Factory-simulator：用于依据基于优化的参照评估制造决策的流程导向环境

Andrzej Wodeckia、Wojciech Godlewski b*、Michał Chojnac

a 华沙理工大学商学院，科希科瓦街79号，02-008 华沙，波兰

b 华沙理工大学博士生院，理工学院广场1号，00-661 华沙，波兰

c 波兰-日本信息技术学院，科希科瓦街86号，02-008 华沙，波兰 * 通讯作者：
Wojciech Godlewski。电子邮箱：wojciech.godlewski4.dokt@pw.edu.pl

摘要

本研究源于与一家电子产品合同制造商关于决策支持的讨论。Factory-simulator 提供了一个确定性制造环境，用于对基于大语言模型的多智能体系统（LLM-MAS）的业务绩效开展受控研究。相同的交付绩效可能来自财务后果不同的决策，因此仅通过交付指标进行评估颇为复杂。其面向流程的模型将采购和生产决策与订单履行及工厂财务联系起来。预设种子的世界和记录的行动支持从共享历史进行回放与分叉，以考察替代决策及其下游后果。线性规划和混合整数规划模型可生成以利润为导向的参考决策。观测受限的比较对象使用已披露的信息；完全预见参考方案则利用未来事件进行诊断性比较。受控示例表明，在信息相同的情况下，交付绩效相同也可能对应不同利润，并且可以区分交付罚金与延期付款。验证内容包括物料与财务平衡、回放和分叉，以及一个小型分配实例；在该实例中，受测的优化控制器达到了通过分析确定的最优解。一个外部三角色大语言模型多智能体系统客户端完成了四次示例运行，共计159个周期，余额核对无误，且记录的行动可被精确回放。匹配世界滚动比较为财务结果提供了背景参照，而完全预见结果提供的是启发式参考点，并非经认证的最优解。该软件使研究人员能够探究智能体组织方式和信息获取权限如何影响业务结果，包括分配采购与生产职责是否会改变相对于集中式控制器的利润。

关键词：基于大语言模型的多智能体系统；业务绩效；确定性制造仿真；基于优化的参考方案；流程管理

必需元数据

Nr	代码元数据描述	元数据
C1	当前代码版本	1.0.0
C2	代码/代码库永久链接 用于此代码版本	https://github.com/mchojna/factory-simulator/tree/ea92c1bfb28228c8929045e1ddcfa9535da2ea27 1.0.0 版本发布；历史源身份与复现 数据见 S1节 .
C3	代码法律许可	MIT
C4	使用的代码版本控制系统	Git
C5	使用的软件编程语言、工具 和服务	Python；FastAPI；SQLAlchemy；DuckDB；NumPy； SciPy/HiGHS
C6	编译要求， 运行环境和 依赖项	Python >=3.14、<3.15；uv >=0.12.10、<0.14。本地 验证：macOS arm64、Python 3.14.10。托管式 持续集成：Ubuntu 测试；macOS/Windows 冻结世界 检查。锁定的依赖项和范围：章节 S1.1–S1.2.
C7	如有，请链接至开发者 文档/手册	https://github.com/mchojna/factory-simulator/tree/ea92c1bfb28228c8929045e1ddcfa9535da2ea27/docs
C8	问题咨询支持邮箱	Wojciech Godlewski; wojciech.godlewski4.dokt@pw.edu.pl

1. 研究动机与意义

近期基准测试评估了大语言模型（LLM）智能体在库存管理和业务运营中的表现 [1,2]。电子制造需要协调采购、生产与交付承诺；半导体供应中断可能沿供应网络扩散 [3]。提前采购或许能保障交付，但也会占用现金。将采购和生产分配给不同的智能体角色，会在这些角色之间形成接口，而相关决策仍相互关联。因此，评估基于大语言模型的多智能体系统（LLM-MAS）[4] 需要追踪其跨职能后果。InvAgent 将大语言模型智能体应用于库存订购 [5]；在 Jannelli 等人的实验中，牛鞭效应最低的配置并非成本最低的配置 [6]。Factory-simulator 将工厂动态与智能体组织方式分开，使研究能够在固定工厂模型、已实现世界、评估目标及时域的同时，改变协调方式和信息获取权限。

此处的流程导向是指跨相互关联的业务职能进行评估 [7]。局部职能优化可能导致整体次优 [8, Chapter 1]。Galbraith [9] 将缓冲库存等冗余资源与额外的信息处理能力区分开来，认为它们都是应对不确定性的方式，但各自会产生不同的成本。这些框架为设计提供了参考：Factory-simulator 会记录库存承诺和仓储成本；研究可以改变披露信息和智能体组织方式。

相关环境采用了不同的参考方案。OR-Gym 将强化学习与优化和启发式方法进行比较 [10,11]。AIM-Bench 使用事后补货基准，并考察信息共享 [2]。InventoryBench 报告基准库存、大语言模型和混合方法的归一化奖励 [12,13]。EconEvals 将基于模型的绩效基准与轨迹分析相结合 [14,15]。schlably 框架通过深度强化学习、调度启发式方法和求解器比较支持生产调度研究 [16]。表1记录了优化参考和信息干预方面的先例。Factory-simulator 将这些能力与包含应收账款的利润以及从记录历史分支的功能相结合。

表*1.* 相关环境与 *Factory-simulator* 中的评估支持。所选环境涵盖生产、供应链和业务运营任务。条目描述有文档记录的能力；未报告的能力不视为不存在。信息对等因协议而异。*S2*节提供了来源和更广泛的比较。

环境	参考或评估目标	信息、控制与诊断
OR-Gym [10,11]	混合整数线性规划和启发式方法；库存任务 递缩时域线性规划和完美— 信息预言机；任务奖励。	库存状态和动作历史；任务— 特定的优化模型。
AIM-Bench [2]	事后补货基准； 订货偏差和库存 成本。	多智能体设置和信息共享 处理；周期级订货行为。
InventoryBench [12,13]	带上限的基库存基线； 运筹学、大语言模型和 混合方法的归一化奖励。	预生成需求和提前期； 历史输入和上下文输入； 推荐或预测流程。
Vending-Bench [1]	长期运行的业务运营； 最终净资产和单次运行 人类基线。	智能体轨迹、工具使用和模拟的 供应商通信。

环境	参考或评估目标	信息、控制与诊断
schlably [16]	深度强化学习智能体与……进行比较 调度启发式方法和求解器； 奖励和与求解器 解的差距。	生成的调度实例、Gym 环境、记录的结果和甘特 图。
SCML [17,18]	供应链世界中谈判智能体的 比较。	可配置上下文与受控世界 条件下的行为比较。
MLPro-MPPS [19,20]	用户配置的生产系统 用于学习和控制。	传感器和组件状态；集成 强化学习与博弈论 。
prodsys [21]	生产配置； 吞吐量、吞吐时间、在制品 和重新配置成本。	带种子的仿真和存储事件，用于 后处理。
SimPROCESD [22]	维护和生产 策略；净生产利润和 质量。	设备状况和资源使用 指标。
Factory-simulator（本研究）	基于线性规划/混合整数线性规划的比较器；完整的 前瞻诊断参考方案； 包含应收款的规划期利润 以及库存残值回收。	已记录世界和历史分叉； 可配置的信息披露，访问权限 由控制框架强制执行；共享材料 和财务核算。

SupChain-Bench 根据真实值 [23,24]评估知识和工具介导的检索。

通用引擎对提交的策略应用相同的规则。研究人员可以追溯相同服务背后的成本，并将利润影响与付款时机区分开来（第 3 节）。

2. 软件描述

2.1 软件架构

引擎负责状态转换和会计核算（图1）。HTTP服务提供 JSON API，并持久化存储世界和运行记录。基准测试层提供基于线性规划/混合整数线性规划的比较器、诊断参考和经济评分。随附的单提示、工具引导式和协同客户端可扩展为针对特定研究的角色。引擎为每次运行提供一个控制器接口。诸如 mango 之类的通用智能体框架提供角色、消息传递以及与外部模拟器的耦合功能 [25]；Factory-simulator 则将智能体组织方式留给外部控制框架。

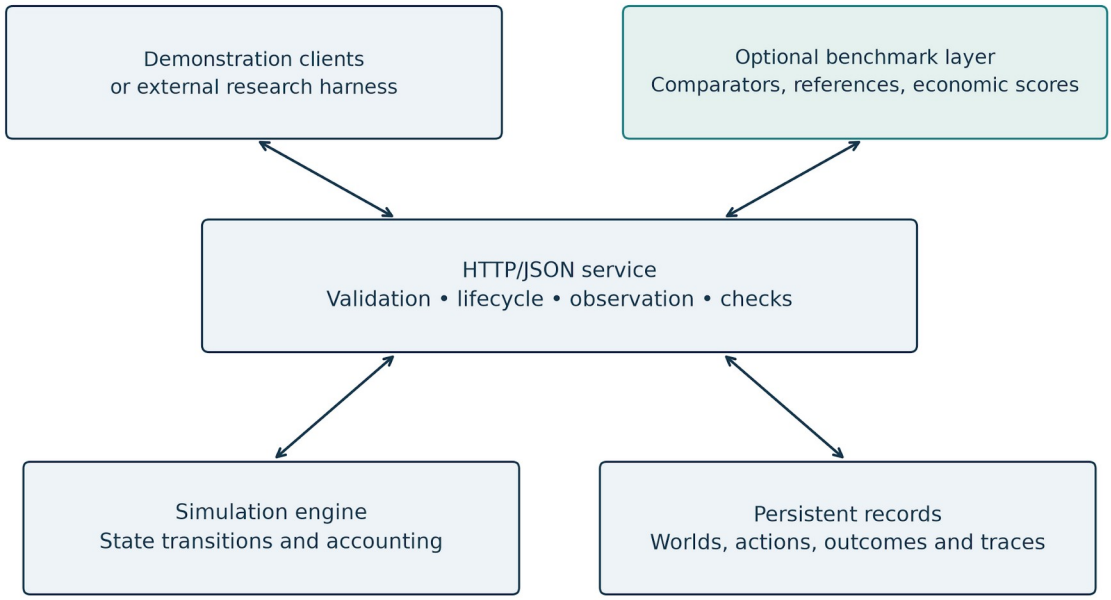

图*1.* 支持决策评估的架构。演示客户端与引擎配套；外部研究控制框架负责定义研究并控制访问。绘图代码由 *Codex*（*OpenAI*，*GPT-6 Astra*）协助编写，使用 *Python/Matplotlib*。

清单在模拟器的建模假设范围内定义工厂实体、配方和参数分布；情景则指定初始条件、时域和扰动。随机种子决定已实现世界。发布版本中包含的电子产品工厂（库工厂）是一个风格化示例，采用合成参数，并未根据公司数据进行校准。它包含三种产品、六种组件和九种材料，以及四家供应商和四位客户。四条生产线和两座由转运路线连接的仓库构成运营约束。可选配方和采购折扣带来了生产量以外的决策选择。

客户端注册清单和情景，启动一次运行，并获取其描述和观测。应用程序接口观测会报告现金、应收款、库存、未完成的客户订单和供应商订单以及生产线状态；在可用且已披露的情况下，还会提供当前价格、中断和预测。客户端提交决策（也可先进行可行性检查）；服务随后推进一个周期。记录和历史分叉仍可通过同一接口访问。S1.1 节包含一段应用程序接口客户端摘录。

2.2 流程模型与执行

物料依赖关系将采购与交付承诺联系起来（图2）。仓库限制仓储，设备状况影响生产，而付款条件则将实体执行与流动性联系起来。只有通过明确的发货决策，客户才能收到货物，因此订单选择也是控制器的职责之一。维护和仓库转运会与资源的其他用途相互竞争。这种结构可能导致某个局部看来有利的动作，对流程的其他环节造成不利影响。

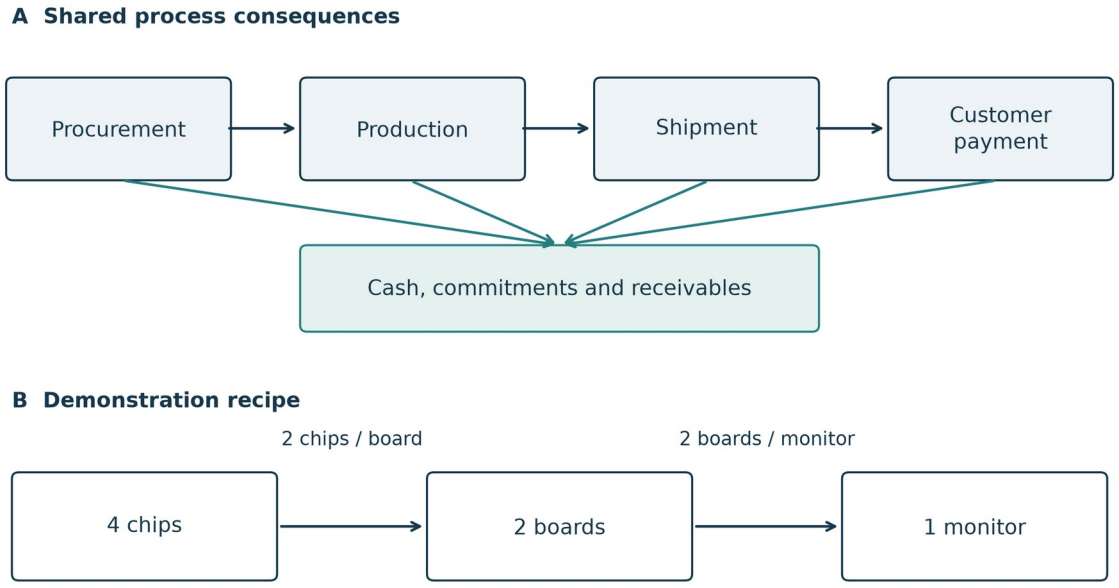

图*2.* 流程关系与简化的物料清单。两阶段链用于第*3.1–3.3*节和集成案例 *A3*；其他集成案例使用规模更大的库工厂。发货和客户付款是两个独立事件。*Codex*（*OpenAI*，*GPT-6 Astra*）协助编写了 *Python/Matplotlib* 绘图代码。

时间按离散周期推进（图3）。交付和事件先于观测发生；随后执行决策，周期才结束。提前期为一个周期的生产会在周期结束时进入库存，并可在下一个周期使用。首先执行发货，随后是取消、调拨和维修。到期的生产计划先于当前生产执行，采购则最后执行。未来的生产计划不会预留资源。每项动作都会根据更新后的状态整体接受或拒绝，因此，彼此竞争的动作需要明确的执行顺序。

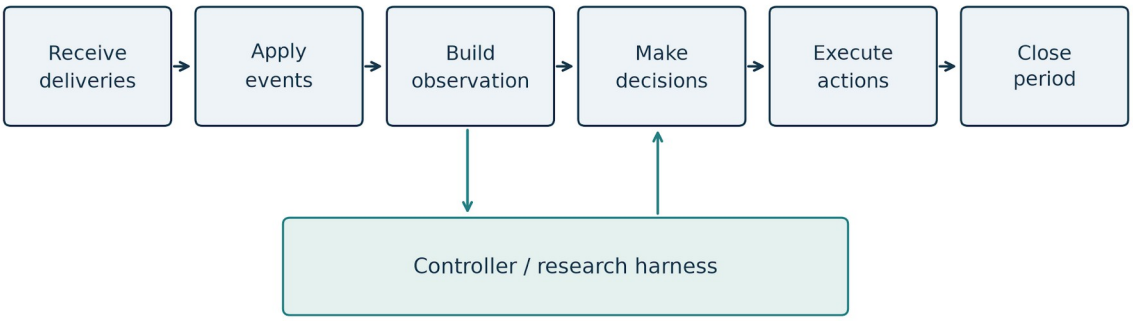

图*3.* 一个决策周期。控制框架接收描述和当前观测，然后返回动作。周期内的执行顺序和生产时间安排均为模型假设。*Codex*（*OpenAI*，*GPT-6 Astra*）协助编写了用于绘图的 *Python/Matplotlib* 代码。

只有当采购成本不超过现金加信用额度再减去现有采购承诺时，信用检查才会批准采购。其他成本可能导致现金低于该上界；库工厂会对债务和超额借款收取利息。货币计算采用十进制算术。这些约定以及固定的行动顺序都是决策问题的一部分，将总体计划转化为动作时必须遵守。

2.3 情景与信息

该库包含一个稳定市场基线和22个分属六大类的情景（表2）。时域可配置；提供的情景均运行52个周期。这些情景将运营约束与采购和履约相关的业务问题联系起来。S3节介绍了每个情景、其设计依据及适用的诊断方法。这些合成商业条件支持针对风格化工厂开展受控研究。工业校准和场景难度校准不在本文范围之内。

表2. 稳定市场基线之外的情景类别。问题表示可能开展的研究，并非已确立的性能结果。

类别（数量）	**示例条件**	**研究问题**
需求（6）	季节性、增长、衰退、 激增和产品生命周期	何时应该增加或减少库存?
供应商（5）	停运、延误、质量损失， 短缺和价格变化	采购应如何调整?
生产线（4）	故障、产能削减和 产量损失	恢复决策如何影响 履约?

