Adaptive Query Scheduling for Mixed Database Workloads with Multiple Objectives

Stefan Krompass[TUM], Harumi Kuno[HPL],
Kevin Wilkinson[HPL], Umeshwar Dayal[HPL], Alfons Kemper[TUM]

[TUM] Technische Universität München
Munich, Germany

[HPL] Hewlett-Packard Laboratories
Palo Alto, CA, USA

Problem statement

- n service classes (i. e., a set of queries)
- $n \cdot m$ objectives (multiple objectives per service class)
- $n \cdot k$ control knobs (to control service per class, e. g., MPL)

Search problem

Find control knobs settings to achieve objectives for all service classes

Problem statement

- n service classes (i. e., a set of queries)
- $n \cdot m$ objectives (multiple objectives per service class)
- $n \cdot k$ control knobs (to control service per class, e. g., MPL)

Search problem

Find control knobs settings to achieve objectives for all service classes

Difficulties

- Large search space
- Queries have different characteristics (resource requirements, variance in resource requirements)
- Service classes have different characteristics (start time, arrival rate, objectives)
- Contention among the queries unknown
- Non-linear relationships between objectives and the control parameters

Difficulties

- Large search space
- Queries have ents,
  variance in re
- Service classe arrival
  rate, objectiv
- Contention a
- Non-linear rel rol
  parameters

In this presentation: Present framework and experiments with algorithm to tackle the search problem

Solution approach

- Base: algorithm devised by Niu et. al: "Adapting Mixed Workloads to Meet SLOs in Autonomic DBMSs"
  $\Rightarrow$ Multi-class, single objective
- Extension: assume relationship between objectives is known in order to solve our problem

Workload Adaptation-Maximize Single Objective

- Goal: maximize overall utility (measure to quantify how well the system meets the objectives)
- Service classes $s_1, \ldots, s_n$, each with single objective
- Idea: assign system resources to service classes by controlling the number of queries a service class may run
- Service class s_i has control knob x_i
- Assumption: $\exists$"system cost limit" X where performance is maximized

Workload Adaptation-Maximize Single Objective

$$\underset{x_1,\ldots,x_n}{\text{maximize}} \; u_1\left(h_1(x_1)\right) + \cdots + u_n\left(h_n(x_n)\right)$$

$$\text{subject to } x_1 + \cdots + x_n = X$$

- Estimation model: control knob setting $(x_i) \rightarrow$ estimated performance
- Utility function: performance value $\rightarrow$ utility (positive if performance $>$ objective, negative otherwise; utility decrease faster for lower performance, utility increase slower with better performance)

Dominance

Definition

Objective *o* is *dominant* for a service class if a set of conditions satisfying *o* implies that the other objectives of this service class are satisfied as well.

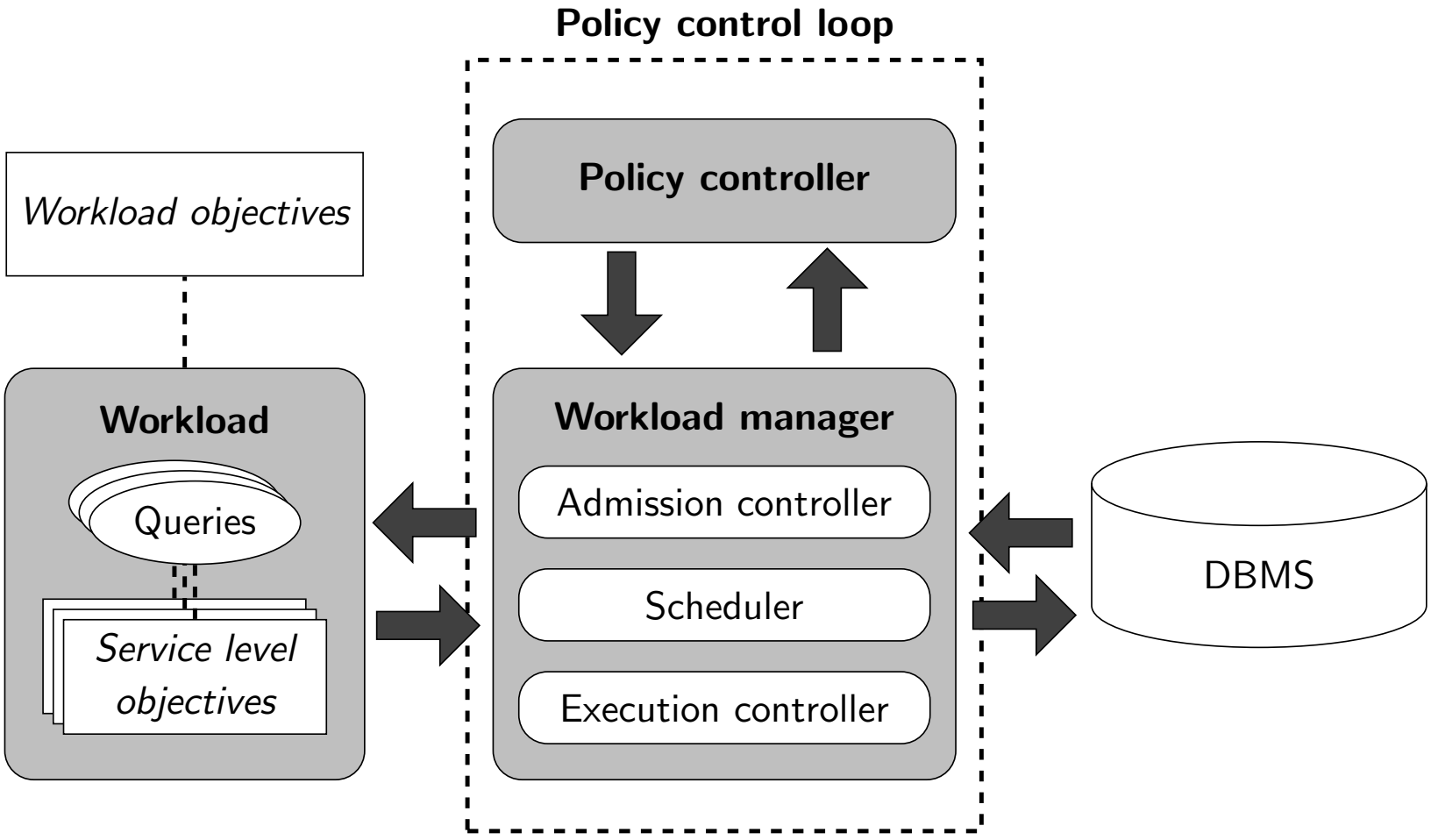
Note

- Dominance holds only for a specified range of control knob settings
- Dominance applies to objectives of a single service class only

