Big Integers for GIS: Testing the Viability of Arbitrary Precision Arithmetic for GIS Geometry

Rizwan Bulbul and Andrew U. Frank
Department of Geoinformation and Cartography,
Technical University Vienna,
Gußhausstraße 27-29/E127-1
A-1040 Vienna, Austria
{bulbul, frank}@geoinfo.tuwien.ac.at

ABSTRACT

A systematic exploration of solutions to represent geometric objects shows that the combination of big integers for metric and convex polytopes for topological information is promising. The viability of this novel approach depends on the effective performance of geometric operations typical for GIS using big integers. We report an experiment to determine whether solutions using big integers are realistic for GIS geometry. Metric computations with big numbers are conceptually simpler and need no testing for approximation problems. Performance penalties in some cases are severe (but much less than what we expected), but we found that they should not cause noticeable effects for users.

GEOMETRIC COMPUTATIONS WITH FINITE PRECISION ARITHMETIC

Current commercial approaches in GIS use floating point numbers. These solutions are complex and therefore difficult to extend, whereas the theory based ones either require much effort when entering the data to produce the data structure or are based on complex algorithms; they seem not compelling for the designers of commercial GIS software.

Geometric calculations in finite precision arithmetic can yield results that contradict geometric reasoning. Years ago, Franklin in a landmark paper (Franklin, 1984), has given instructive examples. Line intersections are most important in GIS (e.g., in overlay computations). Two lines *AB* and *CD* intersect at point *P* as shown in figure 1. We may find that *P* is neither on line *AB* nor on line *CD*.

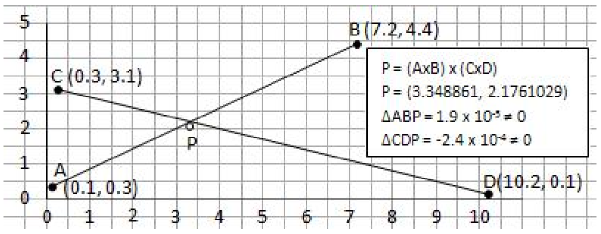

Figure 1: Two lines intersect, but the computed point is not on either line.

SEPARATION OF METRIC AND TOPOLOGICAL PROCESSING

The approaches to metric calculation can be combined with the approaches to representation of geometry (figure 2). Is every combination viable? Which combinations are attractive?

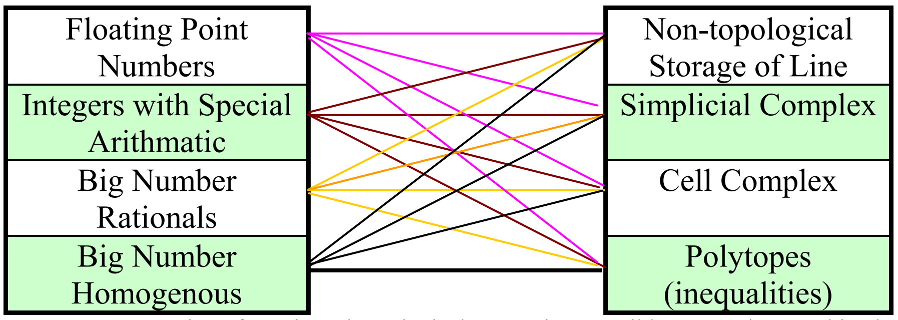

Figure 2: Separation of metric and topological processing; possible approaches combined.

An unexplored instance is big integers combined with polytopes. In order to be easily implementable, it requires homogeneous coordinates represented as big integers. Homogeneous coordinates with big integers are advantageous because the duality in projective geometry can be used and topological relations are computed consistently.

TEST FOR VIABILITY

Computation with big numbers is said to be very slow and complicated (Fortune, 1996) (Yap, 1997) (Li, 2005). In order to test the viability of big integers for GIS geometry, we tested the behavior of geometry with intersecting lines generation after generation whose point coordinates are represented with big integers. Thus, a generation is characterized by a set of points whose coordinate values are big integers and the average length of the points (*length of a point* is the length of the number of digits in any of its coordinate value). The points in one generation create lines which are then intersected to construct new points (of intersection) for next generation. This resembles generations of geometrically constructed points. For assessment we observed;

- the growth of average length of points
- execution times compared to the same computation with floating point numbers
- the ease of programming with big integers

Size of Representation

The result is unfortunately less encouraging: the average length doubles with every following generation of lines intersecting (figure 3).

