Variance and Resource Usage of Computational Experiments

Bachelorarbeit · Universität Basel

Nillan Sivarasa
27.07.2026

Aufbau

1. Einleitung
2. Planning & Benchmarking
3. Infrastruktur
4. Resultate
5. Diskussion & Fazit

Einleitung

Ausgangslage

- Computational Experiments vergleichen Algorithmen über gemessene Runs
- Bei Planning-Experimenten stehen Runtime, Coverage und Failures im Fokus
- Auf Shared HPC-Systemen können Scheduling, Node Placement und Interferenz die Messungen beeinflussen

Hauptfrage

Wenn derselbe Planner Command auf derselben Benchmark-Task wiederholt wird:

Wie stabil ist die gemessene Runtime?

Classical Planning mit Fast Downward

PDDL Task → Fast Downward → Plan oder Failure + Runtime

Planner-Konfigurationen:

UCS	einfache optimale Baseline
A* LM-cut	informierte optimale Search
LAMA First	Satisficing Search
Operator Counting	LP-basierte Heuristic mit CPLEX

Benchmarking auf einem Shared Slurm-System

Reliable Benchmarking braucht

- Resource Limits
- CPU Placement und mögliche NUMA-Effekte
- Isolation von anderen Jobs
- Measurement und vollständige Experimentdaten

Praktische Limitation

BenchExec mit vollständiger cgroup-Isolation konnte auf dem Cluster nicht verwendet werden, weil Slurm Jobs die nötigen Sub-cgroups nicht erstellen konnten.

Finaler Ansatz

Slurm Resource Requests + Apptainer Container + explizite Records der verbleibenden Limitationen

Experimentelle Infrastruktur

Pipeline

JSON Experiment → Slurm Array → Apptainer Container

Raw Outputs → CSV/JSON → Plots

Experiment Settings

```
{
  "tasks": 133,
  "domains": 5,
  "repetitions": 5,
  "limits": {
    "time": "5 min",
    "memory": "4 GB"
  },
  "cpu_counts": [1, 2, 4, 8, 16, 32, 64,
128]
}
```

Wichtige Metriken

- Mean Runtime
- Standard Deviation
- Coefficient of Variation (CV)
- Allocated CPU Time