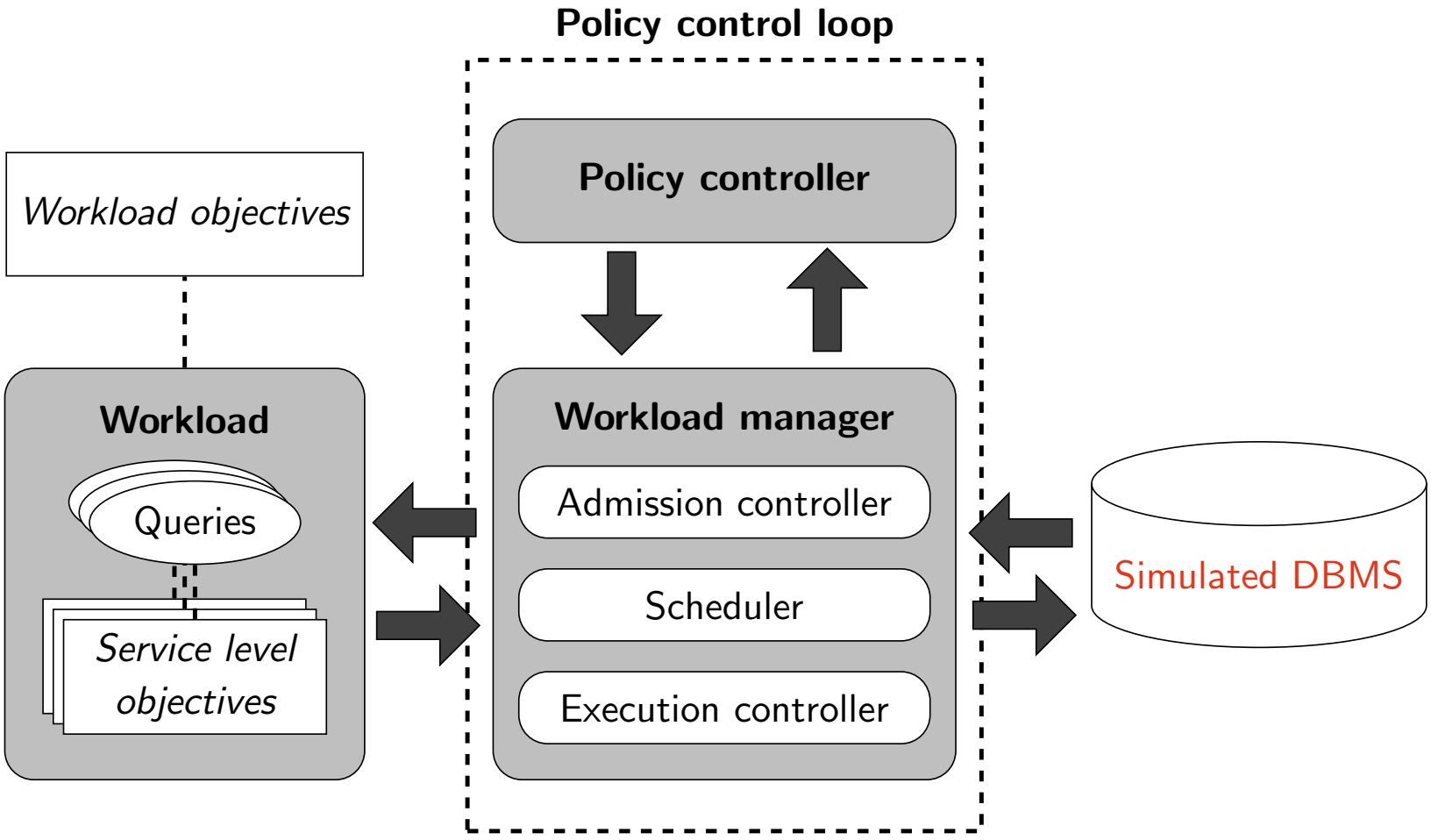
Example

If average response time requirement is satisfied, throughput is also

Framework


Policy control loop
Policy controller
Workload objectives
Workload
Queries
Service level objectives
Workload manager
Admission controller
Scheduler
Execution controller
DBMS

Framework


Policy control loop
Policy controller
Workload objectives
Workload
Queries
Service level objectives
Workload manager
Admission controller
Scheduler
Execution controller
Simulated DBMS

Why a simulator?

- Deterministic
- Repeatable results
- Experiment with varying workloads with varying characteristics
- Easily change system configuration
- Speedup

Experiments

Purpose of the experiments

- Two service classes, each with throughput and average response time objectives
- Control knobs: vary MPL for each service class
- Goal: find MPL settings where each objective is met

Experiments

Experimental setup

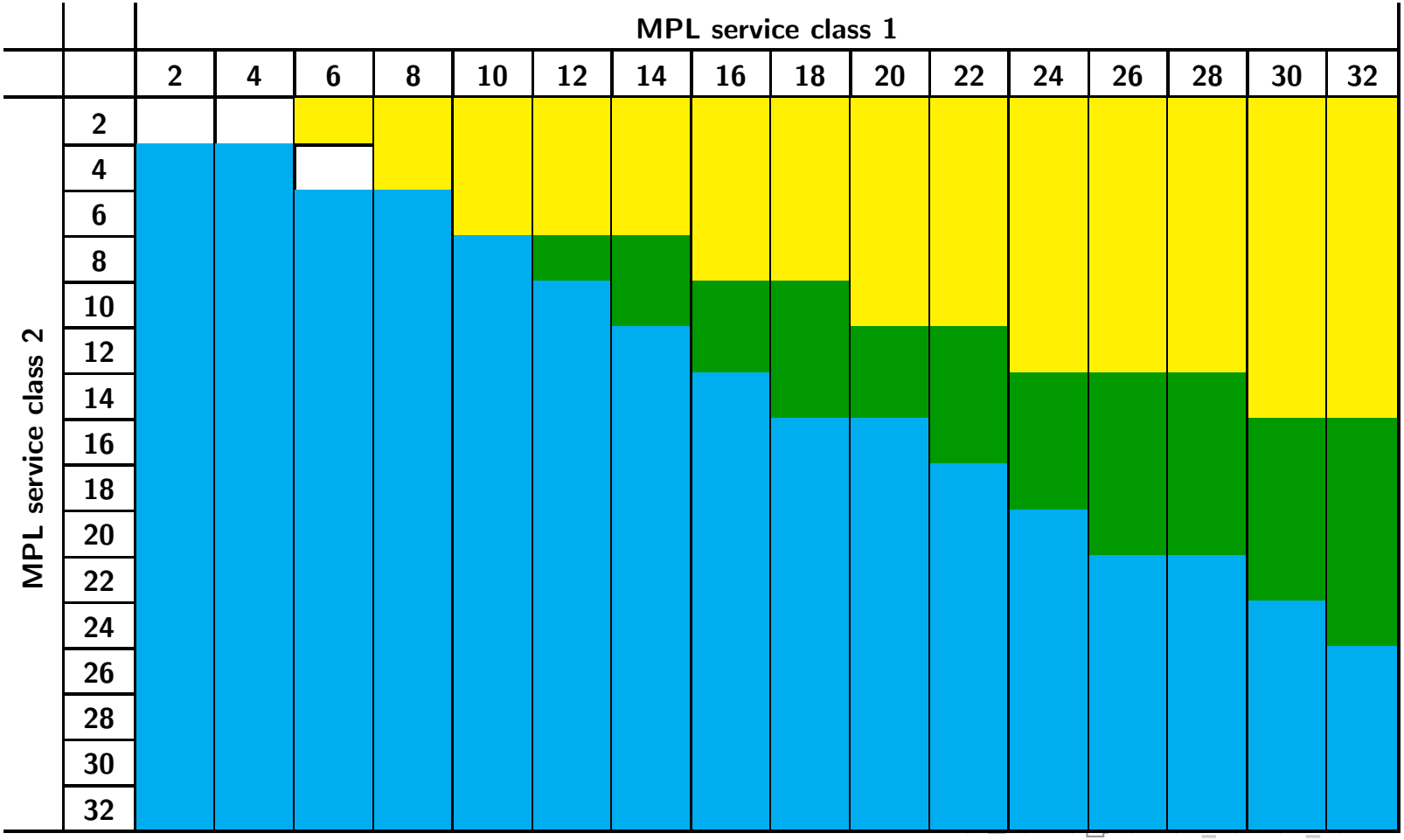
- Database engine models a parallel, shared-nothing architecture; four nodes with eight disks
- Data is partitioned across the disks
- More details on simulated engine in the paper
- Multiple streams per service class, each stream sends queries one after the other with no wait time between two queries
- OLTP-style queries; a query accesses data on a single partition only

Experiment 1

	Service class 1	**Service class 2**
Average response time (sec)	0.25	1.0
Throughput (q/sec)	130	80
Dominant objective	throughput	throughput
Algorithm optimizes for	throughput	throughput

