Welcome to Pairing I

Developed by
Jeppesen Crew Academy

for version 22 of Crew Pairing

Name Ajit Ramachandran
Born 1988
Started at Jeppesen 2012
Language skills English
Located in Göteborg

Head Teacher
Introduction to Pairing, Pairing I

Position
Service Manager, CSS
Product Owner, TD

Academic background
M.Sc. Computer Science

Experience
Support for clients after implementation (SAS, KLM and Saudia)

”I enjoy working at Jeppesen because we solve interesting problems and continuously learn new things about our products”

e-mail ajit.ramachandran@jeppesen.com

Participants presentation

- Name, company
- Role
- Experience
- Your expectations
- Other

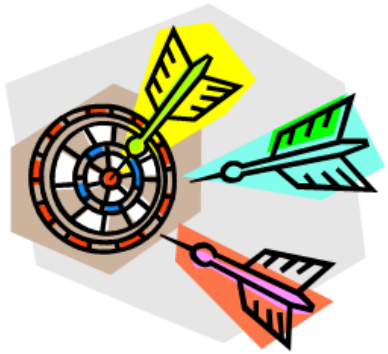
Practical Details

- **Restrooms –** in the corridor, to the right
- **Coffee breaks -** around 10.00 and 14.30 (water and fruit located in the back of the course room)
- **Phones -** are turned off (or kept on silence) and phone calls are taken outside the training room
- **Internet browsing,** typing e-mails etc. is done during breaks.
- **Lunch arrangements**
- **Evaluation**

Course goals

Enable you to:

- Create production trips using Crew Pairing
- Use manual and automatic tools
- Perform simple simulations



Prerequisites

- Pairing Introduction
  (or equivalent knowledge)

Please –

Don't be afraid to ask questions if anything is unclear!

Course material

- Course slides
- Online documentation
- Crew Pairing and Crew Rostering User Guide

Agenda day 1

- Basic functionality in Crew Pairing
- Planning concepts
- Create a standard solution
- **Lunch around 12.30**
- Introduction to optimization
- Influence the solution
- Create a dated solution

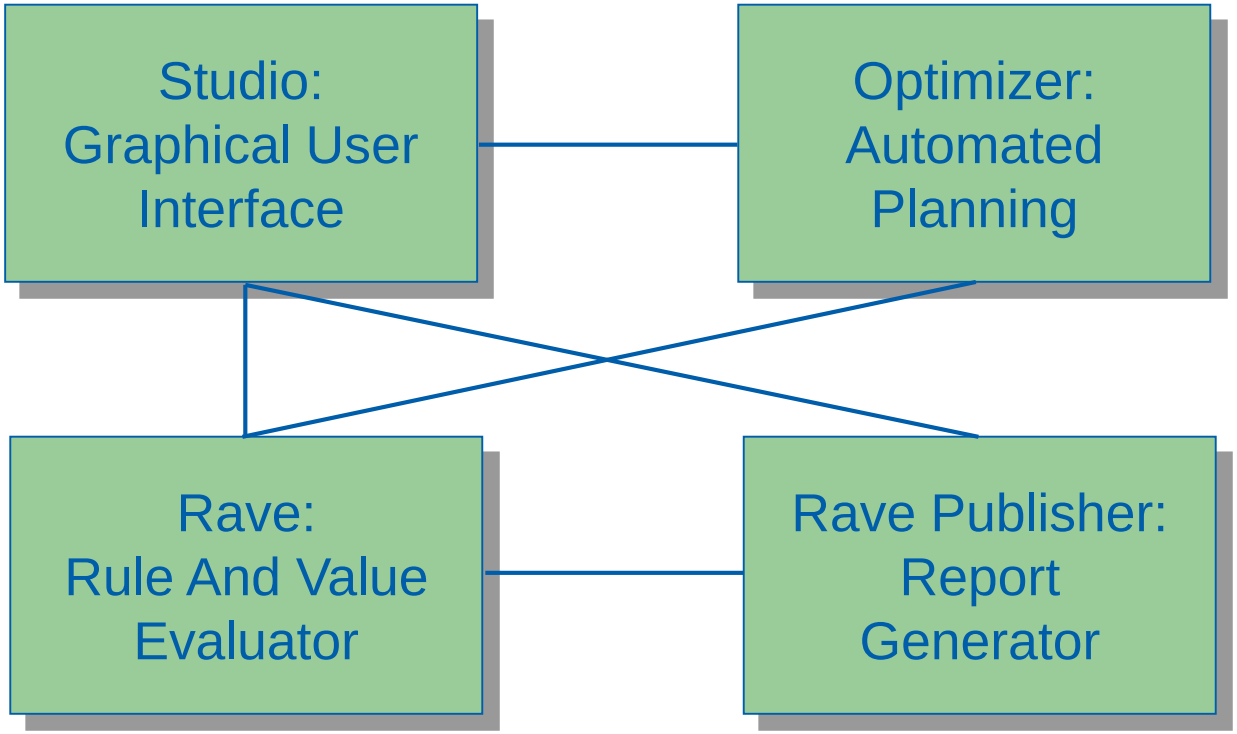
Coffee break around 10.30 and 15.00

Agenda day 2

- Create a dated solution, continuation
- Variable crew need approaches
- Summary and course evaluation

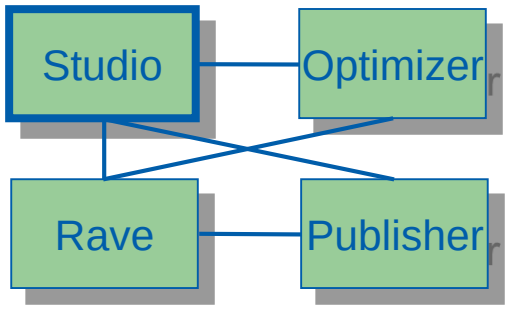
Coffee break around 10.30

Crew Pairing



Studio

Includes tools for:

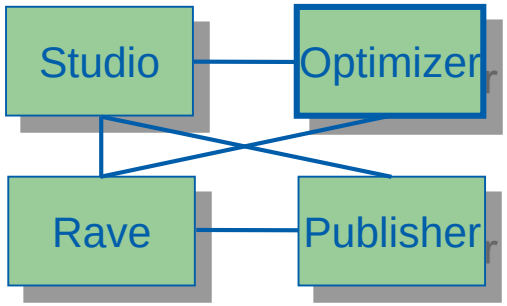


- importing timetable and OAG files
- modifying timetable information
- building rotations
- switching rules on and off
- changing parameters (costs, penalties, global constraints etc.)
- building/modifying duties and trips manually, either according to rules or violating rules deliberately
- creation of reports using Rave Publisher
- sending problems to and getting solutions from Crew Pairing Optimizer

Optimizer

Crew Paring Optimizer solves Crew Pairing problems to optimality (or near optimality) using different optimal or heuristic methods designed for various problems:

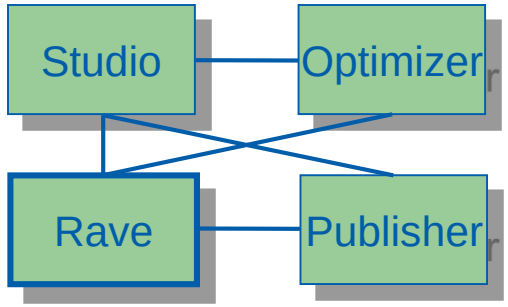
- daily, weekly and dated problems
- short-haul and long-haul problems
- other carrier deadheading
- deadhead optimization



All trips comply with rules as defined in Rave.

Rave

Rave is a programming language for modelling rules, costs and report definitions. It is specifically designed for modelling crew scheduling problems.



Rave:

- is internally developed
- is easy to learn
- allows systems to change with reality
- allows turning rules on/off and parameters to be changed by planners

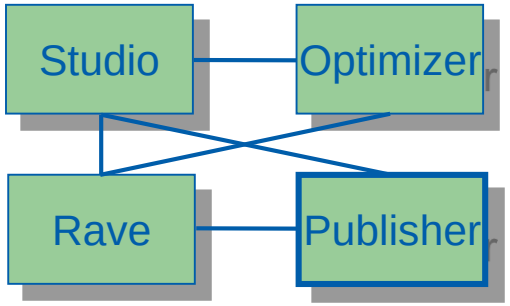
Rule code is administered by rule writers at the customer (or by Jeppesen on consulting basis).

Jeppesen offers several courses in Rave

Rave Publisher

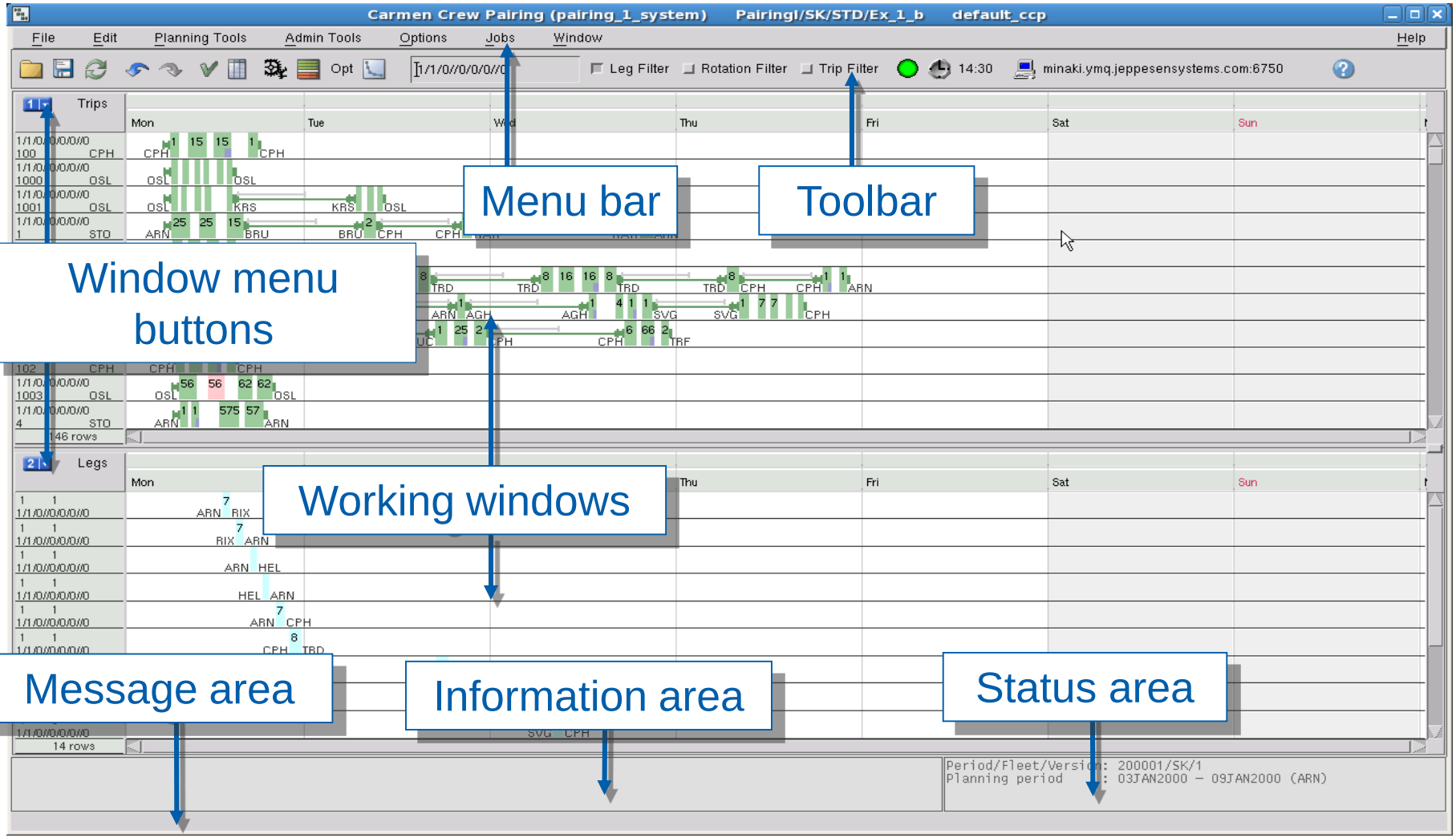
Rave Publisher:

- is an internally developed language for generating reports
- generates graphical, typeset and pure ASCII reports (for system input/output)

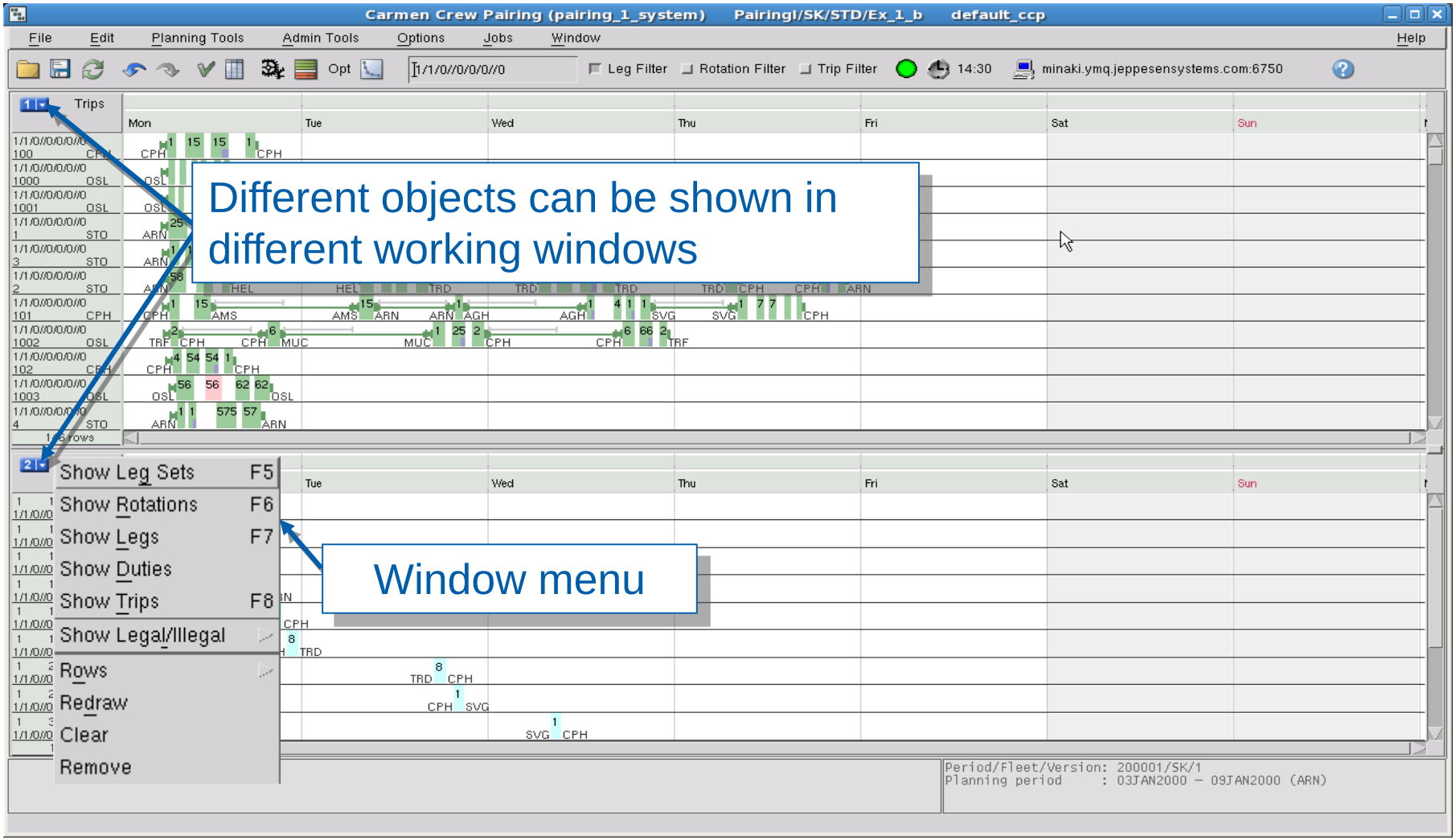


Report code is administered by report writers at the customer (or by Jeppesen on consulting basis)
Jeppesen offers several report courses.

