ITEM #230 - RTB Acceleration for UnalignedAND-BTP

Recursive Tier-wise Segmentation, Anchor→CCC Gating, and Residual Peeling (Fish-Controlled)

Conversation：UnalignedAND-BTP 算法加速

20260111

Authors: Sizhe Tan & GPT-Obot

ITEM #230 — RTB Acceleration for UnalignedAND-BTP

Recursive Tier-wise Segmentation, Anchor→CCC Gating, and Residual Peeling (Fish-Controlled)

Category

DBM-COT · Metric Space Intelligence · UnalignedAND-BTP · Acceleration Framework

Status

Architecture + Algorithmic Workflow (ready for implementation and iterative hardening)

---

0. Motivation

UnalignedAND-BTP（BucketTreeOfPermutation）是 DBM 在 **复杂 Metric Space**（GraphStarmap / SequenceStarmap / ImageStarmap 等）上做**非启发式可控精度匹配**的核心工具，用于：

- **(2A) Best Match / Top-K Matches**：两点之间的 Metric Distance、解释链、对齐映射；
- **(2B) Structure Prediction / Composition**：用一组已知点组合/预测一个大型点（例如蛋白/结构拼装、长序列结构解释）。

但其计算复杂度在大节点规模时迅速爆炸。
RTB 的目的不是替代 BTP 内核，而是提供一个外层"鱼控式加速框架"，把全局巨搜索转为：

分段候选生成 → 片段级 BTP → 结果去重与冲突消解 → 残差剥离 → 递归收敛
同时用 #227 的护栏确保：**宁可慢、宁可候选多，也不静默漏解。**

---

1. RTB 的核心思想（One Sentence）

RTB = 用分段（Segmentation）把搜索域拆小，用锚定（Anchors）把候选召回变稀疏，用 CCC Gate 把噪声 occurrences 降维净化，用 Residual Peeling 把已确认的不重叠匹配从两侧剥离，递归迭代直到收敛，再对残差进行最终 BTP 精扫。

---

2. 统一尺度：Reach/SpanNeed 替代"3× Size"

你原先用"piece size ≥ 3×A"表达"片段要足够大避免边界漏解"，但不同 Starmap 的 size 含义不同。RTB 统一改为：

- **Reach(A)**：匹配影响半径（steps/lags/hops/patch radius 的上界）
- **SpanNeed(A)**：A 的可验证匹配所需最小跨度（例如 Sequence：`len(A)+2*maxLag`）
- **HaloWidth(P)**：片段 core 周围的上下文边界（halo）

不漏解基本条件（PieceWithHalo Contract）：

- `CoreSpan(P) ≥ SpanNeed(A)`
- `HaloWidth(P) ≥ Reach(A)`
- `Union(core(P_i))` 覆盖需要搜索的 B 域

---

3. 两大问题类型与两大加速方向

3.1 Problem 3A：Small A vs Large B（小对大）

目标：在大 B 中找 A 的 best / top-k matches。

RTB 给出两条等价主线（可并存）：

- **(4A) HGP-like 全覆盖切片**：B 全域切片 + halo，逐片跑 BTP；
- **(4B) Anchor→Occurrence→CCC→局部切片**：用 anchors 在 B 中定位 occurrences，再围绕 occurrences 切局部片段跑 BTP。

3.2 Problem 3B：Large A vs Large B（大对大）

目标：大规模匹配与结构对齐（通常需要多块 match 组合）。

RTB 给出两条主线（同样可并存）：

- **(5A) HGP-like + Peeling**：先切 A 成小块，对每块做 3A（4A），收集 matches 后剥离 residual，迭代；
- **(5B) Anchor-like + Peeling**：先切 A 成小块，对每块做 3A（4B），收集 matches 后剥离 residual，迭代。

---

4. RTB(3A) Small-vs-Large：两条路径的精确定义

4.1 4A：HGP-like 全覆盖切片（Exact-Friendly）

步骤：

1. 用 `SpanNeed(A)` 与 `Reach(A)` 在 B 上生成 `PieceWithHalo` 集合 `{P_i}`；
2. 对每个 `P_i` 运行 `BTP(A, B|P_i)`，得到候选 matches；
3. 对 matches 做 `MatchKey` 去重；
4. 取 global Top-K（或最优）。