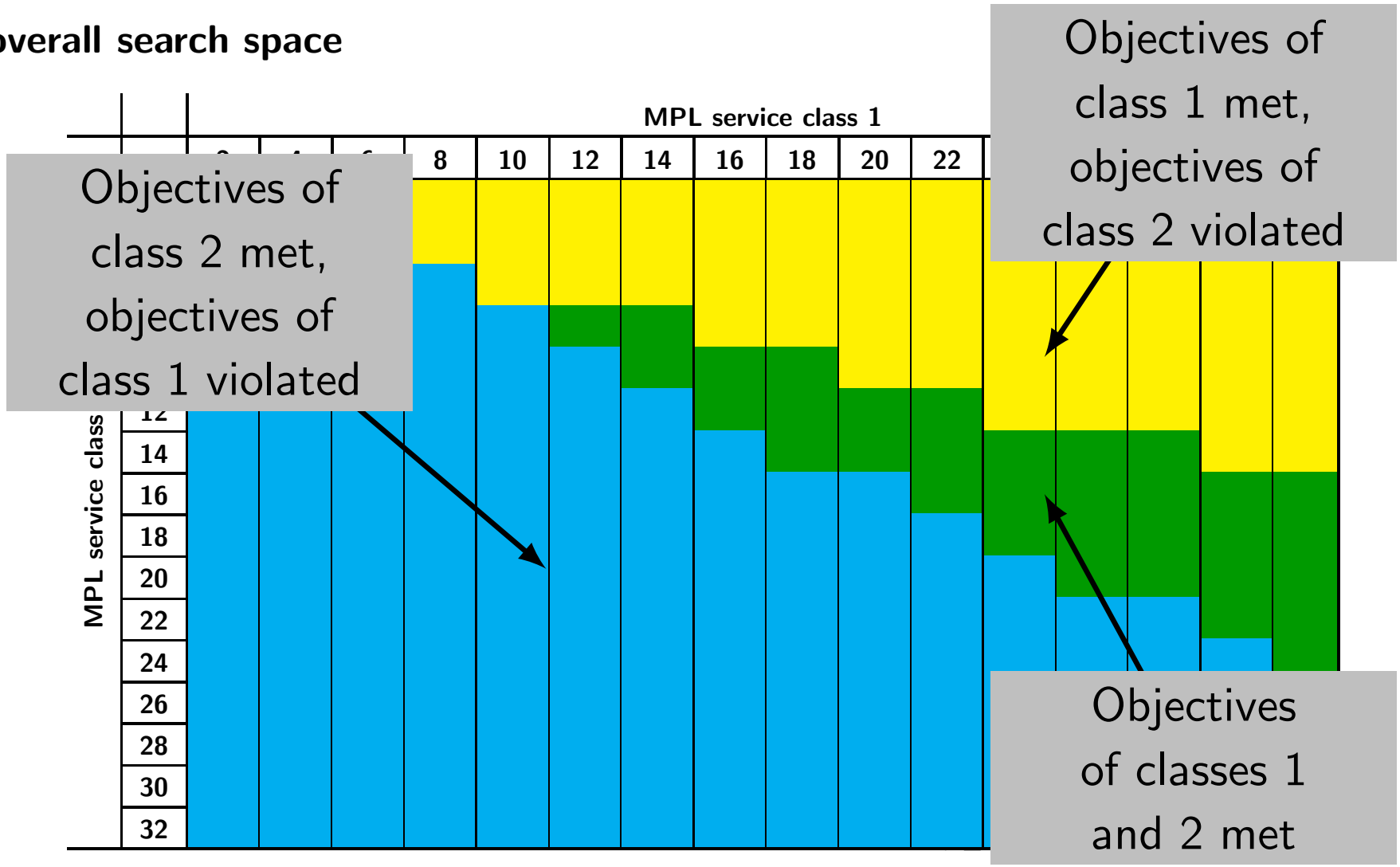
Results

overall search space



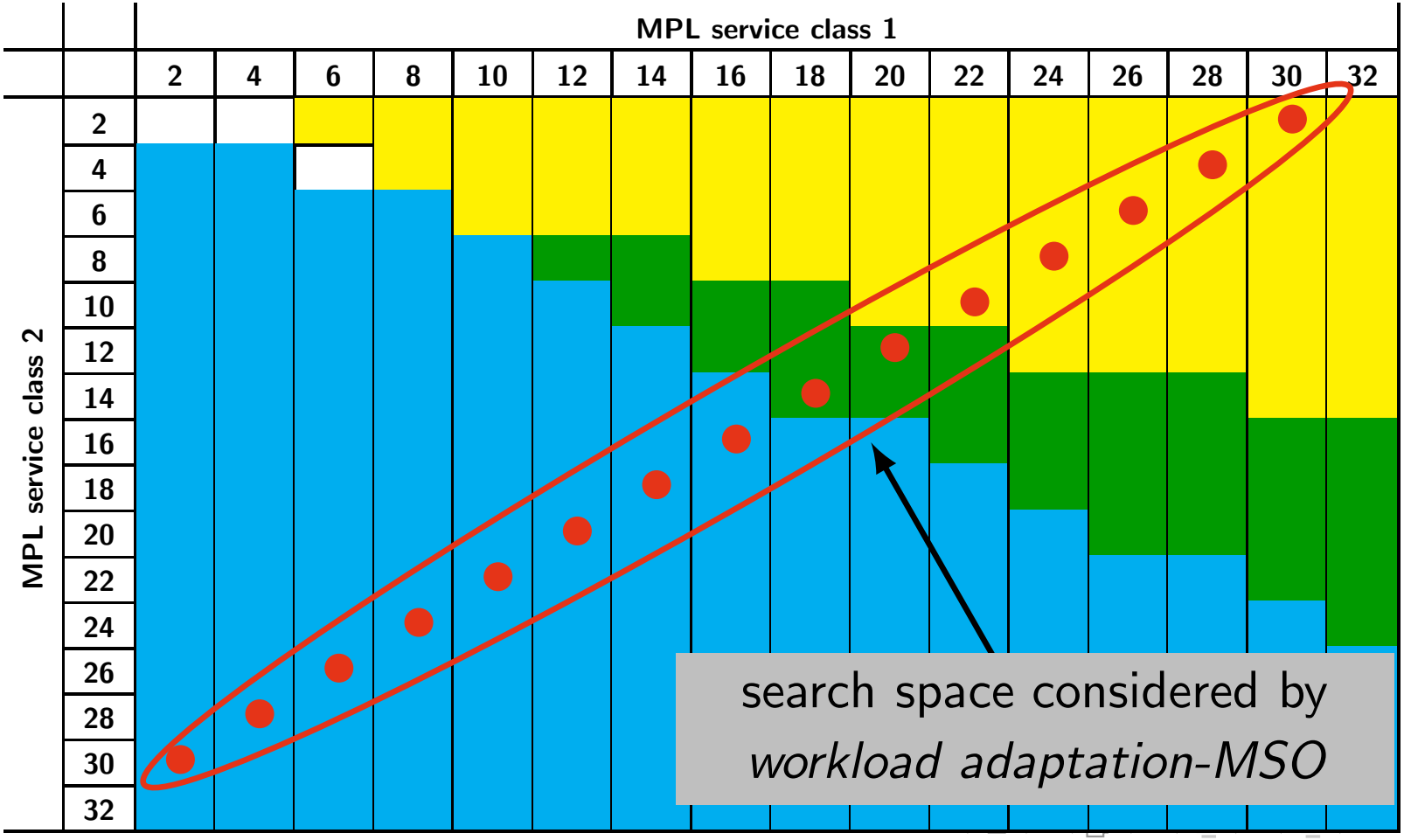
Results

overall search space



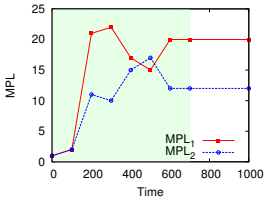
Results

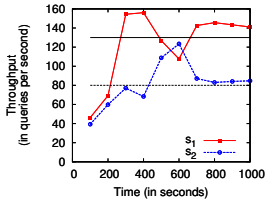
search space considered by workload adaptation-MSO

