## I. RELAXED-CBS PROBLEM STATEMENT, NOTATION, AND FORMULATION

### A. Environment

We are given a directed graph of convex regions

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}),$$

where each vertex $v \in \mathcal{V}$ corresponds to a convex free-space region

$$\mathcal{R}_v = \{x \in \mathbb{R}^d : A_v x \le b_v\},$$

with $A_v \in \mathbb{R}^{m_v \times d}$ and $b_v \in \mathbb{R}^{m_v}$. Directed edges $e = (u, w) \in \mathcal{E}$ encode admissible transitions between adjacent regions. The region graph is assumed fixed; its construction (e.g., by IRIS) is outside the scope of this section.

### B. Robots

We consider $N$ robots indexed by $i \in \mathcal{I} = \{1, \dots, N\}$. Each robot has:

$$x^i_{\text{start}},\ x^i_{\text{goal}} \in \mathbb{R}^d, \qquad v^i_{\text{start}},\ v^i_{\text{goal}} \in \mathbb{R}^d,$$

and must move from its start region $s_i$ to its goal region $g_i$ while remaining dynamically feasible, collision-free, and coordinated with the team.

### C. Trajectories

Within each region $v$, robot $i$ follows an order-$\kappa$ Bézier segment with control-point index $k \in \{0, \dots, \kappa\}$:

- position control points $r^i_{v,k} \in \mathbb{R}^d$,
- time control points $h^i_{v,k} \in \mathbb{R}_{\ge 0}$.

Using Bernstein basis functions $B^\kappa_k(s)$, the local segment is

$$r^i_v(s) = \sum_{k=0}^{\kappa} B^\kappa_k(s)\, r^i_{v,k}, \qquad h^i_v(s) = \sum_{k=0}^{\kappa} B^\kappa_k(s)\, h^i_{v,k},$$

for $s \in [0, 1]$.

### D. Decision Variables

The relaxed-CBS method begins from the same unified GCS+STALC formulation used by the full mixed-integer planner, then relaxes routing variables before candidate extraction and CBS repair. Table I summarizes the main variables.

TABLE I

MAIN VARIABLES IN THE RELAXED-CBS PIPELINE

| Variable | Type | Description |
|---|---|---|
| $y^i_v$ | Binary or $[0,1]$ | Robot $i$ visits region $v$ |
| $y^i_e$ | Binary or $[0,1]$ | Robot $i$ traverses edge $e = (u, w)$ |
| $z^i_v$ | Continuous | Perspective-scaled trajectory variables at region $v$ |
| $\delta^i_v$ | Continuous $\ge 0$ | Cumulative delay of robot $i$ at region $v$ |
| $b^{ij}_v$ | Binary or $[0,1]$ | Both robots $i, j$ visit region $v$ |
| $s^{ij}_T(v)$ | Binary or $[0,1]$ | Temporal separation active at region $v$ |
| $s^{ij}_S(v)$ | Binary or $[0,1]$ | Spatial separation active at region $v$ |
| $\beta^{ij}_v$ | Binary or $[0,1]$ | Local temporal ordering for pair $(i, j)$ at region $v$ |
| $\sigma^{ij}_{v,d}$ | Binary or $[0,1]$ | Spatial separation direction indicator |
| $T_{\max}$ | Continuous $\ge 0$ | Makespan |
| $\mathcal{P}^i$ | Finite set | Candidate discrete paths for robot $i$ |
| $P$ | Tuple | One candidate path per robot |
| $d_i$ | Continuous $\ge 0$ | CBS-added start delay for robot $i$ |

**Notation convention:** Superscripts denote robot indices. Subscripts denote regions, edges, or control-point indices. Calligraphic symbols denote sets.

### E. Problem Statement

We seek a set of continuous trajectories for all robots that minimizes trajectory cost, coordination cost, and makespan while satisfying:

1) region containment and trajectory continuity,
2) velocity and boundary conditions,
3) path connectivity from $s_i$ to $g_i$,
4) STALC traversal and overwatch structure,
5) inter-robot collision avoidance,
6) and, in the relaxed-CBS approximation, recoverability of a feasible rounded joint solution through candidate extraction, reduced re-solves, and CBS delay repair.