优点：

- 容易做到 EXACT（不漏解），因为 coverage 可证明；
- 工程逻辑最直，最适合做“鱼控基线”。

缺点：

- 当 B 极大时片段数多，仍然昂贵。

4.2 4B：Anchor→Occurrence→CCC Gate→局部切片（吞吐提升主力）

步骤：

1. 在 A 中选择 anchor 集合 `S`（稀有度/区分度优先）；
2. 在 B 中检索每个 anchor 的 occurrences：`Occ(B, a)`；
3. 对每个 occurrence 计算局部 CCC：`CCC(A, B@occ)`；
4. `OccurrenceCccGate` 做三态决策：`ACCEPT / DEFER / REJECT`；
5. 对 ACCEPT 的 occurrences，扩张为局部 `PieceWithHalo`，并做 piece merge/union；
6. 在这些局部 pieces 上运行 BTP，合并、去重，得到结果。

关键点（你补充的核心）：

occurrence 找到后必须做 CCC，以**删除低质量 occurrences 添乱**，否则候选爆炸与误导不可避免。

Exactness 注意：

- EXACT 模式下 CCC Gate **不得凭质量 REJECT**，只能基于 `lbHint` 安全剪枝（或 DEFER）；
- RECALL 模式下允许质量阈值 REJECT，但必须输出 coverage/missRisk。

---

5. RTB(3B) Large-vs-Large：Peeling 递归收敛

5.1 5A：切 A + 4A 搜 B + 剥离（更稳）

步骤：

1. 将 Large A 切成多个小块 `{A_j}`（每块可验证，保留 halo 语义）；
2. 对每个 `A_j` 与 Large B 调用 4A，得到候选 matches；
3. 全量收集 matches，做 `MatchKey` 去重；
4. 构建冲突关系（overlap / inconsistent mapping），选择兼容集合 `S`；
5. 对集合 `S` 中满足 **SafePeel** 的 matches 执行剥离：
    - 必须“完全落在 core 内”（不触及边界 halo）才可 peel；
6. 得到 residual A’ 与 B’，重复步骤 1–5，直到无新增 peel；
7. 对最终 residual 执行一次更精细的 3A（通常用 4A 基线）补尾。

5.2 5B：切 A + 4B 搜 B + 剥离（更快）

与 5A 相同，只是每个 `A_j` 的 small-to-large 使用 4B（anchor/CCC）生成更稀疏候选片段。

---

6. “漏解高危点”与 RTB 的防护机制

RTB 最容易出逻辑遗漏的五处，分别对应你要求重点测试的五类：

1. **边界（Boundary）**：
    - 风险：切片或局部片段缺上下文，跨边界真解被静默排除；
    - 防护：`PieceWithHalo` + `HaloWidth ≥ Reach(A)` + core-coverage；触及 halo 的 match 不可 peel。
2. **重复（Duplicate）**：
    - 风险：同一 match 在不同片段/不同 anchor 处反复出现，污染 Top-K 与 peeling；
    - 防护：`MatchKey` canonical 化去重（同 key 保留最优距离/最强证据）。
3. **低质量 Occurrence（Noise）**：
    - 风险：occurrences 极多，噪声占多数；错误过滤又会漏解；
    - 防护：`OccurrenceCccGate` 三态（ACCEPT/DEFER/REJECT）+ EXACT/RECALL 双模式规则。
4. **冲突（Conflict）**：
    - 风险：多个 matches 重叠或映射不一致，错误组合会导致错剥与后续漏解；
    - 防护：ConflictGraph + 兼容集合选择（最小实现可先用 greedy 非重叠）。
5. **剥离（Peeling）**：
    - 风险：过早剥离边界 match，会破坏跨片段的更大真解；
    - 防护：`SafePeel`：**只剥离完全落在 core 内且不与其它 match 冲突**者；触及 halo 一律不剥。

