

# Constraint-Based Local Search for Break Scheduling in Call Centers

Nysret Musliu<sup>1</sup>, Werner Schafhauser<sup>1</sup>

Joint work with Andreas Beer<sup>2</sup>, Johannes Gärtner<sup>3</sup>, and  
Wolfgang Slany<sup>2</sup>

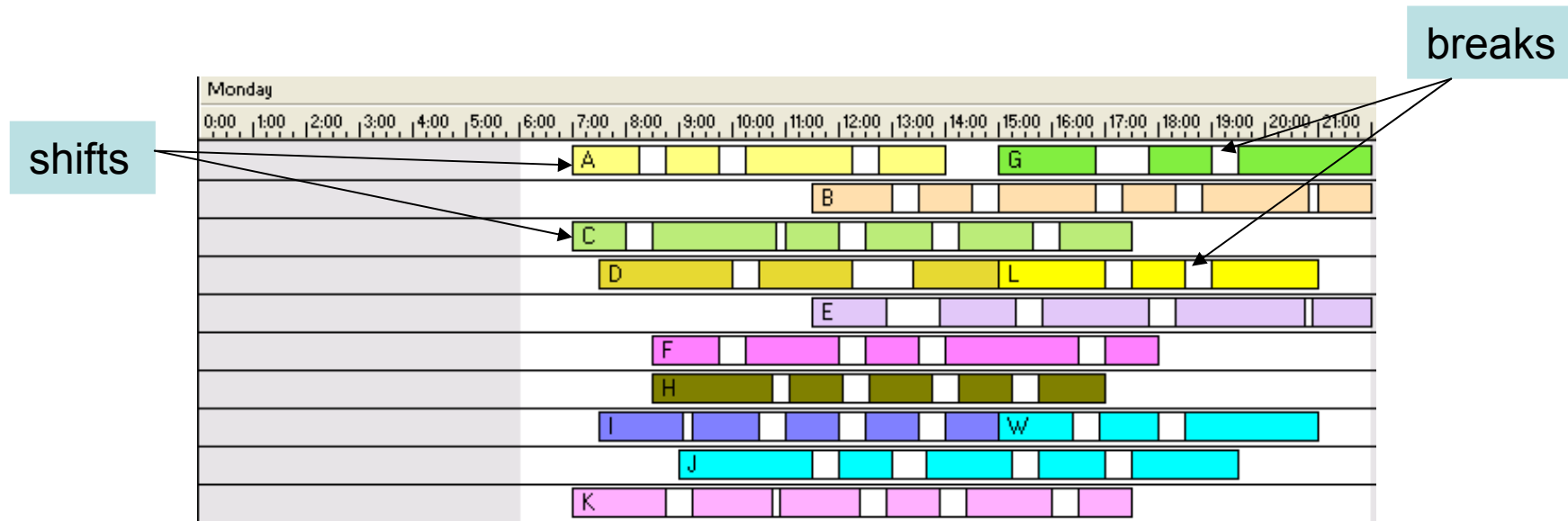
<sup>1</sup>Vienna University of Technology

<sup>2</sup>Graz University of Technology

<sup>3</sup>Ximes GmbH

# Introduction

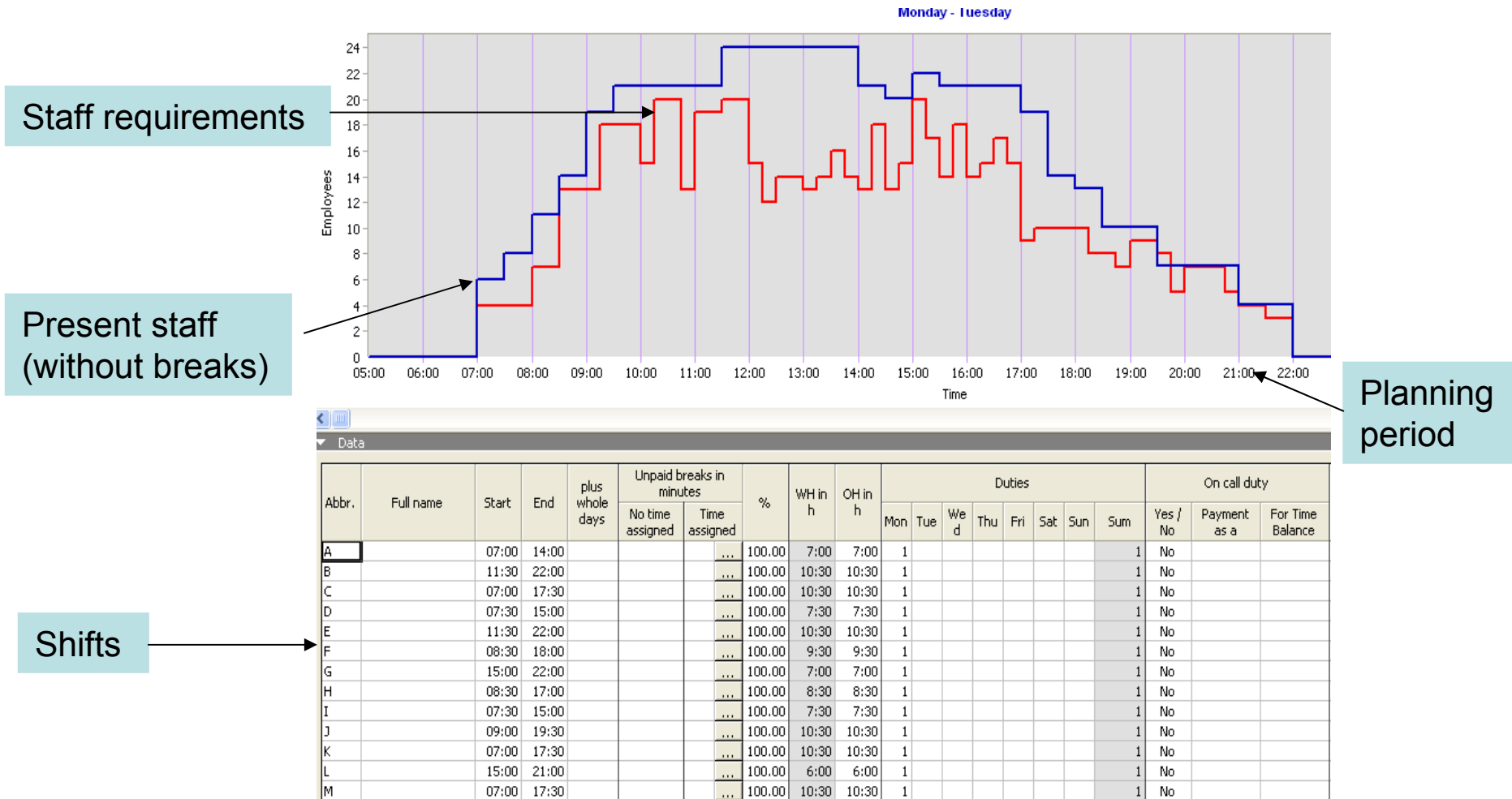
- Shift design and break scheduling are important stages in the general workforce-scheduling problem.



- High-quality solutions for break scheduling problems increase the satisfaction of employees and reduce the costs for companies.
- Break scheduling problems appear in many companies: call centers, airports, plants, ...

# Break Scheduling Problem

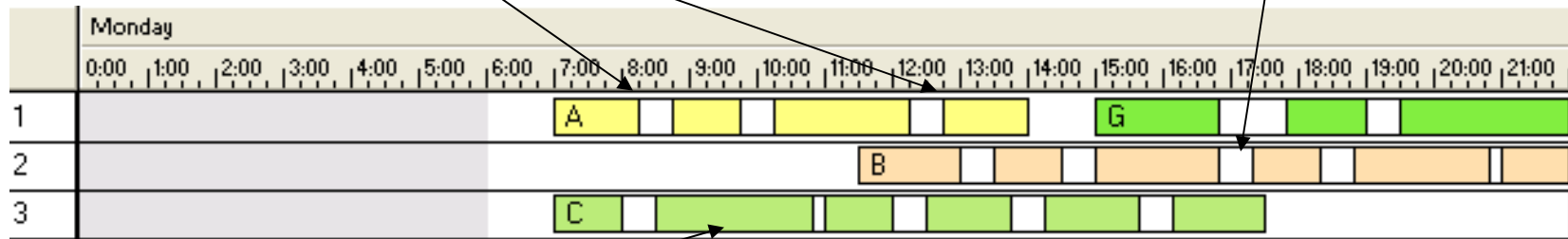
For the break scheduling problem we are given:



# Constraints on Breaks

Distance of break from shift begin and shift end

Time windows of breaks (e.g. lunch break)

