1 Original Algorithm

Algorithm 1 AllPairSearch - bgj3

Require: A list L of N_0 (n-dimensional) lattice vectors, number of repetitions (B_0, B_1, B_2), filter radius $(\alpha_0, \alpha_1, \alpha_2)$, and a goal norm ℓ.

Ensure: A list of reducing pairs in L with a sum/difference shorter than ℓ.

$N \leftarrow \emptyset$
for $i = 0, 1, \ldots, B_0 - 1$ **do**
    Pick a random filter center c_0 from S^{n-1}
    Compute $L_i \leftarrow \{v \in L \mid v \text{ can pass } F_{c_0,\alpha_0}\}$
    for $j = 0, 1, \ldots, B_1/B_0 - 1$ **do**
        Pick a random filter center c_1 from S^{n-1}
        Compute $L_{ij} \leftarrow \{v \in L_i \mid v \text{ can pass } F_{c_1,\alpha_1}\}$
        for $k = 0, 1, \ldots, B_2/B_1 - 1$ **do**
            Pick a random filter center c_2 from S^{n-1}
            Compute $L_{ijk} \leftarrow \{v \in L_{ij} \mid v \text{ can pass } F_{c_2,\alpha_2}\}$
            $N \leftarrow N \cup \{(u, v) \in L_{ijk} \times L_{ijk} \mid \|u \pm v\| < \ell\}$
        end for
    end for
end for
return N

2 Definitions

2.1 Dual Lattice: For a full-rank matrix $\mathbf{B} = (\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$ representing a lattice basis, the lattice generated by this basis is denoted as $\mathcal{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$. The dual lattice of $\mathcal{L}(\mathbf{B})$ is defined as $\mathcal{L}(\mathbf{B}^\vee)$, where $\mathbf{B}^\vee = (\mathbf{b}_0^\vee, \mathbf{b}_1^\vee, \ldots, \mathbf{b}_{n-1}^\vee)$ satisfies the inner product relation:

$$\langle \mathbf{b}_i^\vee, \mathbf{b}_j \rangle = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, $\mathrm{span}(\mathbf{b}_0^\vee, \mathbf{b}_1^\vee, \ldots, \mathbf{b}_{n-1}^\vee) = \mathrm{span}(\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$.

2.2 Voronoi Filter $F_c^{\mathbf{Voronoi}}$ (w.r.t. $\mathbf{B}^\vee$): A vector v passes the Voronoi filter F_c^{Voronoi} if c is the **deterministically determined closest** vector in the dual lattice $\mathcal{L}(\mathbf{B}^\vee)$ to v. This means v is assigned to the Voronoi cell of c using a tie-breaking rule.

3 Algorithm 2: Updated Sieve with Algebraic Pre-filter

Algorithm 2 AllPairSearch - bgj3 (Updated)

Require: A basis $\mathbf{B}$ for the original lattice L_0. A list L of N_0 vectors from L_0. Number of centers for each layer (M_0, M_1, M_2). A goal norm ℓ.

Ensure: A list of reducing pairs (u, v) in L with a sum/difference shorter than ℓ.

$N_{out} \leftarrow \emptyset$
$\mathbf{B}^\vee \leftarrow \text{DualBasis}(\mathbf{B})$ ▷ Compute dual lattice basis
$u_1 \leftarrow \text{FindShortestNonZeroDualLatticeVector}(\mathbf{B}^\vee)$ ▷ Select a short vector for coarse sieving
$K_{\text{coarse\_range}} \leftarrow \{0, \pm 1, \pm 2\}$
$L_{\text{coarse\_temp}} \leftarrow \emptyset$
for each vector $v \in L$ **do**
    $k_{\text{val}} \leftarrow \text{round}(\langle v, u_1 \rangle)$
    if $k_{\text{val}} \in K_{\text{coarse\_range}}$ **then**
        $L_{\text{coarse\_temp}} \leftarrow L_{\text{coarse\_temp}} \cup \{v\}$
    end if
end for
$L' \leftarrow L_{\text{coarse\_temp}}$ ▷ L' is the list after coarse filtering

for $i = 0, 1, \ldots, M_0 - 1$ **do**
    $c_0 \leftarrow \text{GetDualLatticeCenter}(i, \mathbf{B}^\vee)$
    $L_i \leftarrow \{v \in L' \mid \text{FindClosestDualLatticeVector}(v, \mathbf{B}^\vee) == c_0\}$
    for $j = 0, 1, \ldots, M_1 - 1$ **do**
        $c_1 \leftarrow \text{GetDualLatticeCenter}(j, \mathbf{B}^\vee)$
        $L_{ij} \leftarrow \{v \in L_i \mid \text{FindClosestDualLatticeVector}(v, \mathbf{B}^\vee) == c_1\}$
        for $k = 0, 1, \ldots, M_2 - 1$ **do**
            $c_2 \leftarrow \text{GetDualLatticeCenter}(k, \mathbf{B}^\vee)$
            $L_{ijk} \leftarrow \{v \in L_{ij} \mid \text{FindClosestDualLatticeVector}(v, \mathbf{B}^\vee) == c_2\}$
            $N_{out} \leftarrow N_{out} \cup \{(u, v) \in L_{ijk} \times L_{ijk} \mid \|u \pm v\| < \ell\}$
        end for
    end for
end for
return N_{out}

4 Algorithm 3: AllPairSearch with Hybrid Sieve

Algorithm 3 AllPairSearch with Hybrid Sieve

Require: The LLL-reduced basis $\mathbf{B}$. A list L_0 of N_0 vectors. A goal norm ℓ. number of repetitions (B_0, B_1, B_2). Number of coarse-sieve vectors N_{coarse}. Coarse-sieve integer range $K_{\text{rangeCoarse}}(\{0, \pm 1, \pm 2\} or \{0, \pm 1\})$.