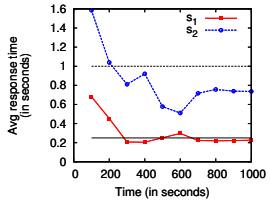


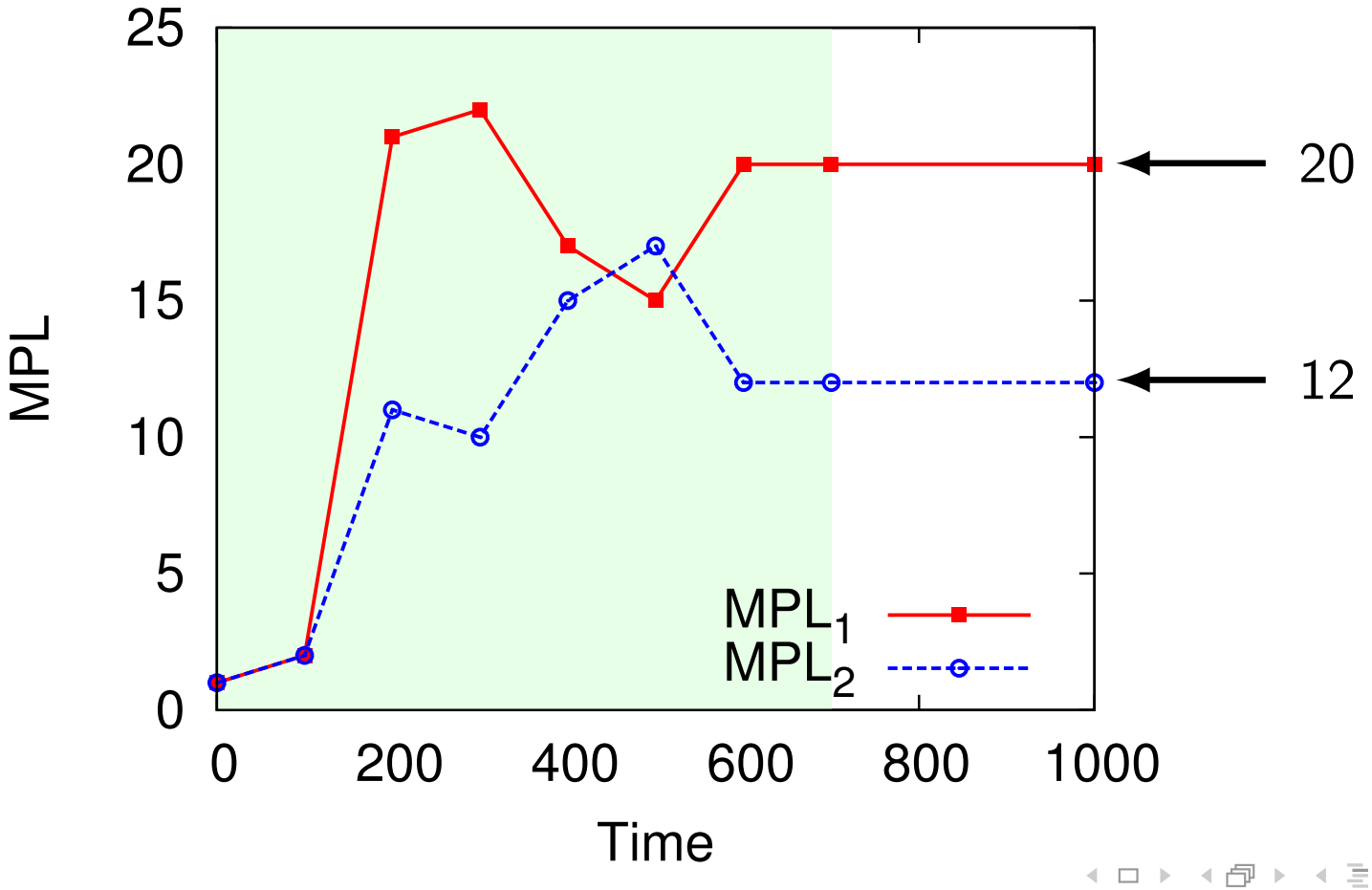
Results

workload adaptation-MSO



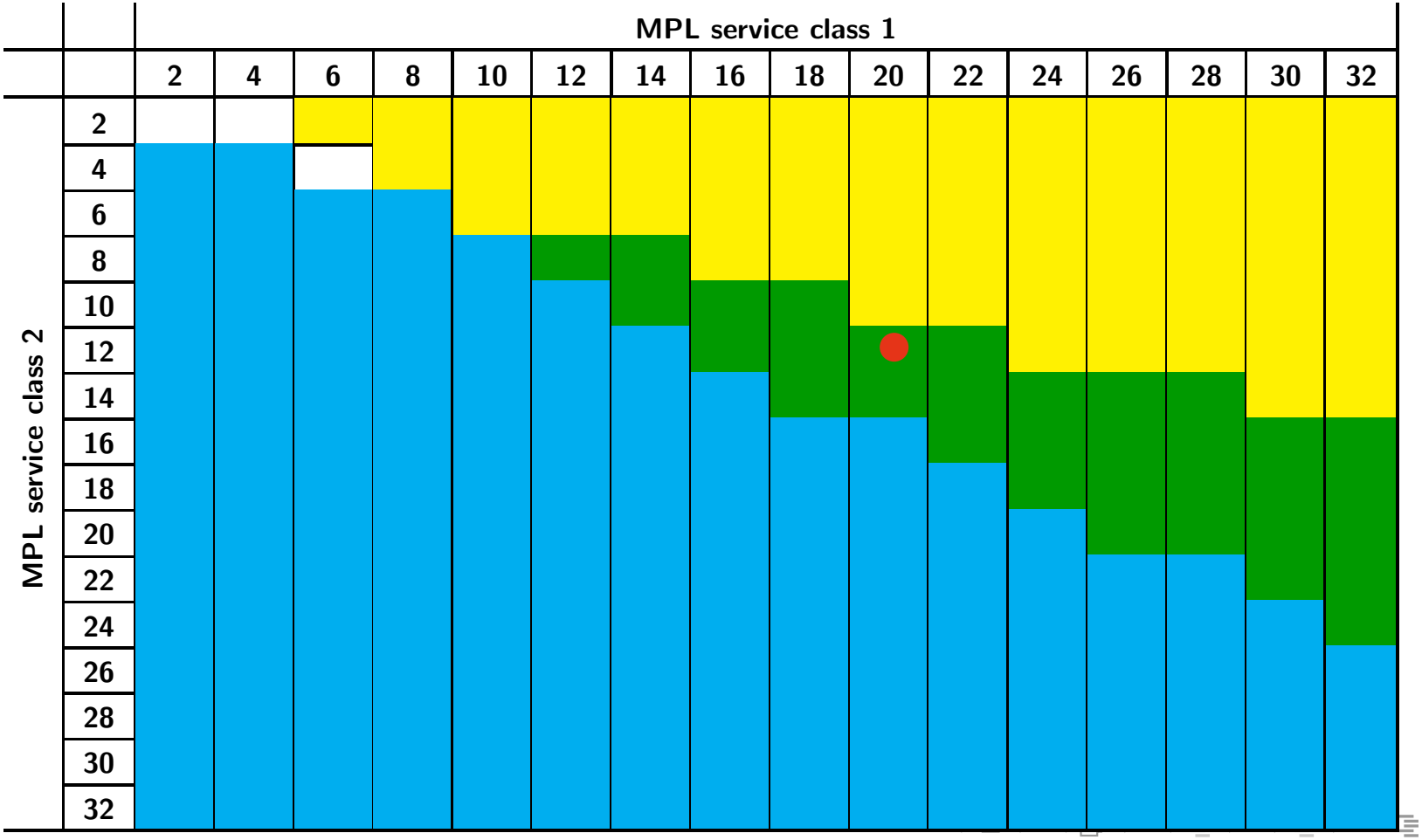






Results

workload adaptation-MSO

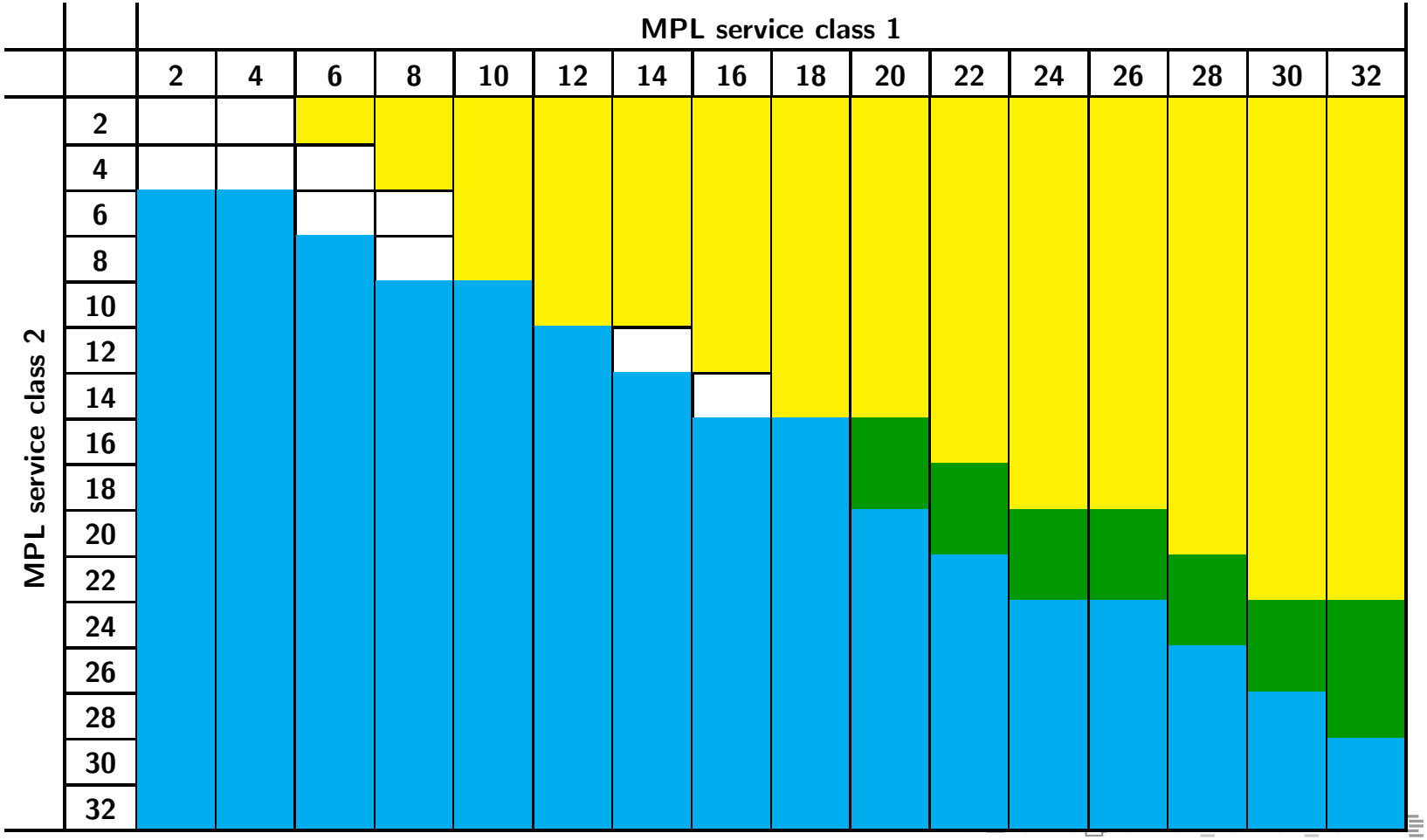


Experiment 2

	Service class 1	**Service class 2**
Average response time (sec)	0.25	0.6
Throughput (q/sec)	100	80
Dominant objective	average response time	throughput
Algorithm optimizes for	throughput	throughput