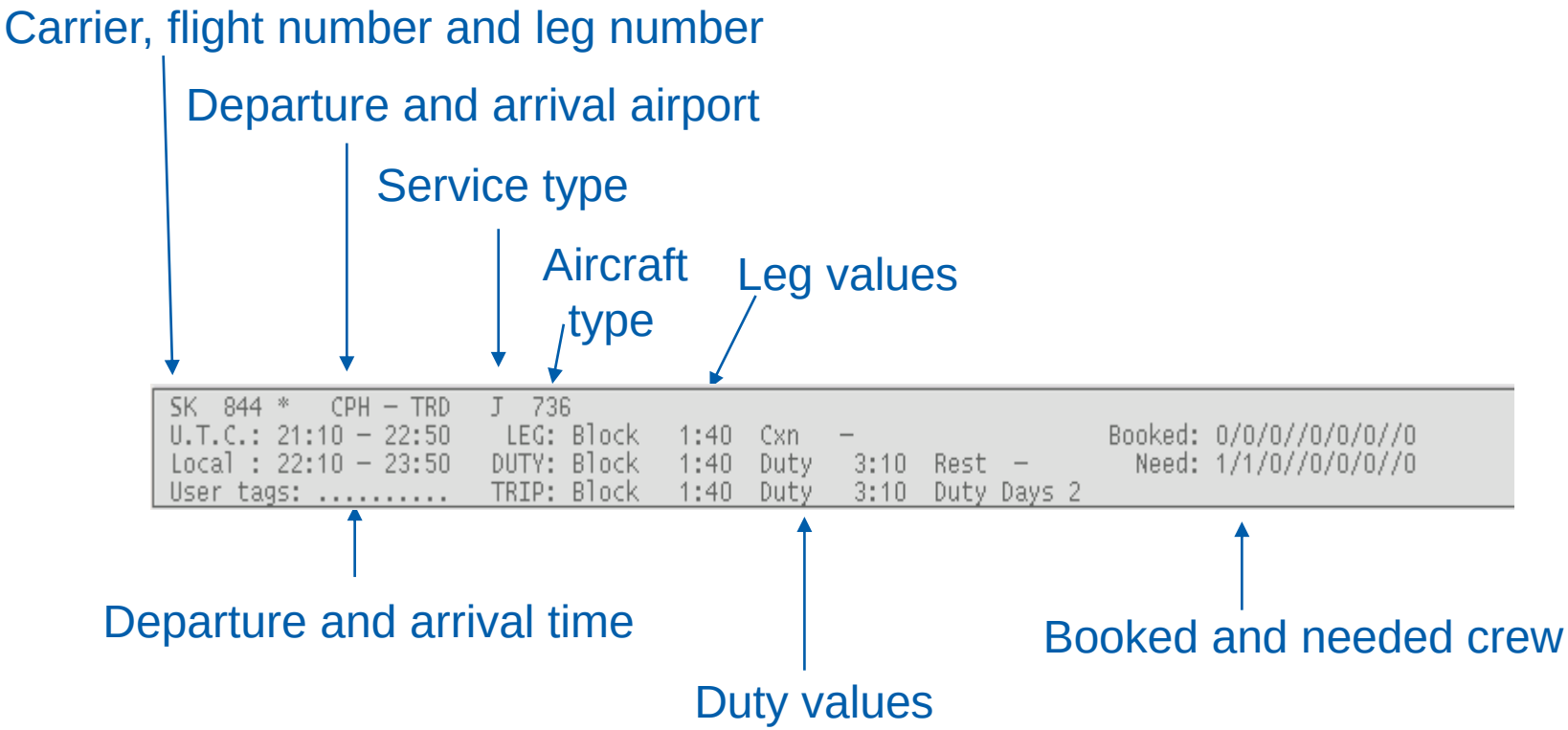
Studio



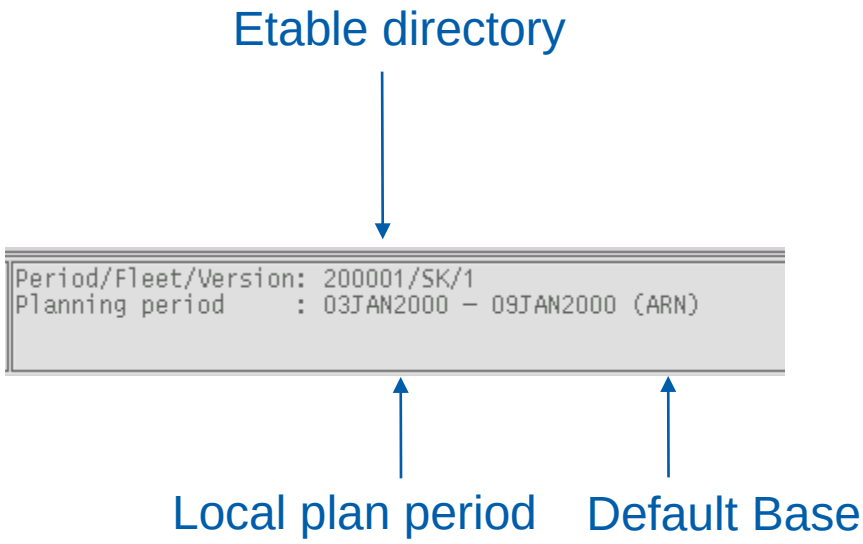
Studio



Information area
default when pointing at a leg



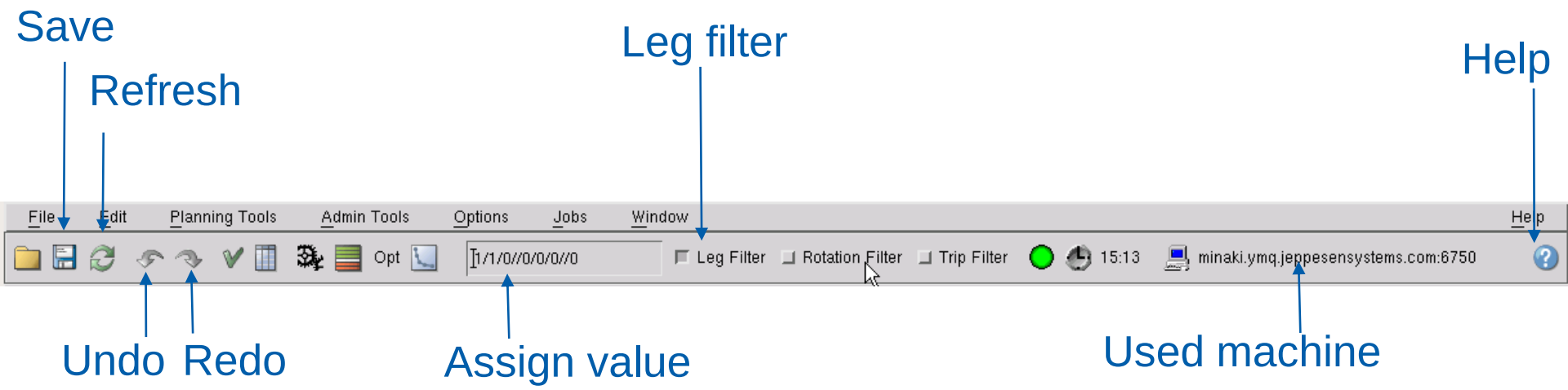
Status area
default



Menu bar

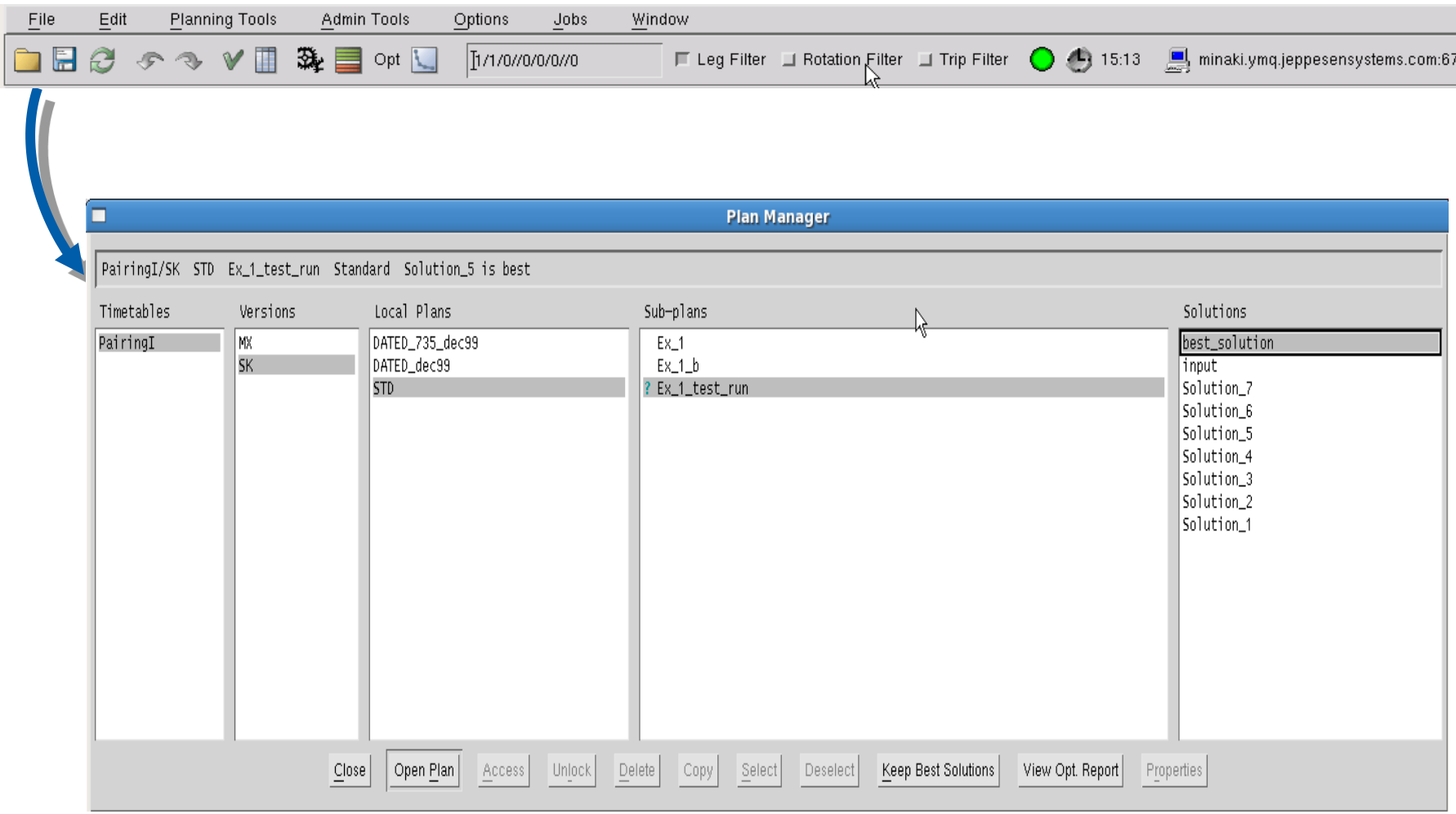
File	New, Open, Save etc.
Edit	Undo, Redo, Repeat
Planning Tools	Airport Manager, parameters, external tables etc.
- Sub-plan	Operations on the loaded Sub-plan
- Local Plan	Operations on the loaded Local Plan
Admin Tools	Rule/report development, system files etc.
Options	Studio settings, colours etc.
Jobs	Start Optimization, Batch Job Manager, Process Manager
Window	Number of windows and their contents
Help	Help etc.

Toolbar

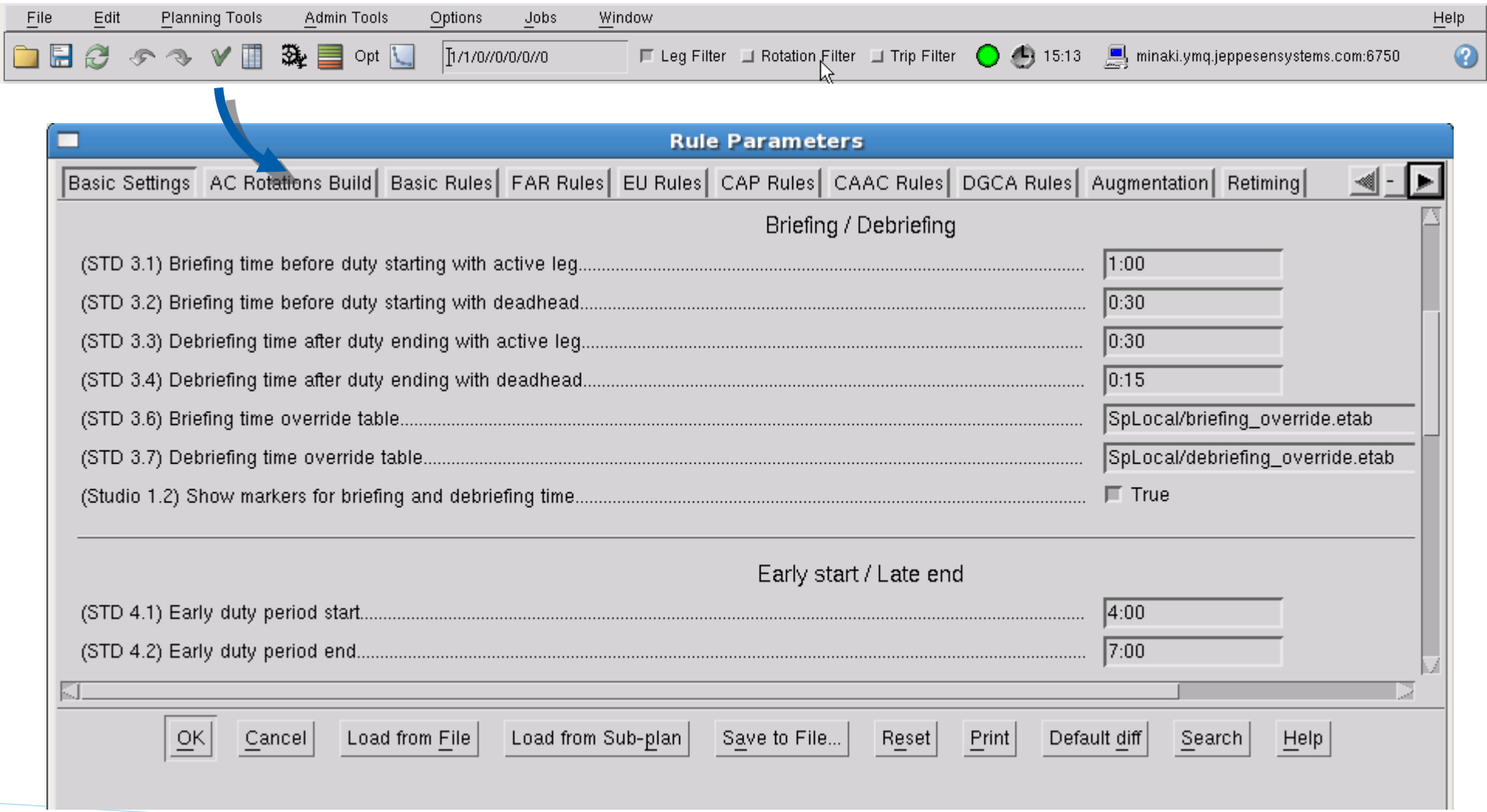


The other toolbar buttons are described in detail soon.