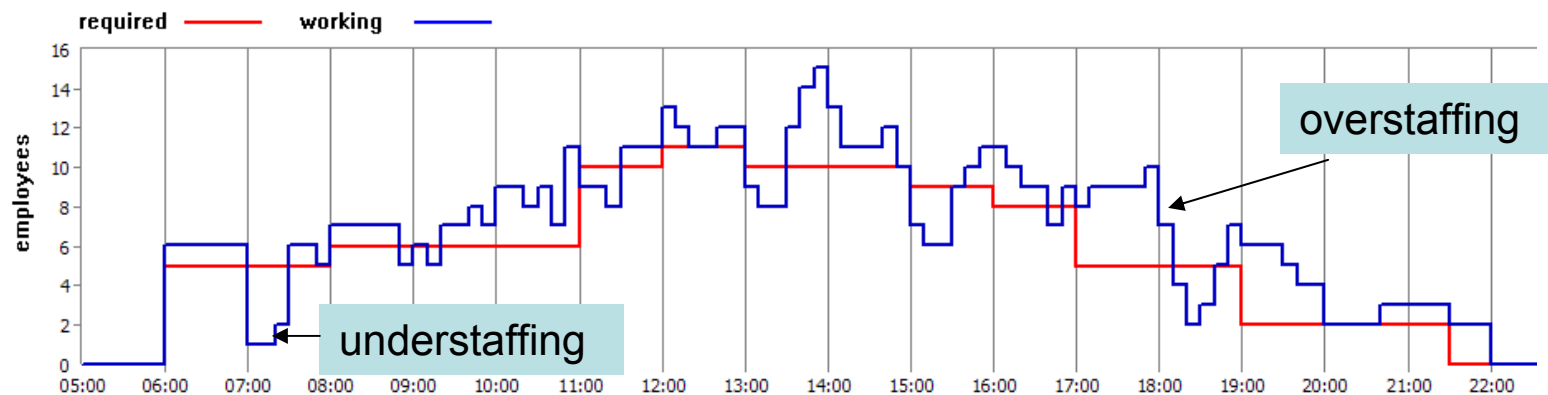


Minimum and maximum distance between breaks

Break quantity per shift

# Problem

- Given the staffing requirements, shifts and the constraints, schedule breaks for each shift such that:
  - overstaffing is minimized,
  - understaffing is minimized,
  - the violation of break constraints is minimized.
- These objectives are usually in conflict with each other.
- Each criteria can have different importance and the decision maker should define its preferences for each criteria.



# Objective Function

$$\text{Violations}(S) = \sum_{i=1}^{14} W_i \cdot C_i \rightarrow \min$$

- Weighted sum of violation degree of each constraint.
- Weight  $W_i$  specifies importance of constraint  $C_i$ .

# Solving Break Scheduling Problems

- The break scheduling problem may be regarded as a constraint satisfaction problem (CSP).
  - CSPs are known to be NP-hard in general.
- We proposed and implemented (meta)heuristic based techniques for solving this problem:
  - Min-Conflicts Heuristics
  - Tabu Search
- Improve a solution step by step by making small changes (=moves).

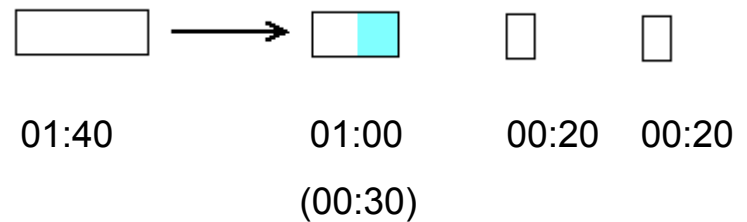
# Problem Representation

- Solution - set of breaks
- Break - variable start, constant length



Distribute the total break time among

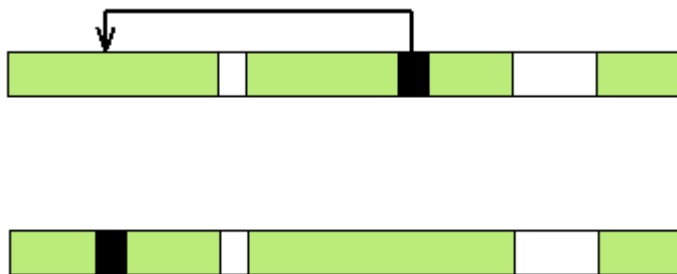
1. Fixed breaks
2. Lunch breaks
3. Monitor breaks  
of minimal or optimal length



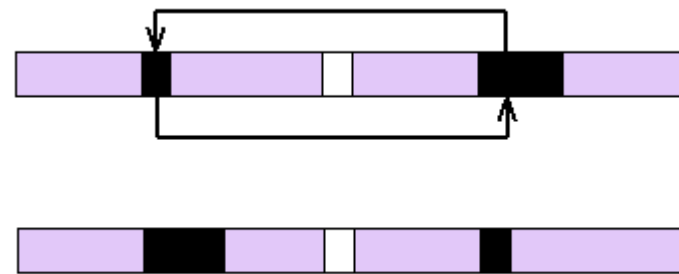
# Moves

- Assignment: assign to a break a new position in its shift.
- Swap: exchange two breaks in a shift.

Assignment

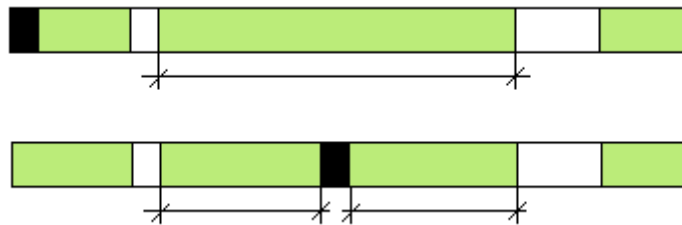


Swap



# Min-Conflicts Based Heuristic

- Improve the solution step by step.
- Select a break in violating a constraint.
- Apply a move minimizing overall constraint conflicts.



- Problem: Local optima!

# Random-Walk

- Goal: escape from local optima.
- Select a break violating a constraint.
- Apply an arbitrary move.



# Min-Conflicts-Random-Walk

While  $\text{Violations}(S) > 0$

**with probability  $1-p$       */\*Min-Conflicts Search\*/***

Select a break  $b$  causing a constraint violation

Apply the best move to  $b$  not increasing  $\text{Violations}(S)$

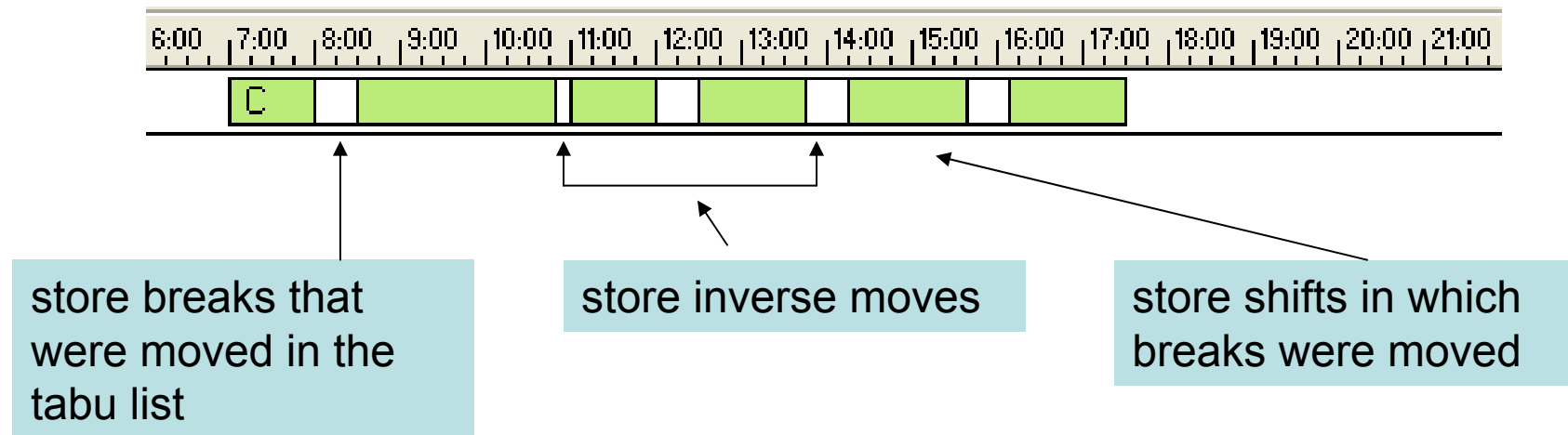
**with probability  $p$       */\*Random Walk\*/***

Select a break  $b$  causing a constraint violation

Apply an arbitrary move to  $b$

# Tabu Search

- The main idea behind tabu search is to store the information about search history in a "memory" denoted a *tabu list*.