类别（计数）	示例条件	研究问题
现金与空间（2）	信贷压力与满载的仓库	哪项动作释放约束性资源?
信息（3）	预测、提前通知和隐藏参数	信息披露如何改变决策?
综合扰动（2）	区域危机；需求激增伴随故障	相互作用的中断如何改变优先级?

季节性促使企业在存储限制下积累库存；提前价格信息可能使消耗信用额度的采购更有利。在仓库满载情形下，中间生产可以腾出空间来存放缺失投入品。需求与产能冲击同时发生时，会造成优先事项之间的冲突。

描述和观测源自底层世界。情景可以隐藏参数组、提供含噪预测，或提前披露中断信息。某些信息情景也会改变需求；已记录协议规定了匹配对照。世界端点要求所请求的运行已完成，但服务不会强制执行针对控制器的访问限制。控制框架必须限制对情景配置以及共享同一世界的已完成运行（例如分叉的父运行）的访问，并提供对可行性检查的同等访问权限。相同的信息意味着采用共同的信息披露机制并使用获准的工具；采取不同动作后，观测可能会有所不同。

2.4 已记录世界与替代决策

对于固定的世界和动作，在兼容的软件环境中，执行具有确定性。外生抽样使用执行前生成的独立命名流。决策会改变结果，但不会改变共享抽样。记录会保留动作、结果和执行轨迹。回放时，首先检查软件包、Python 和 NumPy 的版本兼容性，然后将重新生成的执行结果与记录进行比较。

一次已完成的运行可以在指定周期分叉。服务会回放此前的封闭历史，并在同一外生世界上接受替代延续（第3.2节）。比较内容包括下游策略响应；控制框架必须管理控制器记忆和延续规则。

2.5 根据参考结果评估决策

在比较控制器之前，评估需先明确目标和时域。提供的优化器通过规划窗口模型和终值近似来追求利润；它们并非服务最优参考基准。实际结果衡量的是在单个世界中的表现；要评估策略在不确定性下的表现，必须定义一个世界总体。规划期利润由公式（1）给出。

$$\pi = C_h - C_0 + R_h + V_h. \qquad (1)$$

这里，C0 和 C_h 分别表示期初和期末现金，R_h 表示按名义价值计量的未收客户应收款，V_h 表示仓库存货残值。在制品和在途货物没有期末残值。初始库存不产生采购费用。未交付的供应商订单和未到期的客户订单不计入期末估值（S4.2节）。利润应与服务指标一并报告，并明确分母和时间安排，因为交付率与经济结果回答的是不同问题。

以无决策基线和完全预见参考基准归一化的得分可用作诊断指标，但不代表最优性百分比（S4.2节）。在共同信息披露条件下进行比较时，应在匹配世界中报告相对于观测受限比较器的直接利润和服务差异。

完全预见参考基准根据已实现的未来世界进行规划，反复求解 LP 或 MILP，并使决策适应引擎约束。后期周期的二元选择可以放宽，执行过程中还包括启发式方法。其已实现利润不具有全局最优性的普遍保证。求解器状态和间隙涉及规划程序。相对于该参考方案的利润短缺综合反映了信息和规划过程的差异。

观测受限比较对象使用披露的描述和观测结果，并拥有可行性检查访问权限（2.3节）。它们包括全时域受限控制器、滚动控制器、缓冲滚动控制器、预测对齐控制器和短视控制器（S4节）。知情诊断还会接收隐藏参数均值，但不会接收未来抽样结果（图4）。匹配干预将信息效应与算法选择区分开来。精确整数控制会针对小型程序取消一项规划放宽；引擎级最优解则需要独立证明，如3.3节所示。

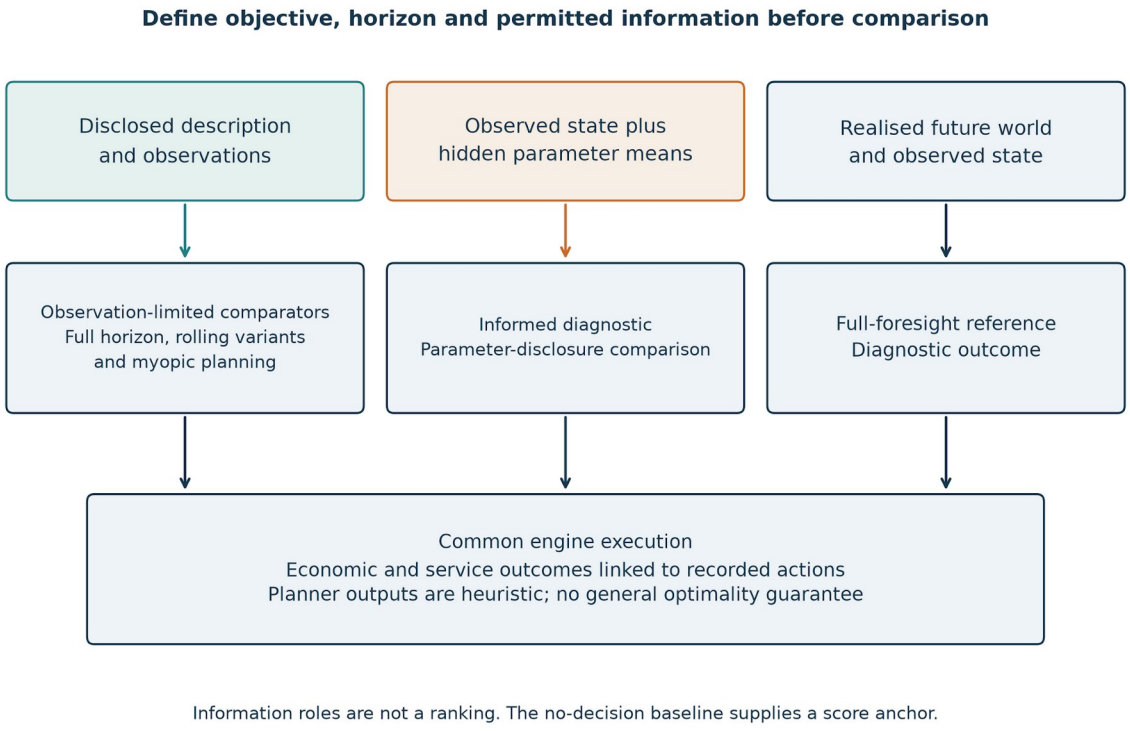

图*4*。比较控制器和诊断参考的作用。可用信息定义了每种作用；各列不表示性能排序。引擎执行决定所有结果。还必须明确可行性检查访问权限。*Codex*（*OpenAI*、*GPT-6 Astra*）协助编写了 *Python/Matplotlib* 绘图代码。

3. 示例

3.1–3.3节使用带有受控决策策略的小型确定性工厂来演示评估逻辑。3.4节总结验证工作；3.5节将这些评估应用于实时LLM多智能体系统。补充材料S1提供输入和复现说明；以S为前缀的章节编号均指该补充材料。

3.1 相同服务，不同经济结果

一家为期十个周期的电子装配厂，每个周期满足四台显示器的需求。每台显示器需要两块电路板，每块电路板需要两枚芯片（图2）。芯片、生产电路板、组装显示器和发货的单位成本分别为0.5、1、2和1；显示器售价为50。产出率为1，供应商提前期和生产提前期均为一个周期。客户立即付款；每件未能满足的需求造成10的销售损失。初始现金为1000，库存包括16枚芯片、8块电路板和4台显示器。芯片、电路板和显示器每单位每周期的仓储成本分别为0.0023、0.0090和0.0270；残值分别为0.1、0.5和2。利息和损耗均为零。第3至第6周期禁止采购芯片，这一信息要么在禁购开始时（第3周期）披露，要么通过第1周期的提前通知披露。

未收到提前通知时，滚动控制器、缓冲滚动控制器和全时域受限控制器均按时交付 40 台监视器中的 24 台，利润为 919.17（图 5）。收到提前通知后，滚动控制器交付 28 台，利润为 1130.96；缓冲滚动控制器和全时域受限控制器均交付 40 台，利润分别为 1732.43 和 1766.08。在这些存储费率下，提前通知使滚动控制器的实际利润增加约 212，使全时域受限控制器的实际利润增加约 847；在此情形中，其影响取决于控制器。在信息相同的情况下，缓冲滚动控制器和全时域受限控制器实现了相同服务，但利润相差 33.66（根据未四舍五入的结果计算）。额外的材料采购和储存费用（部分由残值回收抵消）造成了这一差异。两者的策略在规划时域和期末库存估值方面均有所不同。完全预见参考基准在两种情形下均交付 40 台，利润为 1766.08；在此情形中，收到提前通知的全时域受限控制器与之结果一致。

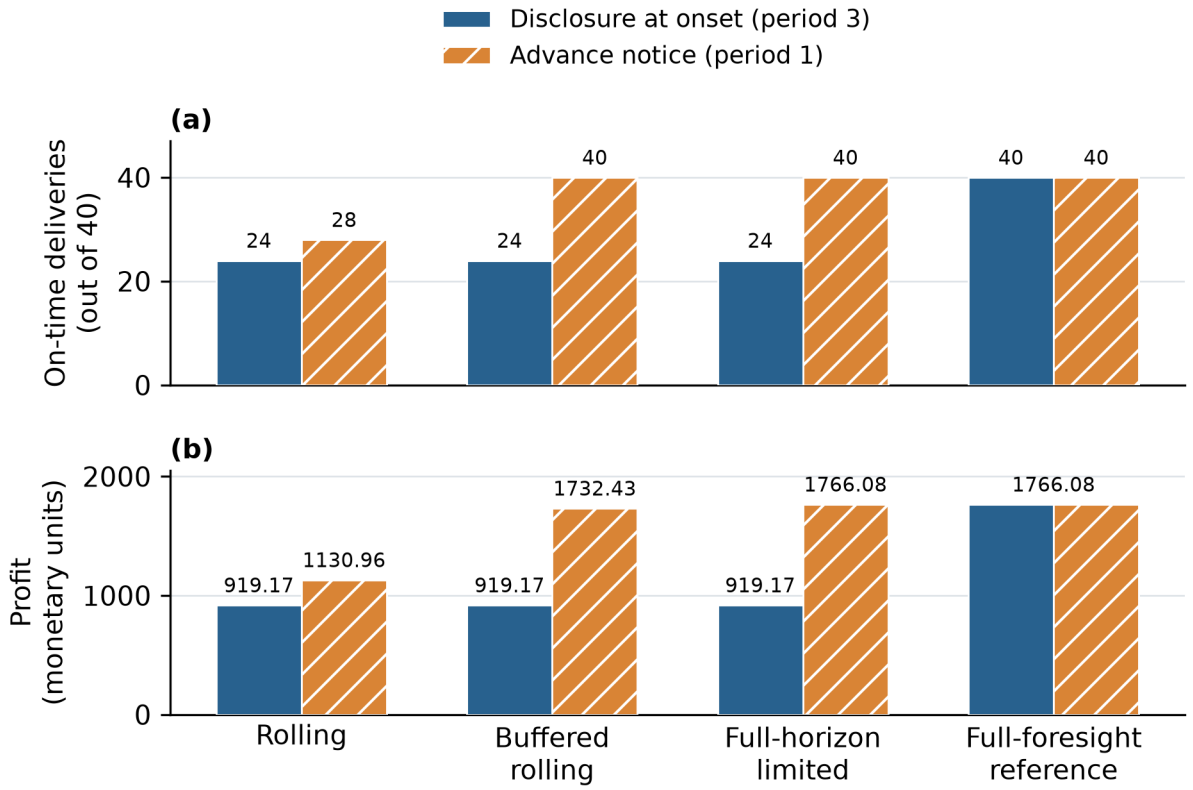

图*5.* 在开始时披露（周期*3*）和提前通知（周期*1*）情况下，正储存成本下的准时交付和利润（第*3.1*节）。在提前通知的情况下，缓冲滚动和全时域受限控制器都交付了全部*40*台显示器，但利润相差*33.66*（根据未四舍五入的结果）。货币单位表示模型货币单位。控制器定义见*S4*节。

收到提前通知后，滚动控制器的交付量取决于存储费率：在上述费率（约为每周期重置成本的 0.45%）以及测试过的 1%、2% 和 5% 费率下，交付量为 28 台；在 0.02%、0.1% 和 0.2% 费率下，则交付 40 台。S5.2 节报告了受控延续、扫描、零成本变体以及核对后的结果。

3.2 沿流程追踪另一种决策

一家为期五个周期的工厂采用相同的配方和单位成本，初始现金为 1000，且没有库存。客户订购了四台显示器，要求在第 4 个周期交付，可容忍延迟一个周期，并在发货后一个周期付款。每件商品每延迟一个周期需支付 2 的延迟罚金；每件商品的销售损失成本为 10。产出率均为 1，提前期为一个周期；仓储费用、利息和损耗均为零。控制器在第 1 个周期购买 16 片芯片，在第 2 个周期生产 8 块电路板，并在第 3 个周期组装 4 台显示器。

既定规则规定，将可用商品发给未完成订单。发货在第 4 个周期进行，付款在第 5 个周期到账。收入 200 减去总成本 28，利润为 172，期末现金为 1172（图6）。从第 3 个周期开始的一个分叉将组装推迟到第 4 个周期，同时保留发货规则。发货因此推迟到第 5 个周期，产生 8 的延迟罚金，且付款时间超出时域。