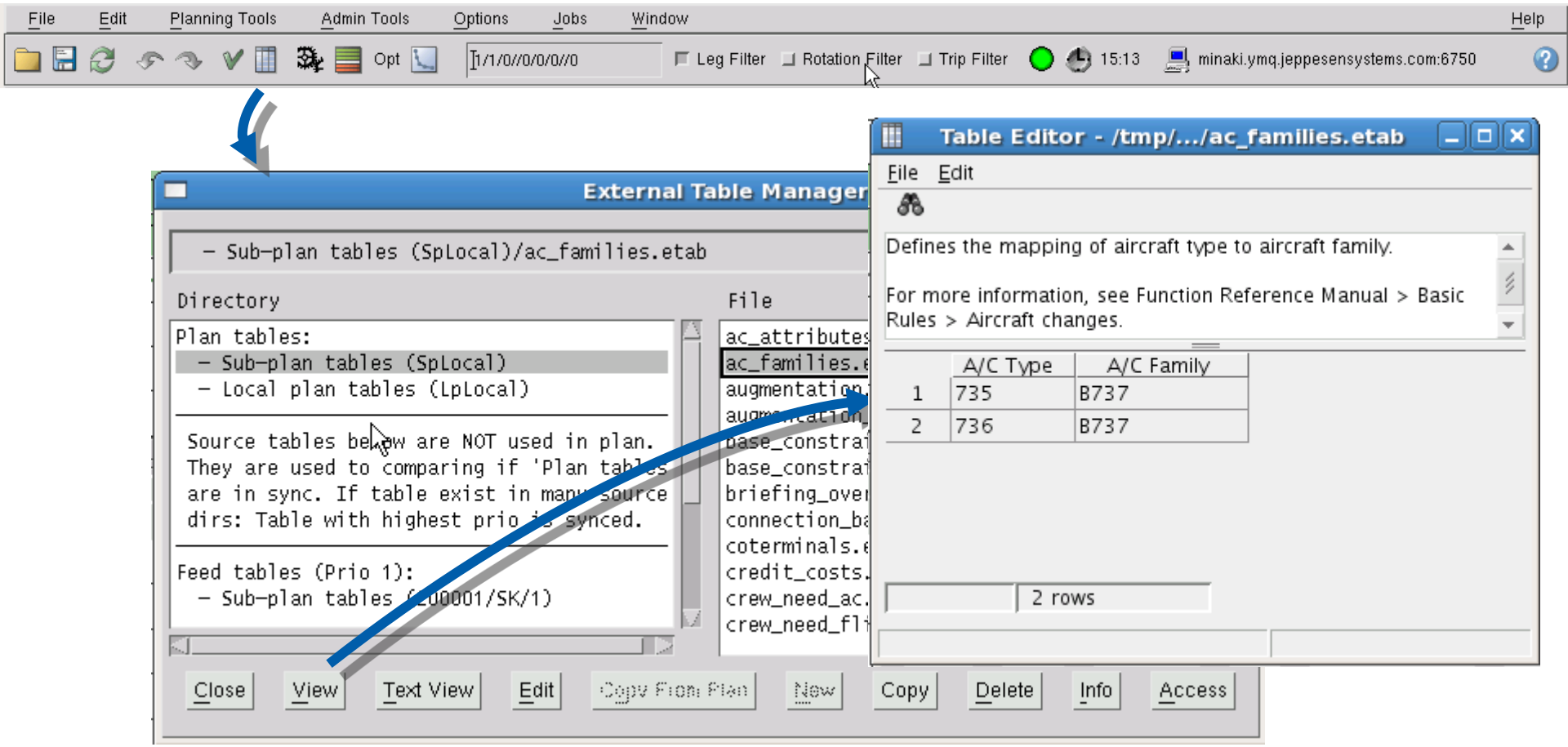
Toolbar – Plan Manager



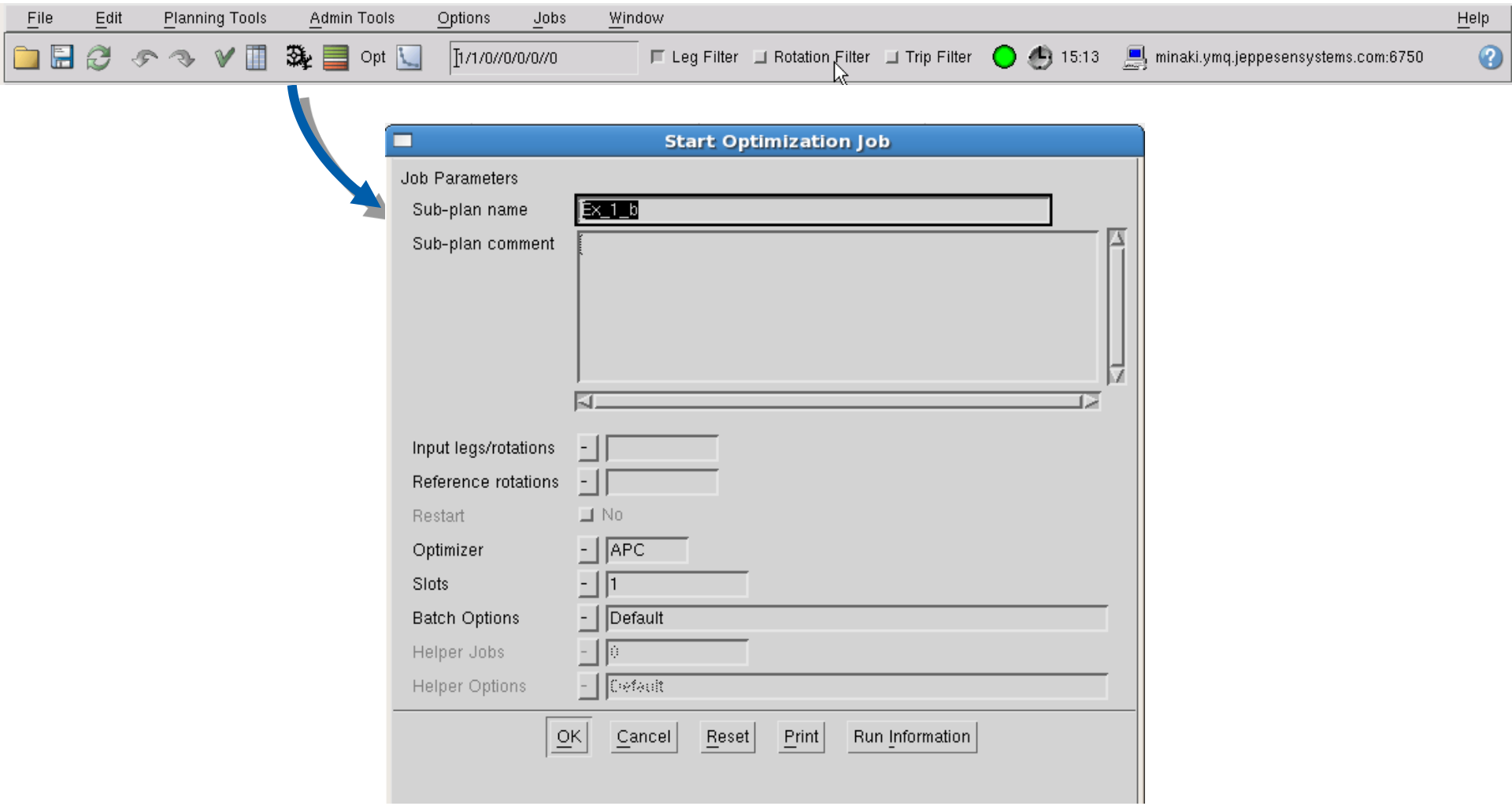
Toolbar – Rule Parameters



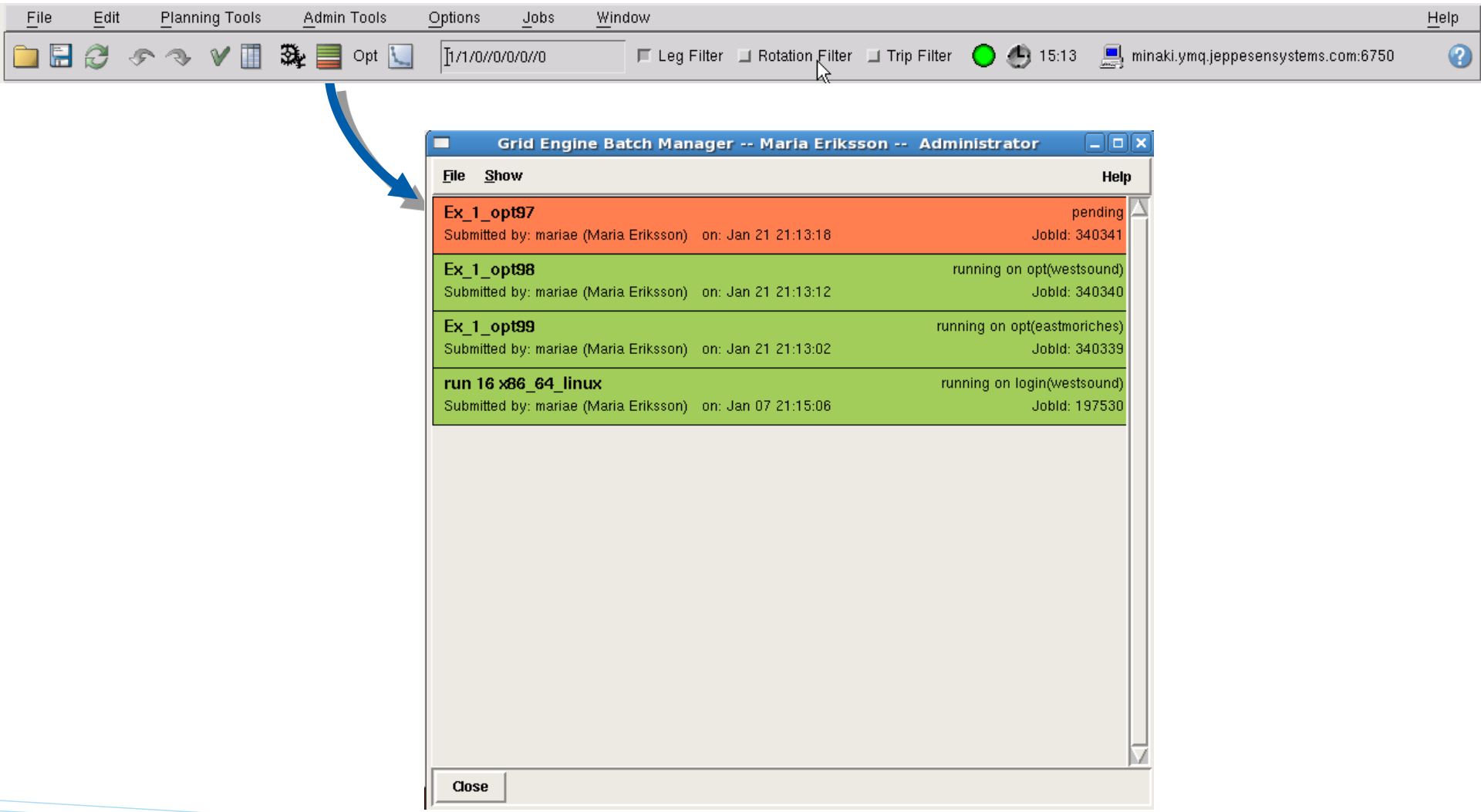
Toolbar – External Table Manager



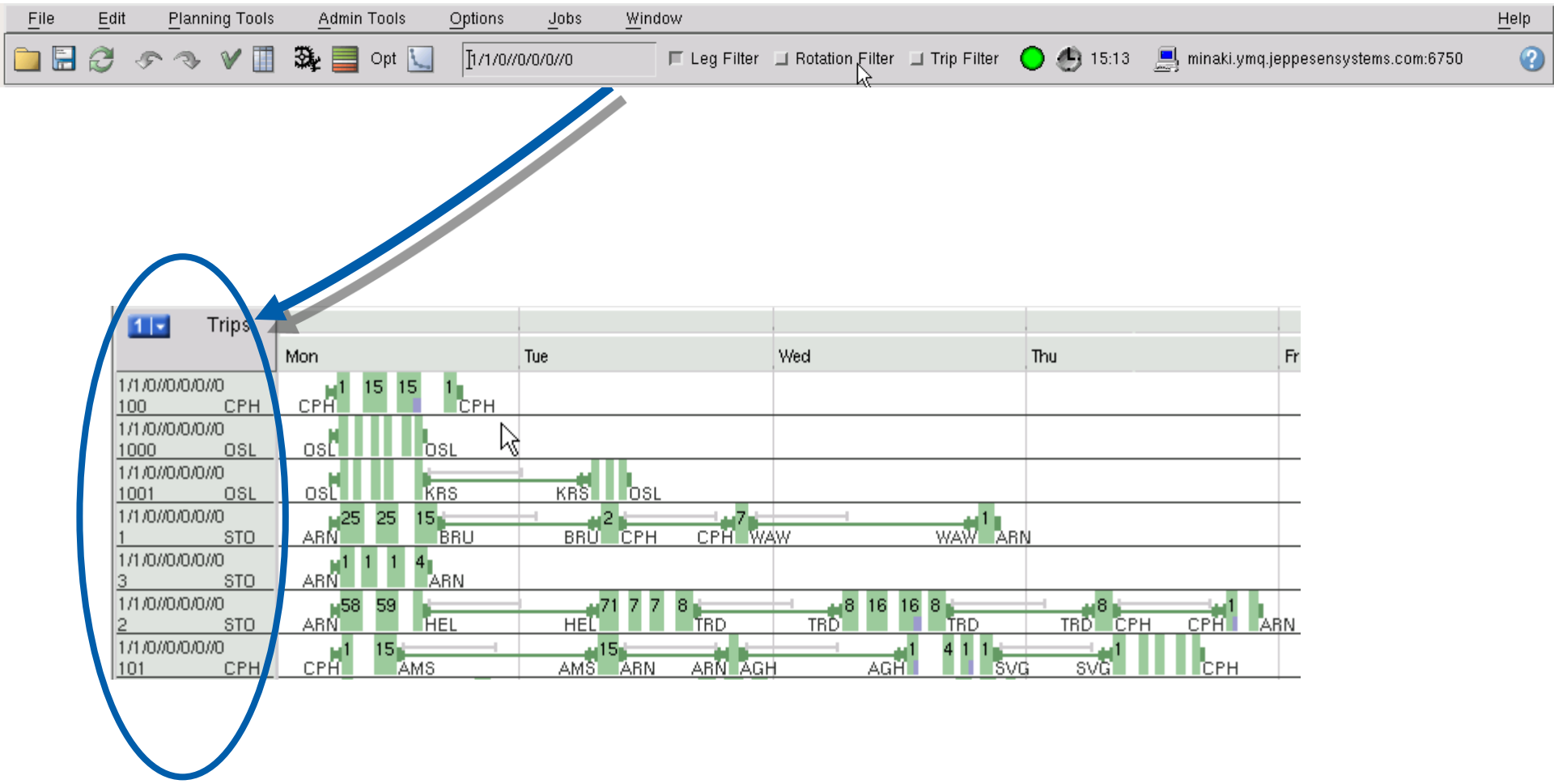
Toolbar – Start Optimization Job