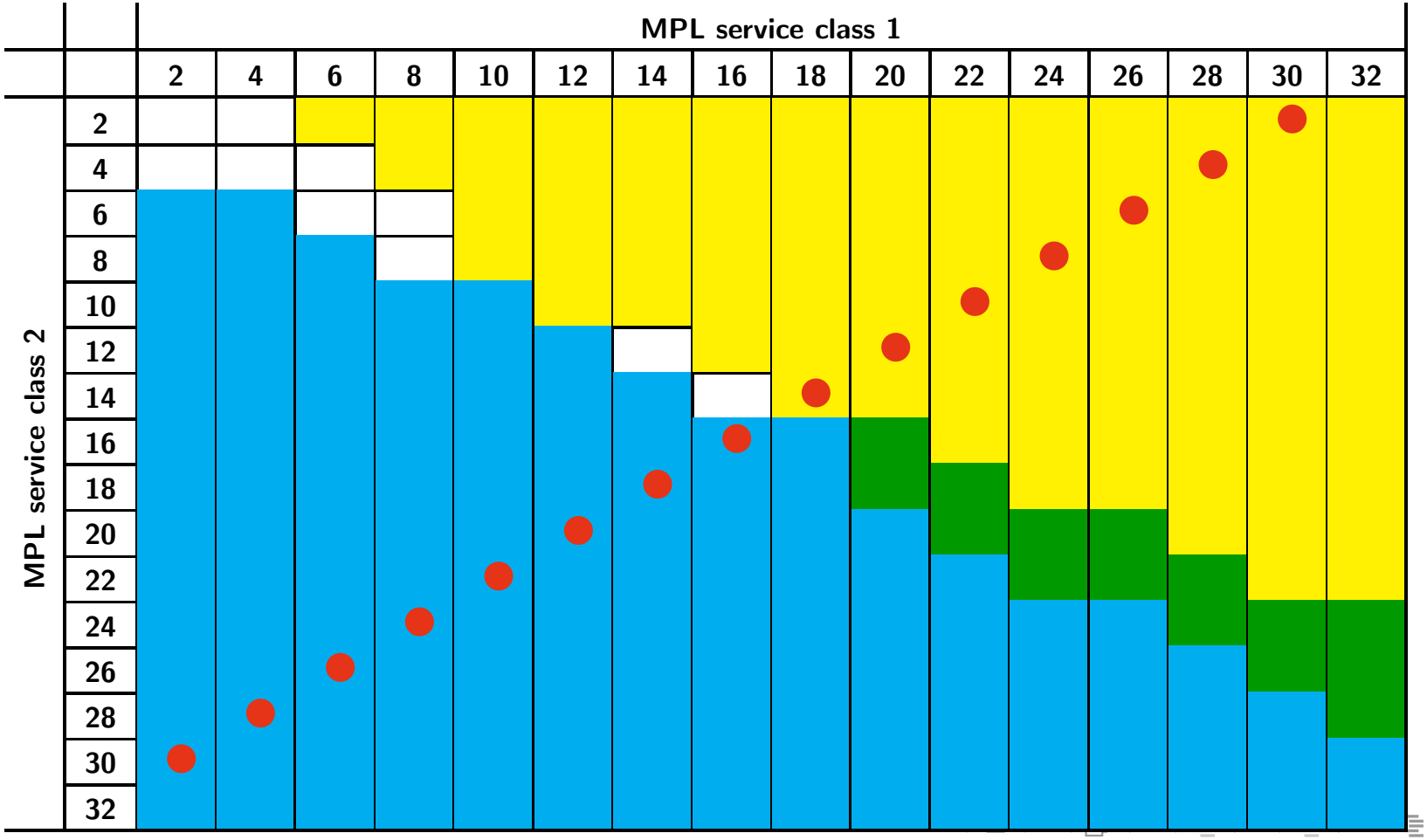
Results

naïve



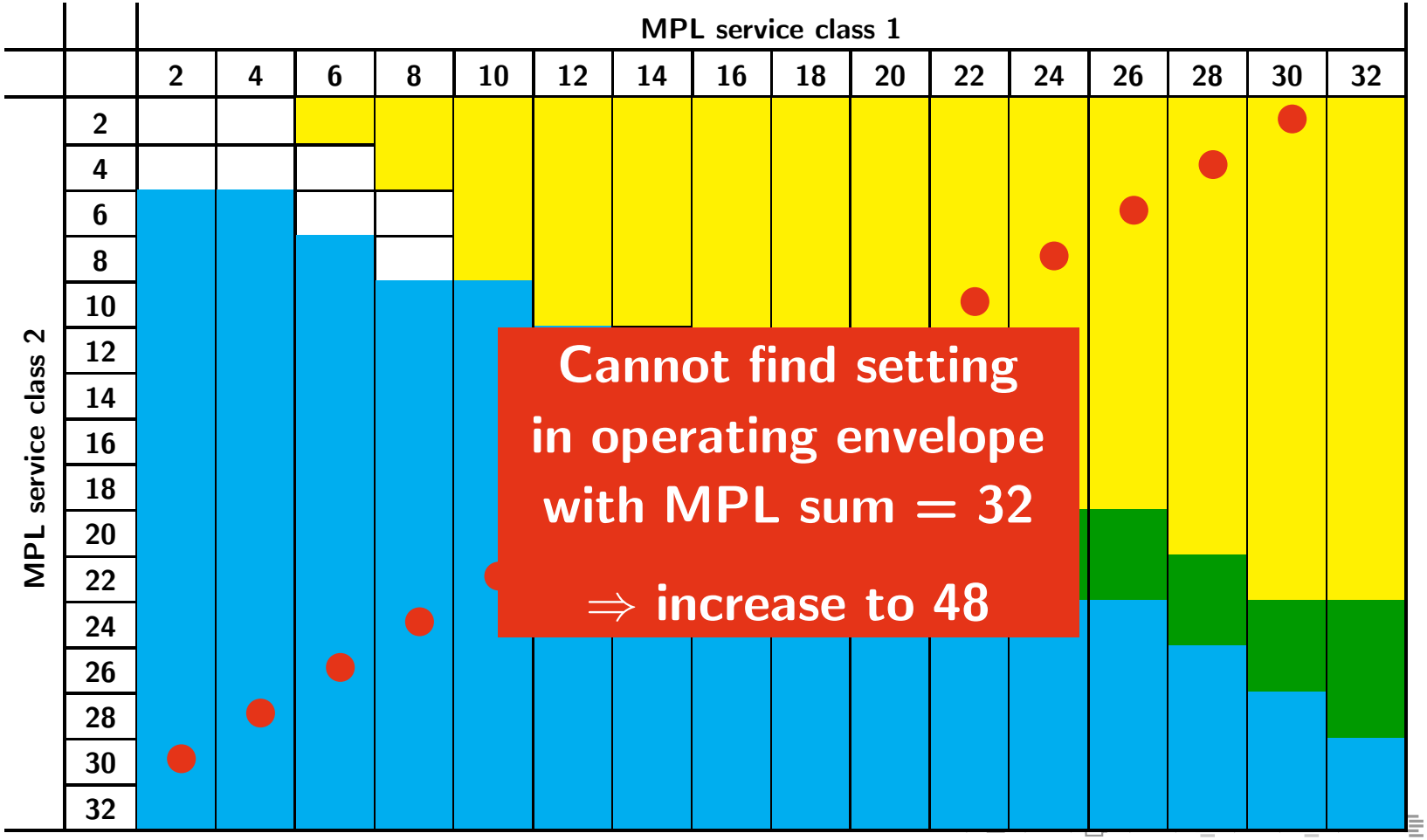
Results

search space considered by workload adaptation-MSO



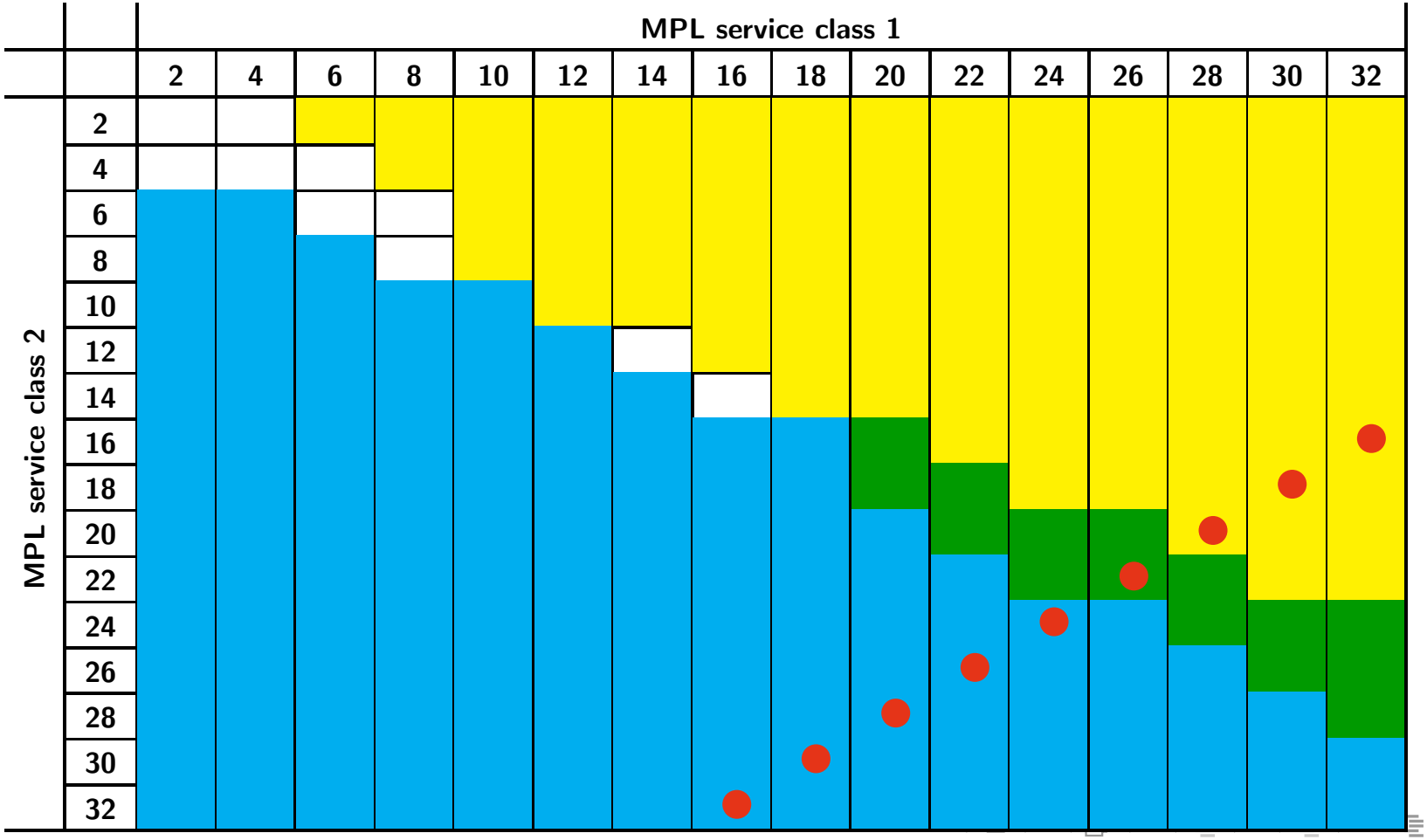
Results

search space considered by workload adaptation-MSO



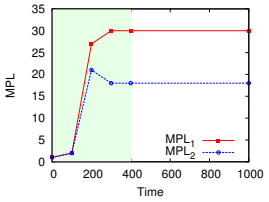
Results

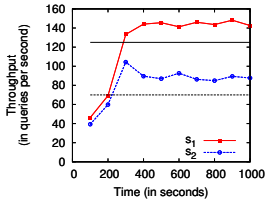
search space considered by workload adaptation-MSO