## II. ROOT RELAXATION

### A. Per-Robot GCS Constraints

For each robot $i$, the root formulation imposes the standard GCS constraints.

**Region containment.**

$$A_v\, r^i_{v,k} \le b_v\, y^i_v \qquad \forall v \in \mathcal{V},\ k \in \{0, \dots, \kappa\}. \tag{1}$$

**Monotone time parameterization.**

$$h^i_{v,k+1} - h^i_{v,k} \ge \varepsilon\, y^i_v \qquad \forall v,\ k \in \{0, \dots, \kappa - 1\}. \tag{2}$$

**Velocity bounds.**

$$v_{\min,d}\, \Delta h^i_{v,k} \le r^i_{v,k+1,d} - r^i_{v,k,d} \le v_{\max,d}\, \Delta h^i_{v,k} \tag{3}$$

for all dimensions $d$, regions $v$, and $k \in \{0, \dots, \kappa - 1\}$.

**Boundary continuity.** For any active edge $(u, w) \in \mathcal{E}$, the end of region $u$ must match the start of region $w$ up to the selected continuity order:

$$r^i_{u,\kappa} = r^i_{w,0}, \qquad h^i_{u,\kappa} = h^i_{w,0}. \tag{4}$$

**Flow conservation.**

$$\sum_{e \in \text{in}(v)} y^i_e = y^i_v = \sum_{e \in \text{out}(v)} y^i_e \qquad \forall v \notin \{s_i, g_i\}. \tag{5}$$

Source and goal nodes obey the corresponding one-in/one-out boundary flow conditions.

**Boundary conditions.**

$$r^i_{s_i,0} = x^i_{\text{start}}, \qquad r^i_{g_i,\kappa} = x^i_{\text{goal}}, \tag{6}$$

with start and goal velocities enforced through first-difference constraints.

### B. Perspective Reformulation

The binary-continuous product $z^i_v = y^i_v \cdot x^i_v$ is handled by the standard GCS perspective reformulation. In the relaxed root, $y^i_v$ and $y^i_e$ are allowed to lie in $[0, 1]$, so the root solve may distribute flow over multiple corridors rather than choosing a single discrete route.

### C. Trajectory Cost

The continuous trajectory cost for robot $i$ is

$$L^i_{\text{GCS}} = \sum_{v \in \mathcal{V}} \left( w_r \sum_{k=0}^{\kappa-1} \left\| r^i_{v,k+1} - r^i_{v,k} \right\|_2 + w_h \sum_{k=0}^{\kappa-1} \left| h^i_{v,k+1} - h^i_{v,k} \right| \right). \tag{7}$$

## III. STALC COORDINATION COSTS

### A. Traversal Cost with Formation Incentives

For each edge $e \in \mathcal{E}$, define the relaxed team occupancy

$$n_e = \sum_{i \in \mathcal{I}} y_e^i. \tag{8}$$

The piecewise-linear traversal cost is

$$C_{\text{trav}}(e, n_e) = \begin{cases} 0 & \text{if } n_e = 0, \\ \bar{w}_e + m_e(a_e - n_e) & \text{if } 0 < n_e \le a_e, \\ \bar{w}_e - r_e(n_e - a_e) & \text{if } n_e > a_e, \end{cases} \tag{9}$$

where $\bar{w}_e$ is the base traversal cost, $m_e$ is the shortage penalty, $a_e$ is the desired team occupancy threshold, and $r_e$ is the teaming reward.

### B. Overwatch Benefit

Each overwatch opportunity $o$ consists of:

- a danger edge $e_o$,
- a set of overwatch regions $\mathcal{V}_{\text{ow}}^o \subseteq \mathcal{V}$,
- a benefit scale $\omega_o$,
- a saturation threshold $\alpha_o$,
- an optional post-threshold slope $\gamma_o$.

