Problem

Given a group of polytopes $\mathcal{P}$, in order to see what the set $\{(P+Q)\cap R|\forall P,Q,R\in\mathcal{P}\}$ is, we can divide the total computing process into 2 parts, computation of Minkowski sums $\{P+Q|\forall P,Q\in\mathcal{P}\}$ and computation of intersection of polytope pairs.

1. Computation of Minkowski sums

Analysis

Let $N_A(x)$ be the nomral cone of polytope A at point x.

For the first part, the computation of Minkowski sums, the essential observation comes from the following sentence,

Assume that there are two polytopes, P and Q, equipted with 2 vertexes, v and u, seperately. $N_P(v)\cap N_Q(u)$ is a full dimensional cone *iff* the point $v+u$ is a vertex of Minkowski sum $P+Q$,

The [Fukuda-Weibel method (https://www.sciencedirect.com/science/article/pii/S0747717104000409)](https://www.sciencedirect.com/science/article/pii/S0747717104000409) enjoys the above observation to produce vertexes of Minkowski sum of a single pair of polytopes. The Fukuda-Weibel method can be used explicitly to compute vertexes of the Minkowski sums. However, because all pairs of a group of polytopes are needed here rather than a single pair, explicitly using the Fukuda-Weibel method can cause redundant computation. Since that whether a point $v+u$ is qualified as a vertex of Minkowski sum $P+Q$ is absolutely decided by that whether their normal cones, $N_P(v)$ and $N_Q(u)$, have an full dimensional intersection or not, we could only focus on those normal cones -- each polytope A with vertex x will induce a normal cone $N_A(x)$ -- to see whether they have a full dimensional intersection or not. In fact, we can move all normal cones of those polytopes to a same space and make them pointed at the same original point, then witness that which pair of those normal cones has a full dimensional intersection, and that will cause that the point, addition of their vertexes, to become a vertex of the corrosponding Minkowski sum. An example was shown in Fig 1.

