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 $L_{ijk} \leftarrow \{v \in L_{ij} \mid v \text{ can pass } F_{c_2,\alpha_2}\}$
      $N \leftarrow N \cup \{(u, v) \in L_{ijk} \mid \|u \pm v\| < \ell\}$
    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 $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} \mid \mathbf{x} \in \mathbb{Z}^n\}$. 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, $\text{span}(\mathbf{b}_0^\vee, \mathbf{b}_1^\vee, \ldots, \mathbf{b}_{n-1}^\vee) = \text{span}(\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{n-1})$.

2.2 Voronoi Filter F_c^{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: the order of coordinates if it is equidistant to multiple dual lattice points.

3 Algorithm2

Algorithm 2: AllPairSearch - bgj3 (Updated)

Require: A basis $\mathbf{B}$ for the original lattice L_0. A list L of N_0 (n-dimensional) lattice vectors from L_0. Number of repetitions (B_0, B_1, B_2). A goal norm ℓ.

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

$N \leftarrow \emptyset$

$\mathbf{B}^\vee \leftarrow \text{DualBasis}(\mathbf{B})$ ▷ Compute dual lattice basis

$u_1 \leftarrow \text{FindShortestNonZeroDualLatticeVector}(\mathbf{B}^\vee)$ ▷ Select a short vector from dual lattice (often the shortest)

$K_{\text{coarse\_range}} \leftarrow \{0, \pm 1, \pm 2\}$

$L_{\text{coarse\_temp}} \leftarrow \emptyset$

for each vector $v \in L$ **do**

  $k_{\text{val}} \leftarrow \text{round}(\langle v, u_1 \rangle)$ ▷ Compute the rounded inner product

  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

$L \leftarrow L_{\text{coarse\_temp}}$ ▷ Coarse filter L using algebraic partitioning

for $i = 0, 1, \ldots, B_0 - 1$ **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 $j = 0, 1, \ldots, (B_1/B_0) - 1$ **do**

    $c_1 \leftarrow \text{GetDualLatticeCenter}(j \cdot B_0 + i, \mathbf{B}^\vee)$ ▷ Center determined by global iteration count

    $L_{ij} \leftarrow \{v \in L_i \mid \text{FindClosestDualLatticeVector}(v, \mathbf{B}^\vee) == c_1\}$

    for $k = 0, 1, \ldots, (B_2/B_1) - 1$ **do**

      $c_2 \leftarrow \text{GetDualLatticeCenter}(k \cdot B_1 + j \cdot B_0 + i, \mathbf{B}^\vee)$

      $L_{ijk} \leftarrow \{v \in L_{ij} \mid \text{FindClosestDualLatticeVector}(v, \mathbf{B}^\vee) == c_2\}$

      $N \leftarrow N \cup \{(u, v) \in L_{ijk} \times L_{ijk} \mid \|u \pm v\| < \ell\}$

    end for

  end for

end for

return N

4 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 shortest non-zero vector from the dual lattice $\mathcal{L}(\mathbf{B}^\vee)$. This is typically solved using a lattice reduction algorithm.
- **FindClosestVector(v, C_{set}):** Returns the vector $c' \in C_{\text{set}}$ that minimizes $\|v - c'\|$. (Note: This is still used by the initial coarse filter for historical reasons or specific design; if the algebraic partitioning is the *only* coarse filter, this function might become obsolete.)
- **GetDualLatticeCenter(index, $\mathbf{B}^\vee$):** Returns the (index+1)-th "short" vector from $\mathcal{L}(\mathbf{B}^\vee)$ based on a pre-

defined ordering (e.g. by increasing norm or lexicographical order of coefficients of basis combination). This function ensures the sequential selection of distinct dual lattice centers for the sieve.

- **FindClosestDualLatticeVector(v, $\mathbf{B}^\vee$):** Returns the unique closest vector $c \in \mathcal{L}(\mathbf{B}^\vee)$ to v. If v is equidistant to multiple dual lattice points, a deterministic tie-breaking rule (e.g. based on the order implied by 'GetDual-LatticeCenter' or a fixed coordinate order) is applied to ensure a single result.