Moves, inverse moves and shifts are kept in the tabu list for a certain number of iterations.

Solutions obtained with these moves are considered tabu -> eliminate cycles during the search.

# COMET – SW Tool for Local Search

- Object Oriented Programming language .
- Constraint-Based Local Search.
- Specify local search algorithms as two components.

1. Problem model

Problem Model

***Break Scheduling Problem***

2. Search method

Search Method

***Tabu Search***

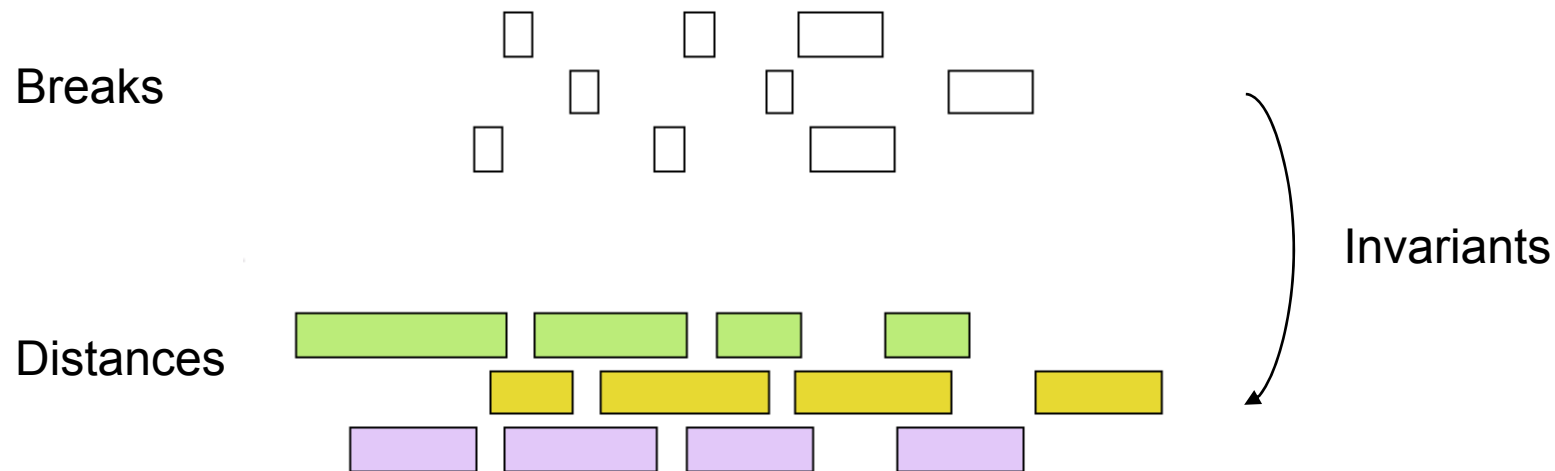
***Min-Conflicts-Random-Walk***

# Problem Model - Concepts

**Incremental variables** are used for representing problem and its properties.

## Invariants

- are expressed in terms of incremental variables.
- specify a relation that is maintained automatically.



# Problem Model - Constraints

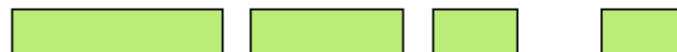
## Constraint (abstract class)

- violations                  violation degree of a constraint
- $\Delta(\text{move})$                   change in violation degree
- violations (var)                  violation degree of variable

Breaks



Distances



MaximumDistance

-Violations

- $\Delta(\text{move})$

-violations(var)



# Advantages of COMET

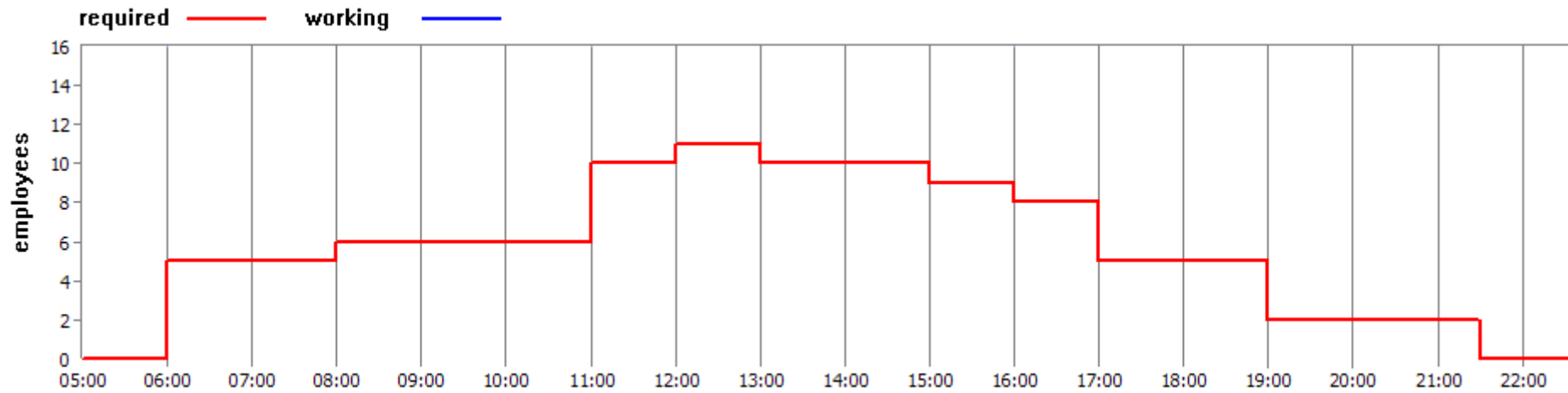
- + Compositional LS algorithms.
- + Concepts and Classes/Interfaces for LS techniques.
- + Problem modeling is separated from search.
- + Reusability of search methods.
- + Reduces effort for developing LS algorithms.

# Application in Call Centers

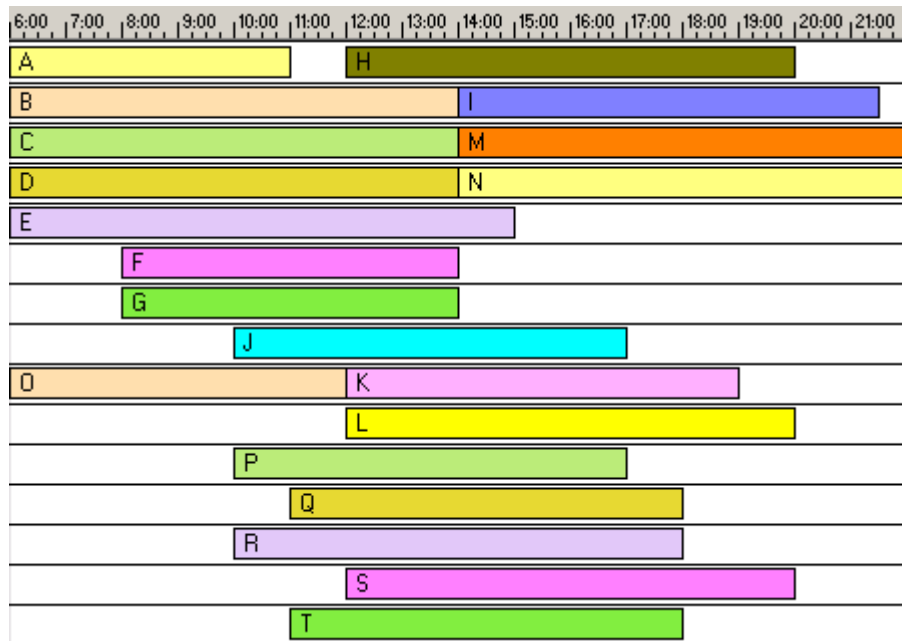
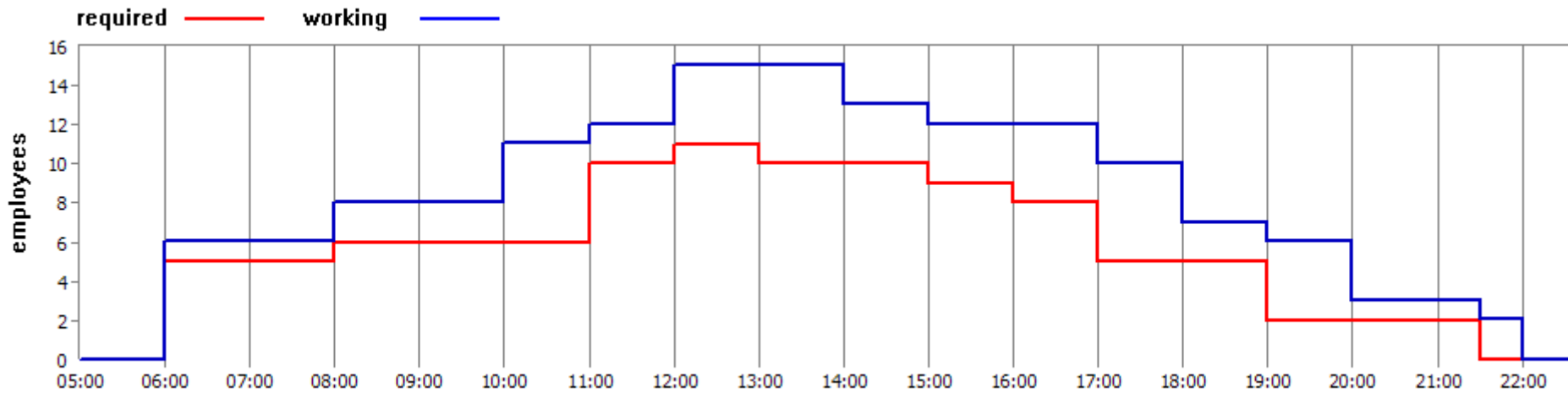
- 1 day planning period
- 15-35 shifts
- 40-120 breaks