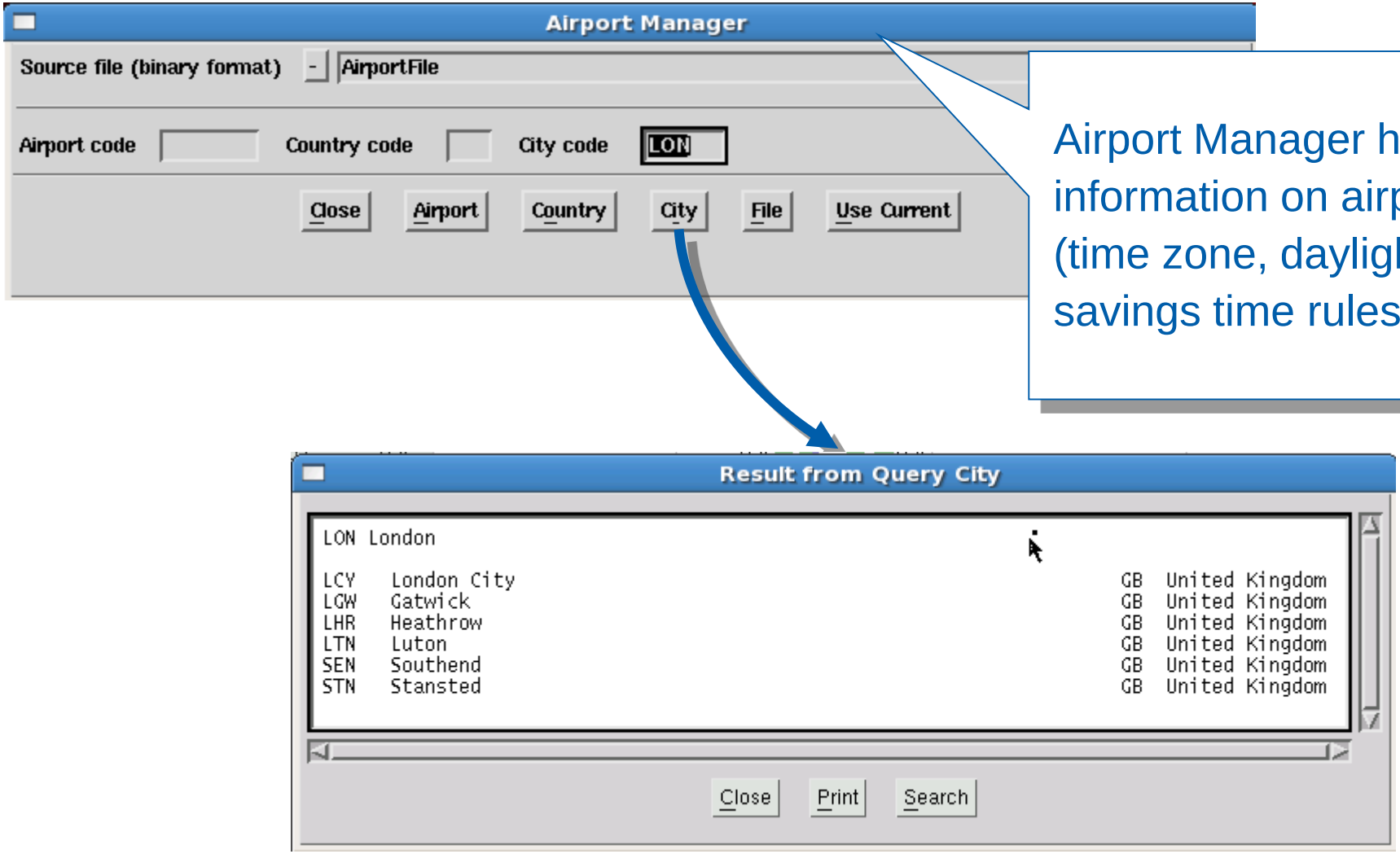
Toolbar – Batch Job Manager



Toolbar – Assign Value

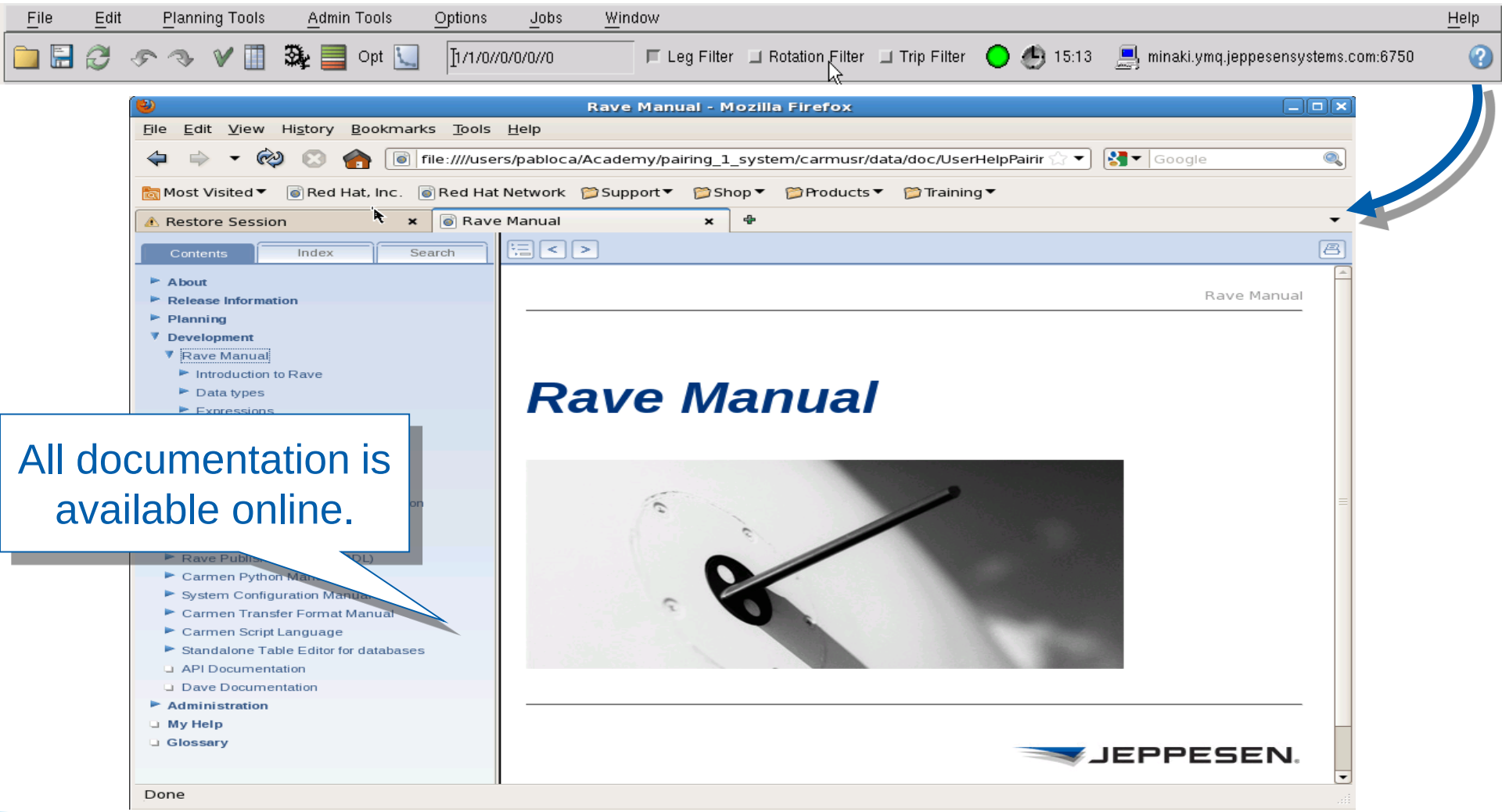


Airport Manager



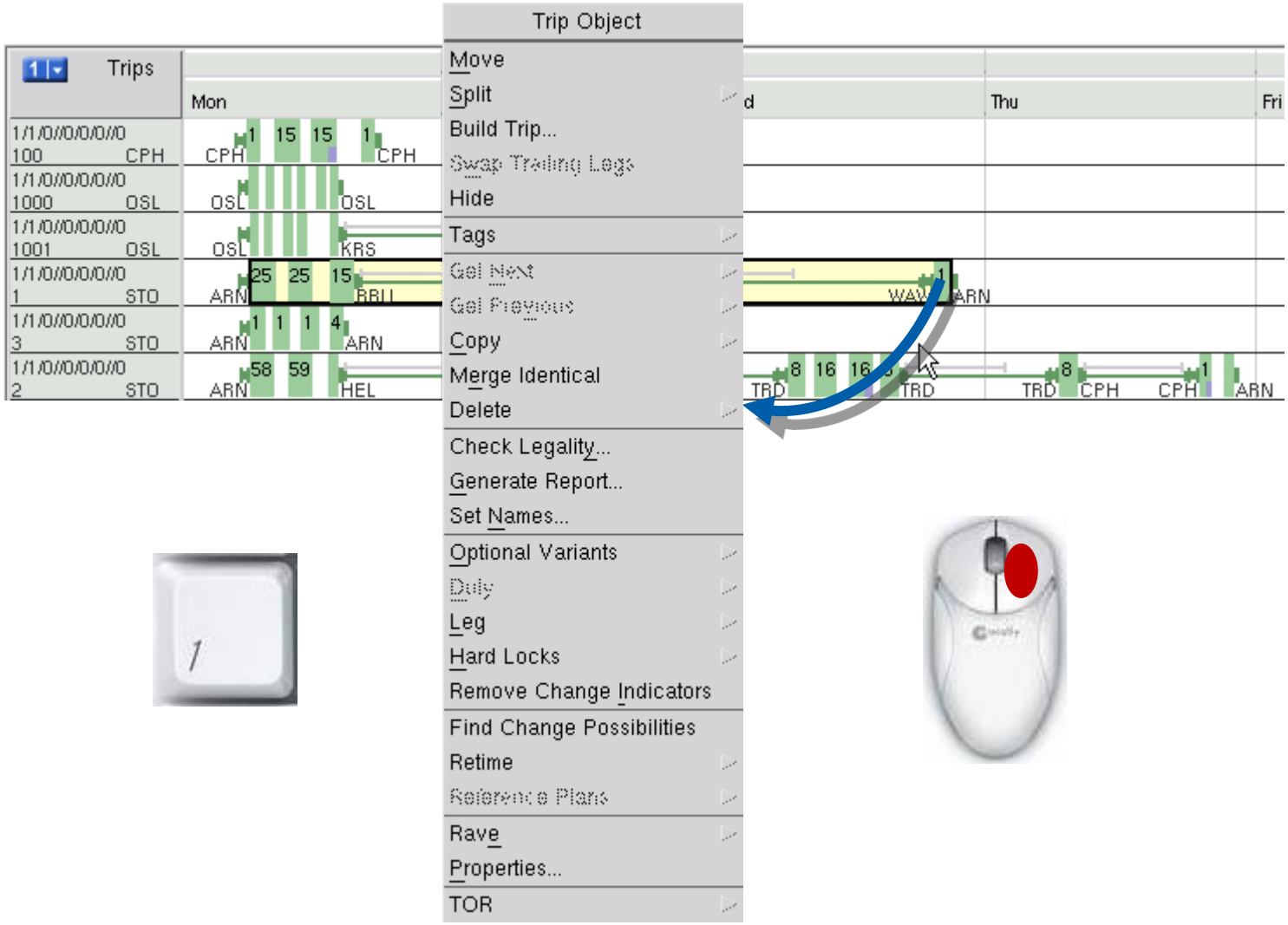
Airport Manager holds information on airports (time zone, daylight savings time rules etc.)

