3-Sum or Problems we do not know how to solve in subquadratic time

Michael Hoffmann <hoffmann@inf.ethz.ch>

17th December 2007

The 3-Sum problem is the following: Given a set S of n integers, does there exist a three-tuple of elements from S that sum up to zero? By testing all three-tuples this can obviously be solved in $O(n^3)$ time. If the tuples to be tested are picked a bit more cleverly, we obtain an $O(n^2)$ algorithm.

Let $(s_1, \ldots, s_n)$ be the sequence of elements from S in increasing order. Then we test the tuples as follows.

```
For i = 1, ..., n − 2 {
    j = i + 1, k = n.
    While k > j {
        If s_i + s_j + s_k = 0 then exit with triple s_i, s_j, s_k.
        If s_i + s_j + s_k > 0 then k = k − 1 else j = j + 1.
    }
}
```

The runtime is clearly quadratic (initial sorting can be done in $O(n \log n)$ time). Regarding the correctness observe that the following is an invariant that holds at begin of the inner loop: There exists no suitable triple that contains s_i and s_ℓ for any $\ell < j$ or $\ell > k$.

Interestingly, this is the essentially the best algorithm known for 3-Sum. It is widely believed that the problem cannot be solved in subquadratic time, but so far this has been proven in some very restricted models of computation only. Moreover, there is a whole class of problems that are equivalent to 3-Sum up to quadratic time reductions; such problems are referred to as **3-Sum-hard**.

Definition 1 *A problem P is 3-Sum-hard if and only if every instance of 3-Sum of size n can be solved using a constant number of instances of P—each of $O(n)$ size—and $o(n^2)$ additional time.*

As an example, consider the Problem **GeomBase**: Given n points on the three horizontal lines $y = 0$, $y = 1$, and $y = 2$, is there a non-horizontal line that contains at least three of them?

GeomBase can be reduced to 3-Sum as follows. For an instance $S = \{s_1, \ldots, s_n\}$ of 3-Sum, create an instance P of GeomBase in which for each s_i there are three points in P: $(s_i, 0)$, $(-s_i/2, 1)$, and $(s_i, 2)$. If there are any three collinear points in P, there must be one from each of the lines $y = 0$, $y = 1$, and $y = 2$. So suppose that $p = (s_i, 0)$, $q = (-s_j/2, 1)$, and $r = (s_k, 2)$ are collinear. The slope of the line through p and q is $(-s_j/2 - s_i)^{-1}$ and the slope of the line through q and r is $(s_k + s_j/2)^{-1}$. The three points are collinear if and only if the two slopes are equal, that is, $-s_j/2 - s_i = s_k + s_j/2 \iff s_i + s_j + s_k = 0$.

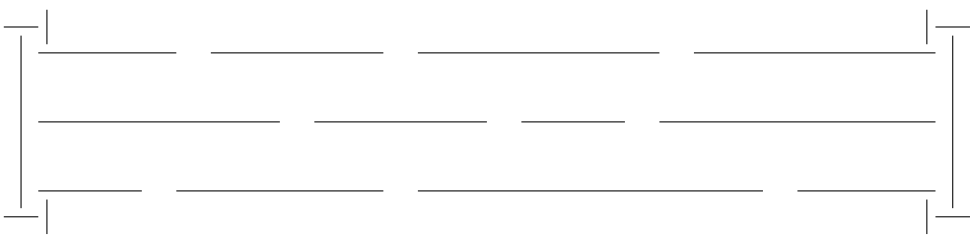
A very similar problem is **General Position**, in which one is given n arbitrary points and has to decide whether any three are collinear. For an instance S of 3-Sum, create an instance P of General Position by projecting the numbers s_i onto the curve $y = x^3$, that is, $P = \{(a, a^3) \mid a \in S\}$.