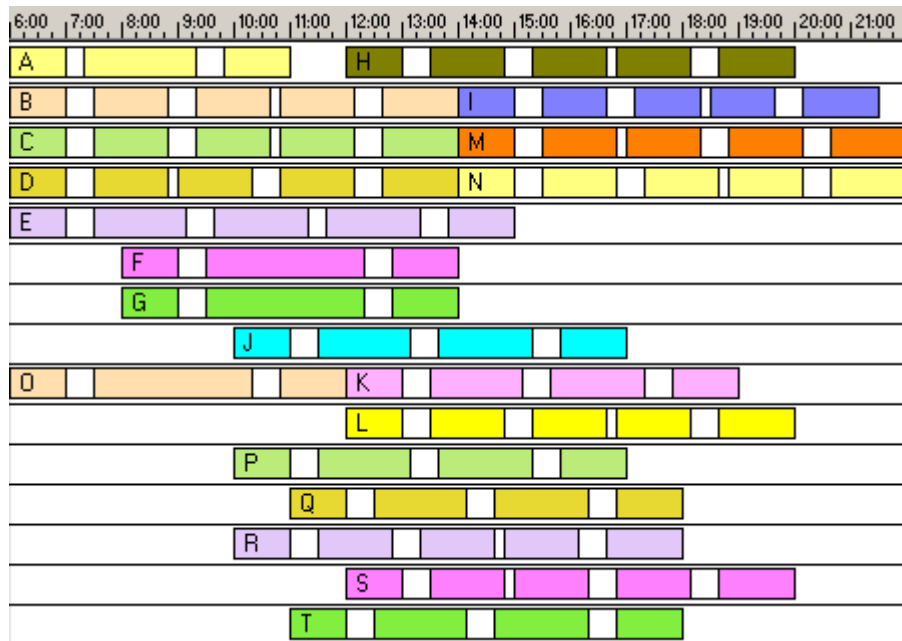
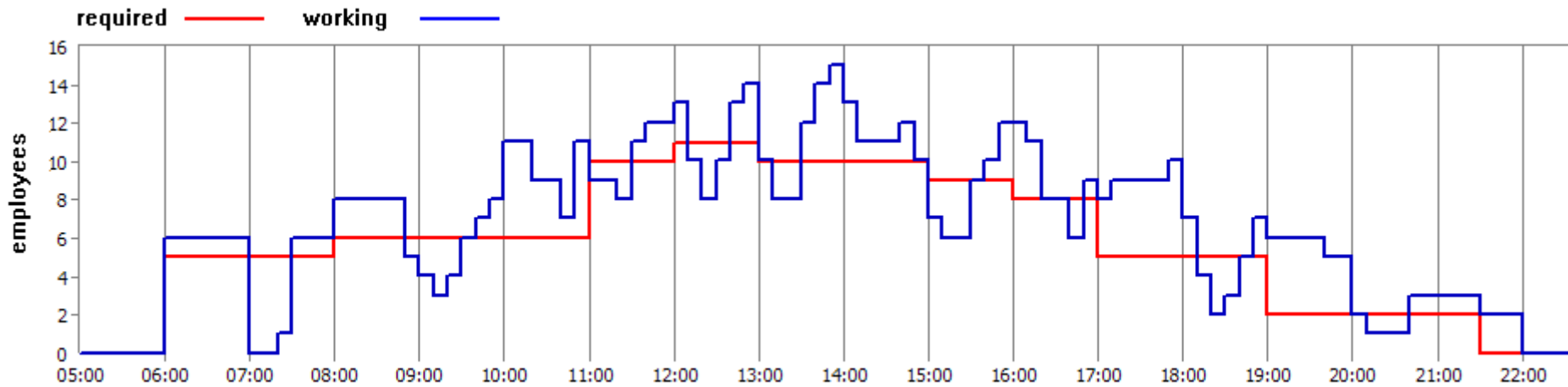
# Staffing Requirements



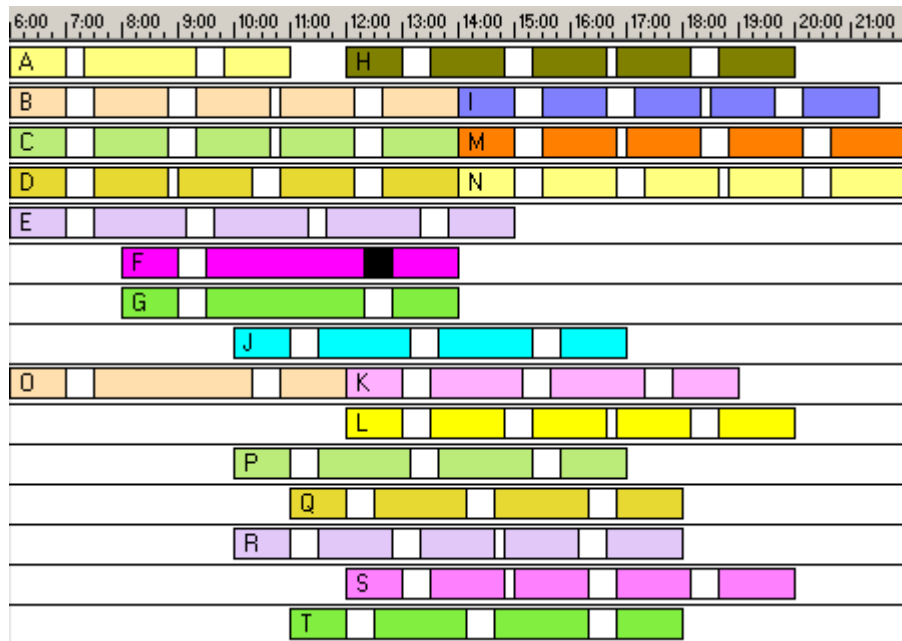
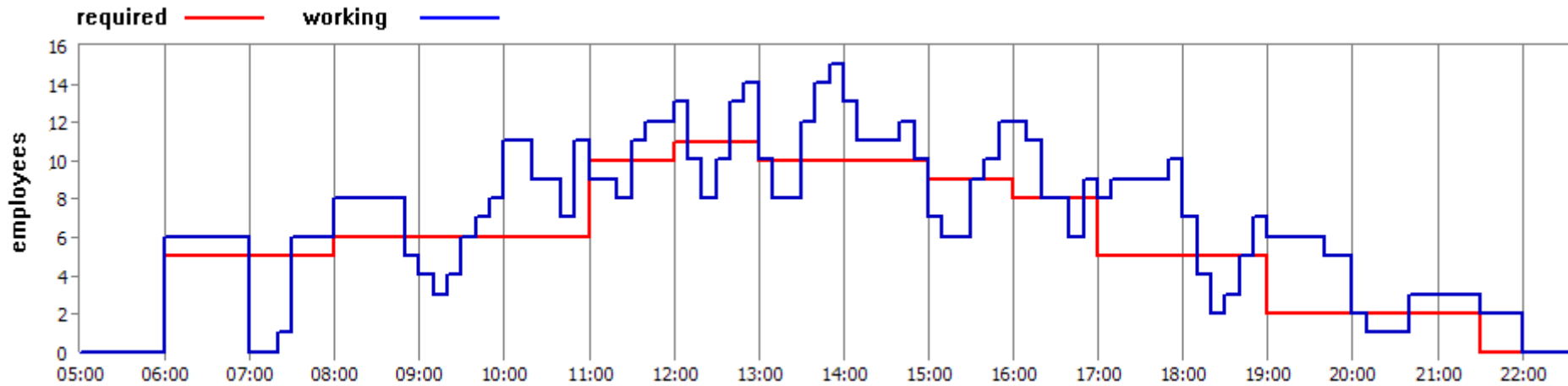
# Shift Plan