Leitfragen

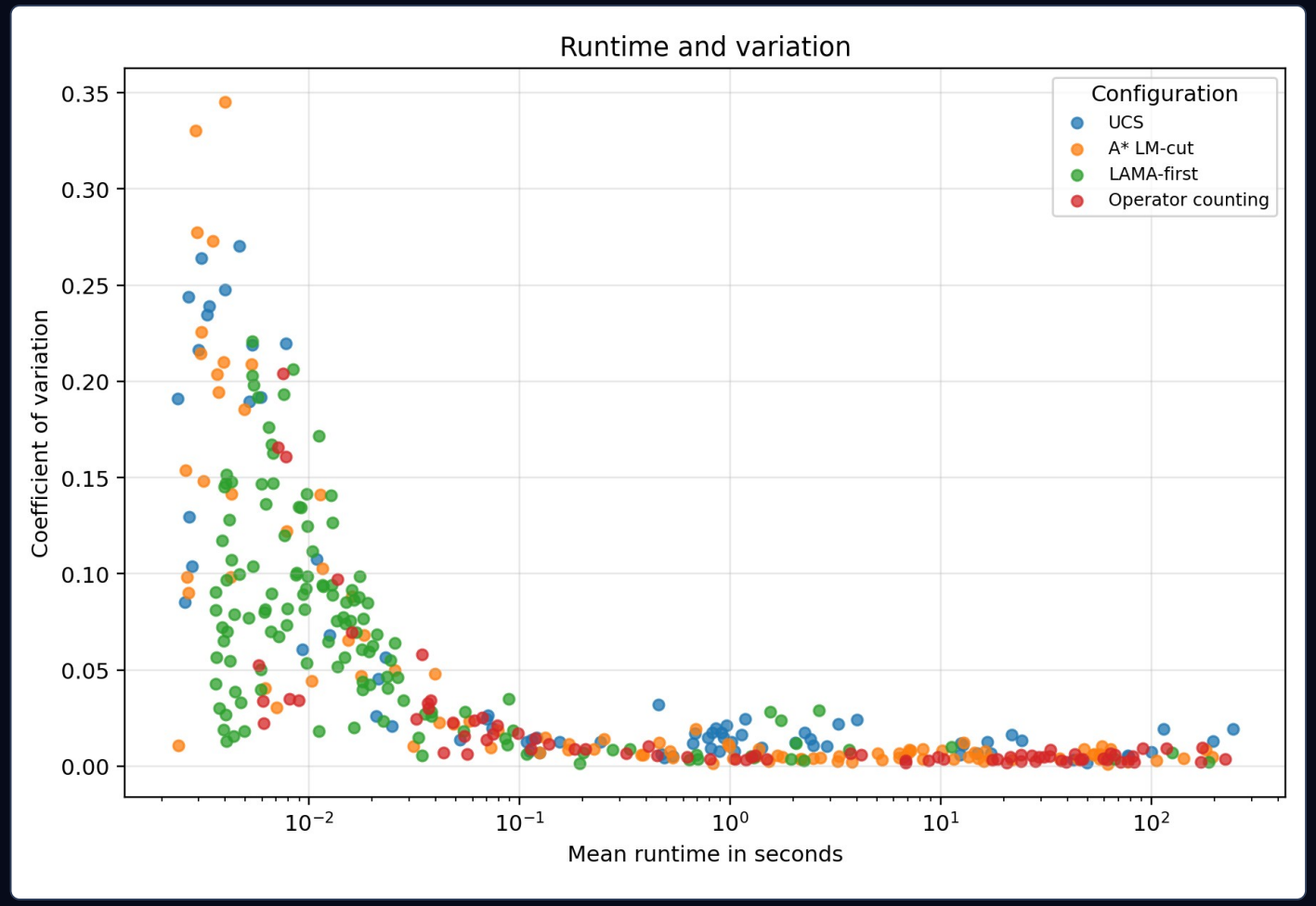
Q1 Wie stark variiert Runtime bei wiederholten Runs?

Q2 Ändert ein kombiniertes UCS + A* LM-cut Setup die Messungen?

Q3 Wie wirken Memory Binding, CPU Allocation und Ressourcen?

Q4 Welche Experimentdaten verbessern Repeatability?

Runtime Variation über wiederholte Runs



Interpretation

Hohe CV-Werte treten hauptsächlich bei sehr kurzen erfolgreichen Tasks auf.

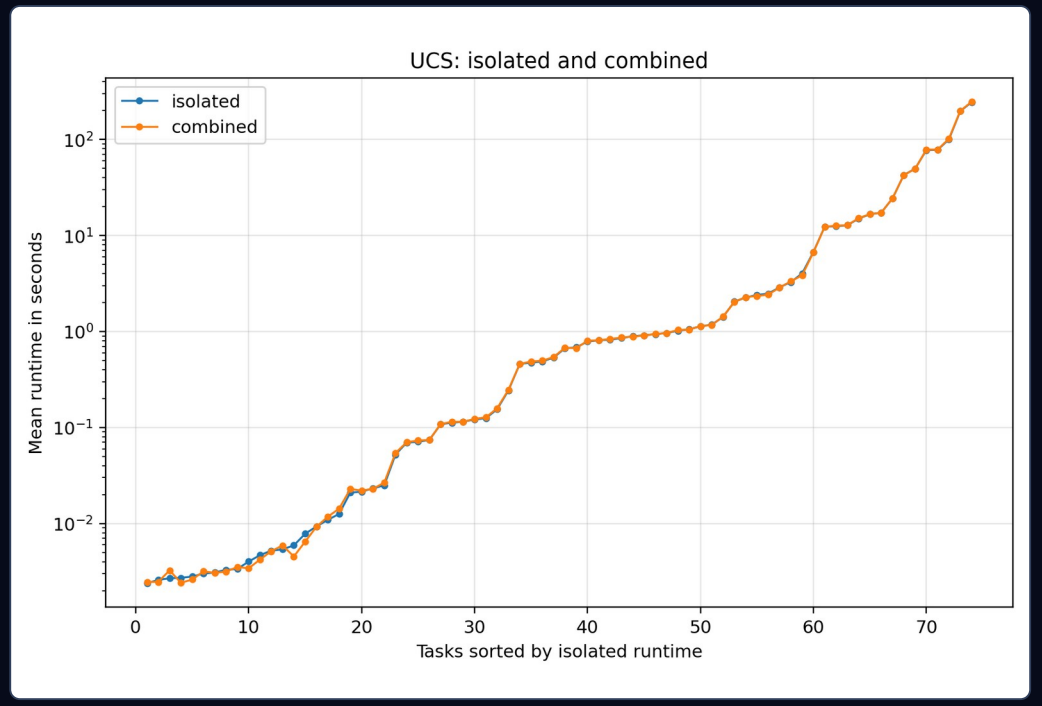
Bei Mean Runtimes über ungefähr einer Sekunde bleibt CV unter etwa 3%.