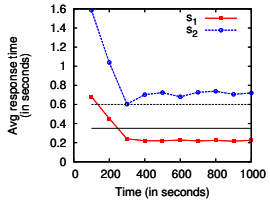


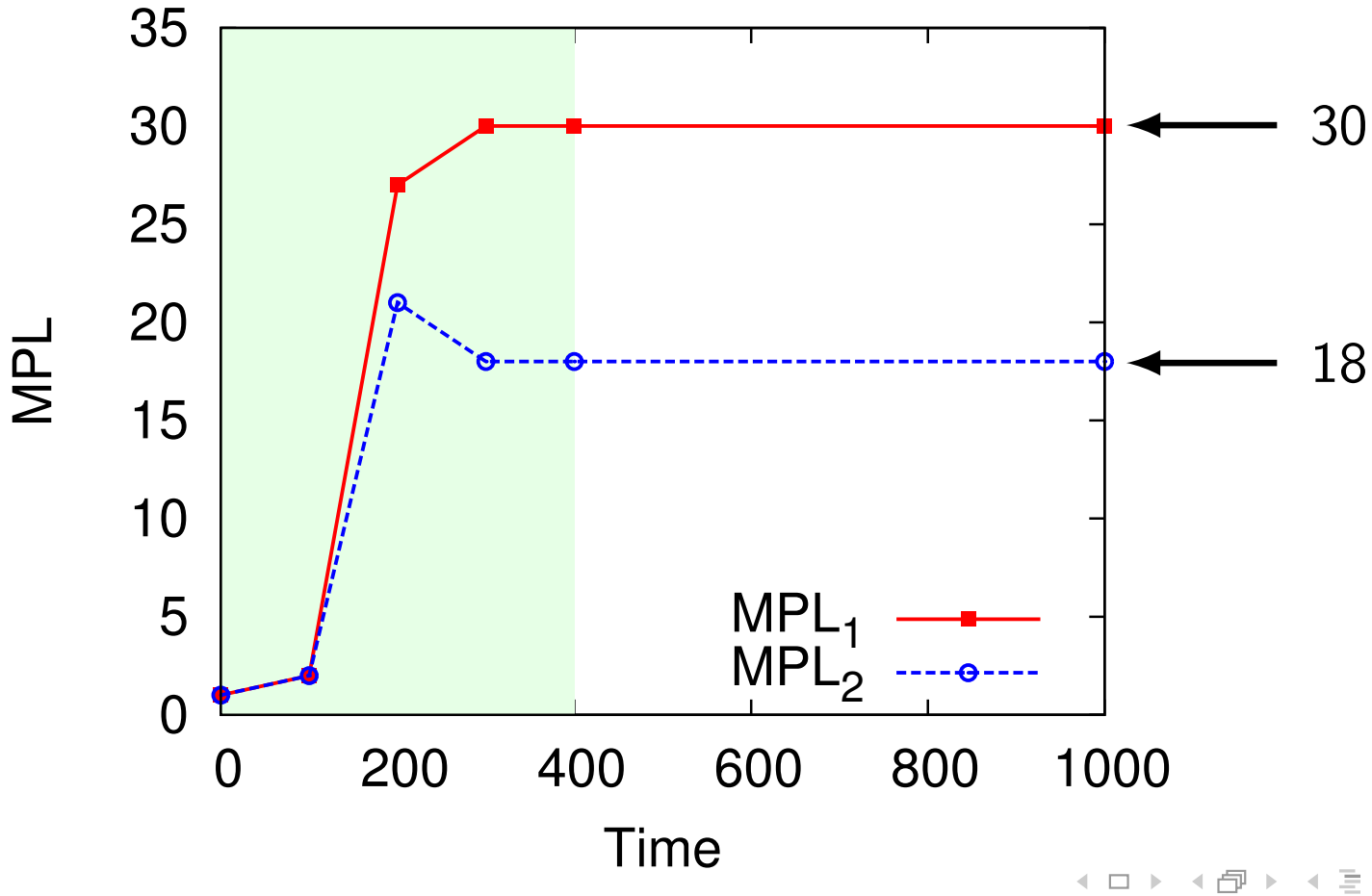
Results

workload adaptation-MSO



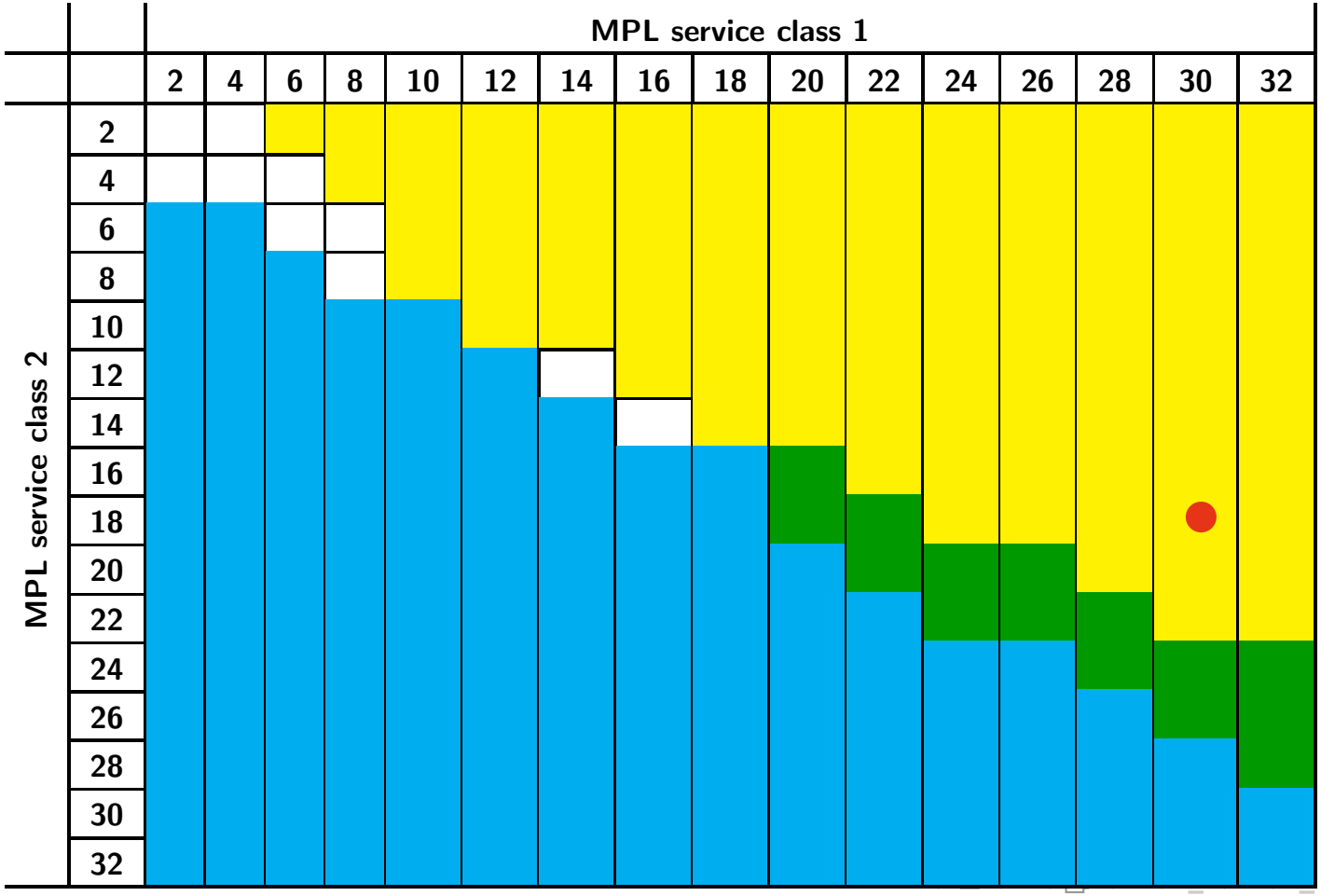






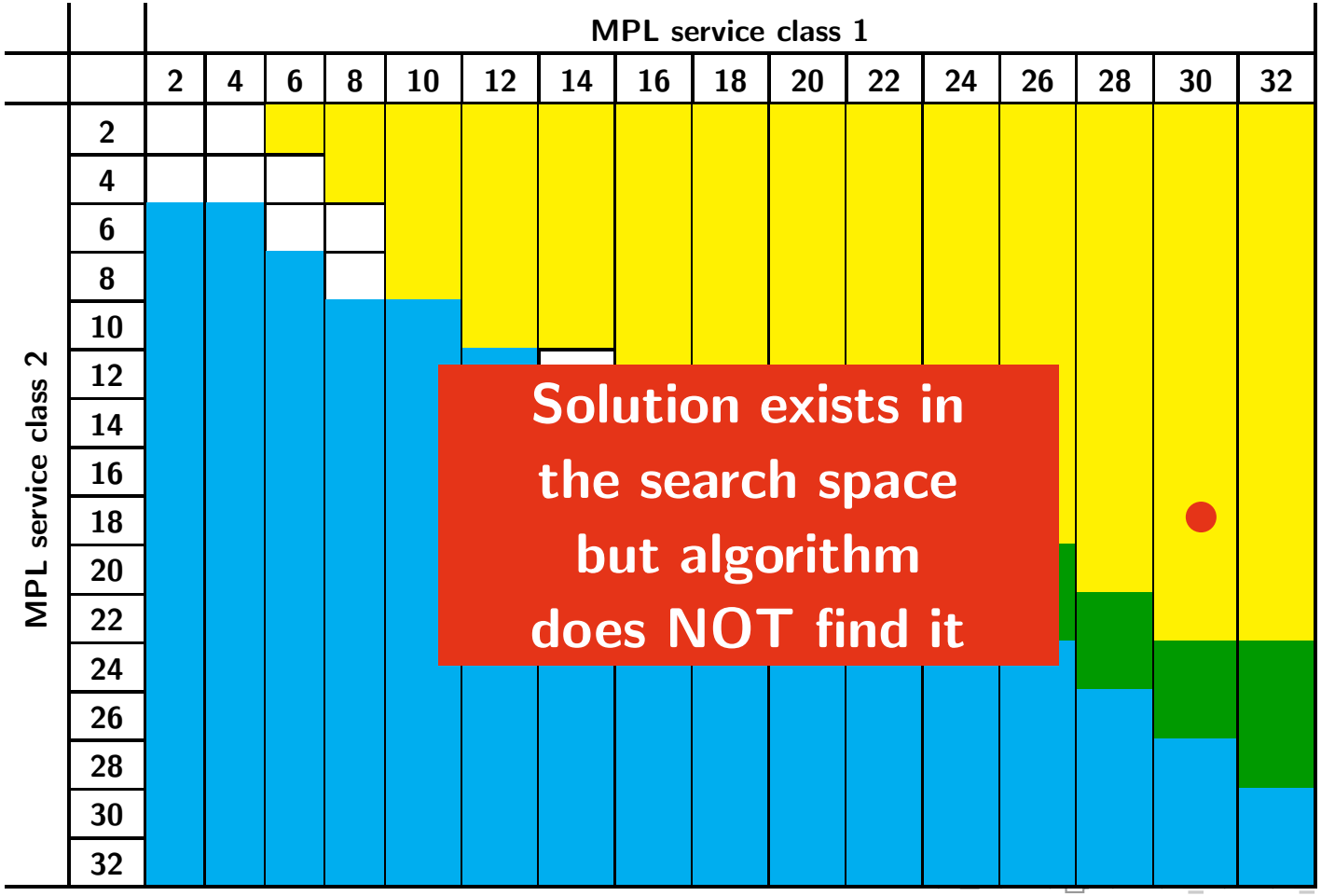
Results

workload adaptation-MSO



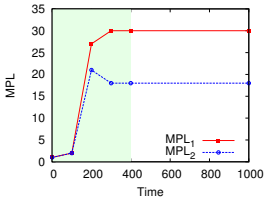
Results

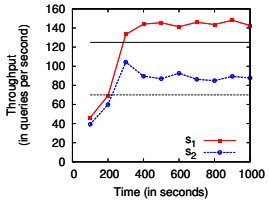
workload adaptation-MSO

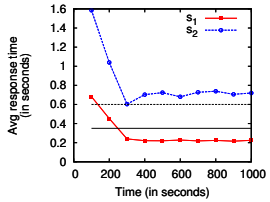