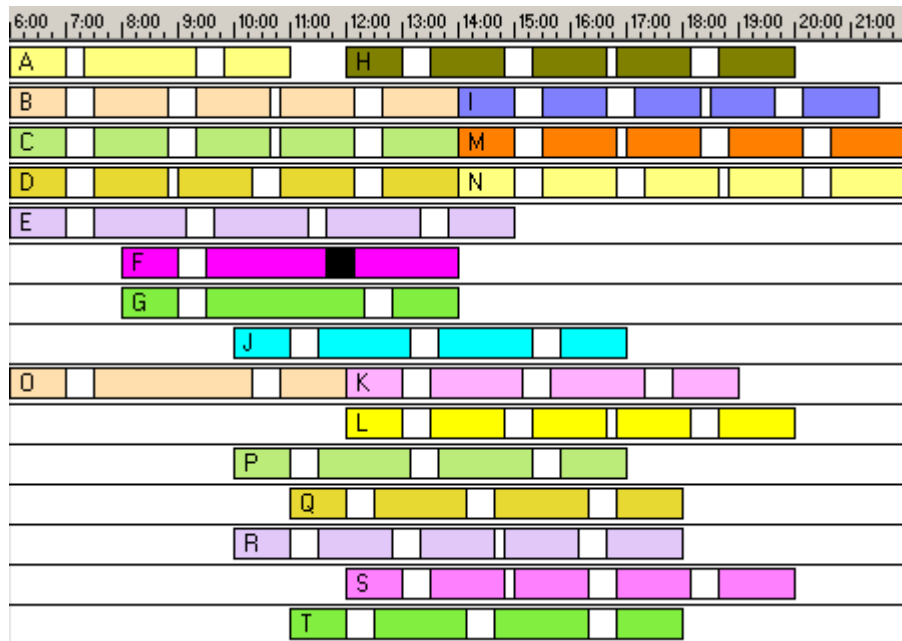
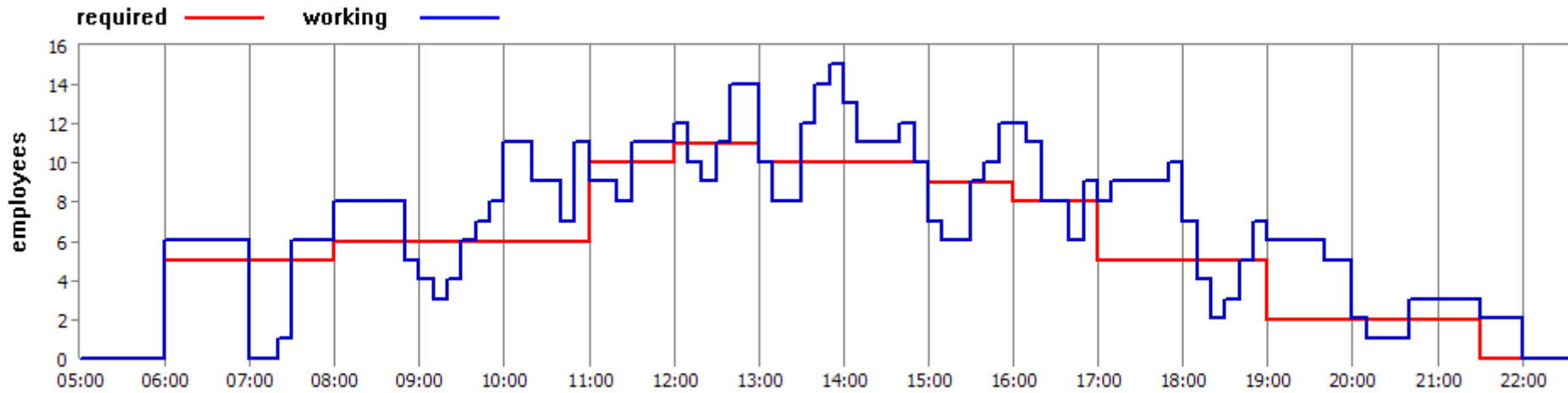
# Shift Plan with Breaks



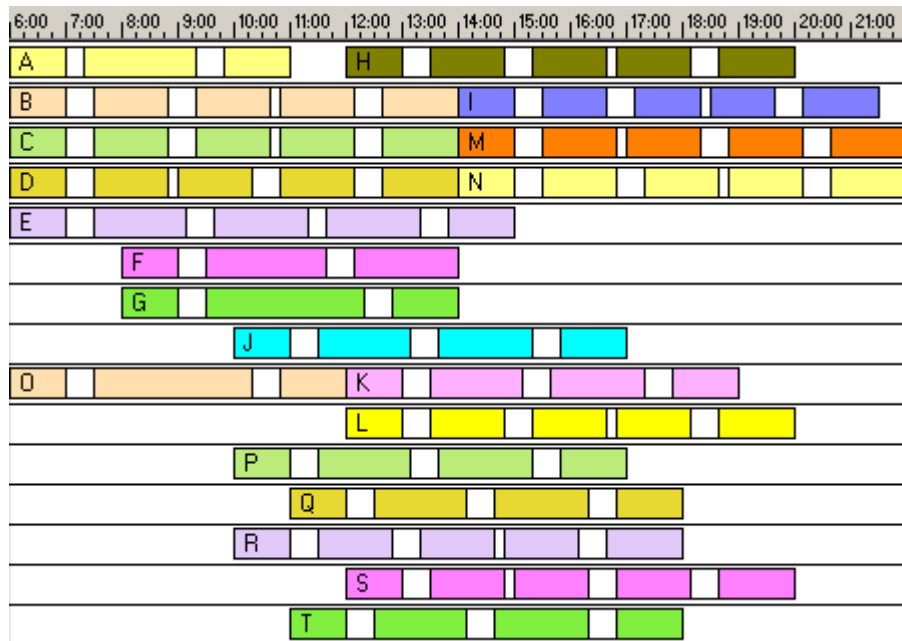
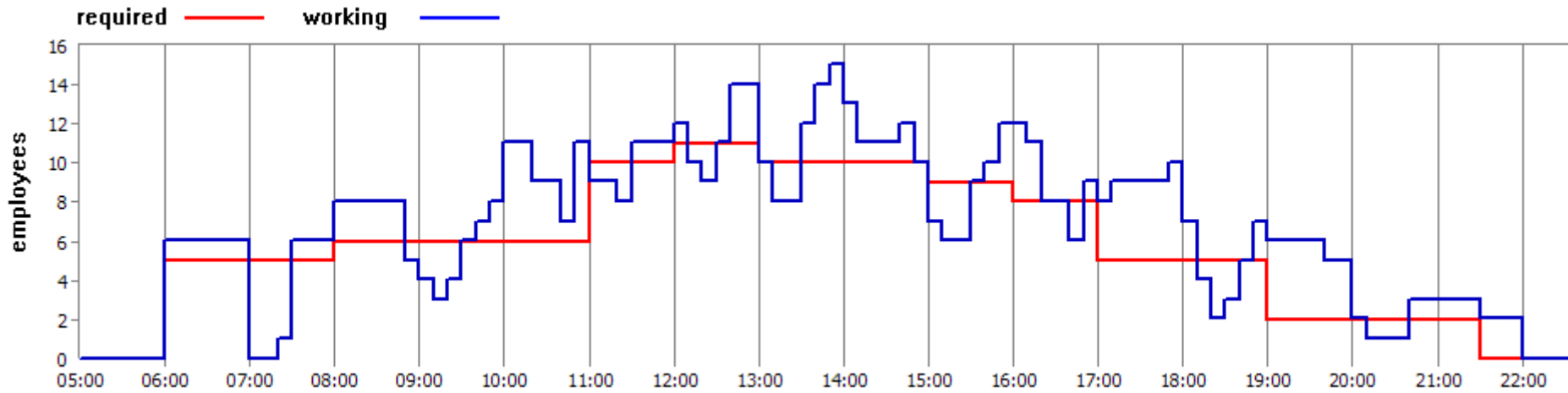
# Iteration 1