---

7. RTB 的流程总图（Mermaid）

```
flowchart TD
  subgraph S3A[RTB for Small A vs Large B]
    A1[Compute Reach(A), SpanNeed(A)]
    A2[Candidate Generation]
    A2a[4A: Full Coverage Pieces\nPieceWithHalo]
```

```
    A2b[4B: Anchors -> Occurrences]
    A3[Occurrence CCC Extraction]
    A4[OccurrenceCccGate\nACCEPT/DEFER/REJECT]
    A5[Pieces (local) Expand + Merge/Union]
    A6[Piece-level Cheap LB Filter]
    A7[Run UnalignedAND-BTP on surviving pieces]
    A8[MatchKey Dedup + Global Top-K]
    A1-->A2
    A2-->A2a-->A6
    A2-->A2b-->A3-->A4-->A5-->A6
    A6-->A7-->A8
  end

  subgraph S3B[RTB for Large A vs Large B (Iterative Peeling)]
    B1[Cut A into blocks A_j\n(with core+halo semantics)]
    B2[For each A_j: call S3A pipeline]
    B3[Collect matches; MatchKey dedup]
    B4[Conflict resolution\n(compatible match set)]
    B5[SafePeel: only core-contained matches]
    B6[Peel from A & B -> residual A', B']
    B7{Any new peel?}
    B8[Final residual S3A refine]
    B1-->B2-->B3-->B4-->B5-->B6-->B7
    B7--Yes-->B1
    B7--No-->B8
  end
```

---

8. 伪代码（读者可快速抓住“回转嵌套逻辑”）

8.1 RTB-SmallLarge(A,B,mode)

```
Reach = estimateReach(A)
SpanNeed = estimateSpanNeed(A)

if useFullCoverage:
  Pieces = genPiecesFullCoverage(B, SpanNeed, Reach)
else:
  Anchors = selectAnchors(A)
  Occs = searchOccurrences(B, Anchors)
  Occs' = gateByCCC(A,B,Occs,mode)  // ACCEPT/DEFER/REJECT
  Pieces = expandAndMerge(Occs', SpanNeed, Reach)

Pieces = cheapLBFilter(Pieces, thresholdK)
Matches = []
for P in Pieces:
  Matches += BTP(A, B restricted to P)

Matches = dedupByMatchKey(Matches)
return topK(Matches)
```

8.2 RTB-LargeLarge(A,B,mode)

```
ResidualA = A; ResidualB = B
AllMatches = []

repeat:
  Blocks = cutIntoBlocks(ResidualA)   // each block carries core/halo
semantics
  CandMatches = []
  for block in Blocks:
    CandMatches += RTB-SmallLarge(block, ResidualB, mode)

  CandMatches = dedupByMatchKey(CandMatches)
  Compatible = resolveConflicts(CandMatches)

  Peelable = [m in Compatible where safePeel(m)]
  if Peelable empty: break

  peel(Peelable, ResidualA, ResidualB)
  AllMatches += Peelable

until no progress

TailMatches = RTB-SmallLarge(ResidualA, ResidualB, mode)
return merge(AllMatches, TailMatches)
```

---

9. 与 ITEM #227 的对应关系（读者导航）

- ITEM #227：定义 EXACT/RECALL 两种模式与 Anti-Leak Contracts（护栏）
- ITEM #228：给出在护栏之内的 **RTB 加速算法全流程**（怎么跑）
- 实现顺序建议：
    1. 先实现 4A（最稳基线），跑通 EXACT；
    2. 再加入 4B 的 anchor + CCC gate（先 DEFER/排序，后 RECALL）；
    3. 最后实现 5A/5B peeling，并用 JUnit5 类高危点回归护栏锁死。

---

10. Closing

RTB 的价值不是“某个小技巧提速 30%”，而是把 UnalignedAND-BTP 这种顶级复杂内核，放进一个 **可控、可证明、可迭代演化** 的鱼控框架里：

- **先不漏，再提速**
- **先护栏，再优化**

- **先骨架，再填肉**

这正是 DBM 在复杂度量空间上做工程落地与科学可重复的关键路径。