Suppose three of the points, say, (a, a^3), (b, b^3), and (c, c^3) are collinear. This is the case if and only if the slopes of the lines through each pair of them are equal. (Observe that a, b, and c are pairwise distinct.)

$$\begin{aligned}
(b^3 - a^3)/(b - a) &= (c^3 - b^3)/(c - b) \iff \\
b^2 + a^2 + ab &= c^2 + b^2 + bc \iff \\
b &= (c^2 - a^2)/(a - c) \iff \\
b &= -(a + c) \iff \\
a + b + c &= 0\,.
\end{aligned}$$

Minimum Area Triangle is a strict generalization of General Position and, therefore, also 3-Sum-hard.

In **Segment Splitting/Separation**, we are given a set of n line segments and have to decide whether there exists a line that does not intersect any of the segments but splits them into two non-empty subsets. To show that this problem is 3-Sum-hard, we can use essentially the same reduction as for GeomBase, where we interpret the points along the three lines $y = 0$, $y = 1$, and $y = 2$ as sufficiently small "holes". The parts of the lines that remain after punching these holes form the input segments for the Splitting problem. Horizontal splits can be prevented by putting constant size gadgets somewhere beyond the last holes, see the figure below. What "sufficiently small" means for the size



of those holes can be expressed in terms of the distance between the closest pair of points (that can be computed in $O(n \log n)$ time, for example, via Delaunay triangulation).

In **Segment Visibility**, we are given a set S of n horizontal line segments and two segments $s_1, s_2 \in S$. The question is: Are there two points, $p_1 \in s_1$ and $p_2 \in s_2$ which can see each other, that is, the open line segment $\overline{p_1 p_2}$ does not intersect any segment from S. The reduction from 3-Sum is the same as for Segment Splitting, just put s_1 above and s_2 below the segments along the three lines.

In **Motion Planning**, we are given a robot (line segment), some environment (modeled as a set of disjoint line segments), and a source and a target position. The question is: Can the robot move (by translation and rotation) from the source to the target position, without ever intersecting the "walls" of the environment?

To show that Motion Planning is 3-Sum-hard, employ the reduction for Segment Splitting from above. The three "punched" lines form the doorway between two rooms, each modeled by a constant number of segments that cannot be split, similar to the boundary gadgets above. The source position is in one room, the target position in the other, and to get from source to target the robot has to pass through a sequence of three collinear holes in the door (suppose the doorway is sufficiently small compared to the length of the robot).

Note that this does not imply any quadratic lower bound for motion planning, as the models for which 3-Sum has proven to be quadratic are not powerful enough to solve Motion Planning. 3-Sum hardness only provides some indication that motion planning "is likely to" require quadratic time.

1 Arrangements of Line Segments

Definition 2 *A (n, s)-Davenport-Schinzel sequence is a sequence over an alphabet A of size n in which*

- *no two consecutive characters are the same and*
- *there is no alternating subsequence of the form $\ldots a \ldots b \ldots a \ldots b \ldots$ of $s+2$ characters, for any $a, b \in A$.*

Let $\lambda_s(n)$ be the length of a longest (n, s)-Davenport-Schinzel sequence of order s.

For example, $abcbacb$ is a $(3,4)$-DS-sequence but not a $(3,3)$-DS-sequence because it contains the subsequence $bcbcb$.