Help

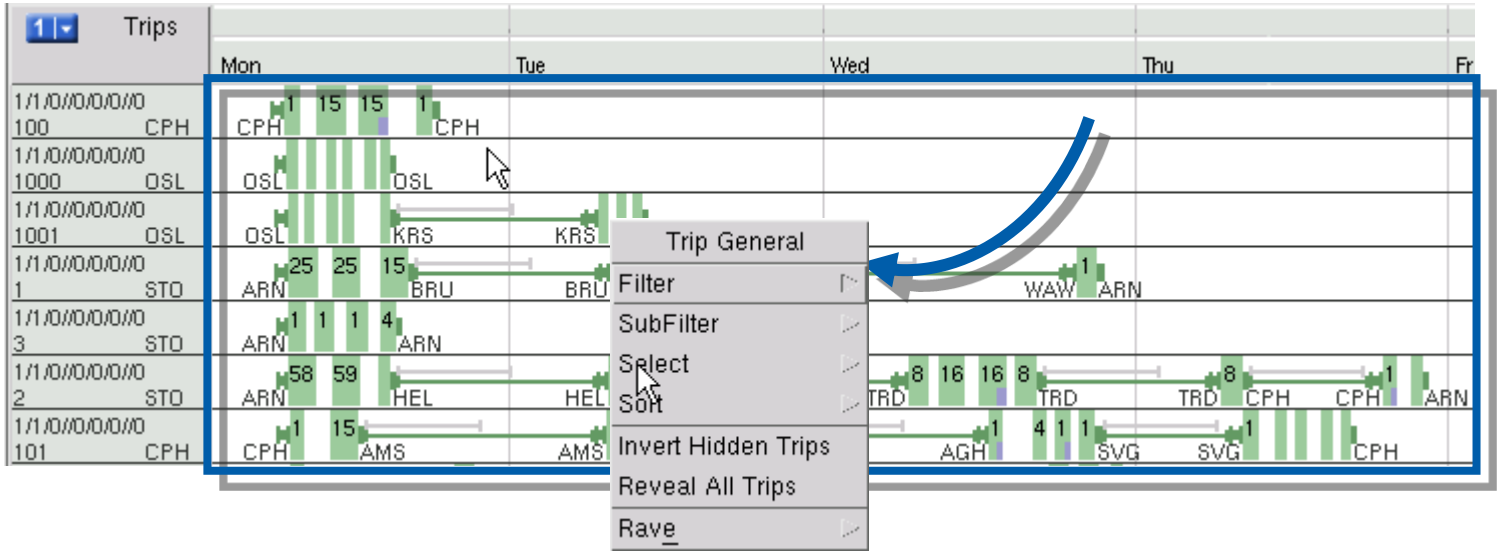


All documentation is available online.

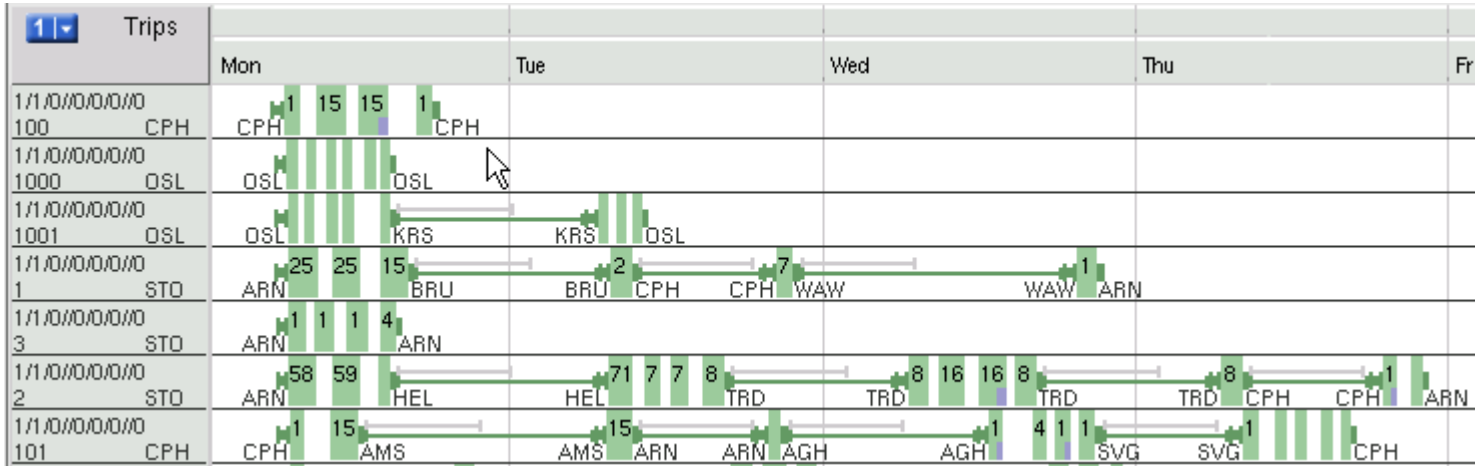
Object pop-up



General pop-up



Cancel command



Manual planning

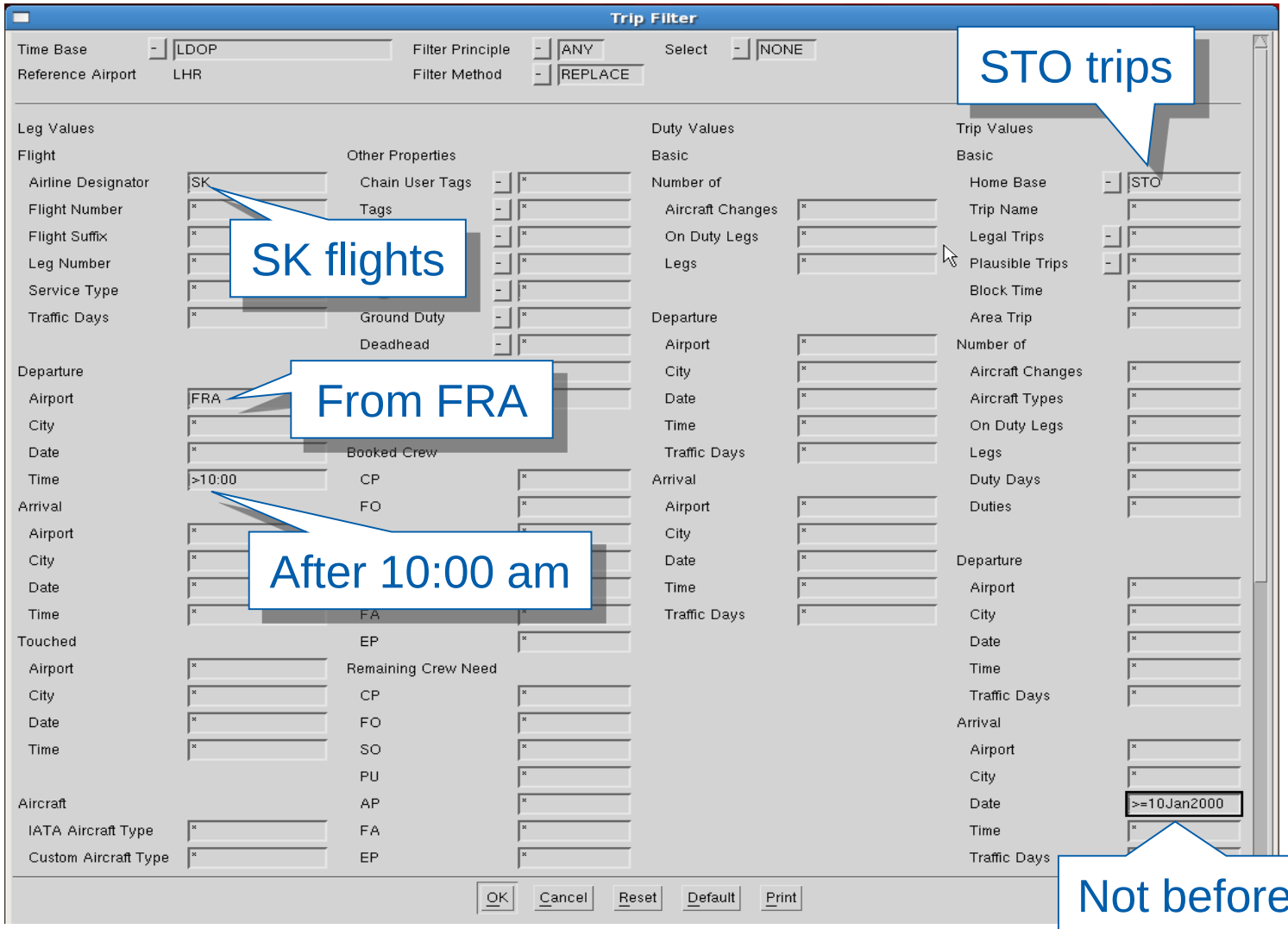
- Filter
- SubFilter
- Select
- Sort
- Drag and drop
- Get Next/Get Previous
- Run reports
- ...

Sorting

Objects in the working windows may be sorted according to any sorting criteria.
For trips:

- departure time
- arrival time
- departure airport
- arrival airport
- crew complement
- …

Filter – form



Filter – form - expressions

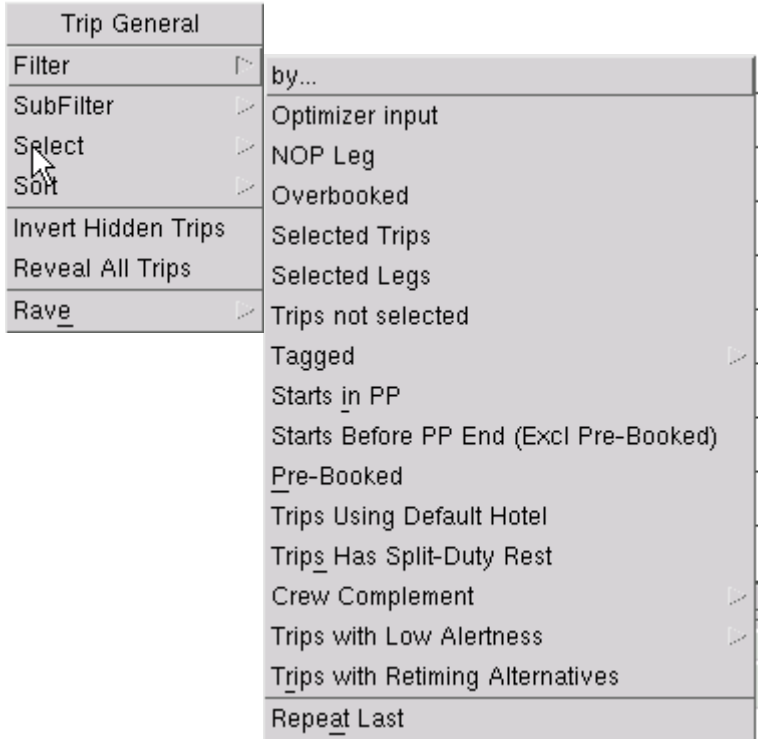
Different expressions can be used in the filter form:

- <, <=, >, >=
- / interval (e.g. 1Jan2008/31Jan2008)
- ! not
- * replacing several characters
- ? replacing one character

For more information, see Help

Filter – fast

Commonly used filter criteria should be included as fast filters in menus.



