数学推导过程

两阶段优化数学模型详解

1. 问题建模基础

首先，我们将一个复杂的、随机的、连续时间的现实世界问题，抽象为一个可计算的离散时间模型。

时间离散化：将机器人的运行时间划分为长度为Δt 的时隙（time slot），记为 t = 0, 1, 2, ...。这个时隙的长度选择至关重要，它需要短到能够捕捉到环境的重要变化（如突发异常），又需要长到足以完成有意义的计算和通信动作。在巡检机器人场景中，Δt 可能在 100 毫秒到 1 秒之间。

核心思想：我们的目标不是在每个时隙都做一个全新的、全局最优的决策（这计算量太大），而是**在每个时隙，基于当前状态，做出一个“足够好”的局部决策，并保证这些局部决策在长期来看是系统最优的**。

2. 系统状态与决策变量

2.1 系统状态 S(t)

系统状态是所有影响决策的环境和内部因素的集合。在时隙 t，状态包括：

C_{total}：总计算资源（如 1000 GOPS - Giga Operations Per Second）。这是一个常量，由硬件决定。

B_{total}：总通信带宽（如 100 Mbps）。这是一个常量。

$E_{remaining}(t)$：剩余电池能量（如 20000 mAh）。这是一个随时间缓慢变化的量。

$Event_{priority}(t) = [\omega_1(t), \omega_2(t), \ldots, \omega_m(t)]$：这是一个向量，是模型动态性的核心输入。其中 $\omega_m(t)$ 表示模块 m 在当前时隙的任务关键性权重。例如：

$\omega_{perception}(t)$：平时为 1，检测到疑似异常时变为 5，确认严重火灾时变为 10。

$\omega_{communication}(t)$：定时传输数据时为 1，传输报警信息时为 10。

这个向量通常由高层的“任务决策模块”根据规则或轻量级推理产生。

$Data_{Backlog}(t) = [Q_1(t), Q_2(t), \ldots, Q_m(t)]$：数据队列积压。例如 $Q_{perception}(t)$ 表示待处理的图像帧数，$Q_{communication}(t)$ 表示待发送的数据包数量。这个状态确保了当某个模块任务繁重时，它能获得更多资源。

2.2 决策变量 X(t)

这是优化问题的输出，即我们要控制的“旋钮”。

$c_m(t)$：在时隙 t 分配给功能模块 m 的计算资源。

$b_m(t)$：在时隙 t 分配给功能模块 m 的通信资源。

物理约束：

$$\sum_{m=1}^{M} c_m(t) \le C_{total} \qquad (1)$$

$$\sum_{m=1}^{M} \mathrm{b}_m(t) \le B_{total} \qquad (2)$$

这两个不等式约束是“硬约束”，意味着资源分配绝不能超出现有物理上限。

3. 性能模型与效用函数

这是连接资源投入与系统产出的桥梁，是整个优化问题的“价值导向”。

3.1 模块性能函数 $\mathrm{P}_m(x_m(t), S(t))$

这是一个将资源投入映射为性能产出的函数。它通常是凹函数（Concave Function），符合“边际效益递减”规律。例如：

感知模块：其性能可以是目标检测的**准确率（Accuracy）**。

$$\mathrm{P}_{\mathrm{perception}}(\mathrm{c}_{perception}(t)) = A_{\max}(1 - e^{-k \bullet c_{perception}(t)})$$

这里，$A_{\max}$ 是准确率上限（如 99%），k 是一个敏感度参数。当 $\mathrm{c}_{\mathrm{perception}}$ 为 0 时，性能为 0；随着算力增加，准确率快速提升并逐渐饱和。这个函数形状很好地模拟了现实：初期增加算力（如使用更复杂的模型）效果显著，但后期再翻倍算力可能只能提升零点几个百分点的准确率。

通信模块：其性能可以是**传输延迟的倒数**，因为我们希望最大化效用。

$$\mathrm{P}_{\mathrm{communication}}(\mathrm{c}_{communication}(t)) = \frac{1}{\tau_{\min} + \dfrac{D_{packet}}{b_{communication}(t)}}$$

这里，τ_{min} 是固定处理延迟，D_{packet} 是数据包大小。带宽 b 越大，传输时间 $D_{\mathrm{packet}}/\mathrm{b}$ 越小，总延迟的倒数就越大，即性能 P 越高。

3.2 系统总效用函数 U(t)

这是我们在每个时隙最终要最大化的目标。我们将其设计为各模块性能的加权和：

$$U(\mathrm{t}) = \sum_{m=1}^{M} W_m(t) \bullet P_m(x_m(t), S(t)) \qquad (3)$$

关键点：权重 $W_{\mathrm{m}}(\mathrm{t})$ 直接与系统状态中的 $\omega_{\mathrm{m}}(\mathrm{t})$（事件优先级）相关。例如，$W_{\mathrm{m}}(\mathrm{t}) = \omega_{\mathrm{m}}(\mathrm{t})$。

设计动机：这个函数完美地体现了我们的优化策略。

当某个模块 m 的优先级 $\omega_{\mathrm{m}}(\mathrm{t})$ 很高时（如通信模块需要报警），它在效用函数 U(t) 中的“话语权”就变重。此时，优化算法会倾向于将更多的资源（$c_{\mathrm{m}}(\mathrm{t})$，$\mathrm{b}_{\mathrm{m}}(\mathrm{t})$）分配给这个模块，因为这样做能最有效地提升 U(t) 的总值。反之，一个低优先级的模块，即使剥夺其部分资源对 U(t) 的负面影响也很小，这些省下来的资源就可以分配给更关键的任务。