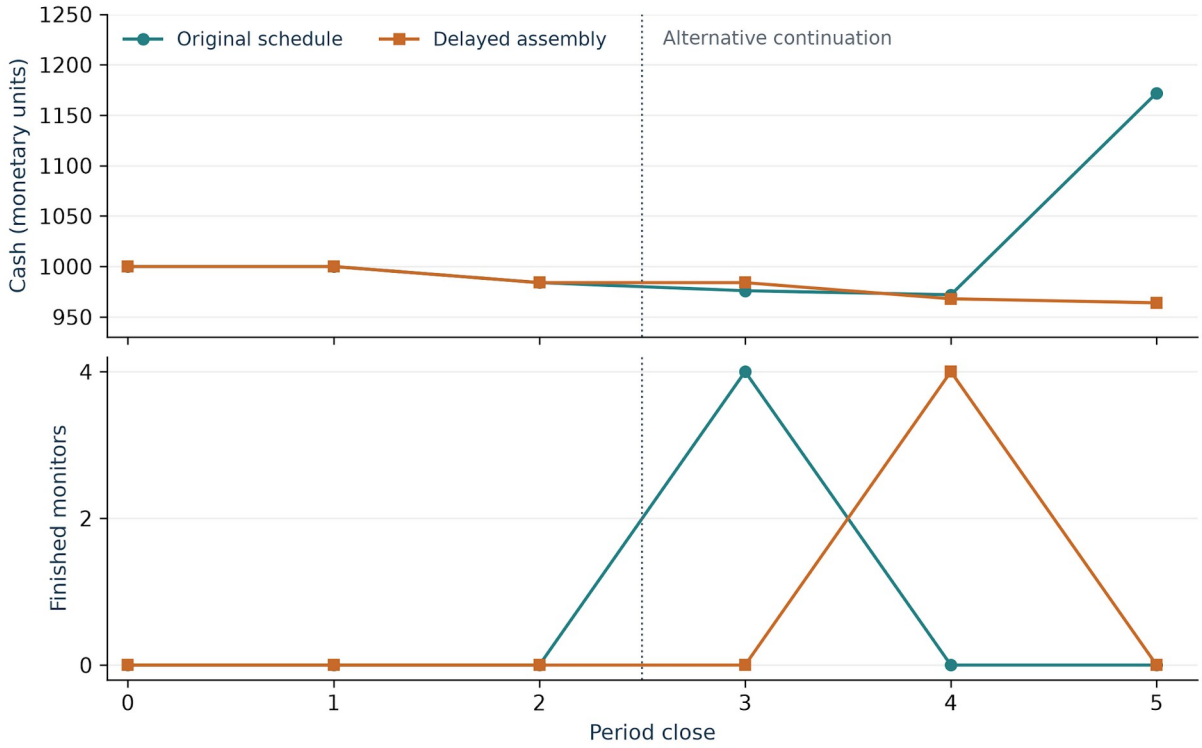

图*6.* 替代装配决策会传递至发货和付款。共同的封闭历史在周期*2*结束。期末现金相差*208*，而利润相差*8*。货币单位指模型货币单位。周期*0*为初始状态；线条连接离散观测。

分叉方案的期末现金为 964，应收款为 200，因此利润为 164。现金差额 208 包括延迟支付的 200 和经济罚金 8。在利息为零且应收款为名义金额的情况下，此示例将交付罚金与付款时机区分开来。比较还纳入了未改变的发货规则所产生的下游响应。无决策运行损失 48：四个单位产生 8 的延迟罚金和 40 的终端销售损失罚金（S5.1节）。

3.3 可核查的最优分配

一个单周期微型实例以 0.001 个显示器为起点，并有两笔数量同为 0.001 的订单，单价分别为 50 和 70。运输成本为每单位 1；未满足的需求将产生 10 的惩罚，残值回收为零。在当前实现精度下，三种可能的分配方式分别是不发货、发给低价订单和发给高价订单，收益依次为 −0.020、0.039 和 0.059。生产在发货后完成，无法提高同期收入。上界为 0 001 。

× 70 − 1 − 0 001 × 10 = 0 059

（）。。。完全预见参考基准、精确整数控制和滚动比较器在此实例中均达到该上界。S5.3节给出了证明和精度检查。

3.4 验证范围

托管式持续集成在 Ubuntu 上通过了测试，并在 macOS 和 Windows 上通过了冻结世界检查（S1.2节）。在分阶段进行的 macOS 运行中，全部 3430 项不同的非压力测试和文档测试均累计通过。测试覆盖引擎行为、应用程序接口处理、回放、分叉和仿真模型集成。七次履约分析运行和 47 次通知分析运行均通过物料和财务对账；七次分配运行均通过财务对账。

遵循 Sargent [26], ，这些结果提供了有针对性的软件验证证据。与软件一同记录的开发性情景校准，与本次验证以及工业验证相互独立。

发布文档报告称，六种情景未能通过针对特定控制器–随机种子组合的结果方向检查。仅凭此类失败，既无法诊断场景机制是否失效，也无法确定任务难度；S3节对此作了区分。

3.5 与基于大语言模型的多智能体控制器集成

代码库中协调式客户端的外部扩展将规划器、采购和审查器角色接入 HTTP/JSON服务，并于2026年11月12日至13日使用低推理强度的 OpenAI gpt-6.1-terra。它允许每个周期最多进行两次额外的审查器修订，并包含一个用于移除被拒绝决策的过滤器。表3报告了全部四次最终运行。补充材料S1记录了最终协议，并在第 S7 节保留了早期尝试。

全部159个周期均已完成，共有483个模型响应，其中包括六次额外修订。过滤器和传输重试均未使用；C1 和 C2 各需要三次额外的审查器修订。周期层面的物料和财务检查未发现差异，回放也在未进行额外模型调用的情况下重现了记录的行动和结果。回放无法重现随机决策生成。

C1 和 C2 使用相同的世界和披露设置，并采用新记忆。C1 最终实现正利润，但现金为负，应收账款余额为 211964.41。C2 支付了3783.17的利息，期末余额低于其信用额度；该额度限制的是采购准入，而非每笔借记。这些记录体现了盈利能力与流动性的区别。

匹配世界的结果提供了背景信息（表4）。带通知中断世界在稳定市场世界的基础上，增加了第 10–41 期的芯片供应中断，并从第 6 期起披露该信息；这一比较并未单独衡量通知的价值。滚动控制器使用已披露观测值并进行可行性检查；完全预见还会使用未来事件。在该世界中，两次完全预见求解达到 30 秒时限，但保留了可用的混合整数线性规划解（S5.5 节）。B3 按时发货的订购单位占 21.71%，而滚动控制器为 65.85%。在带通知中断世界中，C1 发货的订购单位占 54.7%，高于滚动控制器的 45.1%，但按时发货比例为 18.1%，低于滚动控制器的 36.7%；其利润则更高。C2 的利润低于滚动控制器。C1 和 C2 的实际利润相差约 211571 个模型货币单位。这些运行结果不足以确定控制器排名。

表*3*。最终大语言模型多智能体系统运行：*A3* 为三个周期，其余情景为 *52* 个周期。财务数值及其与匹配世界滚动基准的差异均以模型货币单位表示。*A3* 展示了小型工厂整合；未报告滚动比较。

Run	情景	按时 / 订购 (%)	规划期利润	期末现金	Δ 相对于滚动基准的利润
A3	小型工厂	66.67	328.00	1328.00	—
B3	稳定市场	21.71	552071.45	361309.71	−316898.86
C1	提前通知的停机通知	18.11	203626.76	−15502.68	+24167.97
C2	提前通知的停机	15.12	−7944.52	−119243.39	−187403.32

表*4.* 匹配世界视野利润，种子*7*。*C1*和*C2*共享带通知中断世界。无决策和短视行动及现金流在各世界中一致（补充材料*S1*）。短视计划仅针对当前期间，并计入期末库存和在途库存估值（*S4*节）。启发式完全前瞻基准使用额外信息（*2.5*节）。

世界	无决策	短视	滚动控制器	完全预见
稳定市场	−709331.23	−432814.34	868970.31	1098988.81
提前通知的停机	−709331.23	−432814.34	179458.79	892273.68

4. 影响

Factory-simulator 使研究人员能够考察在建模工厂中达到特定服务水平所需的成本，以及这些成本产生的环节。在第3.1节中，两种信息相同的策略均按时交付全部 40 台显示器，但材料采购和仓储费用导致二者利润不同，残值回收则抵消了部分差异。第3.2节追踪了装配延迟对发货和收集的影响。实时集成新增了可审计的大语言模型多智能体系统轨迹，以及与滚动控制的匹配世界比较。

未来研究可以探讨：在同一个世界中，将采购和生产任务分配给不同智能体角色，与集中式控制相比，会如何改变利润？库存策略和信息共享会如何影响工厂利润，以及单独衡量的推理成本（S4.1 节）？从共同前缀分叉，并采用指定的延续策略时，改变一项决策会如何影响结果？Factory-simulator 是计划用于华沙理工大学博士研究的工厂环境，研究方向为大语言模型多智能体系统支持的决策。该应用程序接口也支持启发式方法或人工控制策略。

库工厂经过风格化处理，尚未展示工业校准和独立采用的可行性。离散周期和固定执行顺序限制了其在详细调度应用中的适用性；财务惯例和证据边界详见 2.5节和 S6节。

5. 结论

Factory-simulator 通过基于优化的参考方案和可重放的历史记录，将制造决策与业务结果联系起来。受控示例区分了服务、利润和付款时机。借助外部客户端，实时大语言模型多智能体系统演示完成了四次运行，余额均已核对，并精确重放了已记录的动作。匹配世界参照

为这些结果提供背景。比较不同方案的有效性需要重复研究，而 Factory-simulator 提供了一个共同的环境。

代码和数据可用性

Factory-simulator 1.0.0 依据 MIT 许可发布[27], ，包括外部客户端和复现脚本。数据和报告以知识共享署名 4.0 国际许可公开发布于 GitHub Releases，档案中注明了固定装置例外情况。S1节提供了带版本号的链接和校验和。

CRediT 作者贡献声明

Andrzej Wodecki：概念化、监督、项目管理。Wojciech Godlewski：撰写——初稿、可视化。Michał Chojna：软件（设计、架构和测试）。全体作者：方法论、软件、验证、形式分析、调查、资源、数据管理、撰写——审阅与编辑。

资金

本研究未获得公共部门、商业部门或非营利部门资助机构提供的任何专项拨款。

利益冲突声明

作者声明，他们不存在任何利益冲突。

关于在稿件准备过程中使用生成式 AI 和 AI 辅助技术的声明

作者使用 Claude（Anthropic）和 Codex（OpenAI）进行初稿撰写、英文编辑、参考方案核查和可复现性检查。Codex 协助开发和审查代码，并编写生成图表的脚本。作者审阅并编辑了生成内容，并对本文承担全部责任。

参考文献

[1] Backlund A、Petersson L。Vending-Bench：用于评估自主智能体长期连贯性的基准。arXiv:2502。15840 [预印本]；2025年。https://doi.org/10.48550/arXiv.2502.15840。[2] Zhao X、Xie Y、Chen C、Sun Y。AIM-Bench：评估智能体型大语言模型作为库存管理者时的决策偏差。arXiv:2508。11416 [预印本]；2025年。https://doi.org/10.48550/arXiv.2508.11416。[3] Kravchenko K、Gruchmann T、Ivanova M、Ivanov D。应对系统性中断引发的涟漪效应：来自COVID-19期间半导体短缺的实证证据。《现代供应链研究与应用》2024；6(4)：354–75。https://doi.org/10.1108/MSCRA-03-2024-0011。[4] Guo T、Chen X、Wang Y、Chang R、Pei S、Chawla NV等。基于大语言模型的多智能体：进展与挑战综述。载于：第三十三届国际人工智能联合会议论文集。2024年。第8048–57页。https://doi.org/10.24963/ijcai.2024/890。[5] Quan Y、Liu Z。InvAgent：用于供应链库存管理的基于大语言模型的多智能体系统。arXiv:2407.11384v2 [预印本]；2025年。https://arxiv.org/abs/2407.11384v2。[6] Jannelli V、Schöpf S、Bickel M、Netland T、Brintrup A。供应链中的智能体式大语言模型：迈向自主的多智能体共识寻求。《国际生产研究杂志》2025年。https://doi.org/10.1080/00207543.2025.2604311。

[7] Davenport TH、Short JE。新工业工程：信息技术与业务流程再设计。《斯隆管理评论》1990；31(4)：11–27。https://sloanreview.mit.edu/article/the-new-industrial-engineering-information-technology-and-business-process-redesign/.[8] Rummler GA、Brache AP。改进绩效：如何管理组织结构图中的空白地带。第3版。Jossey-Bass；2012年。[9] Galbraith JR。组织设计：一种信息处理视角。《Interfaces》1974；4(3)：28–36。https://doi.org/10.1287/inte.4.3.28。[10] Hubbs CD、Perez HD、Sarwar O、Sahinidis NV、Grossmann IE、Wassick JM。OR-Gym：用于运筹学问题的强化学习库。arXiv:2008。06319 [预印本]；2020年。https://doi.org/10.48550/arXiv.2008.06319。[11] Hubbs C、Perez Parra H、Sarwar O。OR-Gym。0.5版。0 [软件]。PyPI；2022年。https://pypi.org/project/or-gym/0.5.0/ [访问于11月14日2026]。代码库：https://github.com/hubbs5/or-gym。[12] Baek J、Fu Y、Ma W、Peng T。用于库存控制的大工智能智能体：人类-大语言模型-运筹学互补性。arXiv:2602。12631 [预印本]；2026年。https://doi.org/10.48550/arXiv.2602.12631。[13] Baek J、Fu Y、Ma W、Peng T。InventoryBench [软件]。GitHub；无日期。https://github.com/TianyiPeng/InventoryBench [访问于11月8日。[14] Fish S、Shephard J、Li M、Shorrer RI、Gonczarowski YA。EconEvals：大语言模型智能体经济决策的基准与试金石测试。arXiv:2503.18825v4 [预印本]；2026年。https://arxiv.org/abs/2503.18825v4。[15] Fish S、Shephard J、Li M、Shorrer RI、Gonczarowski YA。EconEvals [软件]。GitHub；无日期。https://github.com/sara-fish/econ-evals-paper [访问于11月8日。[16] Waubert de Puiseau C、Peters J、Dörpelkus C、Tercan H、Meisen T。schlably：用于基于深度强化学习的调度实验的Python框架。SoftwareX 2023；22：101383。https://doi.org/10.1016/j.softx.2023.101383。[17] Mohammad Y。运行受控实验。SCML 0.7.7文档；无日期。https://scml.readthedocs.io/en/latest/tutorials/05.experiments.html [访问于10月29日。[18] Mohammad Y。SCML：ANAC供应链管理联盟平台[软件]。GitHub；无日期。https://github.com/yasserfarouk/scml [访问于11月8日。[19] Yuwono S、Löppenberg M、Arend D、Diprasetya MR、Schwung A。MLPro-MPPS：用于可定制生产系统的高性能仿真框架。Software Impacts 2023；16：100509。https://doi.org/10.1016/j.simpa.2023.100509。[20] Yuwono S、Löppenberg M、Arend D、Diprasetya MR、Schwung A。MLPro-MPPS：基于Python的可定制多用途生产系统。1.1版。0 [软件]。Zenodo；2023年。https://doi.org/10.5281/zenodo.8202434。https://github.com/fhswf/MLPro-MPPS [访问于11月8日。[21] Behrendt S。prodsys：生产系统的建模、仿真与优化[软件]。GitHub；无日期。https://github.com/sdm4fzi/prodsys [访问于10月29日。文档：https://sdm4fzi.github.io/prodsys/Tutorials/tutorial_3_优化_一个_生产_系统/ [访问于11月9日。[22] Drozdov S、Dadfarnia M。SimPROCESD [软件]。美国国家标准与技术研究院；2024年8月15日。https://www.nist.gov/services-resources/software/simprocesd [访问于10月30日。[23] Guan S、Liu Y、Cao L。SupChain-Bench：面向现实世界供应链管理的大语言模型基准。在：Liakata M、Moreira VP、Zhang J、Jurgens D，编。计算语言学协会年会 ACL 2026 论文集。计算语言学协会；2026年。第7526–50页。https://doi.org/10.18653/v1/2026.findings-acl.371。[24] Guan S、Liu Y、Cao L。SupChain-Bench [软件]。GitHub；无日期。https://github.com/Damon-GSY/SC-bench [访问于11月8日。[25] Schrage R、Sager J、Hörding JP、Holly S。mango：基于Python的模块化智能体仿真框架。SoftwareX 2024；27：101791。https://doi.org/10.1016/j.softx.2024.101791。