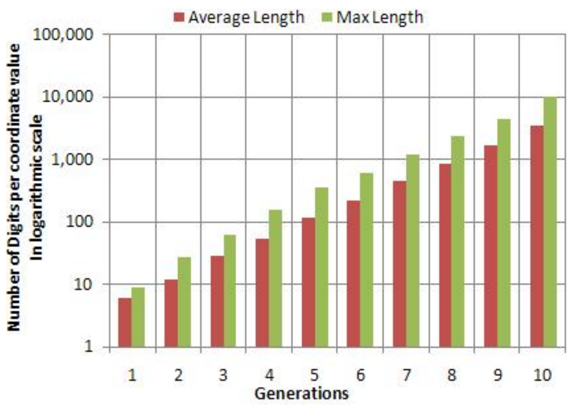

Figure 3: Number of digits per coordinate value.

Performance

The average time for floating point intersections is 1.4 x 10^{-4} seconds, a constant for every round. However, computation with big integers takes the nearly same time for the first 4 rounds and then increasing from round 5 up till > 10^{-2} seconds for round 9 (figure 4).

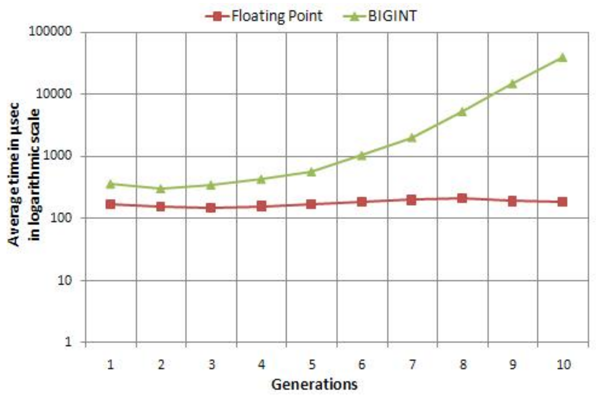

Figure 4: Average time per intersection in µsec.

Ease of Programming

Most modern languages offer ready to use packages for big numbers. Programming with big numbers is definitely easier than when working with *Float* or *Integer* because no consideration to check for overflow is necessary and no hidden bugs.

CONCLUSION AND FUTURE WORK

As a summary, metric computations with big numbers are conceptually simpler and lead to straightforward programs that are guaranteed to be consistent. This leads to more representation options, for example, in case of simplicial complexes the repetition of finite precision computation is not permitted, to avoid inconsistencies by "asking stupid questions". Once we have a precise representation of geometric objects, we can use simplicial complexes by allowing repetitions of computations.

Constructions that go for several generations of constructed points are rare in GIS. Time penalties for such situations are up to 4 or 5 generations of points—based on our test results—negligible. The speed reduction depends therefore on the percentage of "higher generation points".

For applications, which are often constructed geometrically in several generations, the growth in the size of the big numbers representing the coordinates may be problematic. The average size doubles with each generation of computation. Typical GIS operations do not construct long chains of intersections of lines defined as intersections of other lines defined as intersections.

In future, we will explore the representation of convex polytopes using homogenous big integers and geometric operations on these convex polytopes.

BIBLIOGRAPHY

Franklin, W. R., 1984: Cartographic errors symptomatic of underlying algebra problems. In: First international symposium on spatial data handling, Zurich, Switzerland.

Fortune, S. and Wyk, C. J. Van, 1996: Static analysis yields efficient exact integer arithmetic for computational geometry. In: ACM Transactions on Graphics 15(3), pp 223-248.

Yap, C. K., 1997: Robust geometric computation (Handbook of discrete and computational geometry). CRC Press, Inc., Boca Raton, FL, USA.

Li, C., Pion, S. and Yap, C. K., 2005: Recent progress in exact geometric computation. The Journal of Logic and Algebraic Programming, 64, pp 85-111.

Jones, S.P., Hughes, J. and Augustsson, L, 2002: Haskell 98: A non-strict, purely functional language. Available at: http://www.haskell.org/onlinereport/ as on December, 2008.