4. 两阶段优化的数学分解

现在我们来详细讲解如何将这个复杂的随机优化问题分解为两个可求解的阶段。

4.1 第一阶段：静态预算分配（离线规划）

核心思想：忽略环境的瞬时波动，在一个“平均的、典型的”场景 Ξ 下，求一个一劳永逸的、固定的资源预算方案 X_{budget}。这个方案可能不是任何时候都最优，但它是一个安全、稳定且计算高效的基线。

数学表述：

$$\max_{X_{budget}} \quad E_{\lambda\sim\Xi}[U(X_{\text{budget}}, S_\lambda)] \tag{4}$$

s.t.

$$\sum_m c_{\text{m}}^{budget}(t) \le C_{total} \qquad (4a)$$

$$\sum_m \text{b}_{\text{m}}^{budget}(t) \le B_{total} \qquad (4b)$$

$$E_\lambda[\sum_m (\kappa(c_{\text{m}}^{budget})^3 + \eta b_m^{budget}] \le \bar{P}_{avg} \qquad (4c)$$

公式详解：

目标函数 (4)：E[...] 表示数学期望。我们的目标是让这个资源预算 X_{budget} 在所有可能出现的典型场景 λ 上的平均效用达到最大。这是一个随机规划（Stochastic Programming）问题。

约束 (4a), (4b)：与在线约束相同，确保预算总和不超过资源总量。

约束 (4c)：这是一个长期平均功耗约束。$\kappa(\text{c}_m^{budget})^3$ 是计算功耗（经典模型），ηb_m^{budget} 是通信功耗。这个约束确保了按此预算运行，机器人的平均功耗不会超过一个安全阈值 $\bar{P}_{\text{avg}}$，从而保证足够的续航时间。这是第一阶段考虑长期行为的体现。

求解方法：

1、场景生成：收集大量历史运行数据，或通过仿真生成成千上万个不同的典型场景 $\{\lambda_1, \lambda_2, \ldots, \lambda_{\text{k}}\}$。

2、近似：用这 k 个场景的平均值来近似数学期望 E。问题转化为一个确定的非线性规划问题：

$$\max \frac{1}{K}\sum_{k=1}^{K} U(X_{budget}, S_{\lambda_k})$$

3、**求解**：由于X_{budget}是固定的，且问题规模可控，可以使用标准的凸优化求解器（如 IPOPT、CVXPY）或进化算法进行求解，得到全局最优的静态预算X^*_{budget}。

4.2 第二阶段：动态资源调配（在线调整）

核心思想：在每个时隙 t，我们不再重新决定所有资源，而是在第一阶段的最优基线 X^*_{budget}上进行“微调”。这极大地降低了问题的复杂度。

数学表述：

$$\max_{\Delta X(t)}[U(X^*_{budget}+\Delta X(t),S(t))] \qquad (5)$$

s.t.

$$\sum_{m}(c_m^{budget}+\Delta c_m(t))\le C_{total} \qquad (5a)$$

$$\sum_{m}(b_m^{budget}+\Delta b_m(t))\le B_{total} \qquad (5b)$$

$$|\Delta c_m(t)|\le \delta_c,\qquad |\Delta b_m(t)|\le \delta_b \qquad (5c)$$

$$\Delta c_m(t)\ge 0 \quad if \quad \omega_m(t)\ge\theta \qquad (5d)$$

公式详解：

决策变量 (5)：ΔX(t) 是调整量。新的资源分配为 X(t) = X^*_{budget}+ ΔX(t)。我们最大化的是当前时隙 t 的瞬时效用。

约束 (5a), (5b)：微调后的总资源依然不能超标。

约束 (5c)：这是核心约束。δ_c和δ_b 是调整幅度的上限。它保证了：

1、**决策空间小**：优化算法只需要在一个以 X^*_{budget}为中心、边长为 2δ 的小超立方体内搜索，计算速度极快。

2、**系统稳定**：防止资源分配在相邻时隙发生剧烈跳变，避免对软件模块造成冲击。

约束 (5d)：这是一个优先级保障约束。θ 是一个优先级阈值。它规定：如果一个模块的实时优先级 $\omega_m(t)$ 非常高（超过 θ），那么给它分配的资源只能增加，不能减少（$\Delta c_m(t) \ge 0$）。这确保了关键任务（如报警）绝不会因为优化过程而被剥夺资源，实现了软硬约束的结合。

求解方法：

这个问题是一个小规模的、带约束的非线性优化问题。由于搜索空间被严格限制，且效用函数通常是光滑的，可以采用非常高效的算法：

梯度投影法：计算效用函数 U 关于 ΔX 的梯度，然后沿着梯度方向（最速上升方向）走一小步，再将结果投影回由约束(5a)-(5d)构成的可行域内。迭代几次即可收敛。

拉格朗日对偶法：将约束(5a)和(5b)通过拉格朗日乘子放入目标函数中，将原问题转化为一个无约束的对偶问题，然后用梯度上升法求解乘子。

5. **两阶段如何协同：一个总结**

第一阶段（宏观、慢速、离线）：

输入：历史数据、物理约束、长期目标（续航）。

过程：解决一个复杂的随机规划问题。

输出：一个全局的、稳健的资源配置基线 X^*_{budget}。

作用：定下“基调”，保证系统长期稳定和高效。

第二阶段（微观、快速、在线）：

输入：第一阶段基线 X^*_{budget}、实时系统状态 S(t)。

过程：解决一个简单的、局部的、带约束的瞬时优化问题。

输出：当前时隙的微调量 ΔX(t)。

作用：实现“精准打击”，应对瞬时波动和突发事件。

这个框架的精妙之处在于，它通过**第一阶段将复杂问题“简易化”**，从而使得第二阶段的**实时决策变得可行且高效**，最终在理论和实践上都实现了在资源受限环境下，系统性能的持续最优化。