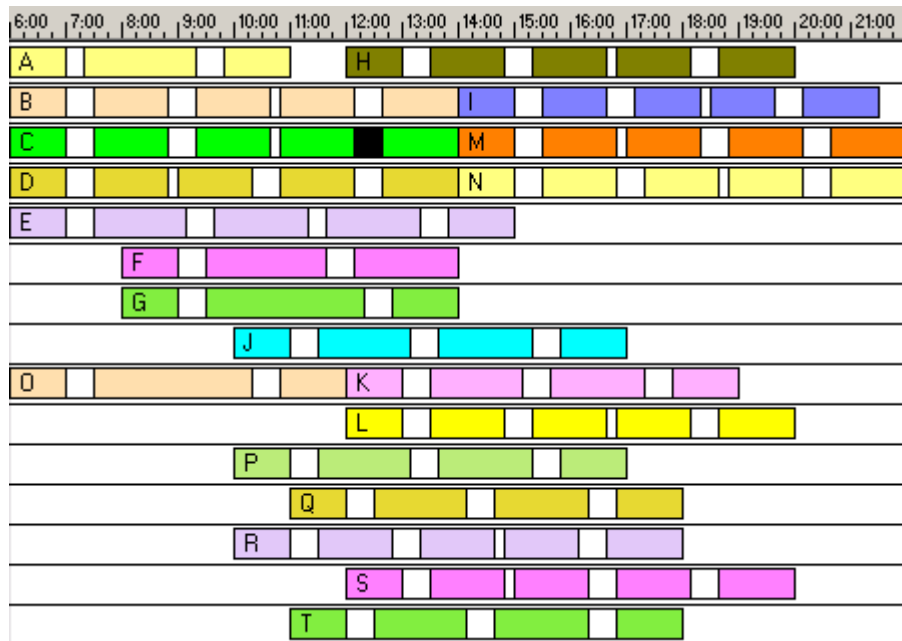
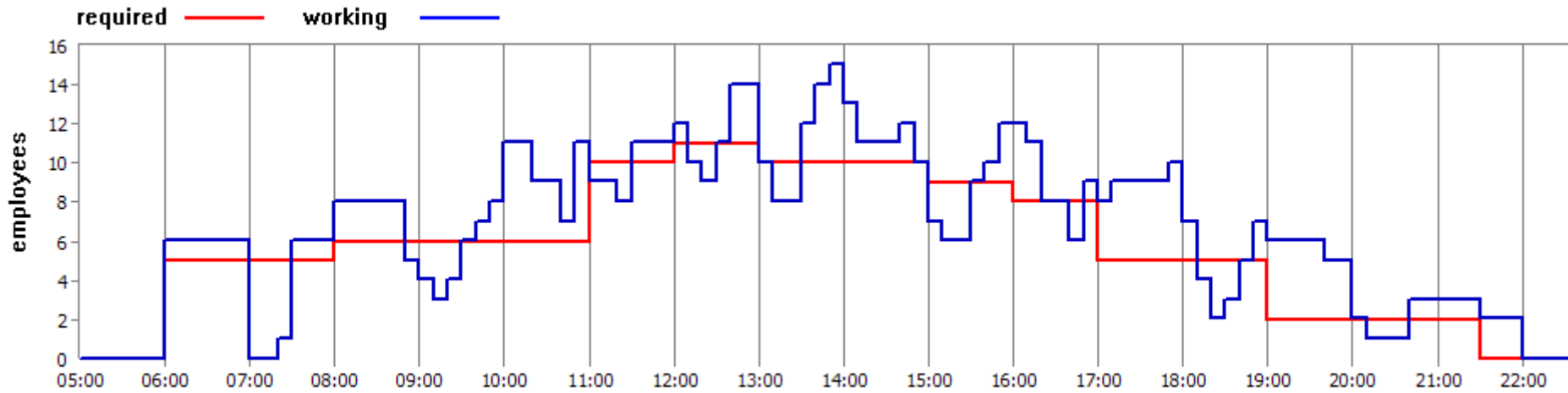
# Iteration 1



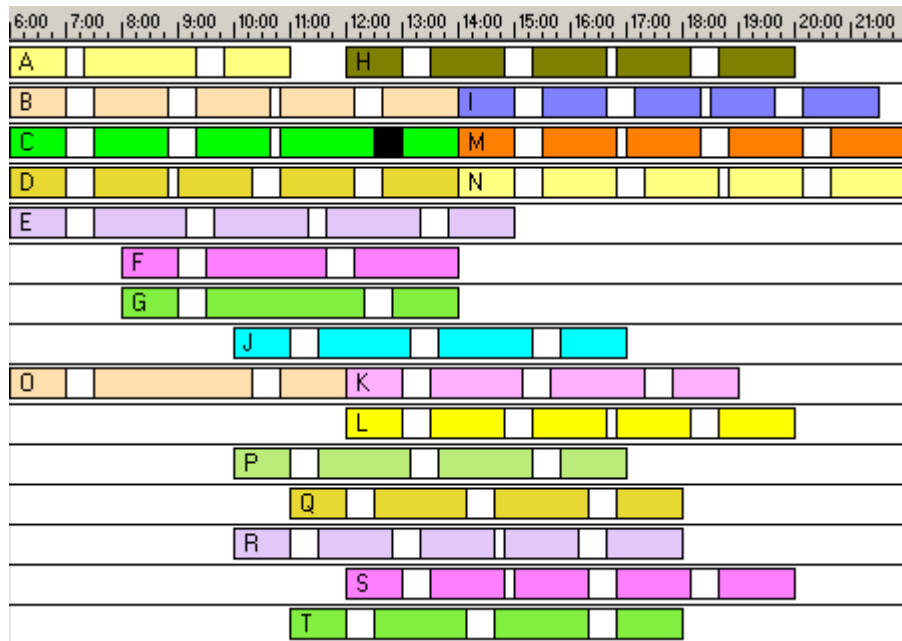
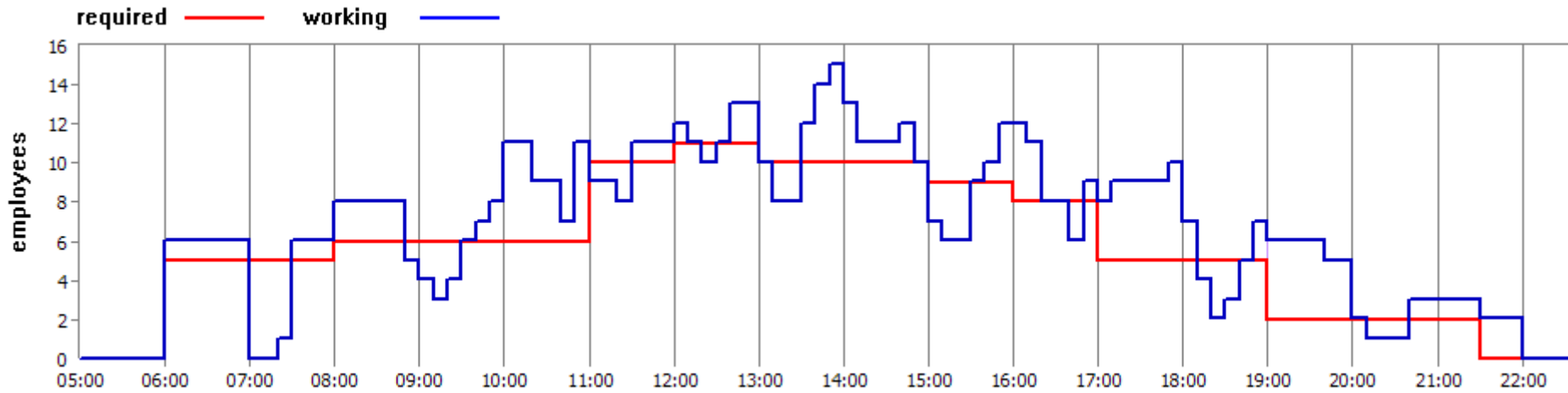
# Iteration 2