Define the relaxed number of overwatch providers as

$$\rho_o = \sum_{i \in \mathcal{I}} \sum_{v \in \mathcal{V}_{\text{ow}}^o} y_v^i. \tag{10}$$

The piecewise-linear overwatch term is

$$C_{\text{ow}}(o, \rho_o) = \begin{cases} 0 & \text{if } \rho_o = 0, \\ -\frac{\omega_o}{\alpha_o}\rho_o & \text{if } 0 < \rho_o \le \alpha_o, \\ -\omega_o - \gamma_o(\rho_o - \alpha_o) & \text{if } \rho_o > \alpha_o. \end{cases} \tag{11}$$

### C. *[Overwatch Timing Constraints]*

The overwatch reward is not purely spatial. It is coupled to the time variables so that coverage lasts for the full traversal of the danger edge. For each overwatcher–crosser pair $(i, j)$ associated with opportunity $o$ and its danger edge $e_o = (u, w)$, the root formulation enforces three Big-M constraints:

$$t_{\text{enter}}^i(v_{\text{ow}}) \le t_{\text{enter}}^j(u) + M(1 - \eta_o^{ij}), \tag{12}$$

$$t_{\text{exit}}^i(v_{\text{ow}}) \ge t_{\text{enter}}^j(u) - M(1 - \eta_o^{ij}), \tag{13}$$

$$t_{\text{exit}}^i(v_{\text{ow}}) \ge t_{\text{exit}}^j(w) - M(1 - \eta_o^{ij}), \tag{14}$$

where $v_{\text{ow}} \in \mathcal{V}_{\text{ow}}^o$ is the active overwatch region and $\eta_o^{ij}$ is the corresponding activation indicator. Intuitively:

1) the overwatcher must arrive before the crosser enters the danger edge,
2) the overwatcher must still be present when the crossing begins,
3) the overwatcher must remain until the crosser exits the danger corridor.

Thus the overwatch benefit activates only when spatial occupancy and temporal coverage are both satisfied.

### D. Makespan

The root solve also includes

$$L_{\text{makespan}} = \lambda T_{\max}, \qquad T_{\max} \ge h_{g_i,\kappa}^i + \delta_{g_i}^i \quad \forall i \in \mathcal{I}. \tag{15}$$

## IV. TEMPORAL-SPATIAL COLLISION AVOIDANCE

### A. Shared-Region Visit Indicator

For each robot pair $(i, j)$ with $i < j$ and each region $v$, define

$$b_v^{ij} \le y_v^i, \qquad b_v^{ij} \le y_v^j, \qquad b_v^{ij} \ge y_v^i + y_v^j - 1. \tag{16}$$

### B. Temporal or Spatial Separation

If both robots visit region $v$, the model must activate either temporal or spatial separation:

$$s_T^{ij}(v) + s_S^{ij}(v) \ge b_v^{ij}. \tag{17}$$

### C. Effective Occupancy Times

Let

$$t_{\text{enter}}^i(v) = h_{v,0}^i + \delta_v^i, \qquad t_{\text{exit}}^i(v) = h_{v,\kappa}^i + \delta_v^i. \tag{18}$$

The cumulative delays $\delta_v^i$ are monotone along the chosen path.

### D. Temporal Separation

If temporal separation is active, a local ordering variable $\beta_v^{ij}$ chooses which robot goes first:

$$t_{\text{exit}}^i(v) + \varepsilon \le t_{\text{enter}}^j(v) + M(1 - s_T^{ij}(v)) + M(1 - \beta_v^{ij}) + M(1 - b_v^{ij}), \tag{19}$$

$$t_{\text{exit}}^j(v) + \varepsilon \le t_{\text{enter}}^i(v) + M(1 - s_T^{ij}(v)) + M\beta_v^{ij} + M(1 - b_v^{ij}). \tag{20}$$

### E. Delay Propagation

For each robot $i$ and edge $(u, w)$,

$$\delta_w^i \ge \delta_u^i - M(1 - y_{(u,w)}^i), \qquad \delta_v^i \le M y_v^i. \tag{21}$$

This ensures waiting introduced earlier in the path propagates to later regions.