[26] Sargent RG。仿真模型的验证与确认。J. Simul. 2013；7(1)：12–24。https://doi.org/10.1057/jos.2012.20。 [27] Wodecki A、Godlewski W、Chojna M。工厂模拟器。版本 1.0。0 [软件]。GitHub；2026。https://github.com/mchojna/factory-simulator/releases/tag/v1.0.0。

补充材料S1

Factory-simulator：一种流程导向的环境，用于依据基于优化的参照评估制造决策

Andrzej Wodecki、Wojciech Godlewski、Michał Chojna

本补充材料记录了 Factory-simulator 1.0.0 及其演示证据。代码维护在代码库中；Factory-Simulator-Reproduction.zip 仅包含数据、报告和图表。该档案可公开获取，网址为

https://github.com/mchojna/factory-simulator/releases/tag/factory-simulator-reproduction。README.txt 提供命令；CONTENTS.tsv、MIGRATION.tsv 和 PROVENANCE.json 标明保留记录及其原始位置。

S1. 软件元数据与复现

元数据项	价值
软件与许可	Factory-simulator 1.0.0；MIT
源代码仓库	https://github.com/mchojna/factory-simulator
已发布代码	ea92c1bfb28228c8929045e1ddcfa9535da2ea27
软件发布版本和数据归档	代码：https://github.com/mchojna/factory-simulator/releases/tag/v1.0.0（提交 ea92c1bfb28228c8929045e1ddcfa9535da2ea27）。数据：https://github.com/mchojna/factory-simulator/releases/tag/factory-simulator-复现。归档 SHA-256：3340137bc74bb754ded7b350db4fcf6c0b5e7b782b2113d5fb09abadb5d67d8f.
语言和环境	所需 Python >=3.14、<3.15。已验证的环境：Python 3.14.10；macOS 26.5.2 arm64；原始复现使用 uv 0.13.6，补充检查使用 0.13.7。
数值软件	NumPy 2.5.3；SciPy 1.18.1；捆绑的 HiGHS 1.12.0
服务与持久化	FastAPI 0.142.2；SQLAlchemy 2.1.3；DuckDB 1.5.6
文档	代码库中的 docs/README.md
执行/测试依赖项	代码库 uv.lock；默认开发依赖。历史安装的版本位于 historical/v06-verification/environment.json。
图表	Matplotlib 3.11.2；可编辑的 SVG 和矢量 PDF，以及 PNG
支持联系信息	Wojciech Godlewski；wojciech.godlewski4.dokt@pw.edu.pl

已发布的标签 v1.0.0 对应提交 ea92c1bfb28228c8929045e1ddcfa9535da2ea27。早期开发快照使用了相同的版本标签；请以 S1.1 节中的确切提交为准。原始科学记录使用的引擎提交为

319489ac6f7c45ae022521681b1a93b5e9e01ecd；validation/repository-2026-11-19/ 下的报告描述了提交 394f68c6f16da1a280cfb0cef3a70d3fa46e066a。这些提交属于早期开发历史，并非公开检出目标。src/ 中的全部 185 个文件以及 uv.lock 均与 historical/source/source-distribution.json 中的历史源文件哈希一致。

REPRODUCTION.json 区分了已发布标签与历史验证；SHA256SUMS.json 涵盖数据负载。回放要求使用兼容的软件包、Python 和 NumPy 版本。受控示例使用独立于模型客户端的代码仓库测试夹具。其求解限制为 30 秒；没有任何示例达到该限制。S5.5 节另行报告库-

世界求解器状态。确定性记录比较不包括时间安排；不声称不同执行之间的完整档案字节完全一致。

S1.1 安装并重新运行演示

克隆代码库，并在安装 Python 3.14.10 与 uv >=0.12.10,<0.14 之前检出确切的复现提交。将 Factory-Simulator-Reproduction.zip 下载到检出目录旁。以下命令使用 POSIX shell、默认锁定的开发依赖项和新的输出目录。无需 API 密钥。请将生成的输出保存在提取后的数据归档之外；切勿在禁用 Python 断言的情况下运行。

```
git clone https://github.com/mchojna/factory-simulator.git

cd factory-simulator

git checkout ea92c1bfb28228c8929045e1ddcfa9535da2ea27

uv sync --locked --python 3.14.10

uv run python -m scripts.reproduction.prepare_reproduction --archive ../Factory-Simulator-Reproduction.zip --output data/reproduction

PACKAGE=data/reproduction/Factory-Simulator-Reproduction

uv run python -m scripts.reproduction.check_recorded_inputs --package-root "$PACKAGE"

uv run python -m scripts.reproduction.replay_saved_actions --case all --package-root "$PACKAGE" --output data/replay-check

uv run python -m examples.coordinated_planning.run --offline --case all --archive ../Factory-Simulator-Reproduction.zip --output data/client-check

uv run python -m scripts.reproduction.article all --data "$PACKAGE" --output data/article-check
```

文章入口点会运行通知矩阵、敏感性和延续检查、订单履约/分叉示例以及微观分配示例，然后渲染所有图表。它会在执行前后验证档案，并阻止外部网络调用。这些示例使用代码库的进程内异步服务器网关接口测试客户端。2026年11月19日，34 个通知案例、12 个敏感性案例、一个受控延续以及 35 个小型示例记录比较均通过；全部六个图表均已生成。JSON 解析后，结果、期间记录和世界保持一致。求解器诊断信息采用文档规定的容差，不包括时间安排。报告位于 validation/repository-2026-11-19/。

以下节选展示了代码库中 scripts/reproduction/article/run_allocation.py 的注册和运行创建过程。完整脚本提供基准客户端 c、清单 m、情景 s、导入项和控制器。这些节选并非独立客户端；请使用上文的完整入口点。

```
def setup(c,m,s):
 mid=b.send(c,'POST','/manifests',m)['id']
 return b.send(c,'POST',f'/manifests/{mid}/scenarios',s)['id']
def play(c,sid,agent_kind,base=None,policy=None):
 r=b.start_run(c,sid,7); check=partial(b.create_check,c,r['id'])
```

play 的其余部分（此处省略）会选择控制器、提交其决策并读取记录的结果。这些节选来自已执行的进程内异步服务器网关接口客户端；它们不构成单独的网络传输测试。

S1.2 验证范围与复现

已发布代码通过了 S1.1 节所标识提交对应的 CI 运行中的全部 13 项任务。Ubuntu 检查包括常规测试和八个属性测试／慢速测试分片；macOS 和

Windows 运行冻结世界选择。持续集成还会检查软件包构建、全新安装和提交前规则；它不会在每个平台上运行完整的测试套件。

档案中的本地报告记录了历史引擎提交版本在分阶段 macOS 运行中通过的 3430 项互不重复的非压力测试项（S1 节）。这一累计覆盖范围独立于持续集成；本文未重新开展压力测试活动。测试选择、命令和报告保存在数据归档的 historical/v06-verification/ 和 historical/verification/ 中；S7.1 描述了执行条件。

端到端测试通过环回 HTTP，覆盖不决策和优化客户端、可行性检查、恢复、分叉及记录检查。模型适配器测试使用模拟传输；实时大语言模型-多智能体系统演示及其检查在 S5.4 节单独报告。

要复现文章结果，请获取代码和数据归档，并按照 S1.1 节及代码库中 scripts/reproduction/README.md 的命令操作。这些内容涵盖档案验证、已记录输入检查、已保存操作回放、离线客户端检查、受控示例重新执行和图表生成，无需进行模型调用。S5.4–S5.5 节区分了回放与新的模型采样和优化。

S1.3 图表与通知示例

```
uv run python -m scripts.reproduction.article figures --data "$PACKAGE" --output data/figure-check
```

请从代码仓库根目录运行该命令。数据归档中包含六张当前图表，格式为 PNG、PDF 和 SVG，且没有重复的渲染素材。绘图代码位于 scripts/reproduction/article/。图1–4 是使用 Python/Matplotlib 绘制的说明性示意图；图6 使用归档的示例记录，图5 使用 notice/recorded-results.json。Codex（OpenAI）协助编写了图1–6 的绘图代码；这些图像由代码渲染，并非由模型直接生成。图1–4 的绘图代码使用了通过 Codex 调用的 GPT-6 Astra。代码/数据迁移未改变上述信息披露。

第3.1节和图5所用的通知数据位于 S1节所指数据归档中的 notice/；可执行流程位于代码库的 scripts/reproduction/article/ 中。

通知命令组会重新运行 34 案例矩阵、另外十二个敏感性案例、一项受控延续和规划诊断。请使用 S1.1 节中的 all 命令，或将其替换为 notice，仅运行此命令组。原 macOS 工作站和 Linux 容器上的历史检查仍按其原有范围保留。11月19日的 macOS 代码仓库验证再次确认结果、周期和世界字段一致；六项重建的规划输入和 42 次诊断求解均通过了文档所述的数值检查。无需模型凭据。S5.2 节提供数值证据，S5.5 节介绍历史 Linux 检查。

S1.4 离线复现与可选的实时执行

如需根据已存储决策重新执行八条最终集成轨迹和参考轨迹，请使用代码库中的 scripts/reproduction/replay_saved_actions.py，具体如 S1.1 节所述。此路径既不调用模型，也不调用优化器；它不同于仅进行档案哈希和算术检查。

归档响应路径通过 python -m examples.coordinated_planning.run --offline 执行代码库中的三角色客户端，并按照 S1.1 节所示直接使用 ZIP。A3 有三个周期；all 会选择 A3、B3、C1 和 C2（共 159 个周期、483 个已保存的响应）。无需单独的客户端ZIP。S5.4 节区分了当前的离线验证与历史实时执行。

对于新的决策，可选的实时路径需要凭据、经过验证的代币价格以及明确的支出上限；相关命令见代码库中的 examples/coordinated_planning/README.md。历史打包客户端于 2026年11月18日完成了三期A3实时检查。代码仓库迁移仅经过离线检查；这并不构成新的实时检查或跨平台检查。

S2. 相关软件与比较依据

比较采用目的性方法，并基于主要文档或原始论文；软件引用核查截至2026年11月8日，文献综述于2026年11月14日扩展，纳入了原始的schlably和mango出版物，以及供应链大语言模型（LLM）和多智能体系统（MAS）研究。主表涵盖包含生产、供应链或业务运营任务的环境。比较维度包括参考或评估目标、报告的结果，以及有文档记录的信息或诊断支持。信息对等要求对每项实验方案进行审查。未提及的能力不归类为缺失。本文未运行任何竞争方案，也未声称进行了穷尽性或性能比较。

InvAgent [5] 研究多级库存环境中的基于大语言模型的订货。Jannelli 等人 [6] 考察了三级供应链中的信息共享、工具使用和谈判，并介绍了一种集成于序贯供应链仿真环境中的开源实现。他们报告了累计成本和牛鞭效应指标，但未报告利润。对于测试的两个 Gemini 模型，牛鞭效应最低的配置并非累计成本最低的配置 [6, 正文表格 2]。Factory-simulator 报告期末现金和规划期利润，后者包含应收账款余额和库存残值。

环境	参考或评估目标	信息、控制与诊断
OR-Gym [10,11]	混合整数线性规划和启发式方法；库存任务 递缩时域线性规划和完美 信息预言机；任务奖励。	库存状态和动作历史；特定任务 优化模型。
AIM-Bench [2]	事后补货基准； 订货偏差和库存 成本。	多智能体设置和信息共享 处理；周期级订货行为。
InventoryBench [12,13]	带上限的基库存基线； 运筹学、大语言模型和 混合方法的归一化奖励。	预生成需求和提前期； 历史输入和上下文输入； 推荐或预测流程。
SupChain-Bench [23,24]	标准答案与检索 目标；准确率与工具使用 成本。	上下文与程序指导条件； 执行错误分析。
Vending-Bench [1]	长期运行的业务运营； 最终净资产和单次运行的 人类基线。	智能体轨迹和工具使用；大语言模型- 生成的供应商回复和需求 参数；含噪销售数据。
schlably [16]	深度强化学习智能体与 调度启发式方法和求解器的比较； 奖励以及与求解器解的差距 。	生成的调度实例、Gym 环境、记录的结果和甘特 图。
SCML [17,18]	谈判智能体的比较 在供应链世界中。	可配置上下文和受控世界 条件，包括外生合同 及其揭示时间。

环境	参考或评估目标	信息、控制与诊断
MLPro-MPPS [19,20]	可复用生产环境 适用于不同学习方法； 模块化、去中心化的散装货物 系统示例。	传感器和组件状态；状态可能 无法被传感器获取；与 强化学习和博弈论集成。
prodsys [21]	生产配置；吞吐量、吞吐时间、在制品和重新配置成本。带种子的仿真和存储事件； 机器、运输资源和工艺模块的重新配置成本，作为优化目标或约束。	
SimPROCESD [22]	维护与生产 策略；净生产利润和 质量。	设备状况和资源使用 指标。
EconEvals [14,15]	采购和定价结果 相对于 OPT；调度阻塞 对相对于随机匹配。	智能体中立工具接口，轨迹 分析以及针对冲突 目标的试金石测试；注释分析假定注释 能够忠实地描述推理过程。