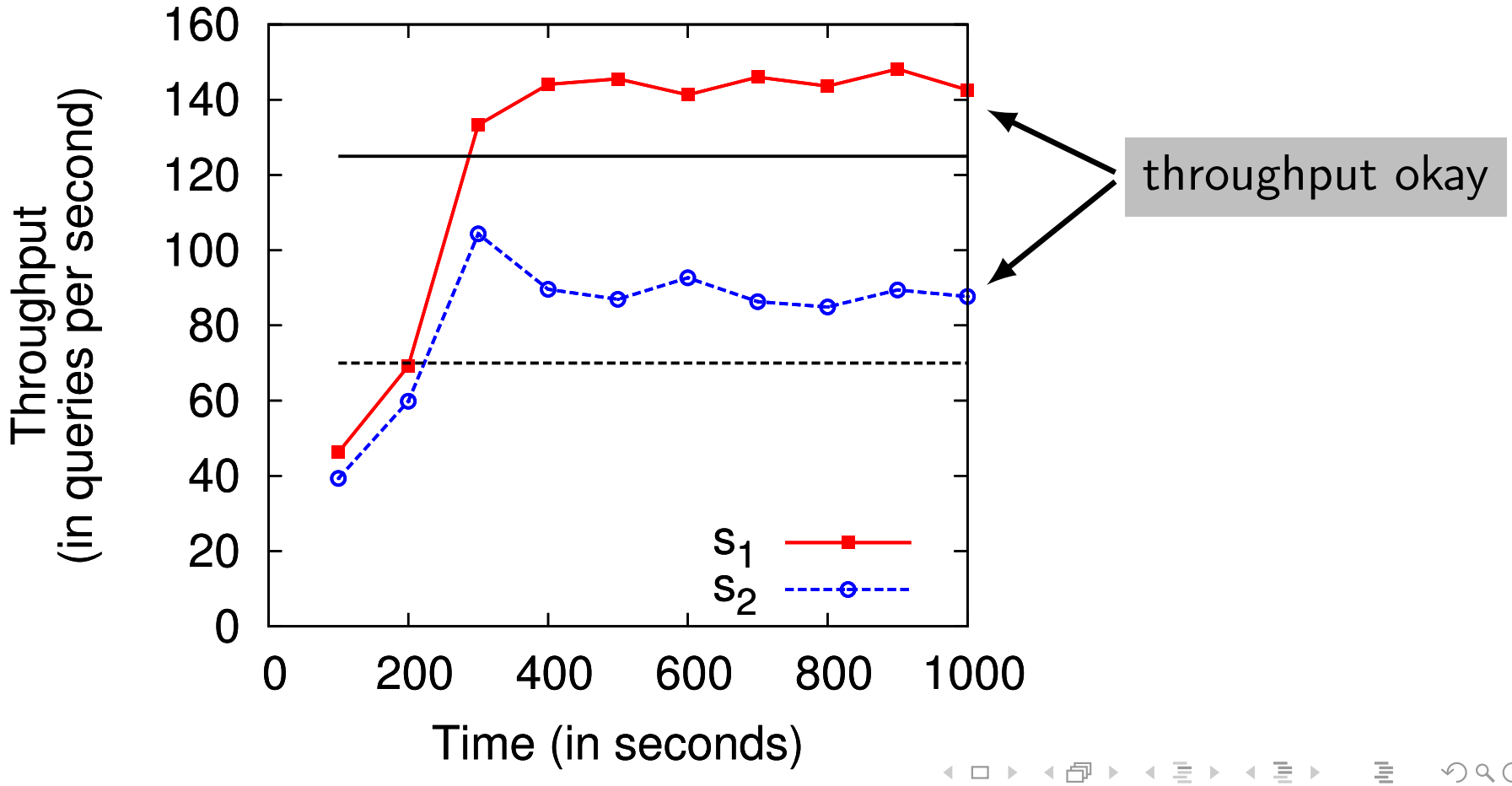
Results

workload adaptation-MSO



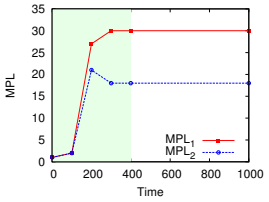


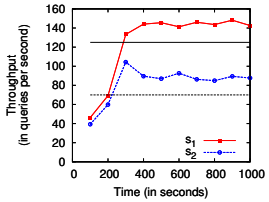


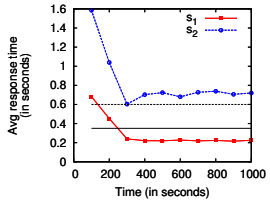


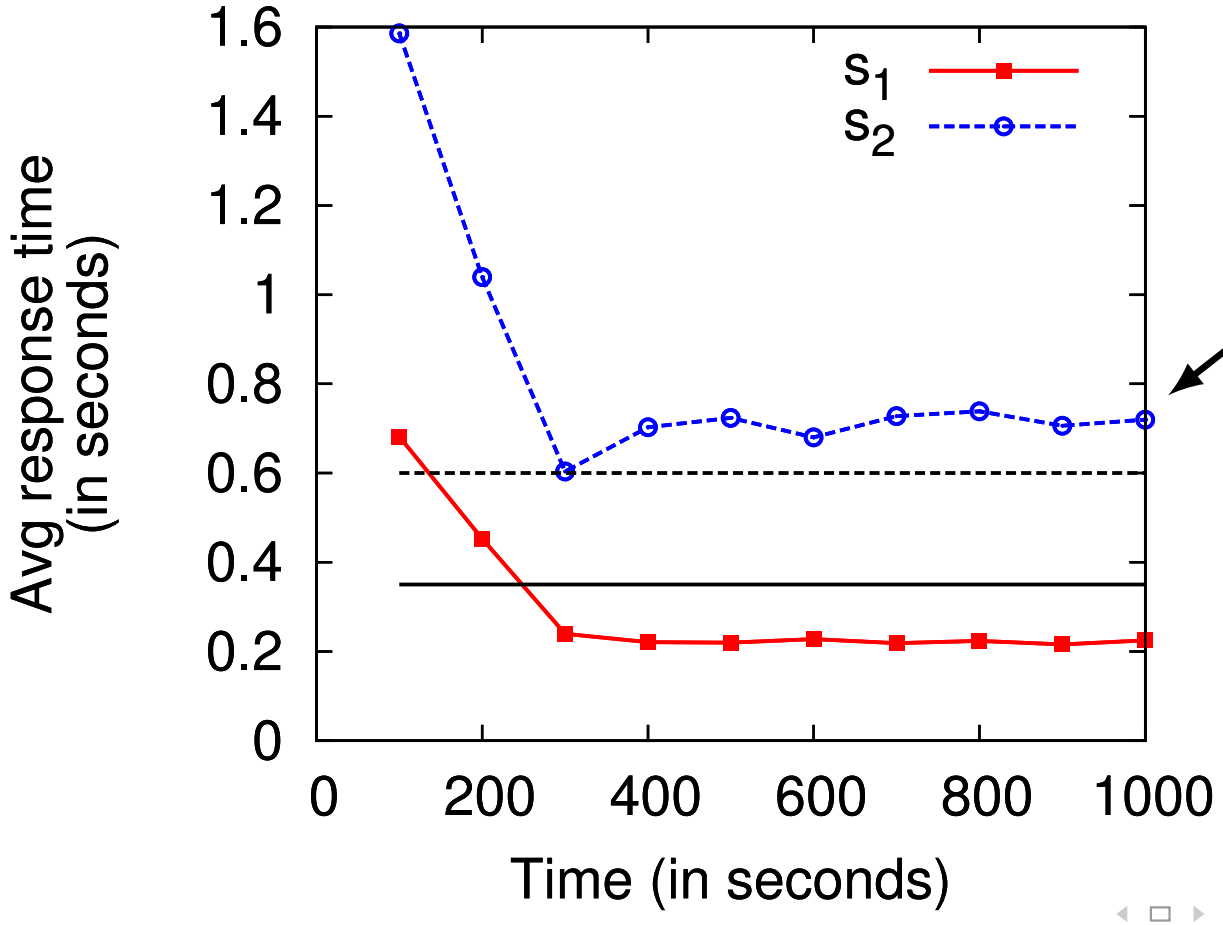
Results

workload adaptation-MSO









average response time of service class s_2 violated

Conclusion and ongoing work

- Presentation of test framework
- Comprehensive search solves the search problem, and gives additional information: Does a solution exist? How many settings satisfy the constraints? $\rightarrow$ prohibitively expensive
  $\Rightarrow$ Need heuristic approach
- Solutions found by *workload adaptation-MSO*are "fragile"
  $\Rightarrow$ Need different set of algorithms to solve the search problem