Cost Function Design & Markov Chains with PERT

Agentic Congress Planner (ACP) - Architecture Document
Prepared by Claudia | Sunrise Gen AI | Feb 2026

Cost Function Design & Markov Chains with PERT

Part A: Cost Function Design

1. Multi-Objective Cost Function

The planner optimizes across competing objectives. The composite cost function:

C_total = w1*C_schedule + w2*C_resource + w3*C_risk + w4*C_quality + w5*C_disruption

Where weights w1..w5 are configurable per congress priority (tuned via historical outcomes).

2. Individual Cost Components

C_schedule -- Schedule Cost

```
C_schedule = SUM over all tasks [
  alpha * max(0, actual_end - due_date)^2         // Tardiness penalty (quadratic)
  + beta * max(0, due_date - actual_end)          // Earliness bonus (linear, diminishing)
  + gamma * (is_critical_path * tardiness_penalty)  // Critical path multiplier
]
```

- Quadratic tardiness: Small delays are tolerable, large delays are catastrophic (non-linear penalty)
- Critical path multiplier (gamma): Delays on CP tasks penalized 3-5x more
- MS Planner mapping: dueDateTime vs simulated completedDateTime

C_resource -- Resource Utilization Cost

```
C_resource = SUM over all assignees [
  delta * max(0, utilization - U_max)^2           // Over-allocation penalty
  + epsilon * max(0, U_min - utilization)         // Under-utilization waste
  + zeta * context_switch_count                   // Multi-tasking penalty
]
```

- Derived from: assignments field -- count concurrent tasks per person per time window
- Historical calibration: Compute each person's U_max from past congresses
- Context switch cost: Each additional parallel task adds ~20% overhead

C_risk -- Risk Exposure Cost

```
C_risk = SUM over all tasks [
  eta * P(delay > threshold) * impact_magnitude
]
```

- P(delay): From Monte Carlo simulation output

- Impact magnitude: Derived from priority field + downstream dependency count in DAG
- High fan-out tasks (many dependents) get higher impact scores

C_quality -- KOL & Speaker Alignment Quality

C_quality = SUM over all speaker assignments [
theta * topic_mismatch_penalty
+ iota * preference_violation_penalty
+ kappa * audience_overlap_penalty
]

- Topic mismatch: NLP similarity between speaker expertise and assigned topic
- Preference violations: Soft constraint from taskDetail.description
- Audience overlap: If two sessions share >40% target audience, penalize parallel scheduling

C_disruption -- Reactive Replanning Cost

C_disruption = SUM over all replan events [
lambda * cascade_depth
+ mu * human_approval_latency
+ nu * plan_delta
]

- Stability premium: Users prefer plans that don't change drastically
- Cascade depth: From dependency DAG traversal

3. Cost Function Calibration from Historical Data

1. Compute actual costs from historical congresses (delays, overruns, speaker issues)
2. Regression: fit weights w1..w5 to minimize |predicted_cost - actual_outcome_cost|
3. Output: Validated weight vector for current year planning

Part B: Markov Chains + PERT Integration

1. Why PERT Alone Is Insufficient

Standard PERT assumes:
- Beta distribution for task durations (optimistic, most likely, pessimistic)
- Independence between tasks (only structural dependency via DAG)
- Static estimates -- no state transitions during execution

Problems for congress planning:
- Task states evolve -- a task "in progress" for too long signals trouble
- External events change the probability landscape
- Resource states matter -- overloaded person affects ALL their tasks

2. Markov Chain Enhancement: State-Based Task Modeling

Each task is modeled as a Markov chain with states:

States S = {Not Started, Planning, In Progress, Blocked, Under Review, Completed, Cancelled}

Mapping to MS Planner:
- Not Started: percentComplete = 0, no assignedDateTime

- Planning: percentComplete = 0, has assignedDateTime
- In Progress: percentComplete = 50
- Blocked: percentComplete = 50 + no progress for > T_block threshold
- Under Review: percentComplete = 50 + checklist mostly done
- Completed: percentComplete = 100
- Cancelled: task deleted or marked in description

3. Transition Probability Matrix (from Historical Data)

Each cell P[i,j] represents probability of transitioning from state i to state j per time period.
Calibrated per task category, assignee historical performance, and congress phase.