软件参考文献将相关代码库与介绍 OR-Gym [10,11], InventoryBench [12,13], MLPro-MPPS [19,20], SupChain-Bench [23,24] 和 EconEvals [14,15]的论文分别列出。SCML 的文档和软件分别引用 [17,18]；prodsys 和 SimPROCESD 则根据其软件来源标注 [21,22]。AIM-Bench 和 Vending-Bench 根据其原始论文进行比较 [1,2]。代码库引用标明了查阅过的来源；未运行任何竞品。在线文档在查阅后可能发生变化。

schlably 框架整合了调度实例生成、Gym 环境，以及将深度强化学习智能体与调度启发式方法和求解器进行比较的功能 [16, 第 2–4] 节。其示例性评估报告了调度结果和求解器差距。这为基于参考方案的生产决策评估提供了先例。Factory-simulator 关注采购、生产和履约所产生的财务后果，包括应收款和库存残值、历史记录分支以及可配置披露。这些是文献所述研究范围上的差异，并不意味着 schlably 无法扩展。mango 提供智能体角色和通信基础设施，包括外部模拟器耦合 [25, 第 3–4] 节。它提供了互补的架构背景；尚未测试与 Factory-simulator 的集成。

发表状态于2026年11月14日进行了核查。EconEvals [14] 被列为 EC 2026 已接收论文之一；其一位合著者的发表列表区分了会议摘要与完整工作论文。此处比较采用完整的 arXiv v4 文本。来源：https://ec26.sigecom.org/program/accepted-papers/ 和 https://yannai.gonch.name/scientific/research.html。参考文献 [1,2,5,10,12] 保留其 arXiv 引文，因为尚未核实有替代发表版本。

InventoryBench 将所比较的方法应用于相同的预生成需求和提前期实现值 [12]。其归一化奖励采用乐观上界，而非经证明可达到的实例最优值。语言模型程序可以获取产品背景信息和日期，而基础运筹学程序不使用这些文本 [12, 附录 E]。这些比较考察的是互补的程序，并未在信息使用完全相同的条件下隔离算法效应。AIM-Bench 报告称，缺货率相近，但与最优订货的偏差各不相同 [2, 发现 V]。这表明有必要考察单一运营指标以外的决策；不过，该比较并不涉及利润或付款时机。

MLPro-MPPS 描述了可复现的仿真、训练，以及在不同学习方法间的复用，为受控比较提供了支持 [19]。传感器可能无法获取组件状态，这为受限观测提供了先例。但这两者都没有确立基于记录历史进行分支的机制。EconEvals 在明确假设智能体笔记忠实反映实际情况的前提下对其进行分析 [14]。我们记录的行动不会揭示控制器的内部推理。

S3. 场景目录

以下目录介绍了全部 23 个随附情景。定义取自代码库中的 docs/library/scenario-library.md 和随附配置，而非示例性微型工厂。所有情景均运行 52 个周期。通用工厂初始现金为零，材料和组件约够三个周期使用，成品库存约够四分之一个周期使用。信用额度为 80000。配送中心的初始库存为：22.500 个工业控制器、23.750 个智能监视器和 20.000 个传感器网关。初始时没有供应商订单。各情景会在说明中列出的地方覆盖这一初始设置。

建议的诊断方法指出了需要检查的机制，但不保证这些机制会产生特定方向的影响。在使用这些案例提出性能主张之前，研究应先明确控制器、匹配的对照组和随机种子样本。

S3.1 基线

1. 稳定市场

52 个周期内需求保持不变，且没有额外干扰。

建立共同的运营参考方案。检查履约、利润、库存和资源利用率。

S3.2 需求

2. 季节性需求

四个各持续 13 个周期的季节分别将需求乘以 0.2、0.6、6 和 0.4。

检查采购和生产是否在高峰期前建立库存，同时避免耗尽仓库空间。跟踪调拨和销售损失。

3. 增长

需求每个周期增长 5%，达到初始水平的约 12 倍。

研究需求增长何时超过产能或流动性。跟踪销售损失，以及生产和采购量。

4. 衰退

需求　　每个周期下降 12%，达到初始水平的约 0.15%。

研究控制器何时减少采购和生产。跟踪滞销库存和期末残值。

5. 需求激增

在第 14–41 周期，智能监测器需求会升至通常水平的八倍，且没有提前通知。

检查对暂时性特定产品峰值的适应情况，包括未完成订单和竞争性资源需求。

6. 需求阶跃

从第 11 个周期起，所有产品的需求都变为通常水平的六倍 d.

比较应对永久性需求变化与临时需求激增时的响应。跟踪履约情况和产能。

7. 产品生命周期

传感器网关的初始状态不含传感器模块或成品库存。S形需求曲线在第 6 期前后升至正常水平的三倍，并于第 20 期结束；陡峭度为 1.4。

考察上市准备，以及淘汰前停止采购的时机。按需求阶段和订单批次检查网关销售情况，并查看剩余投入品。与稳定市场的比较还会改变初始库存。

S3.3 供应商

8. 供应商停供

在第 10–41 周期，亚洲芯片供应商不交付任何货物，欧洲供应商也无法销售微芯片-b。未提供提前通知。

追踪交付失败和采购受阻带来的不同后果。检查物料可用性和下游履约情况。

9. 运输延误

原定于第 12–41 周期交付的首批供应商货物延迟 16 个周期。补货交付仍按原定时间表进行。

研究供应链暴露和延迟可得性。追踪迟到货物、未结承诺及由此造成的销售损失。

10. 质量危机

在第 9–40 周期，欧洲零部件供应商的验收率乘以 0.001。该供应商是 PCB 空白板和连接器引脚的唯一供应商。

检查被拒收的投入品如何沿依赖配方传递。记录接收数量、废品和未完成订单。

11. 材料短缺

在第 10–41 周期，两种芯片类型的最大采购量均变为零。

将采购限制与交付失败分开研究。跟踪被拒绝的采购和库存耗尽情况。

12. 价格变化

在第 8–39 周期内，微芯片采购量的成本在未收到通知的情况下为正常价格的四倍。微芯片-b 的价格保持不变；它只能在控制板替代配方中作为替代品。

考察配方替代，以及在预期价格变化时受信贷约束影响的提前采购。跟踪单价、采购数量、材料选择和付款时机，并同时关注利润。单价上涨时，总支出不一定会增加。

S3.4 生产线

13. 生产线故障

最终装配在第 18 周期完全失去条件，在第 22 至 37 周期没有常规工时或加班工时。

将状态恢复与某个没有可用工时的周期区分开来。检查维修、停机时间和交付损失。

14. 频繁故障

最终装配每个周期还额外面临 80% 的概率，状态会损失 60–90%；并且在第 34–41 个周期没有可用工时。

研究严重中断下反复进行的恢复决策。跟踪维护资源使用和销售损失。

15. 产能削减

第18–41周期内，SMT生产线没有常规工时或加班工时。

评估持续产能中断期间对中间库存及其他工序的依赖程度。

16. 产率下降

第7–40周期投产的SMT产品，合格品数量减少95%。基准产出率为0.975，下降后为0.04875。

检查产生的可用产出很少的资源消耗。针对对应的生产批次，分别跟踪投产数量、合格品产出和废品。仅在生产数量为正时报告良率；生产数量为零时，良率未定义。

S3.5 现金与空间

17. 现金紧张

初始现金为 −120000，比信用额度低 40000，且有四分之一期的成品。

在客户付款足以弥补承诺支出并恢复现金水平之前，采购仍不允许进行。研究发货优先级以及采购能力的恢复情况。

18. 仓库满载

主工厂已满，库存价值集中在 6347.380 个外壳、820.615 个 LCD 面板、1554 个连接器引脚和 2823 个铜线单位；其他投入品均不存在。

线束生产可以为缺失投入品腾出空间。检查腾出空间的动作，以及准时交付、延迟交付、销售损失和流动性。延迟交付减少的同时，履约也可能降低。

S3.6 信息

19. 预测

基准需求是隐藏的。预测覆盖六个周期：提前一个周期的误差最高为 20%，此后每增加一个周期，上界增加至多 4%，到六个周期时达到 40%。

使用不提供预测的隐藏需求控制，以单独考察提供预测的影响。直接与稳定市场比较也会改变参数可见性。在匹配世界中比较决策和业务结果；改进效果取决于控制器如何使用预测。

20. 通知

此情　　文章中将此情景标记为“提前通知的停机”；其代码标识符为 notice　　.

与 supplier-outage 相同的芯片来源中断发生在第 10–41 期，但从第 6 期起，两项限制都可见。

在相同随机种子下与 supplier-outage 对比，以研究提前四个周期披露的影响。检查采购变化和服务；不预设其会带来益处。

21. 隐藏组

需求乘以 2.5。需求参数、供应商交付和检验参数、生产线良率及故障参数均被隐藏。

使用一个保留 2.5 需求乘数且显示参数的对应方案。与稳定市场进行比较，会混淆需求与可见性。

S3.7 综合扰动

22. 区域危机

第 16–41 周期到货的亚洲供应商交付将损失。危机期间应到的首批交付将延迟 16 个周期。

检查交付损失与延迟之间的相互作用。区分已承诺现金、可用物料和最终履约。

23. 激增与故障

第14–41周期，智能监测器需求增至正常水平的八倍；而SMT生产线在第16周期完全失去产能。

研究产能恢复与需求响应之间的协调。比较零部件短缺、维修和客户优先级排序。

发布文档报告称，开发活动涵盖520个世界和3948次运行，包括种子0重复运行，涉及23种情景和随机种子0–19上的三个对照组。发布文档报告称，这3948次开发运行均未违反运行级检查要求，包括物料与财务平衡。有四个世界无法评分，因为参考优化器在其限制范围内未找到解。这些运行级检查并不能证明预期结果方向是否成立。文档列出了六项未通过的方向检查：衰退、产品生命周期、价格变化、仓库满载、预测和产率下降。每项检查都将指定结果与相同控制器和随机种子下的稳定市场进行比较。对于全时域优化器、滚动优化器和对齐优化器，衰退和产品生命周期要求结果严格下降；对于这些优化器以及完全预见参考基准，价格变化、仓库满载和产率下降要求结果严格上升；滚动优化器下的预测则要求结果严格下降。任何可用控制器–随机种子组合中出现持平或相反变化，都会导致方向检查失败。这些描述的是控制器的结果，并非对配置机制是否正确执行的测试。控制器变体见S4节。

单独的难度评估使用缓冲滚动优化器，并根据不同随机种子的配对校准得分差计算结果。与公式（S1）不同，该得分使用同一随机种子下稳定市场参考方案与不决策利润之间的差额作为共同分母。它

等于 1 减去情景特定的参考方案与控制器之间的利润差额，再除以该分母。对于被归类为涉及前瞻的情景，代码不提供难度判定。文档将产率下降报告为难度为“是”，尽管其方向检查未通过。因此，六项未通过的方向检查并不能说明有六个情景低于同一个难度阈值。现有的发布摘要未说明衰退、产品生命周期、仓库满载和预测这几种情景各自的难度结果；本文不对此推断结果。对于价格变化，代码将其归入前瞻类别，而非给出难度判定。

情景目录提供可配置的业务条件。应根据研究问题、控制器和信息制度选择其建议的诊断方法。声称难度经过校准的研究需要单独规定校准方案。现有开发摘要不包含历史上失败的控制器–随机种子记录；因此，其各自原因仍未解决。该软件的开发活动未针对本文重复进行。定义和报告结果见代码库中的 tests/stress/playing.py、tests/stress/report.py 和 docs/benchmark/stress-test.md。针对产品生命周期、价格变化、产率下降、仓库满载和预测所选取的机制检查见 tests/integration/examples/test_scenario_library.py。其覆盖范围不应等同于完整的情景验证或难度校准。对于衰退，情景库集成测试不包含专门的需求下降机制检查；通用测试涵盖其配置和 API 注册。

S4. 优化参考方案与信息边界

控制器	可用信息	规划角色
无决策 no_决策	无需输入决策	动作列表为空；零分锚点。
完全预见参考基准 参考方案_优化器	生成世界和客户 时域内的订单	重复执行全时域线性规划/混合整数线性规划，随后 进行引擎检查。
全时域受限 全时域优化器	描述和当前 观测值	贯穿整个 剩余时域。
滚动控制器 滚动控制器_优化器	描述和当前 观测值	基于最长采购至 销售交付周期链加两个周期。
缓冲滚动 缓冲_滚动_优化器	与滚动控制器相同	期末库存乘数为 1.2，而不是 1.01.
预测对齐 对齐_优化器	描述和当前 观测	滚动窗口；来自开放订单的需求 订单和预测。
短视 短视_优化器	描述和当前 观测值	含终端条件的当前期计划 库存和在途库存估值。
知情诊断 知情_优化器	观测加上……的均值 原本隐藏的参数	从……恢复参数均值 非隐藏对应项；不会揭示 未来抽取结果。

控制器	可用信息	规划角色
精确整数参考控制 参考方案_optimizer(..., exact=True)	与完全预见相同，如同参考方案	每个建模周期中的整数选择；仅限于最多包含 100 个二元选择的小型程序。

常规基准放宽了每个项目第一周期之后的二元选择。其数学解经过取整，并转换为有序动作。可行性检查可能会拟合数量或删除动作。模型与执行不一定一致，因此较小的求解器差距并不是引擎层面最优性证书。精确整数控制消除了一种放松，但并不能证明每个引擎机制都得到了精确表示。

对于结束时间早于运行时域的规划窗口，滚动控制器和缓冲滚动控制器分别按重置成本的 1.01 倍和 1.20 倍评估符合条件的期末库存，评估上限为根据窗口之后的数据计算出的预期使用量。全时域受限规划覆盖剩余运行时域。在运行时域结束时，适用清单中的残值，而非中间窗口估值。此处的“受限”是指使用描述和当前观测值，而非已实现的未来世界。

对于 S5.1–S5.3 节中的受控演示，每次求解的时间限制为 30 秒。实现还设定了 1000 个节点的节点限制，以及绝对间隙停止容差 0.01 货币单位。记录的求解均找到了可行解，且未因达到时间或节点限制而停止。修订后的通知矩阵报告的间隙均不超过 4.55 × 10 13。经⁻ 检查的优化器运行报告中没有拟合或丢弃的动作。S5.2 节规定了其单独规划诊断所用的更严格容差。

受限控制器可以看到已披露参数和观测，包括通知示例中可见的中断持续时间。管理性世界端点不是信息安全接口。实验必须在其控制框架或部署中限制访问。代码库包含用于单提示、工具引导和协同规划的演示客户端；研究配置由研究人员负责。确定性轨迹展现的是引擎执行过程，而不是模型的私有推理。语言模型测试框架还应记录模型配置、提示词和每个动作，区分对已存储动作的回放与新的采样，并在适用时报告控制器的计算预算或令牌预算。

S4.1 指定决策评估研究

以下协议说明现有引擎和控制器接口如何支持未来的研究。它并非大语言模型或多智能体系统实验的报告。在查看比较结果之前，先定义目标和时域。明确期末估值和每项服务指标，包括其分母是按订单、单位还是价值计算，以及涵盖收到的订单、到期订单还是已完成的发货。报告物理或财务单位。通知演示中统计了需求为 40 个监视器单位且准时发货的数量；这并不表示有 40 个不同订单。在 Vending-Bench 中，工具会使模拟时间推进不同的时长，消息上限可能导致不同的经过时域 [1]。因此，研究应区分决策次数、模拟时长和计算预算。

进行控制器比较时，应固定工厂规格、情景和外生世界。使用一组定义好的随机种子进行跨世界推断。说明已披露参数、预测可用性和允许的端点。观测受限比较器获得与受测控制器相同类型的描述和观测，并拥有等效的可行性检查机会。然而，其轨迹仍可能产生不同的观测。不同动作之后的实际状态无需完全相同。