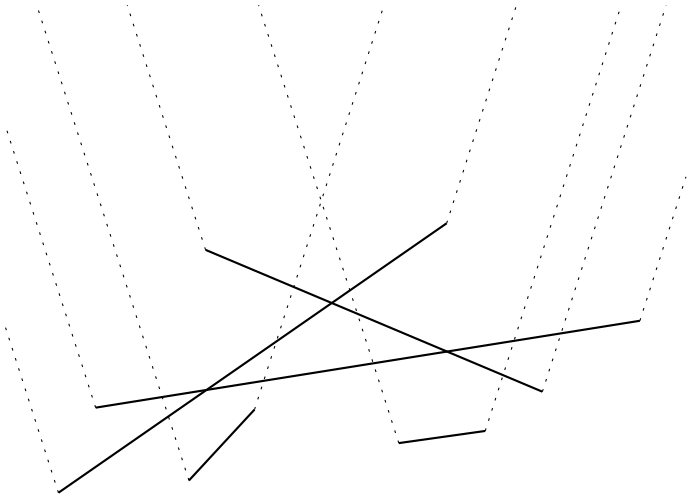
Let $\mathcal{F} = \{f_1, \ldots, f_n\}$ be a collection of real-valued continuous functions defined on a common interval $I \subset \mathbb{R}$. The *lower envelope* $\mathcal{L}_\mathcal{F}$ of $\mathcal{F}$ is defined as the pointwise minimum of the functions f_i, $1 \leq i \leq n$, over I. Suppose that any pair f_i, f_j, $1 \leq i < j \leq n$, intersects in at most s points. Then I can be decomposed into a finite sequence $I_1, \ldots, I_\ell$ of (maximal connected) pieces on each of which a single function from $\mathcal{F}$ defines $\mathcal{L}_\mathcal{F}$. Define the sequence $\phi(\mathcal{F}) = (\phi_1, \ldots, \phi_\ell)$, where f_{ϕ_i} is the function from $\mathcal{F}$ which defines $\mathcal{L}_\mathcal{F}$ on I_i.

Observation 3 *$\phi(\mathcal{F})$ is an (n, s)-Davenport-Schinzel sequence.*

In the case of line segments the above statement does not hold because a set of line segments is in general not defined on a common real interval.

Proposition 4 *Let $\mathcal{F}$ be a collection of n real-valued continuous functions each of which is defined on some real interval. If any two functions from $\mathcal{F}$ intersect in at most s points then $\phi(\mathcal{F})$ is an $(n, s+2)$-Davenport-Schinzel sequence.*

Proof. Let I denote the union of all intervals on which one of the functions from $\mathcal{F}$ is defined. Consider any function $f \in \mathcal{F}$ defined on $[a, b] \subseteq I = [c, d]$. Extend f on I by extending it using almost vertical rays pointing upward, from a use a ray of sufficiently small slope, from b use a ray of sufficiently large slope. For all functions use the same slope on these two extensions such that no extensions in the same direction intersect. Denote the resulting collection of functions totally defined on I by $\mathcal{F}'$. If the rays are sufficiently close to vertical then $\phi(\mathcal{F}') = \phi(\mathcal{F})$.



For any $f \in \mathcal{F}'$ a single extension ray can create at most one additional intersection with any $g \in \mathcal{F}'$. (Let $[a_f, b_f]$ and $[a_g, b_g]$ be the intervals on which the function f and g, respectively, was defined originally. Consider the ray r extending f from a_f to the left. If $a_f \in [a_g, b_g]$ then r may create a new intersection with g, if $a_f > b_g$ then r creates a new intersection with the right extension of g from b_g, and if $a_f < a_g$ then r does not create any new intersection with g.)

On the other hand, for any pair s, t of segments, neither the left extension of the leftmost segment endpoint nor the right extension of the rightmost segment endpoint can introduce an additional intersection. Therefore, any pair of segments in $\mathcal{F}'$ intersects at most $s + 2$ times and the claim follows. □

Next we will give an upper bound on the length of Davenport-Schinzel sequences for small s.

Lemma 5 $\lambda_1(n) = n$, $\lambda_2(n) = 2n - 1$, *and* $\lambda_3(n) \leq 2n(1 + \log n)$.

Proof. $\lambda_1(n) = n$ is obvious. $\lambda_2(n) = 2n - 1$ is given as an exercise. We prove $\lambda_3(n) \leq 2n(1 + \log n) = O(n \log n)$.

For $n = 1$ it is $\lambda_3(1) = 1 \leq 2$. For $n > 1$ consider any $(n, 3)$-DS sequence σ of length $\lambda_3(n)$. Let a be a character which appears least frequently in σ. Clearly a appears at most $\lambda_3(n)/n$ times in σ. Delete all appearances of a from σ to obtain a sequence σ' on $n-1$ symbols. But σ' is not necessarily a DS-sequence because there may be consecutive appearances of a character b in σ', in case that $\sigma = \ldots bab \ldots$.