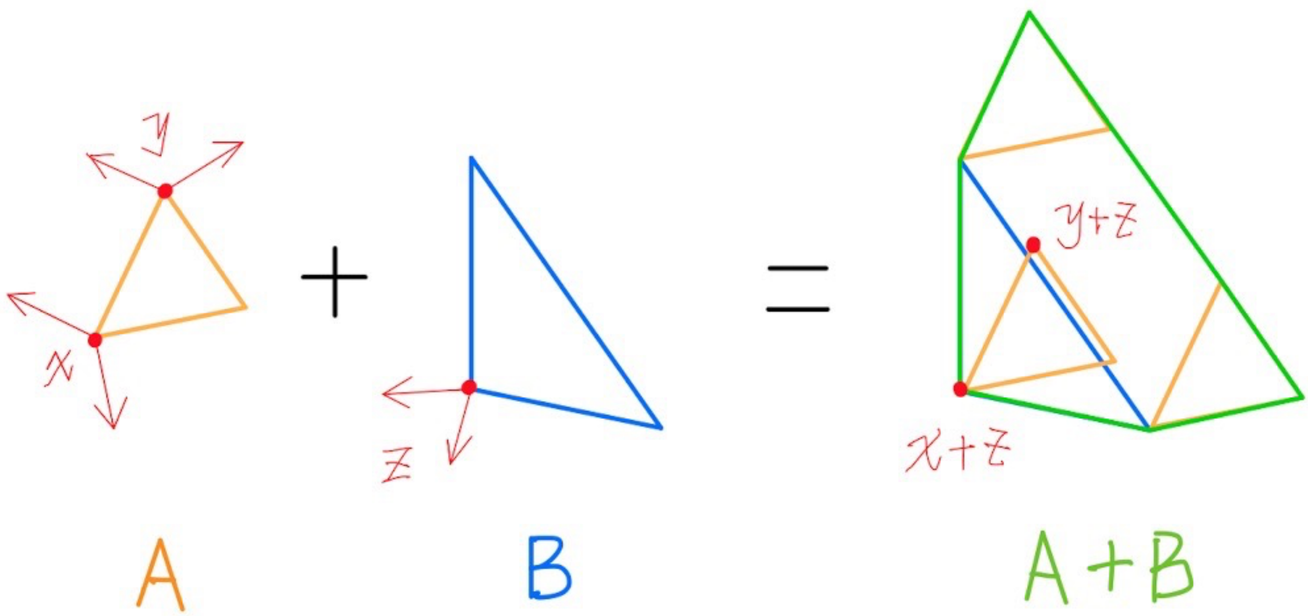

Fig 1. polytope A with vertex x and vertex y induces normal cone $N_A(x)$ and normal cone $N_A(y)$, polytope B with vertex z induces normal cone $N_B(z)$. Since $N_A(x)$ has a full dimensional intersection with $N_B(z)$, point $x + z$ is qualified as a vertex of $A + B$. Since $N_A(y)$ does not have a full dimensional intersection with $N_B(z)$, point $y + z$ is **not**qualified as a vertex of $A + B$.

Comparing to the Fukuda-Weibel method, our method is more scalable.

Procedures (Algorithm)

After the above analysis, we could design a algorithm to efficiently compute the Minkowski sums.

Given a group of polytopes $\mathcal{P}$, computation of Minkowski sums $\{P + Q | \forall P, Q \in \mathcal{P}\}$ can be performed by the following 2 steps.

1. Create all normal cones of the group of polytopes $\mathcal{P}$, that is, set $\{N_P(v) | \forall P \in \mathcal{P}, \forall v \in Vertex(P)\}$.
2. Search for every pair of the above normal cones that has an full dimensional intersection, and add a vertex into the vertex set of corrosponding Minkowski sum. (For example, suppose that we found $N_{P_1}(v)$ and $N_{P_2}(u)$ have a full dimensional intersection, then we would add $v + u$ into the vertex set of Minkowski sum $P_1 + P_2$.)

The pseudo code looks as follows.

```
For p in P:
    For v in ver(p):
        Create (p, v, N_p(v))
        collect it to the set {(p, v, N_p(v))}

For (a, x, N_a(x)) in {(p, v, N_p(v))}:
    For (b, y, N_b(y)) in {(p, v, N_p(v))}:
        If (N_a(x) has an full dimensional intersection with N_b(y)):
            Add x + y into the vertex set of a + b
```

Implementation

We define the following structures to support the above pseudo code.

1. List `self.P[i]` is to support the group of polytopes $\mathcal{P} = \{P_i\}$.
2. List `self.normalcones[i]` is to support set $\{N_P(v) | \forall P \in \mathcal{P}, \forall v \in Vertex(P)\}$. Inside, Tuple `(i, v, N)` to support (p, v, N_p(v)), where `i` is the index of polytope $P_i \in \mathcal{P}$, `v` is the value of vertex v_j, and the `N` is the normal cone of polytope P_i in terms of v_j. We use `polyhedron()` from SageObject to wrap `N`, and we can explicitly use the `plyhedron().intersection()` to compute the intersection of two polytopes in the following.
3. Matrix `self.Minkowski_sum[i][j]` is to store the values of its vertexes. i and j are the indexes of polytopes P_i and P_j, seperately.

Time Complexity

Here is a rough estimation of time complexity on the current method.

Suppose that the number of the group of polytopes is n, and each polytope is a k dimensional simplex, which means they have exactly k vertexes for each.

Since the normal cone is essential part of this method, it is predictable to see that the time complexity of it has a close relationship with the number of normal cones, that is, $k * n$.

There are 3 subparts, construction of normal cones, intersection detetion of cones and accumlation of vertices of minkowski sums. The time complexity of each is as follows:

$k * n * T$ (construction of normal cones)

$(k * n)^2 * I$ (intersection detetion of cones)

$(k * n)^2$ (accumlation of vertices of minkowski sums)

where T is the time complexity of construction of a single normal cone, I is the time complexity of obtainning intersection of 2 polyhedra (cones) and testing whether the intersection is full dimensional.

Note that, since that the H-representation of a prime cone is exactly the V-representation of its dual (normal) cone, and that we have already know the V-representation of the prime cone, currently, T actually depends on the process from transfering V-representation of a cone to its H-representation. specifically, the actual time is decided by internal method in `Polyhedron()` in SageObject.