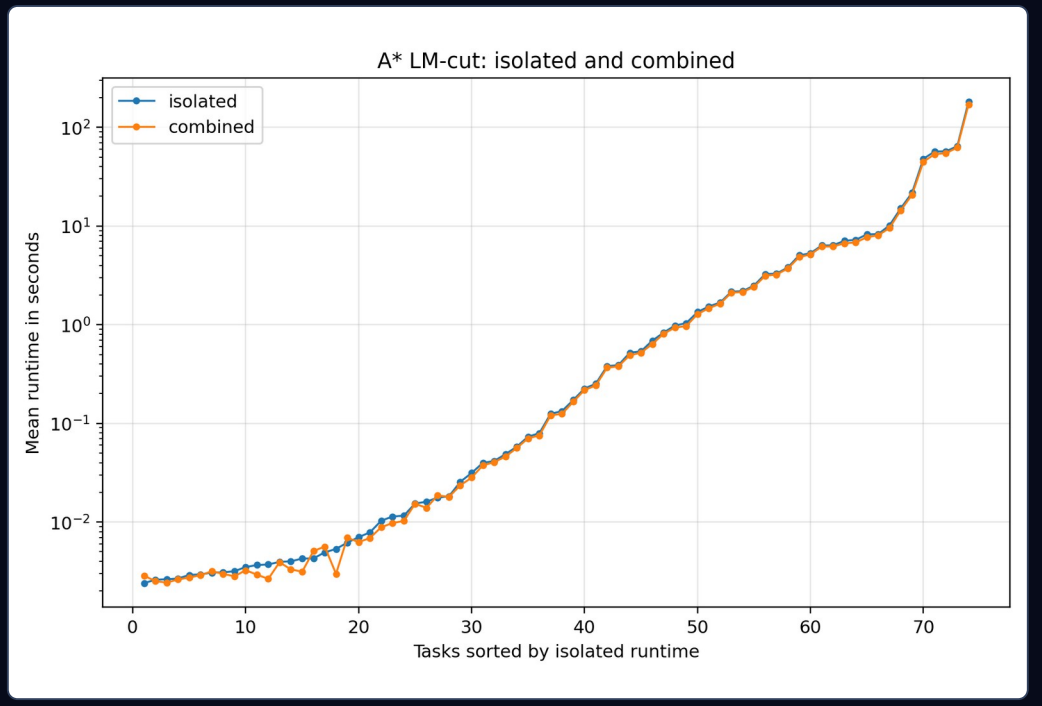
Eine globale Runtime-Verteilung vermischt Task Difficulty mit Measurement Variation.

Antwort

Varianz pro Task und Konfiguration analysieren.