报告比较器规划窗口、假设和求解器预算。全时域观测受限比较器根据披露信息估计未来状况；它无法看到已实现的未来。知情诊断恢复隐藏分布均值。全前瞻参考方案接收已实现世界和未来客户订单。其结果用于诊断，并不保证在受测控制器所拥有的信息下能够达到的最优结果。应将这些区别与任何归一化得分一并说明。

信息干预会在保持决策程序和物理世界不变的同时改变信息披露。比较不同算法属于另一种对比。若要研究这两个因素，应分别为其指定匹配条件，并在相关情况下分析它们之间的交互作用。将受限控制器的利润从完全预见参考基准的利润中扣除，本身并不能区分信息效应与规划或执行效应。

对于分叉，应指定干预期间以及哪些动作会发生变化。恢复或有意重新定义外部控制器的记忆和通信状态；引擎会重现工厂历史，但不会重现任意控制器状态。说明后续动作是回放还是重新生成。除了财务记录，还应检查被拒绝的操作和物料平衡。由此产生的差异描述的是模型下的该延续方案；它既不能揭示语言模型的内部推理，也不能确立跨世界的策略排序。

存档控制器配置和提交的动作。对于随机控制器，除了匹配世界之外，还需要重复运行或采用其他合理的不确定性处理。计算预算和 API 成本应分别报告，或纳入明确声明的研究目标；引擎计算的工厂利润不会自动计入外部大语言模型服务的费用。记录求解器状态和得分排除项。所展示的流程提供了执行证据；工业应用则需要针对预期的工厂和决策目的进行验证。

S4.2 财务解释与终止条件

规划期利润是公式（1）中由模型定义的财务结果。初始现金需予以扣除；初始库存的购置价值不计入成本。仓库中剩余库存按指定的残值率估值。在制品和在途货物不计期末残值；已发生的成本仍计入。未结清的客户应收款按其全部名义价值计入，包括时域结束后到期的付款。尚未交付的供应商数量无需付款，且不计入终止资产或负债价值。在期末结算时，所有已到交付周期但尚未完成的客户订单，均按未发货数量计收缺货销售损失惩罚；尚未到期的订单不产生收益，也不产生期末惩罚。此前发生的成本和惩罚仍保留在分类账中。比较不同控制器时，应固定初始条件和估值规则。若 V0 是对初始仓库存货的共同估值，且 π′ = π − V0，则 π′1 − π′2 = π1 − π2。对控制器、不决策基线和参考方案均扣除 V0，也不会改变公式（S1）。绝对结果会发生变化，因此解读报告中的规划期利润时，必须考虑所述禀赋约定。比较不同情景时，还需考虑初始禀赋、时域和估值假设的差异。不同基准的终止约定也不相同。OR-Gym 的库存任务不对剩余库存计入终止价值 [10]；InventoryBench 在其既定目标中将销售利润减去持有成本后求和，不设单独的终止库存估值项 [12]。Vending-Bench 则将库存按批发价值计入最终净资产 [1]。因此，这些基准的数值财务得分不能直接互换。

无决策基线 π0 和完全预见参考基准 πr 定义了诊断评分

$$s = (\pi - \pi_0) / (\pi_r - \pi_0). \qquad \text{(S1)}$$

当 πr > π0 时，0 表示基准利润，1 表示参考利润；否则该评分未定义。由于参考方案采用启发式方法，评分可能超过 1。该评分既不是最优性百分比，也不是信息价值度量。

S5. 演算示例与数值证据

S5.2 为正文第3.1节（提前通知）提供支持；S5.1 为第3.2节（替代装配）提供支持；S5.3 为第3.3节（可检验最优解）提供支持；S5.4 和 S5.5 为第3.5节（实时 LLM-MAS 集成和匹配世界参照）提供支持。所有分发的演示记录均基于已确定的 1.0.0 快照生成。

S5.1 订单履行与分叉

第一个示例使用没有持续需求的小型工厂，初始库存为空，现金为 1000。一个包含四台显示器的订单须在第 4 周期交付，宽限期为一个周期，付款条件为一个周期。每块电路板需要两枚芯片，每台显示器需要两块电路板。单位成本分别为：芯片 0.5、电路板生产 1、显示器生产 2、发货 1；显示器价格为 50。逾期交付罚金为每单位每周期 2，缺货销售损失惩罚为每单位 10。延迟发货会产生 4 × 2 = 8；无决策运行在最终结算时会产生该罚金以及 4 × 10 = 40 。生产线每周期产能为十小时，每块电路板耗时 0.1 小时，每台显示器耗时 0.5 小时。供应商提前期和生产提前期均为一个周期；产出率为 1，损耗、仓储费用和利息均为零。

结果	原始计划	延迟分叉	无决策
发货期	4	5	None
期末现金	1172	964	952
应收款	0	200	0
惩罚	0	8	48
利润	172	164	−48

第 3 个周期的分叉保留第 1–2 个周期，并将显示器组装从第 3 个周期移至第 4 个周期。发货规则保持不变。相等性检查涵盖已关闭的前缀、重复的父运行、回放后延续方案，以及将该分叉与相同替代方案的新运行进行比较。隐藏生产线良率会改变其在控制器描述中的存在状态，但对于这一固定策略，物理记录保持不变。

七次运行通过了汇总物料与财务核对。前缀摘要为 f5400475079e261332723e0c7279f64528fe6671b136a3ebba48d57e572e4097。完整父运行、重复父运行以及回放-延续方案的摘要为 b0354f7459b7dd998726c4992550b8eec89e7b506f3882c65de29c43db00d6a1。序列化采用 json.dumps(value, default=str, sort_keys=True, indent=2)，并编码为 UTF-8。前缀哈希涵盖包含两行周期数据的列表；完整哈希涵盖期间响应对象。相关过程位于代码库中的 scripts/reproduction/article/run_demonstrations.py；记录位于数据归档中的 examples/order-fulfilment/。

S5.2 提前通知

十周期工厂、随机种子和事件安排详见第3.1节。第3至第6周期的采购受阻要么在发生时披露，要么在第1周期披露。主示例采用的仓储费用为每个芯片每周期0.0023、每块电路板每周期0.0090、每台显示器每周期0.0270。这些

费用约为重置成本的0.45%（芯片为0.46%）；它们是构造性输入，并非工业校准估计。“类库”这一标签指的是代码库 docs/benchmark/optimizers.md 中记录的、约为重置成本0.41–0.51%的存储费率；该示例并未复现所有库工厂参数。匹配世界仅在移除中断可见日期后保持一致。全部34次矩阵运行以及额外的受控延续均通过了物料和财务对账，所有提交的决策均已接受。

表 *S5.2a*。正向存储费用。各单元格报告准时交付数量（共 *40* 件）及利润（货币单位）。

控制器	开始时披露	第1期通知
无决策	0/40；-388.5680	0/40；-388.5680
滚动控制器	24/40；919.1720	28/40；1130.9552
缓冲滚动	24/40；919.1720	40/40；1732.4280
全时域受限控制器	24/40；919.1720	40/40；1766.0840
完全预见参考基准	40/40；1766.0840	40/40；1766.0840

在提前通知且收取正向存储费用的情况下，缓冲滚动和全时域受限控制器均按时发运全部40台监视器。缓冲滚动产生96的材料付款和3.572的仓储费用，期末残值回收为8。全时域受限控制器产生56的材料付款和1.916的仓储费用，残值回收为零。收入、生产和运输成本相同。因此，利润差额为40 + 1.656 − 8 = 33.656。两者的策略在规划窗口和期末库存估值方面均有所不同，具体如S4节所述。在此实例中，全时域受限控制器与完全预见参考基准取得了相同的实际结果。

表*S5.2b*。零仓储费用。单元格报告准时交付数量（共*40*件）及以货币单位计的利润。

控制器	开始时披露	第1期通知
无决策	0/40；-386.4000	0/40；-386.4000
滚动控制器	40/40；1768.0000	40/40；1719.2000
缓冲滚动	40/40；1768.0000	40/40；1719.2000
全时域受限控制器	24/40；920.0000	40/40；1768.0000
完全预见参考基准	40/40；1768.0000	40/40；1768.0000

将 JSON 值标准化后，无论是无决策、滚动策略还是参考方案，在两种信息披露机制下，零仓储费运行的档案记录均得到精确复现，包括期间记录、最终结果和生成的世界。滚动策略在无提前通知时获利 1768，在有通知时获利 1719.2，且两种情况下均交付全部 40 台显示器。在无通知运行中，它在第 2 期购买了 80 个芯片，此时阻塞尚不可观测。对记录的第 2 期规划问题重新求解，并将采购量固定为 16 或 80 时，得到的目标值相同，均为 1081.44。仅将这项采购改为 16，同时保持第 2 期的其他动作及之后的滚动策略不变，会使准时交付量降至 24，利润降至 920。因此，零成本下滚动策略的结果取决于如何从价值相同的规划解中进行选择。

仅对芯片收取正费用（每单位每周期 0.0001）会改变返回的无通知计划，但不会消除规划中的平局：芯片可以转化为电路板，而无需支付仓储费。采用第3.1节中使用的正费用时，记录的无通知第2期方案中，采购量为 16 时的目标值为 1079.856，采购量为 80 时的目标值为 1079.5984。这表明在该方案中，这两种选择之间存在严格偏好，但并不能证明完整方案具有唯一性。

表*S5.2c*。滚动敏感性。除非另有说明，费率均为每周期重置成本的百分比；单元格报告*40*次交付中的准时交付数量及利润。

储存费率	费用：芯片／电路板／显示器	开始时披露	第1期通知
Zero	0 / 0 / 0	40/40；1768.0000	40/40；1719.2000
仅芯片：0.0001	0.0001 / 0 / 0	24/40；920.0000	40/40；1719.1744
0.02%	0.0001 / 0.0004 / 0.0012	24/40；919.9632	40/40；1735.8432
0.1%	0.0005 / 0.002 / 0.006	24/40；919.8160	40/40；1735.2160
0.2%	0.001 / 0.004 / 0.012	24/40；919.6320	40/40；1734.4320
约 0.45%	0.0023 / 0.0090 / 0.0270	24/40；919.1720	28/40；1130.9552
1%	0.005 / 0.02 / 0.06	24/40；918.1600	28/40；1129.6800
2%	0.01 / 0.04 / 0.12	24/40；916.3200	28/40；1127.3600
5%	0.025 / 0.1 / 0.3	24/40；910.8000	28/40；1120.4000

滚动敏感性扫描采用的仓储费用分别为重置成本的 0.02%、0.1%、0.2%、1%、2% 和 5%，并与约 0.45% 的主案例进行比较。不提供提前通知时，所有测试的正费用案例均交付 24 台显示器。提供通知时，测试的 0.02%、0.1% 和 0.2% 案例交付 40 台；主案例以及测试的 1%、2% 和 5% 案例交付 28 台。这些抽样费率无法确定一个普遍阈值。在 0.2% 的费率下，在记录的第 2 期规划问题中，仅将终端估值乘数从 1.01 改为 1.00，就会反转其对采购量为 176 和 32 的偏好。支持采购量为 176 的目标值差异分别为 +0.672 和 −0.768。这项诊断针对的是该固定规划状态；它并非对乘数进行的全流程消融。

六组记录的规划输入均已精确重建。在引入诊断约束之前，原始程序复现了记录中的目标值和当前采购量。42 次规划求解保留了发布程序的二元结构。诊断比较采用相对间隙目标 0、绝对间隙目标 1e-9 和比较容差 1e-7，并进行约束和整数性残差检查。这些结果确立的是规划程序及指定延续方案的性质，而非一般意义上的引擎层面最优性。Source 1.0.0 未作更改。完整记录、输入清单和求解器报告见数据归档的 notice/ 目录。复现脚本位于代码仓库的 scripts/reproduction/article/ 下（S1.3 节）。

另外12次运行将提前通知敏感性评估扩展至缓冲滚动和全时域受限控制器，仓储费分别设为每周期重置成本的0.02%、0.1%、0.2%、1%、2%和5%。这些运行

采用种子7、10个周期和30秒求解时限。所有运行均以已接受的操作完成，且物料与财务余额均已核对一致；它们的世界彼此匹配，也与相应的滚动运行相匹配。结合重复原始矩阵中的零值和类似图书馆的设置，两种控制器在测试的八种存储设置下均提供了全部40个监视器。在这个构造实例中，它们的利润差额介于32.07和50.00之间，计算时未进行舍入。这些抽样设置并不能证明该结果在连续区间内始终成立，也不能证明其适用于其他工厂。

表 S5.2d。等服务比较的提前通知敏感性。每一行中，两个控制器均按时交付 40 台显示器，共 40 台。费率为每周期重置成本的百分比；类似图书馆的行采用 0.0023、0.0090 和 0.0270 的费用，约为 0.45%。利润差额使用未四舍五入的数值。

仓储费设定（%）	缓冲滚动控制器利润	全时域受限控制器利润	利润差额
0	1719.20	1768.00	48.80
0.02	1735.84	1767.92	32.07
0.1	1735.22	1767.58	32.36
0.2	1734.43	1767.15	32.72
≈0.45（类似库）	1732.43	1766.08	33.66
1	1728.16	1763.76	35.60
2	1720.32	1759.52	39.20
5	1696.80	1746.80	50.00

S5.3 可独立验证的最优值

这个单周期微型实例初始有 0.001 台监视器和两笔各为 0.001 台的到期订单。价格分别为 50 和 70；运输成本为 1，未满足数量每单位将受到 10 的惩罚。没有持续需求、利息或期末残值回收。现金和信用额度均为零。正数库存的 0.001 个芯片和 0.001 块电路板满足配置验证，但在发货决策之前无法转化为可销售的监视器。生产在发货阶段结束后完成。

应用程序接口允许的数量精度最多为小数点后三位。验证接受 0.001，拒绝 0.0005，记录见 historical/v06-verification/environment.json。因此，非零库存只能分配给低价订单或高价订单；剩下的选择是不发货。引擎的三种结果为 −0.020、0.039 和 0.059。没有期初应收款、供应商订单或其他收入来源。在这里，维修不会产生成本，其他动作也无法提高同期收入。

更一般地，设 x 和 y 分别为发往低价客户和高价客户的发货量，其中 x + y ≤ 0 001 49 + 69 − 10 0 002 − − = 59 + 79 − 。发货和销售 0 020 79 × 0 001 − 0 020 = 0 059 损失带来的利润为 x y（. x y），x y。该利润至多为 . . .。任何额外的正成本都会进一步降低利润。将全部库存发往高价订单即可达到这一上界。因此，该证明也为假设的连续松弛提供了上界。

常规基准、精确整数参考和滚动控制器在引擎执行中的得分均为 0.059。包括基线和枚举备选方案在内的七次运行均通过

财务对账。不发货策略既作为无决策基线执行，也作为枚举备选方案执行。这解释了运行次数。结果见 examples/reference/micro-*.json。这是针对刻意构造的小规模案例进行的合理性检查，并非对场景库中参考最优性的验证。