Claim: There are at most two pairs of consecutive appearances of the same character in σ'. Indeed, such a pair can be created around the first and last appearance of a in σ only. If any intermediate appearance of a creates a pair bb in σ' then $\sigma = \ldots a \ldots bab \ldots a \ldots$, in contradiction to σ being an $(n, 3)$-DS sequence.

Therefore, one can remove at most two characters from σ' to obtain a $(n-1, 3)$-DS-sequence $\tilde{\sigma}$. As the length of $\tilde{\sigma}$ is bounded by $\lambda_3(n-1)$, we obtain $\lambda_3(n) \leq \lambda_3(n-1) + \lambda_3(n)/n + 2$. Reformulating yields

$$\frac{\lambda_3(n)}{n} \leq \frac{\lambda_3(n-1)}{n-1} + \frac{2}{n-1} \leq 1 + 2\sum_{i=1}^{n-1} \frac{1}{i} = 1 + 2H_{n-1}$$

and thus $\lambda_3(n) \leq 2n(1 + \log n)$. □

Constructing lower envelopes.

Theorem 6 *Let* $\mathcal{F} = \{f_1, \ldots, f_n\}$ *be a collection of real-valued continuous functions defined on a common interval* $I \subset \mathbb{R}$ *such that no two functions from* $\mathcal{F}$ *intersect in more than* s *points. Then the lower envelope* $\mathcal{L}_{\mathcal{F}}$ *can be constructed in* $O(\lambda_s(n) \log n)$ *time. (Assuming that computing an intersection between any two functions can be done in constant time.)*

Proof. Divide and conquer. For simplicity, assume that n is a power of two. Split $\mathcal{F}$ into two almost equal parts $\mathcal{F}_1$ and $\mathcal{F}_2$ and construct $\mathcal{L}_{\mathcal{F}_1}$ and $\mathcal{L}_{\mathcal{F}_2}$ recursively. The resulting envelopes can be merged using line sweep by processing $2\lambda_s(n/2)+\lambda_s(n) \leq 3\lambda_s(n)$ events. (The first term accounts for events generated by the vertices of the two envelopes to be merged. The second term accounts for their intersections, each of which generates a vertex of the resulting envelope.) Observe that no sorting is required and the SLS structure is of constant size. Therefore, the sweep can be done in time linear in the number of events.

This yields the following recursion for the runtime $T(n)$ of the algorithm. $T(n) \leq 2T(n/2) + c\lambda_s(n)$, for some constant $c \in \mathbb{N}$. Observe that $k\lambda_s(n/k) \leq \lambda_s(n)$, for $k \mid n$, because any k DS-sequences on an alphabet of size n/k can be concatenated to a single DS-sequence on an alphabet of size n by using paiwise disjoint (parts of the) alphabets for each of the k sequences. It follows that $T(n) \leq c\sum_{i=1}^{\log n} 2^i \lambda_s(n/2^i) = c\sum_{i=1}^{\log n} \lambda_s(n) = O(\lambda_s(n) \log n)$. □

Remarks. The upper bound is not tight. It can be shown that $\lambda_3(n) = \Theta(n\alpha(n))$, where $\alpha(n)$ is the inverse Ackermann Function. More precisely, Hart and Sharir have shown in 1986 that $\frac{1}{2}n(\alpha(n) - 4) \leq \lambda_3(n) \leq (68n - 32)\alpha(n) + (70n - 32)$.

The Ackermann function is defined on $\mathbb{N} \times \mathbb{N}$ as follows. $A(1, n) = 2n$, $A(k, 1) = 2$, for $k \geq 2$, and $A(k, n) = A(k - 1, A(k, n - 1))$, for $k, n \geq 2$. The inverse Ackermann Function $\alpha(n)$ is then given by $\alpha(n) = \min\{k \in \mathbb{N} \mid A(k, k) \geq n\}$. As $A(4, 4)$ is a tower with 65536 2's, $\alpha(n) \leq 4$ for all practical purposes.

$\lambda_s(n)$ is almost linear even for larger values of s. For example, $\lambda_4(n) = \Theta(n2^{\alpha(n)})$.