Isoliertes versus Kombiniertes Experiment-Setup





Interpretation

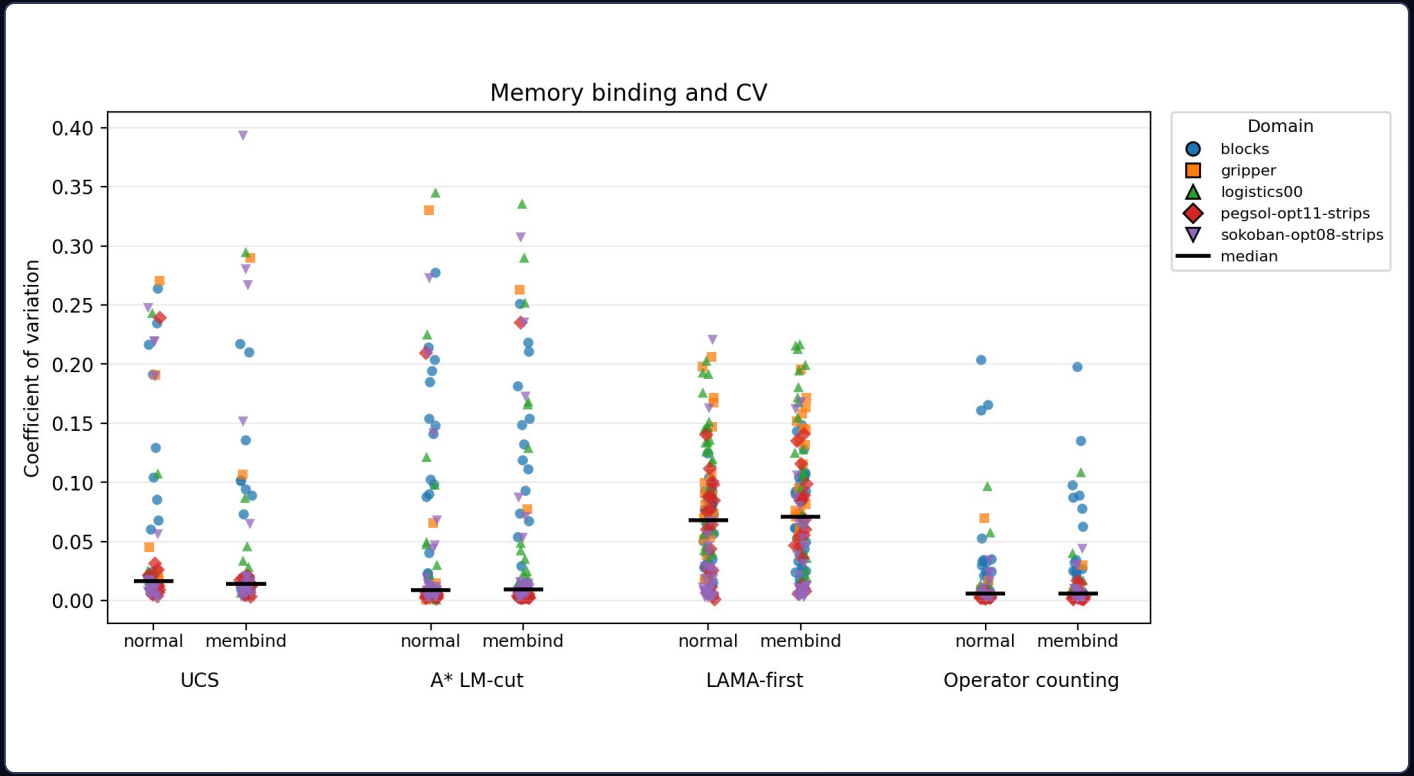
Isolierte und kombinierte Runs verlaufen sehr ähnlich.

Antwort

Einzelne Task-Werte ändern sich, aber nicht systematisch.

Kein klarer globaler Effekt auf Runtime oder CV.

Memory Binding



Interpretation

Keine Reduktion durch Memory Binding.

Median-CV-Werte sind grob ähnlich.

