# **Pre-Coding Essentials (Component: crates/vm\_algo/src/mmp.rs, Version/FormulaID: VM-ENGINE v0) — 46/89**

## **1) Goal & Success**

Goal: Mixed-Member Proportional (MMP) helpers: compute **target seats** from vote totals, derive **deficits/top-ups** against local seats, and handle **overhang** per policy.

Success: Pure integer/rational math; deterministic results; respects params (mlc\_topup\_share\_pct, target\_share\_basis, mlc\_correction\_level, overhang\_policy, total\_seats\_model). Outputs sum to the intended total under the chosen policy.

## **2) Scope**

In scope: seat-target apportionment from vote shares; top-up computation; minimal iterative expansion when total seats must grow to satisfy overhang policy.

Out of scope: reading ballots/locals (caller passes counts), PR within units (other modules), reporting.

## **3) Inputs → Outputs**

Inputs:

vote\_totals: BTreeMap<OptionId, u64> (party/national list votes)

local\_seats: BTreeMap<OptionId, u32> (already awarded “local” seats)

base\_total\_local: u32 (sum of local seats across correction scope)

params: &Params (reads **VM-VAR-013..017**, 015 fixed to natural\_vote\_share)

method\_for\_targets: AllocationMethod (e.g., Sainte-Laguë or D’Hondt) for **target seat apportionment**

correction\_level: country or region (affects which totals you pass per call)

Outputs:

TargetSeats: BTreeMap<OptionId,u32> (apportioned total seats per option at scope)

TopUps: BTreeMap<OptionId,u32> where topup = max(0, target - local)

FinalSeatTotals: BTreeMap<OptionId,u32> (local + topups)

effective\_total\_seats: u32 (after any expansion)

## **4) Entities/Tables (minimal)**

## **5) Variables (used here)**

## **6) Functions (signatures only)**

rust

CopyEdit

use std::collections::BTreeMap;

use vm\_core::{

ids::OptionId, variables::{Params, AllocationMethod, OverhangPolicy, TotalSeatsModel},

rounding::{ge\_percent\_half\_even},

};

pub struct MmpOutcome {

pub targets: BTreeMap<OptionId, u32>,

pub topups: BTreeMap<OptionId, u32>,

pub finals: BTreeMap<OptionId, u32>,

pub effective\_total\_seats: u32,

pub overhang\_by\_option: BTreeMap<OptionId, u32>,

}

/// Compute the intended total seat count given local seats and a top-up share %.

/// If share = s%, total ≈ local / (1 - s). Uses half-even when rounding.

pub fn compute\_total\_from\_share(local\_total: u32, topup\_share\_pct: u8) -> u32;

/// Apportion total seats to options from vote totals using the chosen method.

/// (Typically Sainte-Laguë for proportional targets.)

pub fn apportion\_targets(

total\_seats: u32,

vote\_totals: &BTreeMap<OptionId, u64>,

method: AllocationMethod,

) -> BTreeMap<OptionId, u32>;

/// Given targets and local seats, compute top-ups and apply overhang policy.

/// May expand total seats if policy demands (see params.VM-VAR-014/017).

pub fn compute\_topups\_and\_apply\_overhang(

targets: &BTreeMap<OptionId, u32>,

local\_seats: &BTreeMap<OptionId, u32>,

overhang\_policy: OverhangPolicy,

total\_seats\_model: TotalSeatsModel,

method\_for\_targets: AllocationMethod,

vote\_totals: &BTreeMap<OptionId, u64>,

) -> MmpOutcome;

/// One-shot convenience orchestrator for a correction scope (country or region).

pub fn mmp\_correct(

vote\_totals: &BTreeMap<OptionId, u64>,

local\_seats: &BTreeMap<OptionId, u32>,

params: &Params,

method\_for\_targets: AllocationMethod,

) -> MmpOutcome;

## **7) Algorithm Outline (implementation plan)**

**Total from top-up share**

Let L = Σ local\_seats, s = VM-VAR-013 / 100. Intended **total** T = round\_half\_even( L / (1 - s) ).

Guard: if s = 0, T = L; if s = 100, invalid (reject in params validation).

**Target apportionment**

Apportion T seats to options from vote\_totals using method\_for\_targets (default recommended: Sainte-Laguë).

Deterministic: options ordered by (order\_index, id); all math integer; quotient comparisons via cross-multiplication.

**Top-up deficits**

For each option i: deficit\_i = max(0, target\_i - local\_i).

overhang\_i = max(0, local\_i - target\_i) (diagnostic).

**Overhang policy + total model**

**allow\_overhang:** keep T for targets; set topup\_i = deficit\_i. Final totals = local + topup. Effective total may **exceed T** by Σ overhang; report that delta.

**compensate\_others:** keep overall seats fixed at T. Set topup\_i = deficit\_i for non-overhang options; if Σtopups > T - L, **scale by discrete apportionment**: re-apportion the available top-up seat pool across non-overhang options by vote share (or by deficit\_i weights), using the same method\_for\_targets. Overhang options get zero top-ups; others may not fully reach target.

**add\_total\_seats:** expand total seats minimally so that after apportionment target\_i >= local\_i for all i. Algorithm:

Start with T0 = T. While ∃ i with target\_i(Tk) < local\_i, set Tk+1 = Tk + 1, recompute targets; stop when all target\_i >= local\_i.

Then topup\_i = target\_i - local\_i; effective\_total = Tk. (This is the standard “expanding house size to clear overhang”.)

**Assemble outcome**

finals\_i = local\_i + topup\_i. Store overhang\_by\_option. Return MmpOutcome.

**Correction level**

If VM-VAR-016 = region, callers run this per region and later aggregate; if country, run once nationally. (Library is agnostic—just operate on the passed maps.)

## **8) State Flow**

Pipeline: after **ALLOCATE (locals)** and **AGGREGATE** to correction scope, call mmp\_correct to compute top-ups; then continue to gates/frontier and packaging.

## **9) Determinism & Numeric Rules**

Integer/rational math only; **half-even** rounding only where specified (total-from-share step).

All apportionment uses stable option ordering; no RNG is used in MMP.

## **10) Edge Cases & Failure Policy**

L = 0 with s > 0: T = 0 (no seats to apportion) → all zeros.

vote\_totals sum to 0: apportion returns zeros; all top-ups zero; overhang may exist only if locals > 0 (then **allow\_overhang** yields finals=locals; **compensate\_others** gives no top-ups; **add\_total\_seats** expands until target>=local which may require large growth—guard with sane cap in params or fail if exceeding limit).

Options appearing in local\_seats but not in vote\_totals: treat votes=0. Options with votes but no locals get pure top-ups.

Ensure Σ finals equals L + Σ topups and matches intended effective total as per policy.

Protect against overflow: use u128 for intermediate products (e.g., L \* 100).

## **11) Test Checklist (must pass)**

**Baseline:** L=100, s=30% → T≈143 (half-even); Sainte-Laguë apportioning of T with simple votes; deficits compute; totals consistent.

**Overhang allow:** party X local=60, target=50 ⇒ overhang=10; finals→X=60; effective total > T by 10.

**Compensate others:** same inputs with compensate\_others keep total at T; verify non-overhang parties’ top-ups are re-apportioned and Σ finals = T.

**Add total seats:** iterative growth yields first Tk where all target>=local; verify minimality (dropping Tk-1 violates some target>=local).

**Zero votes:** all targets/top-ups zero; finals=locals under allow\_overhang; others per policy.

**Determinism:** permuting input map orders yields identical outcomes.