problem).

The goal if this English translation is to discuss the motivation for such work rather than the work in itself. Still you may the code in the French version of this paper.

2 From technical aspects towards philosophical aspects of work

Many may ask : why go beyond the technical ideas and waste time on philosophical aspects that can not help us on our concrete work and solve something ? Answer : too many people never think that their work may have any higher meaning beyond the material utility (and the visible achievements). How many great researchers do not have enough stamina to remain strong in the long run ? How many great people lose faith when they no longer have any promotion in sight (nor other stimulating reward) ? A better philosophical understanding may help one defeat such problems by casting light on the more idealistic meanings behind all arts and sciences. Except the very few (scientists) who achieve fame, fortune or any other kind of (pseudo) glory, most people work without realizing that they are also the representatives of a sublime creative force without which the world would become dry and empty.

If we completely ignore such things, if we are given a wrong philosophical framework and wrong motivations, then our work is eventually doomed. Such a huge problem is always rooted in the "prevailing philosophy" of the age and it was very well described by a 19th century philosopher [1] that we cite below, knowing that certain things have improved since his time :

> *"Woe to the time when in philosophy impudence and nonsense supplant insight and understanding for the fruits assume the taste of the soil in which they have grown. What is loudly, publicly, and universally praised, is read and is thus the mental pabulum of the generation that is arriving at maturity ; but this has the most decided influence on its lifeblood and subsequently on its creations. Thus the prevailing philosophy of an age determines its spirit, and so if there now prevails a philosophy of absolute nonsense ; if absurdities invented and advanced under bedlamite twaddle pass for great thoughts, then the result of such sowing is the pretty race of men such as we now have before us. They are without intellect, love of truth, honesty, taste, and are devoid of any noble impulse or of an urge for anything lying beyond material, including political, interests. From this we can explain how the age when Kant philosophized, Goethe wrote, and Mozart composed, could be followed by the present one of political poets and even more political philosophers, of hungry men of letters who earn a living in literature by falsehood and imposture and of ink-slingers of all kinds who wantonly ruin the language."*

The importance of the “prevailing philosophy” of the current age in Computer Science can now be analyzed through the lens of the above citation. Let us go back to the initial question : what is the right place of the “implementation details” in Operations Research and Computer Science ? My intuition tells me that many let us understand that we should not devote too much time on such aspects, because they were not really worthy of our noble minds and it is useless to publicly discuss them too much. Such a conception of life can generate a fracture in the scientific world, *i.e.*, a gap between :

— theory senior leading experts who have never seen an algorithm in execution and who limit themselves to trying to understand everything in very broad terms. Such guys – especially the very old ones – may lack contact with any down-to-earth thing, a bit like the senior bureaucrats or senior high-ranking government officials.

— the programmers who run (optimization) algorithms every day but do not have the good conceptual tools to clearly understand the broad design principles that support their work. Such guys – especially the very young ones – may not (yet) have the best analytical framework to make all distinctions necessary to debunk the countless misconceptions and traps of the system. This limits their capacity to get a very insightful vision over their field of work.

Post-scriptum note : long after first writing about the gap between the above groups, I found an ally in Christopher Strachey, the founder of the Programming Research Group at Oxford. [1]

I have already addressed these issues during the previous edition of this conference in an article "About an implementation "detail" : a few ideas to speed up the execution and the C++ programming (in Operations Research)" [2]. This article is only two pages long (the Roadef limit), but I completed it to make a full argument of a dozen pages uploaded at `http://cedric.cnam.fr/~porumbed/soft/`. Those who live a peaceful life in the luxury of some theory will surely not rush to read me because that’s how the story started :

> *“The execution speed of any practical algorithm clearly depends on a constant complexity factor δ closely related to the quality of the implementation. After ten years of talks at this annual (Roadef) congress, I finally realized that all of my talks invariably end with a line like “And finally, I will spare you the implementation details because I prefer to focus on the essentials and on the high level concepts”. Despite its merits, this sentence is too simplistic. It is likely to divert a fast reader from the truth.*
>
> *Like a backdoor in software, an unspoken consequence is that the (constant complexity factor associated with) the quality of the implementation has no impact on the total computational cost and can be ignored. This constant complexity factor δ is also invisible when calculating theoretical complexities such as $O(n^2)$, $O(n^2 logn)$, etc. But imagine a salesman saying that the cost to pay is 1000\$ multiplied by some constant factor δ=3 or δ=6 that has no importance in absolute terms. With all due respect to all rich people for whom 1000\$ or 3000\$ is more or less the same, I would not let myself be convinced so easily. I am even very grateful to my destiny for having spared me the emptiness of an existence in such disproportionate luxury.”*

Références

[1] Arthur Schopenhauer. On Philosophy at the Universities. pp 139-197, Parerga & Paralipomena, downloaded from `archive.org`, a direct link can be found at `cedric.cnam.fr/~porumbed/soft/schopenhauer.pdf`

[2] Daniel Porumbel. About an implementation "detail" : a few ideas to speed up the execution and the C++ programming (in Operations Research). Roadef 2019, Le Havre

1. He wrote : “It has long been my personal view that the separation of practical and theoretical work is artificial and injurious. Much of the practical work done in computing, both in software and in hardware design, is unsound and clumsy because the people who do it have not any clear understanding of the fundamental design principles of their work. Most of the abstract mathematical and theoretical work is sterile because it has no point of contact with real computing. One of the central aims of the Programming Research Group as a teaching and research group has been to set up an atmosphere in which this separation cannot happen.”