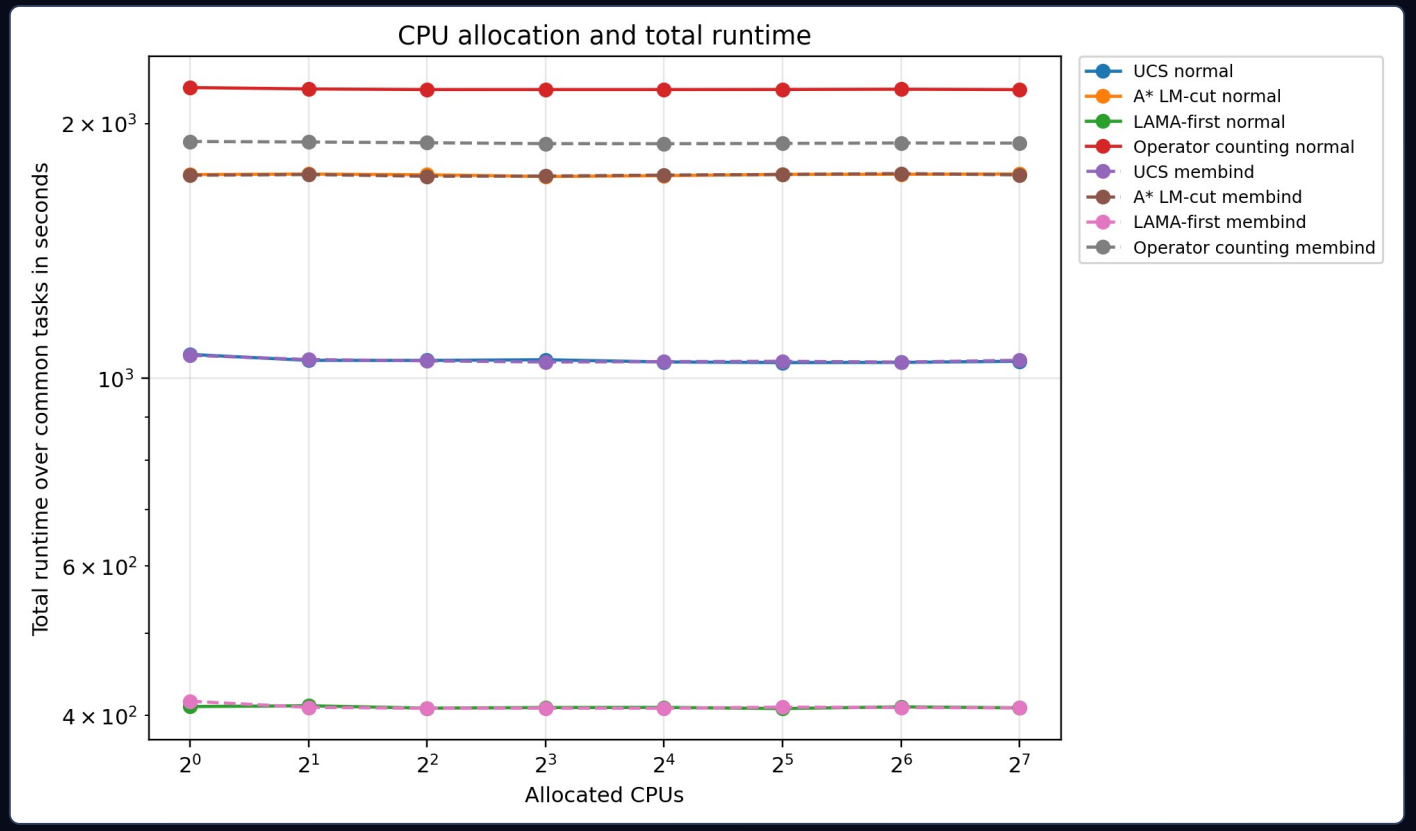
Setup Effekte sind kleiner als Task- und Konfigurationsunterschiede.

Antwort

Memory Binding zeigte im getesteten Setup keinen klaren Stabilitätsvorteil.

NUMA-Effekte werden damit nicht allgemein ausgeschlossen.

CPU Allocation



Interpretation

Fast Downward Commands sind Single-Process Runs.