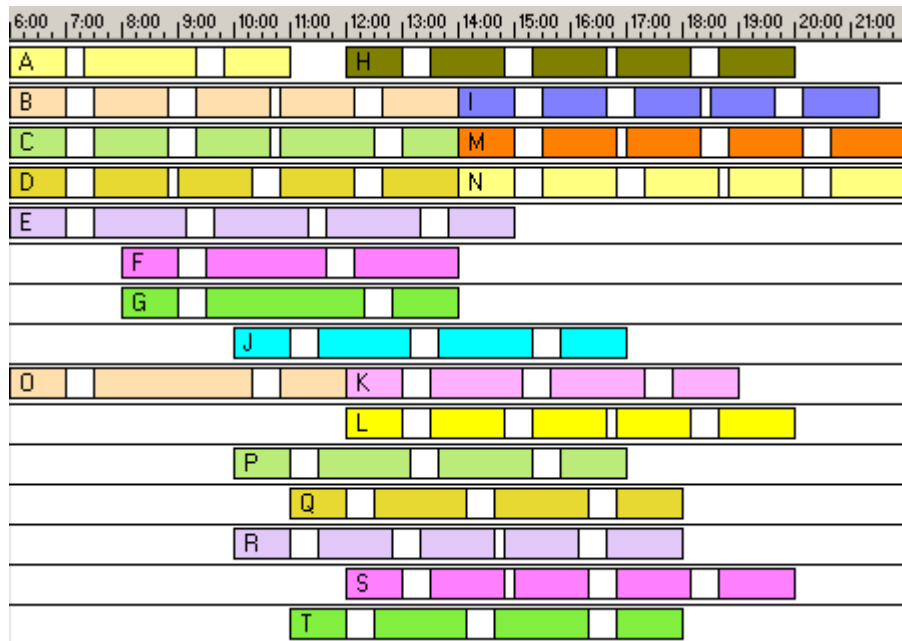
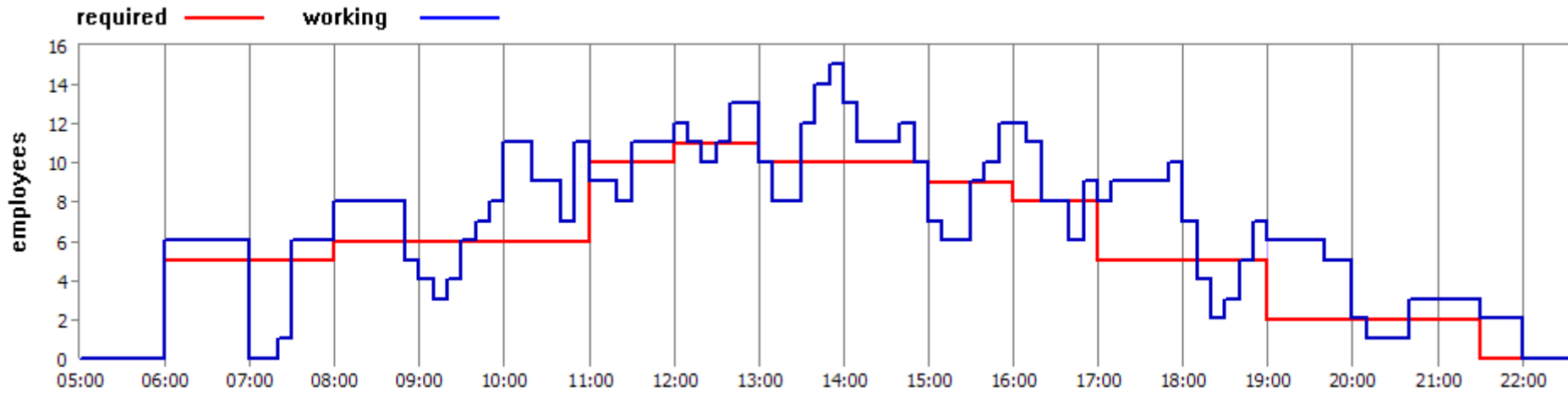
# Iteration 2



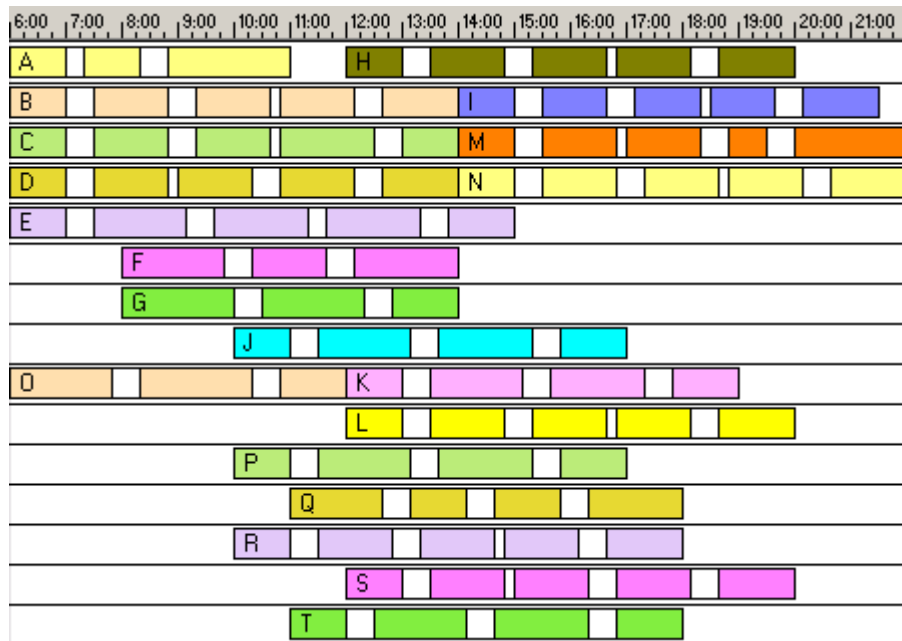
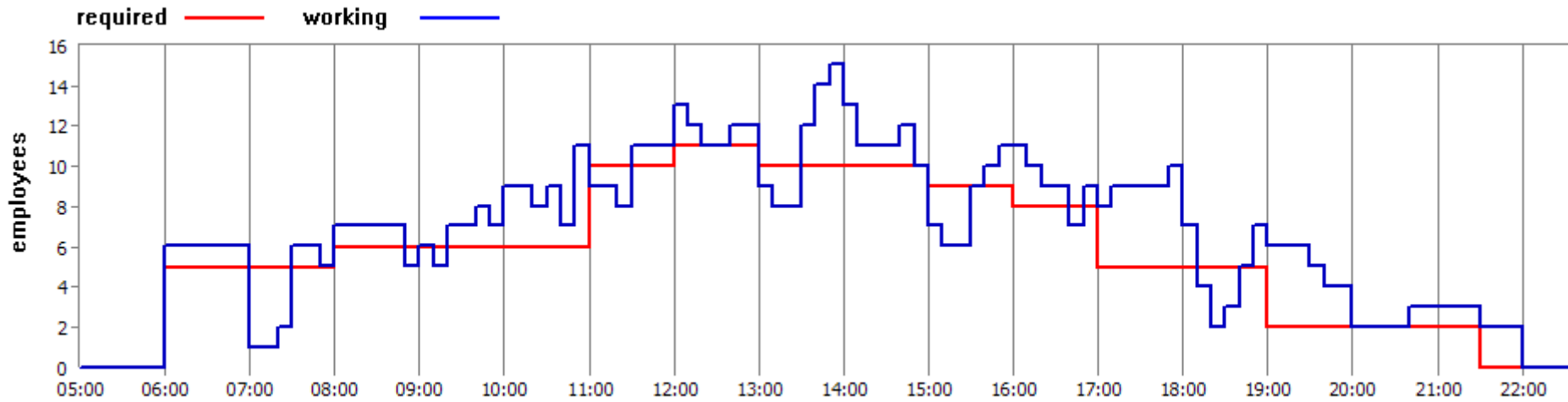
# Iteration 2