Example transitions:

- Not Started -> Planning: 0.7
- Planning -> In Progress: 0.8
- In Progress -> Under Review: 0.4
- In Progress -> Blocked: 0.15
- Under Review -> Completed: 0.7
- Blocked -> In Progress: 0.6
- Completed and Cancelled are absorbing states

4. PERT-Markov Hybrid Model

Standard PERT:
  Duration ~ Beta(a, m, b) where a=optimistic, m=most_likely, b=pessimistic

Enhanced PERT-Markov:
  At each time step t:
    1. Current state s_t of each task (from Markov chain)
    2. Time spent in current state: tau
    3. Conditional duration estimate:
       D_remaining | s_t, tau ~ Updated_Beta(a', m', b')
       where a', m', b' are adjusted based on:
         - Current state (blocked tasks get inflated pessimistic)
         - Dwell time (longer in IP -> right-shift distribution)
         - Resource state (assignee overloaded -> inflate)

5. Absorbing Markov Chain Analysis

Fundamental Matrix: N = (I - Q)^(-1)
Where Q = transient state transition submatrix

N[i,j] = expected number of times process visits state j, starting from state i
Expected time to completion from state i = SUM(N[i,:])

This gives us:

- Expected task duration accounting for blocked/review loops
- Probability of cancellation vs completion from any state
- Variance of completion time (from N matrix properties)

6. Markov-Modulated PERT Network

Traditional PERT Network:

Task A (duration_A) -> Task B (duration_B) -> Task C (duration_C)

Markov-Modulated:

Task A [MC_A state] -> Task B [MC_B state] -> Task C [MC_C state]

Each task's duration is conditioned on:

1. Its own Markov chain state
2. Predecessor's terminal state (completed normally vs rushed vs partial)
3. Shared resource Markov chains (person availability state)

Dependency types become richer:

- Finish-to-Start (FS): Standard -- B starts when A completes
- State-conditional FS: B's duration changes based on HOW A completed
- Resource-coupled: A and B share assignee -> joint Markov chain

7. Combined Simulation Algorithm

```
Initialize:
  For each task: set state = Not Started
  For each task: fit PERT params (a,m,b) from historical data
  Build transition matrices per task category

For each simulation run (1..10000):
  t = 0
  While not all tasks absorbed:
    t += 1
    For each active task:
      new_state = sample(transition_matrix[current_state])

      If entering "In Progress" and not duration_sampled:
        duration = sample(Beta(a, m, b))
        If assignee.state == Overloaded:
          duration *= resource_penalty_factor

      If new_state == Blocked:
        block_time = sample(Exponential(lambda_block))
        duration += block_time

      Check completion: if time_in_progress >= duration -> transition

    Inject external disruptions at time t

  Record: total_time, critical_path, cost_function_value

Output: distributions, percentiles, risk metrics
```

8. When to Use What

- Pure PERT: Quick estimates, well-understood tasks (no state awareness)
- Pure Monte Carlo: Duration uncertainty exploration (no state transitions)
- Markov Chain: State-based progression modeling (memoryless assumption)
- PERT + Markov + MC: Full system -- state-aware, uncertainty-modeled, dependency-coupled

9. Addressing the Memoryless Limitation

Standard Markov chains are memoryless. For congress planning, dwell time matters. Solutions:

- Semi-Markov Process: Transition probabilities depend on time spent in current state
- Phase-type distributions: Represent non-exponential durations as sequences of exponential phases
- Practically: Discretize time into periods (e.g., daily). Transition matrix changes based on days_in_state -- this is a time-inhomogeneous Markov chain