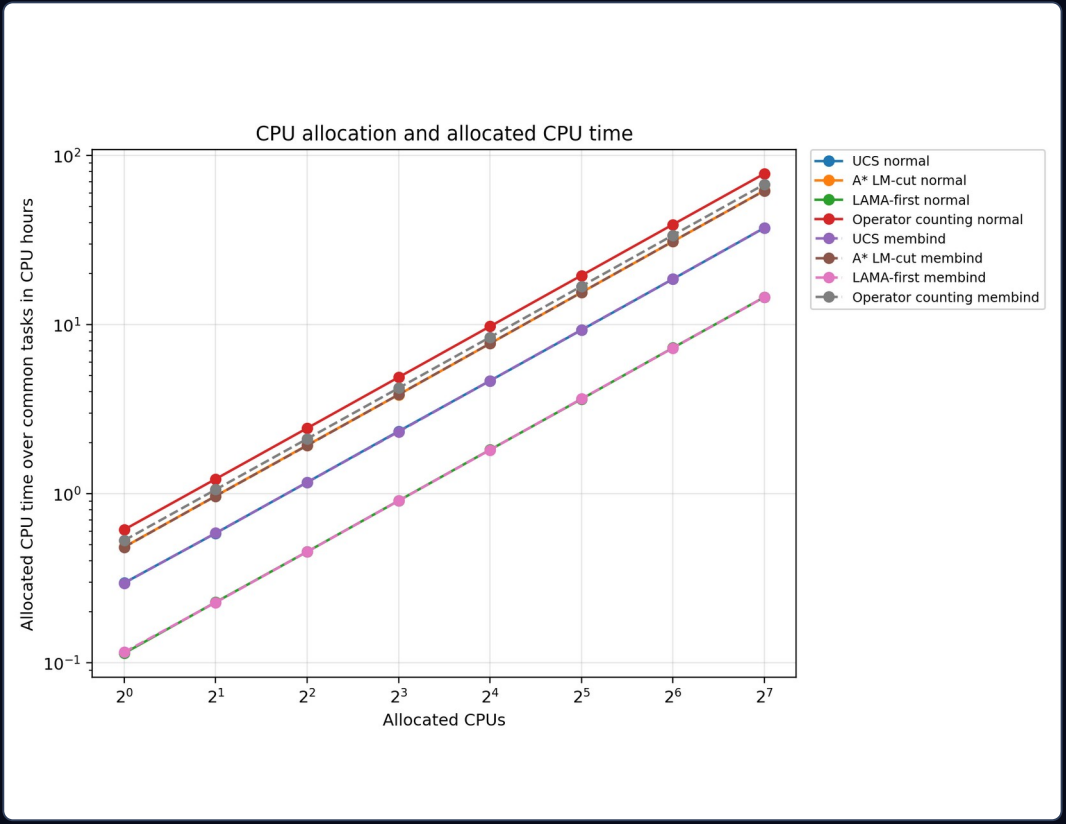
Runtime-Linien bleiben von 1 bis 128 Requested CPUs flach.

Antwort

Zusätzliche CPUs reservieren Kapazität ohne Runtime-Vorteil.

Nur so viele CPUs anfordern, wie genutzt werden können.

Resource Usage



Interpretation

Allocated CPU Time wächst fast proportional mit der Requested CPU Count.

128-CPU-Gruppen dominieren CPU Hours und das Estimated-kWh-Ranking.

kWh-Werte sind Schätzungen, keine direkte Energy Measurement.

Antwort

CPU Allocation beeinflusst vor allem die reservierten Ressourcen.

Records machen das Experiment nachvollziehbar.

Diskussion

Limitationen

- Keine BenchExec/cgroup-Isolation auf dem Cluster
- Resultate hängen von Tasks, Limits, Configurations und Cluster Setup ab
- Direkte Energy Measurement nicht verfügbar

Einordnung

- Container verbessern die Repeatability der Software-Umgebung
- Sie entfernen Scheduling-, Hardware- oder Interference-Effekte nicht
- Der klarste Setup-Effekt liegt im Resource Accounting, nicht in der Planner Runtime

Fazit

Antwort auf die Hauptfrage

Die Runtime ist bei längeren erfolgreichen Tasks relativ stabil. Hohe relative Schwankungen treten vor allem bei sehr kurzen Tasks auf.

Kernaussagen

- Kombinierte UCS + A* LM-cut Runs zeigten keinen klaren Runtime- oder CV-Shift
- Memory Binding verbesserte die Stabilität im getesteten Setup nicht klar
- Excessive CPU Allocation erhöht CPU-Time stark, ohne klaren Runtime-Vorteil
- Runtime Stability und Resource Accounting müssen getrennt interpretiert werden

Future Work

Mehr Repetitions · Direct Energy Accounting · Tests über Node Types oder Partitions