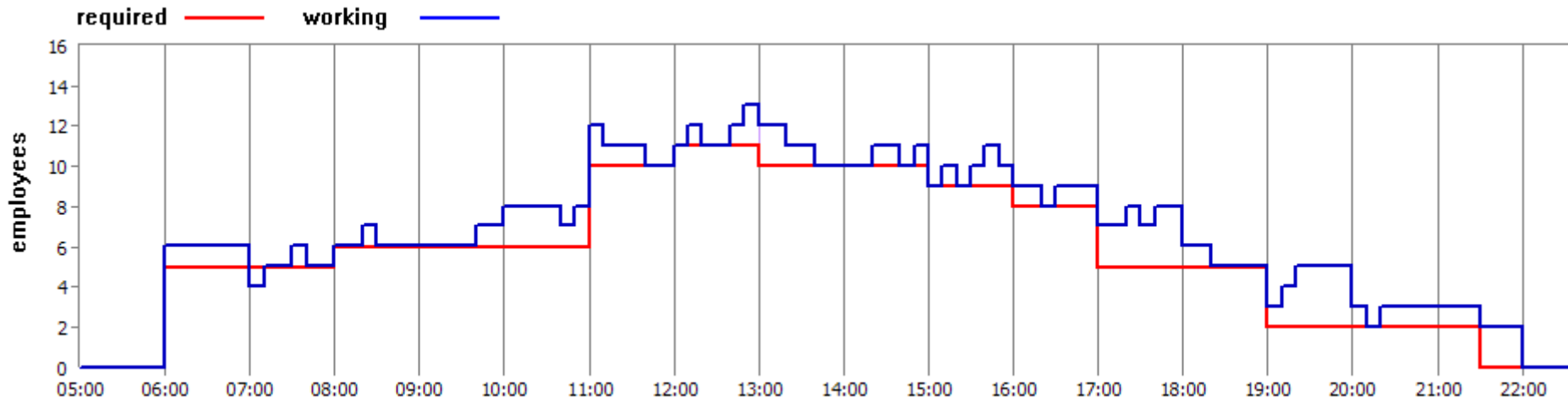
# Iteration 2



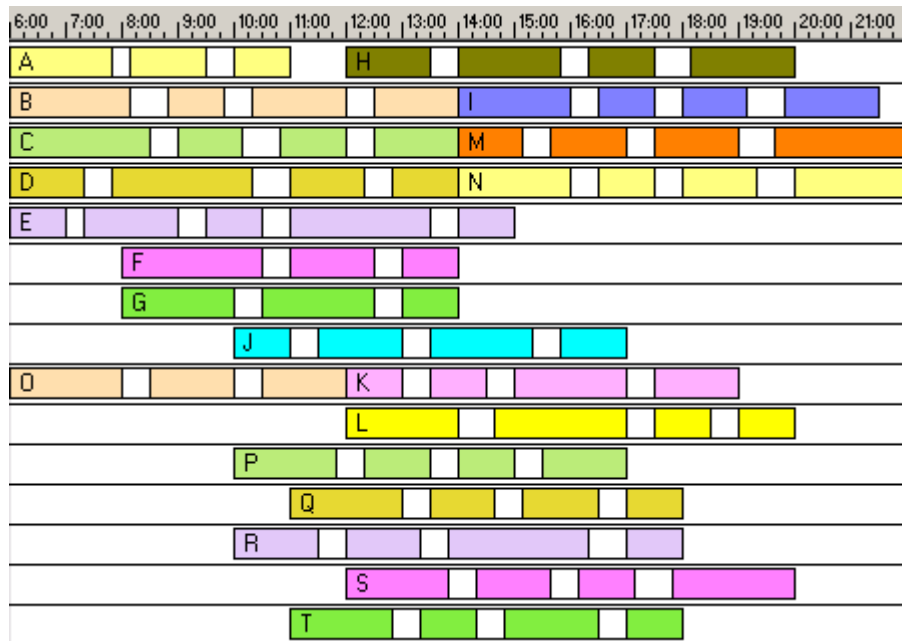
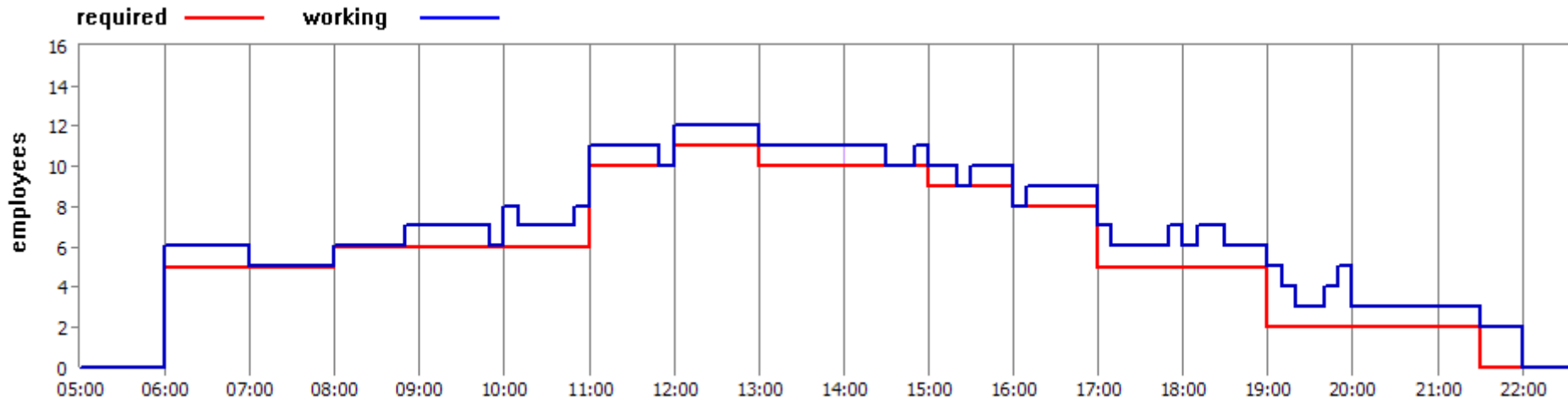
# After 10 Iterations



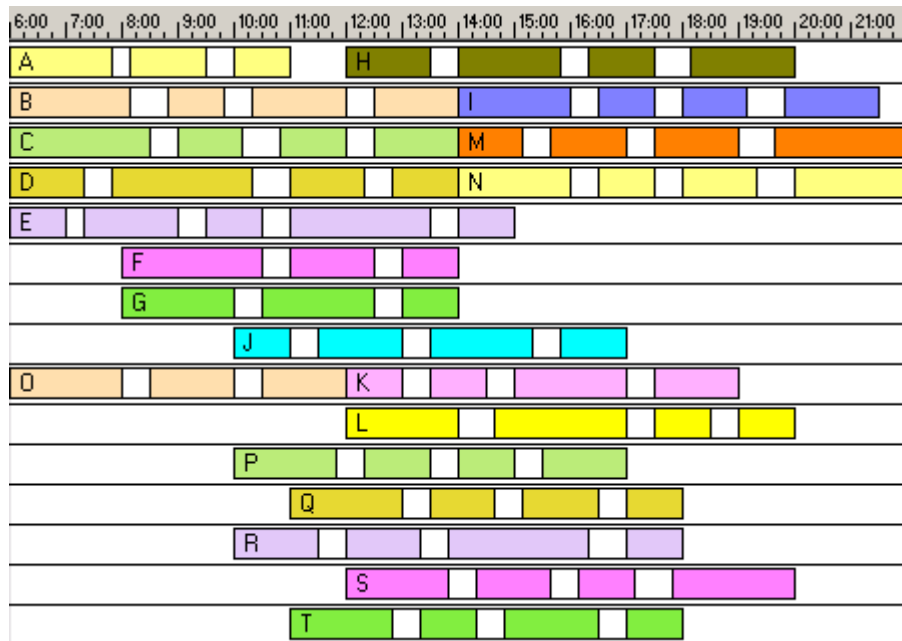
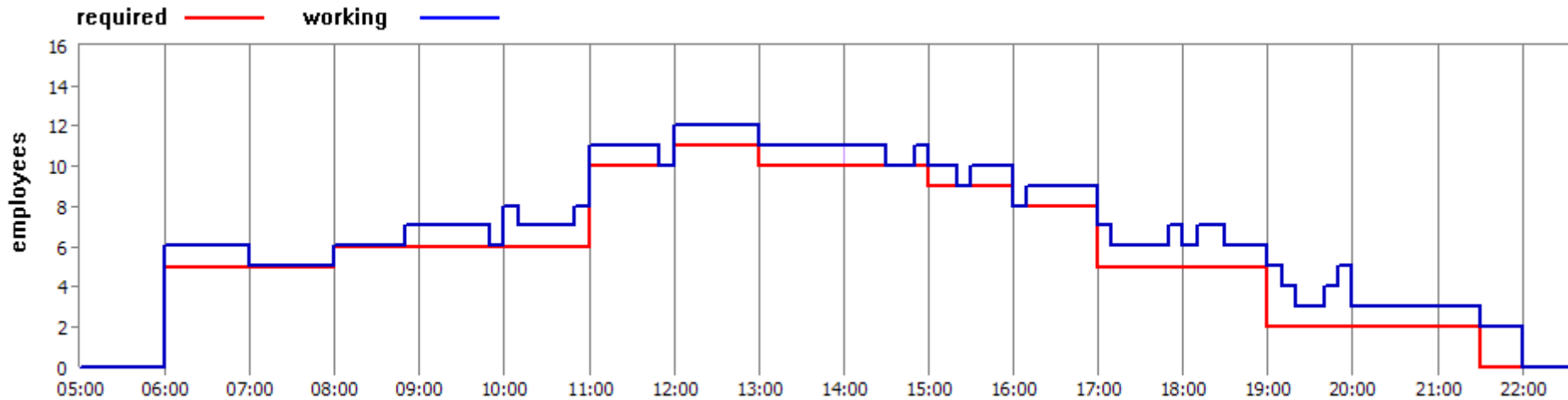
# After 100 Iteration



# Best Solution



# Quality



Constraint	Violations
Optimum break length	12 ✓
Excess	94 ✓
Squared deviation	128 ?
Others	0 ✓

**Close to optimal!**

# Conclusions and Future Work

- The proposed methods have been used successfully for real life problems and have produced very good solutions within a short amount of time.
- The use of COMET as a tool for the implementation of our algorithms makes it possible to extend our methods easily for different problems.
- For future work we will extend our methods to solve similar problems appearing in other working areas than call centers.
- Furthermore we will investigate the application of other meta-heuristic methods to solve break scheduling problems.