S5.4 实时LLM多智能体系统集成

目的与配置

最终演示包括一次三周期小型工厂运行（A3）、一次52周期稳定市场运行（B3），以及两次52周期通知情景运行（C1和C2）。所有运行均使用模拟器随机种子7，并采用全新的控制器记忆。A3使用代码库中的示例固定配置；B3和两次C运行使用随附的电子工厂清单，以及稳定市场或通知情景。C1和C2的世界记录与描述记录完全一致。情景标签表示业务条件，而非经过校准的难度等级。这些是集成案例，并非用于比较效果的实验。

引擎采用S1节所标识的、未经修改的1.0.0源代码快照。历史演示客户端扩展了代码库中的协同示例；当前代码库包含其集成实现。规划器提出计划，采购角色审查采购事项，审查器则利用可行性反馈修订决策列表。同一个托管模型依次服务于各个角色；这是角色分离式编排，并非对独立训练模型的比较。客户端会收到披露的描述、当前观测，以及前两个已结束周期的摘要，其中包括现金、库存和已执行结果。管理性世界记录在执行后用于审计，不会作为额外的模型输入提供。客户端可以进行可行性检查；服务本身并不强制实施信息隔离。

2026年11月12日至13日的实时调用使用了 OpenAI gpt-6.1-terra，推理_effort=low、服务_tier=default 和 max_completion_tokens=8192。记录的返回模型标识符与请求的标识符一致。这些记录没有可用的不可变提供商快照后缀，因此保留了调用日期和返回的标识符。每个请求的输入上限为 16384 个以空白字符分隔的单词，并非令牌数。最终的每周期调用额度为五次，包括三次角色调用和最多两次额外审查器修订。客户端为每次运行都保留空白的初始记忆。完整的请求正文保留了提示词、角色上下文和设置；响应记录则保留了用量和返回文本。实时提供商重新采样预计无法重现归档决策。

客户端的历史变更

与公开示例相比，本地适配器会应用所选模型和 8192 令牌的输出上限；如果拟定的最终计划仍不可行，客户端还会追加最多两次审查器修订。附加指令要求根据未变更的观测和提供的验证结果，修订完整的有序决策数组。指令明确涉及固定设置和批次约束，以及现金、信用额度、材料、仓储费和生产线产能。如果两次修订均未成功，确定性过滤器会移除当前被拒绝的操作，并重新检查剩余的有序计划；如果先前已排程的生产仍有未解决问题，则会停止运行。最终四次运行中均未调用过滤器。两次 C 运行都需要三次额外的审查器调用，分别解决了不同周期的问题；没有任何周期需要两次额外修订。

本地传输对每次逻辑调用最多允许两次尝试，连接/读取/写入/连接池超时分别为 10/240/20/5 秒，并设置了包含等待时间在内的 600 秒逻辑截止时间。重试资格仅限于指定的瞬态故障，同时采用有界等待和持久预算预留；含糊的部分响应会停止执行。模拟器的决策提交不会被盲目重试。

最终系列未使用任何传输重试或超时，因此无法提供关于重试效果的经验估计。测试这些分支的是离线测试，而非这些实时运行。设置文件保留了继承的适配器字段，包括 timeout_seconds =120 和 retries=0；单独记录的传输_策略则规定了实际的本地 v3 传输限制。

观测结果与检查

四次运行均已完成：共 159 个周期、483 次模型响应，其中包括额外的六次审查器修订。每个周期的利润和最终利润均通过了现金、客户、供应商和商品余额核对；没有已执行的最终决策被拒绝。角色交接和修订规则均保持一致。这些检查核对的是记录的流量，并非独立验证每个引擎成本系数。已保存操作回放与期间记录、最终结果和世界相符（S5.5 节）；但它无法重现随机模型决策。表 S5.4a 区分了不同类型的证据。

下列财务数值以模型单位表示；发货率采用单位分母，并非订单数量率。时域结束前，未结订单不一定都已发货或已流失。

A3：规划期利润 328；初始现金 1000；期末现金 1328；应收账款余额 0；库存残值 0。订购单位数 12，已发货 8，准时发货 8，延迟发货 0，流失 4。发货量/订购量 66.67%；准时发货量/发货量 100.00%；按时交付/订单量 66.67%。

B3：规划期利润 552071.4520155；初始现金 0；期末现金 361309.7147587；应收账款余额 181597.1341；库存残值 9164.6031568。订购单位数 14398.221，发货量 11281.293，准时发货量 3125.294，延迟发货量 8155.999，损失量 2749.433。发货量/订购量 78.35%；按时发货量/发货量 27.70%；按时交付/订单量 21.71%。

C1：规划期利润 203626.7626826；初始现金 0；期末现金 -15502.6834614；应收账款余额 211964.414；库存残值 7165.032144。订购单位数 14398.221，发货量 7871.577，准时发货量 2607.455，延迟发货量 5264.122，损失量 6198.043。发货量/订购量 54.67%；按时发货量/发货量 33.12%；按时交付/订单量 18.11%。

C2：规划期利润 -7944.5234898；初始现金 0；期末现金 -119243.3940266；应收账款余额 94956.9375；库存残值 16341.9330368。订购单位数 14398.221，发货量 6053.889，准时发货量 2177.267，延迟发货量 3876.622，损失量 7976.837。发货量 / 订购量 42.05% ；准时发货量 / 发货量 35.96% ；准时发货量 / 订购量 15.12%。

一项结构审计涵盖了全部 483 条已记录的逻辑模型请求（A3：9；B3：156；C1 和 C2：各 159 条）。每条提供的描述都与运行起始记录相匹配，且每条观测都与引擎在相应周期的记录相匹配。在同一世界上运行的大语言模型多智能体系统和滚动运行均接收到了完全相同的起始描述。经解析的 JSON 消息负载不包含随机种子或世界字段，也不存在到达期晚于请求期的客户订单。在 C 的两次运行中，中断通知首次出现在第 6 期；中断持续 32 个周期，从第 10 期到第 41 期。这项检查针对的是已记录的信息路径，而非控制器之间的一般访问隔离或完整的信息等价性。代码库中的 scripts/reproduction/check_recorded¯inputs.py 无需模型调用或引擎执行即可复现这些检查；其当前报告位于 validation/repository-2026-11-19/input-audit.json。

C1 在第 6 期（信用额度）、第 12 期（可用产线工时）和第 40 期（数量）进行了额外的审查器修订；C2 则在第 2 期和第 6 期（信用额度）以及第 25 期（数量）进行了额外修订。C1 在第 29 期和第 31–34 期选择了空计划；C2 则在第 26 期、第 29–33 期和第 51–52 期选择了空计划。这些计划由模型生成，并非由过滤器强制产生。仅凭这些计划的存在，无法证明其意图或决策质量。C2 的现金余额降至 -80000 以下，但信用规则限制的是采购准入，而不是之后的每一笔借记（正文第 2.2 节）。库工厂对正现金采用 0.0003 的每期利率，对 80000 信用额度内的债务采用 0.0015 的每期利率，

对超额债务采用 0.004 的每期利率。C2 支付了 3783.1685 的利息，并获得了 527.2157 的利息收入。其首次跌破限额发生在第 6 期：一笔到期的 11278.3829228 重大付款使交货验收期间的现金余额降至 -80458.9960183。第 52 期，23420.1615 的客户付款不足以抵消 25721.51 的失销罚款、277.5305636 的仓储费和 275.8701 的利息，导致期末现金为 -119243.3940266。因此，信用准入检查并不能保证所有后续现金余额都具有更低的上界。应收款按名义价值计价；S5.1节中的零利息示例特意将付款时机与融资成本分离开来。

执行前已选择较低的推理设置，以确保此次集成演示不超出货币预算。该设置并非针对控制器性能进行优化，这些运行也没有比较不同的推理设置。

运行选择与离线检查

报告了全部四次最终运行，包括负利润C2。较早的配置和中断记录仍保存在档案和S7.2节中；开发序列并未估算未变更客户端的完成概率。

历史客户端检查包括272项通过的客户端/传输检查、5项报告错误回归检查和18项续接保护检查；这些检查不计入3430项模拟器测试。

历史客户端验证与代码库集成

较早的打包客户端在调整路径、导入项和预算初始化的同时，保留了历史提示词和决策规则。11月15日，该客户端在macOS arm64上使用Python 3.14.10、NumPy 2.5.3、SciPy 1.18.1和uv 0.13.7进行了验证；验证环境是新建的虚拟环境，并使用了已有的依赖缓存。在套接字被阻断的情况下，4个案例在159个周期内消耗了483条归档响应；重建的请求、决策、周期、结果和世界均相符，且没有余额违规。该客户端的213项选定测试和2项凭证门控测试特定于该版本，与较早选取的测试存在重叠，并非新增的模拟器测试。报告保存在validation/client-offline-2026-11-18/。修订后的数据ZIP文件不含客户端代码；请使用S1节中的代码库提交。对历史复现提交394f68c的检查于11月19日进行，148项已有选定测试和4项新增路径测试均通过；此外，全部4个离线案例和8次已保存操作回放也再次得到相符结果。报告位于validation/repository-2026-11-19/。未重复进行提供商调用，也未重复运行完整模拟器/压力测试套件。其他平台上的历史检查并不能验证当前代码库客户端的实时启动器。

2026年11月18日，刚解压的打包客户端在 macOS arm64 上的新虚拟环境中，使用现有依赖缓存，完成了一次三周期A3实时检查：在低推理强度下收到九次 gpt-6.1-terra 响应，利润为328模型货币单位，未发生重试、额外修订、过滤或余额违规，估算费用为0.04美元。记录、精确成本计算以及与历史A3的比较见 validation/client-live-2026-11-18/；此次检查不能替代表3中的A3，也不能证明随机可复现性、完成52周期或跨平台运行。

表 *S5.4a.* 集成证据及其范围。计数代表不同的检查，不应相加。

证据	范围与解释	记录 / 复现
实时演示	A3、B3、C1 和 C2：159 个周期，483 条响应；展示执行，而非控制器排名。	integration/2026-11-12/ 和 integration/2026-11-13/；historical/integration/；S5.4 节
精选代码测试	213 项选定测试，外加两项针对其他示例的凭证门控测试；包括模拟失败处理。	validation/client-offline-2026-11-18/（历史选择）
记录输入审计	已核查 483 个请求，依据是描述和观测；记录的信息路径的证据。	scripts/reproduction/检查_recorded_inputs.py 中的代码库；validation/repository-2026-11-19/input-audit.json
已保存操作回放	八条轨迹，367 个周期，包括四个优化参考方案；检查引擎已存储决策的执行。	仓库已保存操作实用程序；validation/repository-2026-11-19/replay-check/；S5.5 节
归档响应执行	四条大语言模型多智能体系统轨迹，159 个周期；检查编排、重建的请求、决策和引擎输出。	validation/repository-2026-11-19/client-check/；仓库客户端 README.md
打包实时检查	一次 A3 次运行，3 个周期，9 个请求，2026年11月18日；用于测试实时路径，而非 52 周期或跨平台运行。	validation/client-live-2026-11-18/

数据、成本与复现

数据归档包含提示词、配置、决策、世界、周期、请求/响应日志、审计和报告，不含凭据或可执行代码。最终运行位于 integration/2026-11-12/ 和 integration/2026-11-13/；较早的尝试位于 historical/integration/。代码仅通过代码库提供。S1节列明了已发布代码、数据发布和归档校验和。新的托管模型调用需要凭据，且可能产生费用。归档完整性检查与引擎重放是两项不同的工作；代码库 README 提供了相关命令。

API 成本与模型单元工厂利润分开计算。按用量估算并四舍五入后，A3 为 0.02 美元，B3 为 4.48 美元，C1 为 4.15 美元，C2 为 3.81 美元；四舍五入前计算的总额约为 12.47 美元。开发系列中的成功响应总计约为 15.89 美元。确切用量、费率和计算过程保留在 historical/integration/ 中；保守的预算分类账包含 B2 超时带来的不确定性。这些估算并非经过核对的发票。模型单元利润不包含这些成本，因此这些运行无法证明在计入推理和协调成本后仍具有净经济收益。

S5.5 集成演示的匹配世界优化参考方案

在大语言模型集成运行之后，又指定了四条参考轨迹，以便为其结果提供背景：稳定市场和提前通知的中断场景下的滚动控制器与完全预见，

种子7，每种均运行52个周期。每种都使用未更改的1.0.0发布版本。观测受限滚动控制器接收披露的描述、观测结果和可行性检查响应。完全预见接收所有客户订单，以及匹配的已归档无决策运行中的已实现的未来世界；精确整数模式处于禁用状态。这是在额外信息条件下可实现的诊断结果，而不是经过认证的引擎最优解。已归档无决策结果和短视结果作为背景保留，而不是仅选作比较对象。

已归档的稳定市场世界与带通知中断世界仅在中断情况上有所不同。带通知中断世界新增了 apac-chips 供应商中断和 microchip-b 材料短缺，两者均从第 10 期开始，持续 32 个周期，并从第 6 期起披露。客户需求及所有其他世界字段均相同。比较这两个世界无法单独隔离提前信息的价值。为此需要使用相同随机种子的匹配供应商中断对照，但本次演示未运行该对照。

该协议为每次求解分配 30 秒，为每条参考轨迹分配 5400 秒，其中包括记录和验证时间。未完成的运行将不会获得规划期利润或得分，且不会自动延长预算；四次运行均在该协议规定的时间内完成。执行在一台交互式 macOS arm64 工作站上进行，同时运行的优化器最多为两个，期间短暂运行了一个已保存操作回放。系统负载已被记录。这些时间并非孤立的运行时间基准测试，新的有时间限制的优化可能会受到计算负载的影响。

已完成的参考运行通过进程内异步服务器网关接口测试客户端执行服务路线，并禁用了外部网络连接。运行过程中未产生模型调用或应用程序接口费用。

稳定市场，滚动控制器：利润 868970.31344082；期末现金 679674.99717842；按时交付/订单量 65.85%；得分 0.872800。52 份周期级求解报告，52 个可用解，0 次时间限制停止和 0 次节点限制停止；报告的最大绝对程序间隙为 0.0019675466464832425 模型单位。状态代码 [0]；固定二进制线性规划状态代码 [0]。包括审计和已记录动作的重放在内，总执行时间为 140.44 秒。

在稳定市场世界中，订单会在到期日期前一到两个周期到达。滚动控制器将未完成订单与预期未来需求结合起来；在其运行过程中，第7期至第52期这40个周期都记录到了延迟发货。尚未测试按时交付与完全预见之间出现差异的原因。订单和发货事实来自已归档记录；预期需求规划来源由发布版本代码设定。

稳定市场，参考方案：利润 1098988.810187；期末现金 848010.4440806；按时交付/订单量 99.58%；得分 1.000000。52 份周期级求解报告，52 个可用解，0 次时间限制停止和 0 次节点限制停止；报告的最大绝对程序间隙为 6.430782377719879e-05 模型单位。状态代码 [0]；固定二进制线性规划状态代码 [0]。包括审计和已记录动作的重放在内，总执行时间为 548.35 秒。

提前通知的停机，滚动控制器：利润 179458.79328454；期末现金 -35284.99310586；按时交付/订单量 36.74%；得分 0.554937。52 份周期级求解报告，52 个可用解，0 次时间限制停止和 0 次节点限制停止；报告的最大绝对程序间隙为 0.0013620160752907395 个模型单位。状态代码 [0]；固定二进制线性规划状态代码 [0]。包括审计和已记录动作的重放在内的总执行时间为 100.60 秒。