### F. Spatial Separation

If spatial separation is chosen, the model projects control points onto candidate separation directions and enforces a minimum gap. Let

$$p_{v,k,d}^i = n_{v,d}^\top r_{v,k}^i \tag{22}$$

be the projection onto candidate direction $n_{v,d}$. Define the projection envelopes

$$\max_{v,d}^{i} = \max_k p_{v,k,d}^i, \qquad \min_{v,d}^{i} = \min_k p_{v,k,d}^i. \tag{23}$$

Then, for an active direction indicator $\sigma_{v,d}^{ij}$, the planner enforces

$$\max_{v,d}^{i} - \min_{v,d}^{j} + d_{\min} \le M_d(1 - \sigma_{v,d}^{ij}) + M_d(1 - b_v^{ij}), \tag{24}$$

or the symmetric reverse ordering.

### G. Cross-Edge Temporal Avoidance

If enabled, the formulation also protects against head-on or crossing conflicts on adjacent transitions. For edge $(u, w)$ and pair $(i, j)$, define a cross-edge visit indicator and a local ordering variable. The same Big-M style temporal precedence constraints are applied to the effective exit time from one side and the effective entry time from the other side.

## V. COMPLETE ROOT OBJECTIVE

The relaxed root minimizes

$$\begin{aligned}\min \sum_{i\in\mathcal{I}} L^i_{\mathrm{GCS}} + \sum_{e\in\mathcal{E}} C_{\mathrm{trav}}(e, n_e) + \sum_{o} C_{\mathrm{ow}}(o, \rho_o) \\ + \lambda T_{\max} + \sum_{i,v} c_\delta\, \delta^i_v + \sum_{i,j,v} c_T\, s^{ij}_T(v),\end{aligned} \tag{25}$$

subject to the GCS constraints (1)–(6), the STALC constraints (9)–(15), and the collision-avoidance constraints (16)–(24).

## VI. RELAXED ROUNDING AND CBS REPAIR

### A. Candidate Path Extraction

After solving the relaxed root, the method reads the fractional edge flows and extracts a small candidate set

$$\mathcal{P}^i = \{P^i_1, \dots, P^i_{K_i}\}$$

for each robot. These candidates are sampled from the relaxed edge-flow graph. Extraction is biased by a heuristic risk score:

1) edges with larger relaxed flow are preferred,
2) edges passing through regions strongly used by other robots are penalized,
3) edges shared with or reversed against other robots' relaxed flow are penalized,
4) regions favored or disfavored by the relaxed collision binaries contribute additional soft penalties.

### B. Joint Candidate Combinations

The method forms joint combinations

$$P = (P^1_{k_1}, \dots, P^N_{k_N}) \in \mathcal{P}^1 \times \cdots \times \mathcal{P}^N.$$

These combinations are ranked heuristically before exact evaluation:

1) sum the candidate path scores across robots,
2) add any combination-level penalty implied by relaxed collision preferences,
3) keep only the most promising combinations up to a fixed cap.

### C. Reduced Exact Re-Solve

For each candidate combination, build the reduced graph

$$\mathcal{V}(P) = \bigcup_{i=1}^{N} \{v : v \text{ lies on } P^i_{k_i}\}, \qquad \mathcal{E}(P) = \bigcup_{i=1}^{N} \{e : e \text{ lies on } P^i_{k_i}\},$$

then solve the exact continuous multi-robot trajectory problem on this restricted graph. The reduced problem retains:

- the same per-robot GCS trajectory variables,
- the same STALC traversal and overwatch costs,
- the same overwatch timing constraints,
- the same temporal-spatial collision constraints,
- the same delay variables.

### D. Collision Validation and Local Repair

Each reduced solution is validated by sampling pairwise robot distances over time. If the solution is not collision-free, the method may:

1) try alternate candidate combinations,
2) locally reroute the colliding robots over different extracted paths,
3) promote specific observed local conflicts back to binary and rerun rounding.

### E. CBS Delay Repair

In the relaxed-CBS variant, any residual conflicts are finally handled by a CBS-style delay search on the extracted continuous trajectories:

1) find the earliest sampled same-region or cross-edge conflict,
2) branch on delaying one robot or the other,
3) shift that robot's entire trajectory forward in time,
4) revalidate and continue until a collision-free solution is found or the node budget is exhausted.

This final repair layer changes timing only; it does not introduce new geometric corridors.