Ensure: A list of reducing pairs (u, v) in L with a sum/difference shorter than ℓ.

$N_{out} \leftarrow \emptyset$

$\mathbf{B}^{\vee} \leftarrow \text{DualBasis}(\mathbf{B})$
$R_{enum\_c} = 1.2\lambda_1(\mathcal{L}(B^{\vee}))$
$C_{\text{dense\_pool}} \leftarrow \text{EnumerateShortDualVectors}(\mathbf{B}^{\vee}, N_{\text{enum\_c}}, R_{enum_c})$ ▷ Enumerate the $N_{\text{enum-c}}$ shortest dual vectors
Sort $C_{\text{dense\_pool}}$ by increasing norm, lexicographical order if same norm.

$U_{\text{coarse}} \leftarrow \{C_{\text{dense\_pool}}[0], C_{\text{dense\_pool}}[1]\}$ ▷ Select the two shortest vectors u1, u2
$L_{\text{filtered}} \leftarrow L_0$
for $\mathbf{u}_1, \mathbf{u}_1 \in U_{\text{coarse}}$ **do**
    $L_{\text{temp}} \leftarrow \emptyset$
    for each $\mathbf{v} \in L_{\text{filtered}}$ **do**
        if $(\langle \mathbf{v}, \mathbf{u}_1 \rangle \in K_{\text{rangeCoarse}} \mid\mid \langle \mathbf{v}, \mathbf{u}_2 \rangle \in K_{\text{rangeCoarse}})$ **then**
            $L_{\text{temp}} \leftarrow L_{\text{temp}} \cup \{\mathbf{v}\}$
        end if
    end for
    $L_{\text{filtered}} \leftarrow L_{\text{temp}}$ ▷ Iteratively shrink the list (AND logic)
end for
$L' \leftarrow L_{\text{filtered}}$ ▷ The final list after multi-vector sieving

$C_0 \leftarrow C_{\text{dense\_pool}}[0 \ldots B_0 - 1]$
$C_1 \leftarrow C_{\text{dense\_pool}}[0 \ldots (B_1/B_0) - 1]$
$C_2 \leftarrow C_{\text{dense\_pool}}[0 \ldots (B_2/B_1) - 1]$

for each $\mathbf{c}_0 \in C_0$ **do**
    $L_i \leftarrow \{\mathbf{v} \in L' \mid \text{FindClosestVector}(\mathbf{v}, C_0) == \mathbf{c}_0\}$
    if $|L_i| \leq 1$ **then continue**
    end if
    for each $\mathbf{c}_1 \in C_1$ **do**
        $L_{ij} \leftarrow \{\mathbf{v} \in L_i \mid \text{FindClosestVector}(\mathbf{v}, C_1) == \mathbf{c}_1\}$
        if $|L_{ij}| \leq 1$ **then continue**
        end if
        for each $\mathbf{c}_2 \in C_2$ **do**
            $L_{ijk} \leftarrow \{\mathbf{v} \in L_{ij} \mid \text{FindClosestVector}(\mathbf{v}, C_2) == \mathbf{c}_2\}$
            if $|L_{ijk}| > 1$ **then**
                $N_{out} \leftarrow N_{out} \cup \{(u, v) \in L_{ijk}^2 \mid u \neq v, \|u \pm v\| < l\}$
            end if
        end for
    end for
end for
return N_{out}

5 Helper Functions (Conceptual)

- **DualBasis(B):** Computes the basis $\mathbf{B}^\vee$ for the dual lattice $\mathcal{L}(\mathbf{B}^\vee)$.
- **FindShortestNonZeroDualLatticeVector($\mathbf{B}^\vee$):** Returns a shortest non-zero vector from the dual lattice $\mathcal{L}(\mathbf{B}^\vee)$.
- **GetDualLatticeCenter(index, $\mathbf{B}^\vee$):** Returns the (index+1)-th ”short” vector from $\mathcal{L}(\mathbf{B}^\vee)$ based on a predefined ordering (e.g., by increasing norm or lexicographical order of coefficients of basis combination).
- **FindClosestDualLatticeVector(v, $\mathbf{B}^\vee$):** Solves the CVP for v in the infinite lattice $\mathcal{L}(\mathbf{B}^\vee)$.
- **EnumerateShortDualVectors($\mathbf{B}^\vee, N_{\textbf{enum-c}}, R_{enum\_c}$):** Runs a lattice enumeration algorithm (e.g., KFP) to find all vectors in $\mathcal{L}(\mathbf{B}^\vee)$ with norm up to $1.2R_{\text{enum}}$.
- **GenerateDenseCenters($\mathbf{B}^\vee, U_{\textbf{all\_short}}$):** Creates a large, sorted list of candidate centers. This can be a combination of vectors from $U_{\text{all\_short}}$, vectors from the basis $\mathbf{B}^\vee$ itself, and short integer linear combinations of the basis vectors.
- **FindClosestVector($\mathbf{v}, C_{\textbf{set}}$):** Finds the vector in the **finite set** C_{set} that is closest to $\mathbf{v}$. This is a simple linear scan, not a CVP problem.