提前通知的停机，参考方案：利润 892273.6753527；期末现金 627924.9001535；按时交付/订单量 82.99%；得分 1.000000。52 份周期级求解报告，52 个可用解，2 次时间限制停止，0 次节点限制停止；报告的最大规划间隙绝对值为 59.132628857158124 模型单位。状态代码 [0, 1]；固定二进制线性规划状态代码 [0, 1]。包括审计和已记录动作的重放在内的总执行时间为 511.53 秒。

在通知参考期第 1 周期，混合整数线性规划报告了一个最优规划程序解，但随后固定二进制线性规划耗尽了剩余时间预算。在第 2 周期，混合整数线性规划和固定二进制线性规划均耗尽了各自的时间预算。求解器保持不变，并在每种情况下保留了可用的混合整数线性规划当前最优解，这是通过其固定线性规划超时回退机制实现的。第 2 周期的间隙 59.132628857158124 是绝对规划目标间隙，而非百分比或引擎级间隙。失败的线性规划细化保留在求解报告中，而未计为成功的线性规划求解。

每个52期参考方案均通过了平衡核对和公式（1）；回放与周期、结果和世界相符。描述和世界与相应的无决策及大语言模型情形相符；C1/C2共用一组参考对。求解器差距针对的是规划程序，而非引擎层面的全局最优性。

每个世界中的无决策利润均为−709331.2306；档案中的短视结果均为−432814.3424318。对于每个基线，两个世界中的全部52次决策提交、执行结果、收到的交付和财务账簿均一致。这解释了为何尽管事件和观测存在差异，利润仍然相同，但这并不意味着对于其他控制器而言，这些情景可以互换。公式（S1）定义于S4.2节：在稳定市场中，B3的得分为s=0.697555，滚动控制器的得分为0.872800；在带通知中断世界中，C1的得分为s=0.570027，C2为0.437927，滚动控制器为0.554937。两个分母均为正。带通知中断的得分使用了有时间限制的启发式锚点：两个完全预见规划步骤均达到求解限制，详情见上文。该得分是世界内利润的仿射缩放，保留了利润排序；其数值取决于锚点，并非最优性百分比，也不是对信息价值的估计。

无决策损失包括失销罚款376982.56、延迟交付惩罚274633.248、利息60586.1173和仓储成本24563.5156，扣除残值回收27434.2103。这个负基线使得即使利润为负，也可能得到正分数，例如 C2。对于固定的完全预见利润，基线越负，对应于固定利润差额的得分差就越小。因此，在表3中，与滚动控制器相比的直接匹配世界差值是主要的财务比较指标。B3、C1 和 C2 的差值分别为 −316898.86142532、 +24167.96939806 和 −187403.31677434。这些差值采用模型货币单位；它们并非扣除模型推理成本后的净收益。

表3以按时发运单位数除以所有订购单位数。B3 的按时发运率为 27.70%，C1 为 33.12%，C2 为 35.96% ；这些条件比率的分母各不相同。单凭任何一个比率，都无法确定哪个控制器创造了更多商业价值。滚动控制器和大语言模型控制器在规划流程和计算预算方面也有所不同。本次小规模集成演示并未估算它们在不同世界或模型响应下的预期性能。

references/ 目录包含轨迹、基线背景、求解器报告、审计、重放判定和原始协议。历史封装器不是修订后的存档的入口。请使用仓库输入审计和已保存操作重放命令；迁移期间并未重新优化这四个 52 期参考项。

重新执行存档中的动作

仓库已保存操作实用程序使用临时数据库并屏蔽外部套接字，不会导入模型客户端或优化器。它会检查描述、提交已存储决策，并逐一比较每个周期、最终结果和完整世界。运行专用世界端点仅在运行完成后返回完整世界。

2026年11月13日，解压后的软件包中的全部八个案例均通过了检查，其 JSON 规范化的周期、结果、世界和周期计数完全一致：A3（3 个周期）以及 B3/C1/C2/B-rolling/B- reference/C1-rolling/C1-reference（各 52 个周期），总计 367 个周期。一个增加已归档

A3 利润增加 1 未通过。573 个源文件与发布哈希值匹配。此次由作者运行的检查使用了现有的 Python 3.14.10/macOS arm64 环境，而不是全新安装的环境。检查执行了已存储决策，未进行模型调用或优化求解。

这验证的是已存储决策重放路径，而非新鲜的控制器效能或随机抽样。伴随效用修正和验证报告保存在 historical/replay/validation/；模拟器未作改动。

按照 S1.1 节中的说明完成安装和解压后，从代码仓库根目录执行已保存的决策，并将结果写入新的输出目录。这会使用代码库实现在已归档记录上执行：

```
uv run python -m scripts.reproduction.replay_saved_actions --case all --package-root "$PACKAGE"
--output data/replay-check
```

```
Individual cases can be selected with --case A3  B3  C1  C2  B-rolling  B-reference  C1-rolling ,,,,,,
or C1-reference.
```

```
The output directory must not already exist.
```

README.txt 提供当前的设置说明和各个案例的命令。请求/响应记录位于 integration/，求解器报告位于 references/。下文的历史作者运行检查和 Linux 检查，与发布持续集成及 11月19日的代码仓库验证各不相同。

应我们的要求，一位人工智能助手（Claude，Anthropic）在 Ubuntu 24.04.5 x86_64 上，于新建的虚拟环境中（uv sync --locked）运行了八次已保存操作回放。期间记录、最终结果和实现的世界完全一致。重新执行 34 个通知矩阵案例和 12 个仓储费敏感性案例后，相同字段也完全一致；差异仅在时间安排和求解器诊断信息，包括一个目标值和上界相差约 2 × 10 13。此历史检查‑针对的是较早的完整归档，而非修订后的纯数据发行包或当前代码库客户端。范围和命令详见 historical/linux-check/linux-check-report.md。

S6. 证据边界

实现声明和数值证据均指向已识别的源代码快照。相关软件比较基于一手资料；经典流程管理参考文献为建模依据提供支持。

本文没有人体参与者实验、经校准的工业工厂或已得到展示的应用案例。预期应用并非已完成的研究。S4.1 节提出了一份研究方案；它并未描述随代码库提供的试验活动。S5.4–S5.5 节提供了作者生成的集成记录和参考记录。这些记录描述了观察到的行为，但未对 LLM-MAS 的比较效能、信息价值或场景难度进行估计。

S7. 执行溯源

本节将分阶段验证和客户端开发历史与复现说明分开保留。归档报告保留了所有尝试；最终客户端系列则完整报告。

S7.1 分阶段本地验证

2026年11月8日至9日的本地验证累计覆盖了 macOS arm64/Python 3.14.10 上的 3430 个不同的非压力测试项目。13 项初始失败中，10 项涉及本地权限，3 项涉及过时的 uv。

修正环境后，12 项测试通过；在使用全新的 Python 安装目录后，另有 1 项通过，期间未更改源代码。这是分阶段覆盖，并非一次不间断运行，也不是托管式持续集成；压力测试未纳入其中。S1.2节列出了命令和报告，其中包含完整的测试选择、失败原因、环境和时间安排。

S7.2 客户端开发与运行选择

最初的小型工厂尝试因无法获得应用程序接口配额，在结束任何周期之前便停止了。原始本地适配器在 3 个周期内完成了 A；而 B、C1 和 C2 分别在 11、3 和 7 个已结束周期后停止，原因是最终计划不可行。支持修订的 v2 客户端完成了 A2，随后 B2 在 18 个周期后因第 19 个周期的采购调用超时而停止；该系列未启动 C 运行。超时处理 v3 客户端完成了 A3 和 B3。之后发生的 Decimal 序列化错误导致控制台报告中断，但 B3 的完整证据此前已保存。仅针对报告的修正恢复了 C1 和 C2，没有重复运行 A3 或 B3，也没有改变它们的决策逻辑。C 运行均已完成。此前的结果和中断情况均保留在随附记录中；此处完整报告最后一个系列，而不是挑选利润最高的运行。这段开发历程无法估算一个未作改动的客户端的完成概率。

S7.3 参考预检和配套脚本检查

两次参考预检在第 1 个周期之前，因 Decimal与JSON字符串比较而停止；只有审计包装器发生了变化。这两条记录仍保存在 references/ 中。

11月14日对 run_allocation.py 和 make_figures_1_4_6.py 的检查均在现有 macOS 环境中成功退出：七次分配运行与存档结果一致，周期和世界（不包括求解器时间安排）也与存档结果一致，图表则根据存档输入生成。范围和报告见 historical/entrypoint-checks.json。

参考文献

参考文献编号接续正文中的文章；为方便起见，此处转载参考文献 [1–27] 。

[1] Backlund A, Petersson L. Vending-Bench：用于评估自主智能体长期连贯性的基准。arXiv:2502.15840 [预印本]；2025年。https://doi.org/10.48550/arXiv.2502.15840。[2] Zhao X, Xie Y, Chen C, Sun Y. AIM-Bench：评估智能体型大语言模型作为库存管理者时的决策偏差。arXiv:2508.11416 [预印本]；2025年。https://doi.org/10.48550/arXiv.2508.11416。[3] Kravchenko K, Gruchmann T, Ivanova M, Ivanov D. 应对系统性中断引发的涟漪效应：来自 COVID-19 期间半导体短缺的实证证据。《现代供应链研究与应用》2024;6(4):354–75。https://doi.org/10.1108/MSCRA-03-2024-0011。[4] Guo T, Chen X, Wang Y, Chang R, Pei S, Chawla NV, 等。基于大语言模型的多智能体：进展与挑战综述。载于：第三十三届国际人工智能联合会议论文集。2024年。第8048–57页。https://doi.org/10.24963/ijcai.2024/890。[5] Quan Y, Liu Z. InvAgent：一种用于供应链库存管理的基于大语言模型的多智能体系统。arXiv:2407.11384v2 [预印本]；2025年。https://arxiv.org/abs/2407.11384v2。[6] Jannelli V, Schöpf S, Bickel M, Netland T, Brintrup A. 供应链中的智能体式大语言模型：迈向自主的多智能体共识寻求。《国际生产研究杂志》2025。https://doi.org/10.1080/00207543.2025.2604311。[7] Davenport TH, Short JE. 新工业工程：信息技术与业务流程再设计。《斯隆管理评论》1990;31(4):11–27。
https://sloanreview.mit.edu/article/the-new-industrial-engineering-information-technology-and-business-process-redesign/。

[8] Rummler GA, Brache AP. 改进绩效：如何管理组织结构图中的空白地带。第3版。Jossey-Bass；2012年。[9] Galbraith JR. 组织设计：一种信息处理视角。《Interfaces》1974;4(3):28–36。https://doi.org/10.1287/inte.4.3.28.[10] Hubbs CD, Perez HD, Sarwar O, Sahinidis NV, Grossmann IE, Wassick JM. OR-Gym：面向运筹学问题的强化学习库。arXiv:2008.06319 [预印本]；2020年。https://doi.org/10.48550/arXiv.2008.06319。[11] Hubbs C, Perez Parra H, Sarwar O. OR-Gym。0.5版。0 [软件]。PyPI；2022年。https://pypi.org/project/or-gym/0.5.0/ [访问日期：11月14日2026]。代码库：https://github.com/hubbs5/or-gym。[12] Baek J, Fu Y, Ma W, Peng T. 用于库存控制的人工智能智能体：人类-大语言模型-运筹学互补性。arXiv:2602.12631 [预印本]；2026年。https://doi.org/10.48550/arXiv.2602.12631。[13] Baek J, Fu Y, Ma W, Peng T. InventoryBench [软件]。GitHub；无日期。https://github.com/TianyiPeng/InventoryBench [访问日期：11月8日2026]。[14] Fish S, Shephard J, Li M, Shorrer RI, Gonczarowski YA. EconEvals：评估大语言模型智能体经济决策的基准与试金石测试。arXiv:2503.18825v4 [预印本]；2026年。https://arxiv.org/abs/2503.18825v4。[15] Fish S, Shephard J, Li M, Shorrer RI, Gonczarowski YA. EconEvals [软件]。GitHub；无日期。https://github.com/sara-fish/econ-evals-paper [访问日期：11月8日2026]。[16] Waubert de Puiseau C, Peters J, Dörpelkus C, Tercan H, Meisen T. schlably：用于基于深度强化学习的调度实验的 Python 框架。《SoftwareX》2023;22:101383。https://doi.org/10.1016/j.softx.2023.101383。[17] Mohammad Y. 运行受控实验。SCML 0.7.7 文档；无日期。https://scml.readthedocs.io/en/latest/tutorials/05.experiments.html [访问日期：10月29日2026]。[18] Mohammad Y. SCML：ANAC供应链管理联盟平台 [软件]。GitHub；无日期。https://github.com/yasserfarouk/scml [访问日期：11月8日2026]。[19] Yuwono S, Löppenberg M, Arend D, Diprasetya MR, Schwung A. MLPro-MPPS：用于可定制生产系统的高性能仿真框架。《Software Impacts》2023;16:100509。https://doi.org/10.1016/j.simpa.2023.100509。[20] Yuwono S, Löppenberg M, Arend D, Diprasetya MR, Schwung A. MLPro-MPPS：基于 Python 的可定制多用途生产系统。1.1版。0 [软件]。Zenodo；2023年。https://doi.org/10.5281/zenodo.8202434。https://github.com/fhswf/MLPro-MPPS [访问日期：11月8日2026]。[21] Behrendt S. prodsys：生产系统的建模、仿真与优化 [软件]。GitHub；无日期。https://github.com/sdm4fzi/prodsys [访问日期：10月29日2026]。文档：https://sdm4fzi.github.io/prodsys/Tutorials/tutorial_3_优化_一个_生产_系统/ [访问日期：11月9日2026]。[22] Drozdov S, Dadfarnia M. SimPROCESD [软件]。美国国家标准与技术研究院；2024年8月15日。https://www.nist.gov/services-resources/software/simprocesd [访问日期：10月30日2026]。[23] Guan S, Liu Y, Cao L. SupChain-Bench：为现实世界供应链管理任务评测大语言模型。载于：Liakata M, Moreira VP, Zhang J, Jurgens D，编者。计算语言学协会年会 ACL 2026 论文集。计算语言学协会；2026年。第7526–50页。https://doi.org/10.18653/v1/2026.findings-acl.371。[24] Guan S, Liu Y, Cao L. SupChain-Bench [软件]。GitHub；无日期。https://github.com/Damon- GSY/SC-bench [访问日期：11月8日2026]。[25] Schrage R, Sager J, Hörding JP, Holly S. mango：模块化的、基于 Python 的智能体仿真框架。《SoftwareX》2024;27:101791。https://doi.org/10.1016/j.softx.2024.101791。[26] Sargent RG. 仿真模型的验证与确认。《J. Simul.》2013;7(1):12–24。https://doi.org/10.1057/jos.2012.20。

[27] Wodecki A, Godlewski W, Chojna M. 工厂模拟器。版本 1.0.0 [软件]。GitHub；2026。
https://github.com/mchojna/factory-simulator/releases/tag/v1.0.0.