Select-and-operate

Select

- Left mouse button ON ANY LEG
  - Use Ctrl to extend the selection (or to deselect)
  - Double-click to select leg ON LEG

Rubber-banding possible when start beside object

Pop-up menus

- Right mouse button
  - On leg: Object menu
  - Beside legs: General menu

Only operates on selected objects

Drag-and-drop

- Between and inside windows
  - Drag with left mouse button
  - (Drag with right mouse button if want to use menu)

Select-and-operate – Left Margin

Not supported in Leg Set and Leg window

Rubber-banding not possible

Select rows

- Left mouse button
  - Use Ctrl or Shift to extend the selection

Drag-and-drop (same result left and right mouse button)

- Within a window -> “sort”
- To another window:

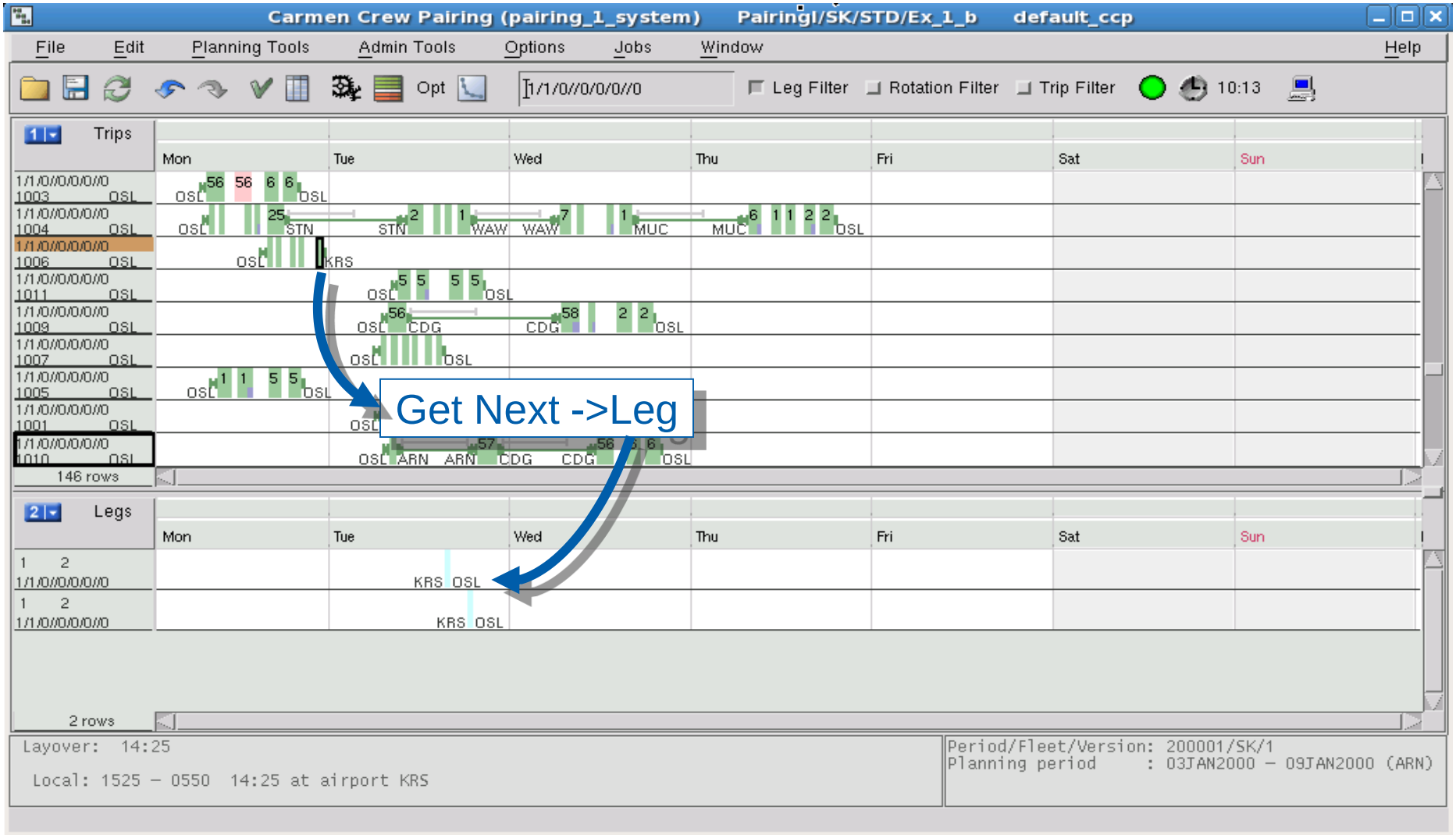
  drag -> “move”

  Ctrl + drag -> “copy”

Select-and-operate – Useful keys

Delete	Trip, Duty: delete to single legs
	Legs: delete from sub-plan
	Object pop-up menu
	General pop-up menu
Ctrl-A	Select all legs in current main window
Ctrl-M	Select with form
Ctrl-Z	Undo

Manual planning

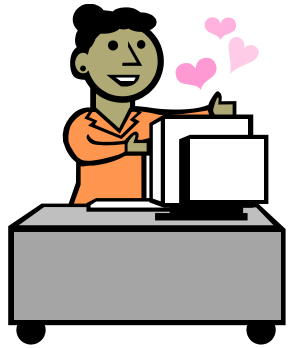


Exercise 1

Basic functionality

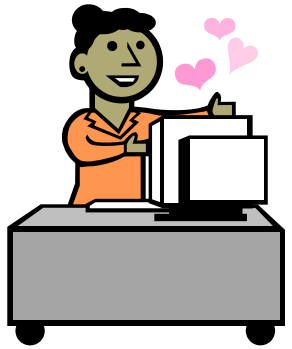
Purpose

- Get familiar with working screen and menus
- Learn to use manual planning tools and reports



Exercise 1 – summary

Summary of exercise 1



Planning concepts – local plan

Contains

- all the timetable data: Leg Sets
- the aircraft rotations (turns): Rotations

Many sub-plans can use the same local plan

Leg Set = a set of legs that are equal except for operation date

Rotation = a chain of legs associated with an aircraft

Planning concept – sub-plan

- Contains all the legs (references) you want to plan right now
- Is a sub-selection of the local plan (sub-problem)
- Contains these planning objects: legs, duties, trips

A sub-plan contains one planning problem.

Flight Deck: one sub-plan per fleet
Cabin: typically one sub-plan for international operations and one for domestic operations

A sub-plan can be a scenario of any of the planning problems

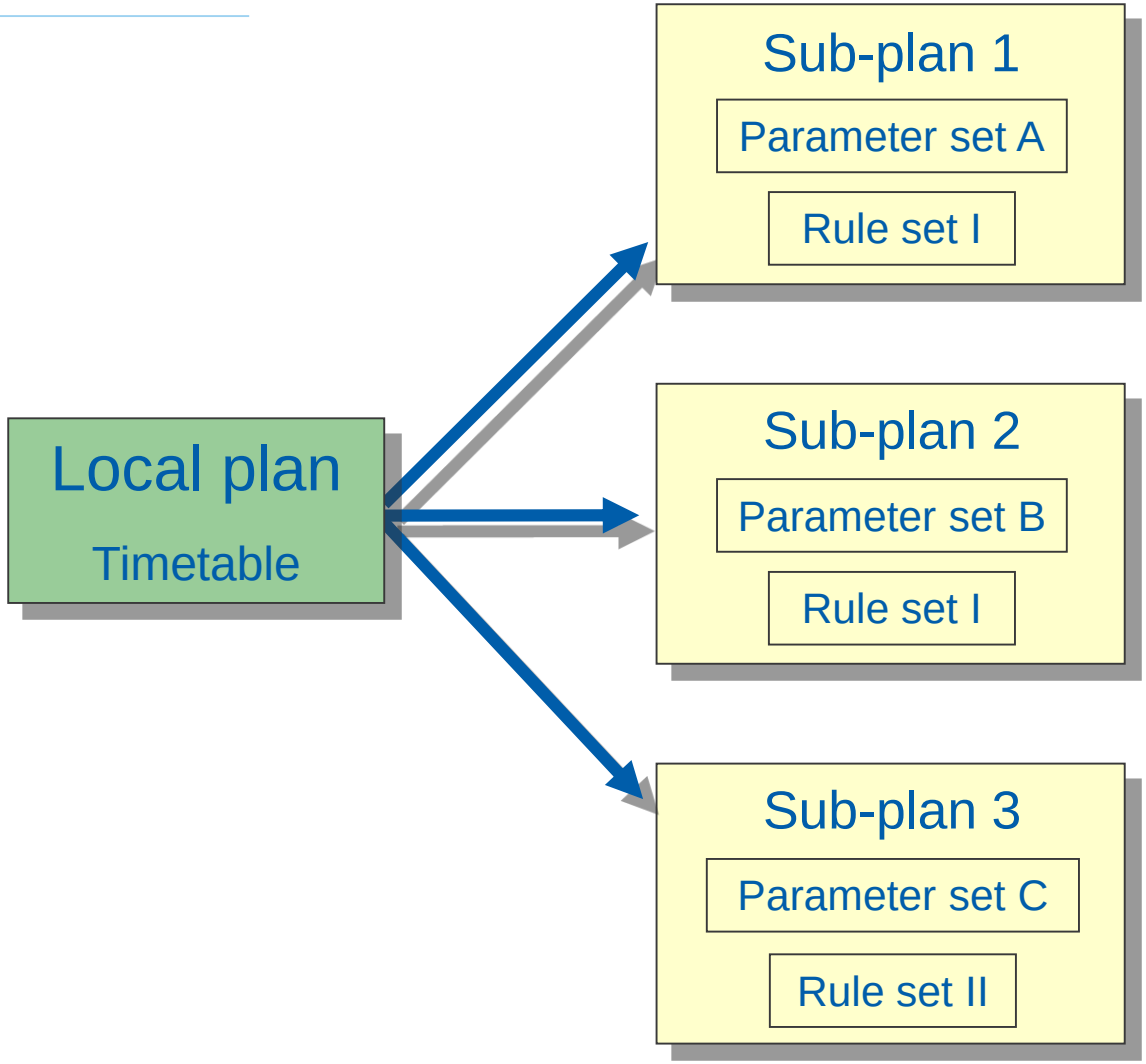
Planning concept – rule set

- Rave is a programming language
- Must be compiled for the system to understand
- A rule set is a compiled file with Rave definitions
- Includes rules, parameters, costs, crew need etc.
- Rule sets for long-/short haul, flight deck/cabin etc.

Planning concept – parameter set

- A text file where good parameter settings are saved
- Rules can be turned on and off
- Cost values (parameters) can be changed
- The behaviour of the optimizer can be controlled
- Often loaded in the start of the planning process
- Often there are ‘monthly’ parameters that need to be changed manually

Plan structure



Planning concept – crew categories

A flight needs crew carrying out various tasks. Therefore, crew is divided into **crew categories**.

Crew categories differ between airlines. A typical example is:

Flight Deck crew: Captain / First Officer / Second Officer

Cabin crew: Purser / Assistant purser / Cabin attendant

Planning concept – crew need

Each flight activity has a crew need, describing the required amount of crew in different crew categories.

Example (varies between airlines)

A leg has crew need:

1 / 1 / 0 / / 1 / 1 / 3

Flight Deck crew Cabin crew

This leg is covered by:

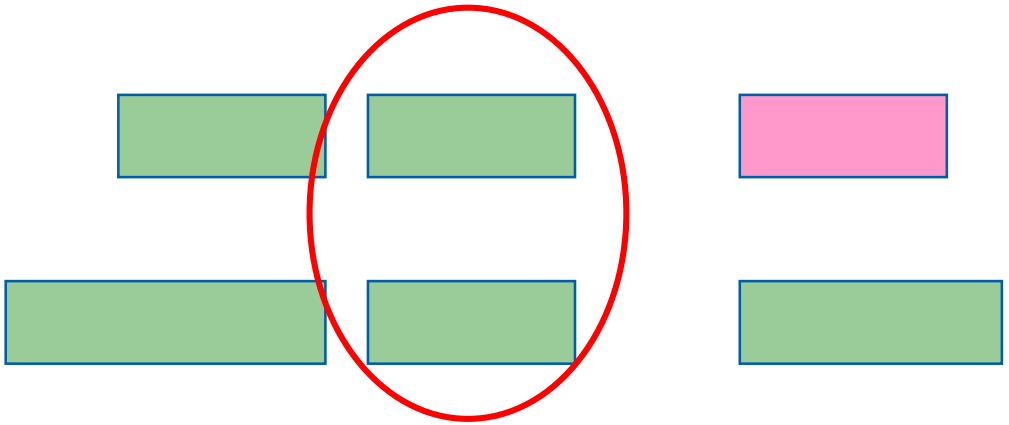
1 captain/1 first officer/0 second officer//
1 purser/1 assistant purser/3 cabin attendants

Planning concept – crew need

The crew need (1/1/0) can be covered by several trips:

Captain 1/0/0:

First Officer 0/1/0:



Planning concept – crew complement

Need (leg) – the total number of crew required on the leg
Example: 1/1/0//1/1/3

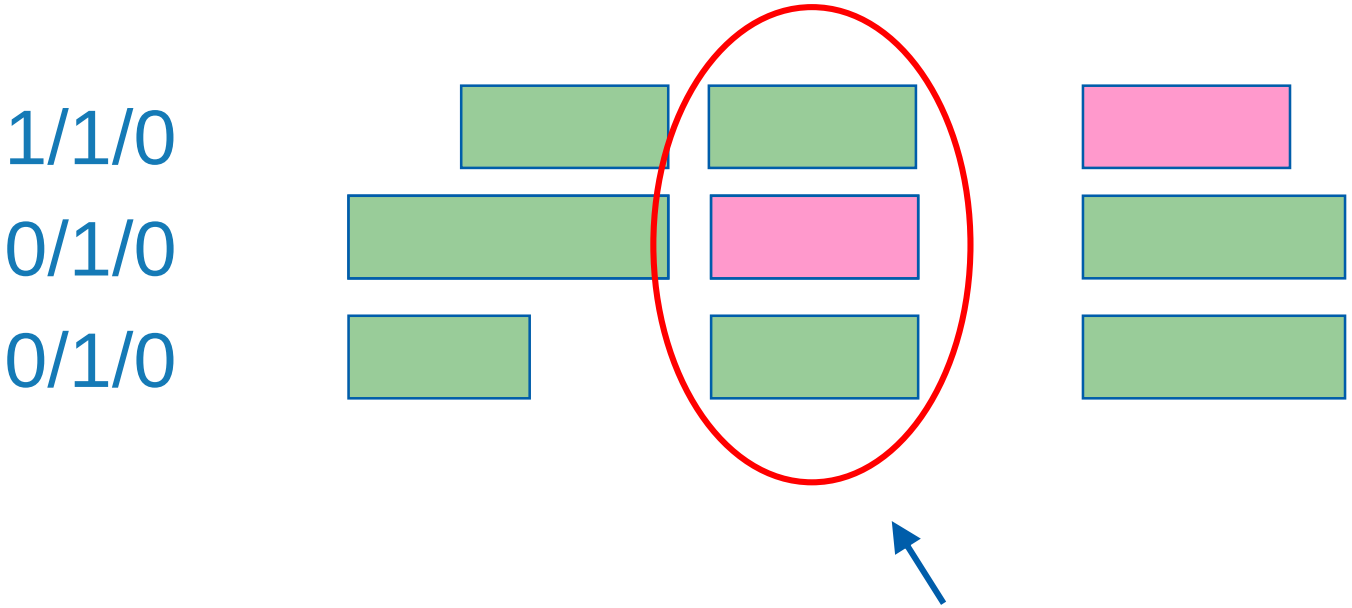
Booked (leg) – the sum of the crew complements for trips containing the leg (as on duty).
Example: 0/0/0//0/0/3

Crew complement (trip) – the number of crew the trip is constructed for

Example: 0/0/0//1/1/2

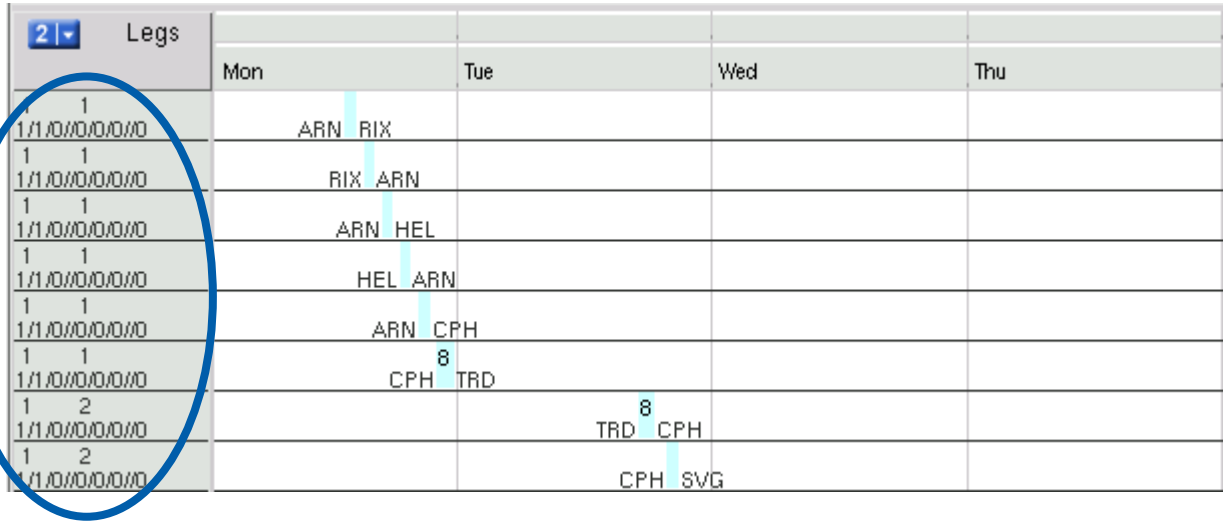
Planning concept – booked value

The leg **booked** value, is the sum of all crew complement values for all trips the leg is part of:

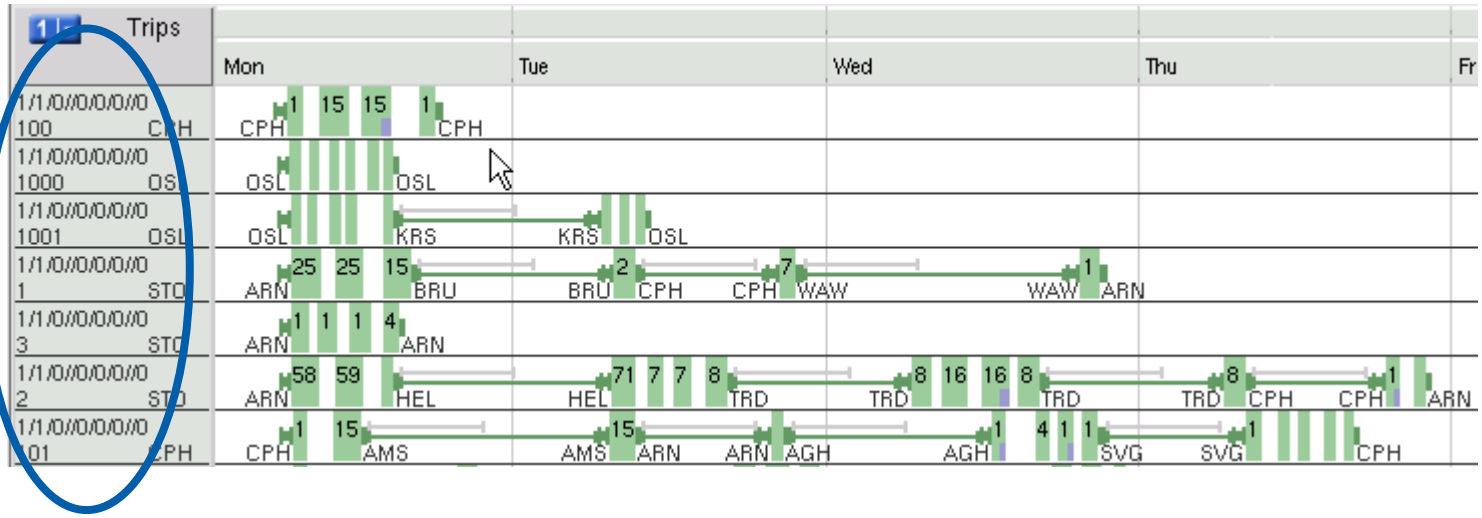


1/1/0 + DH + 0/1/0 = 1/2/0

Planning concept – remaining need



Planning concept – Crew Complement

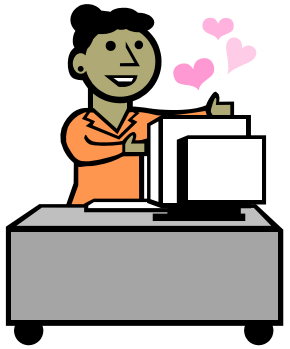


Exercise 2

Crew complement

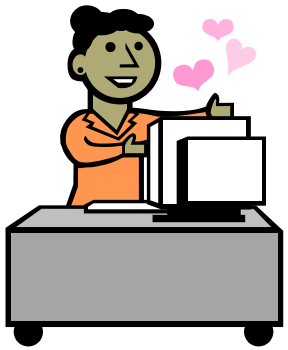
Purpose

To understand how the crew complement works



Exercise 2 – summary

Summary of exercise 2



Looking back

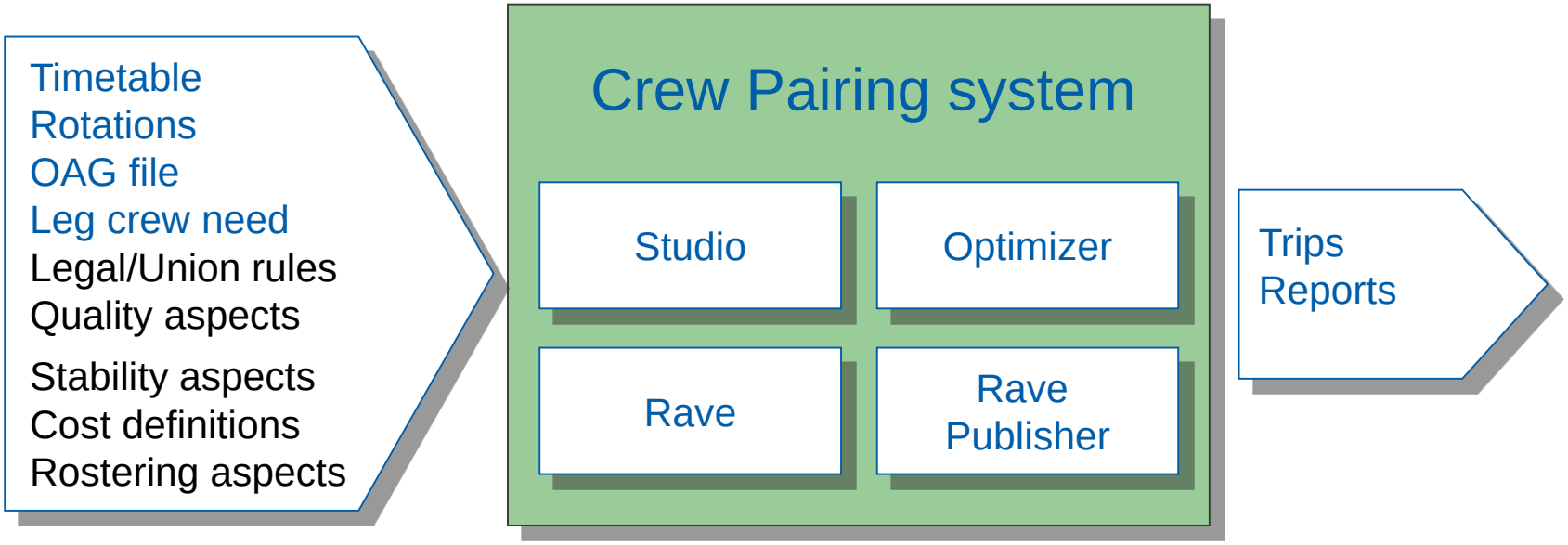
- The modules of the Jeppesen system
- The user interface of the Jeppesen system:
  - menus
  - filtering
  - selecting
  - reporting
- Planning concepts

Looking ahead

- Input to the Crew Pairing problem
- Create a standard solution:

1. Create a standard local plan
2. Build rotations
3. Load rules and parameters
4. Create a sub-plan by adding legs
5. Create trips:
   - manually in Studio
   - automatically with the optimizer
6. Verify the solution

Input and output



Input – timetable

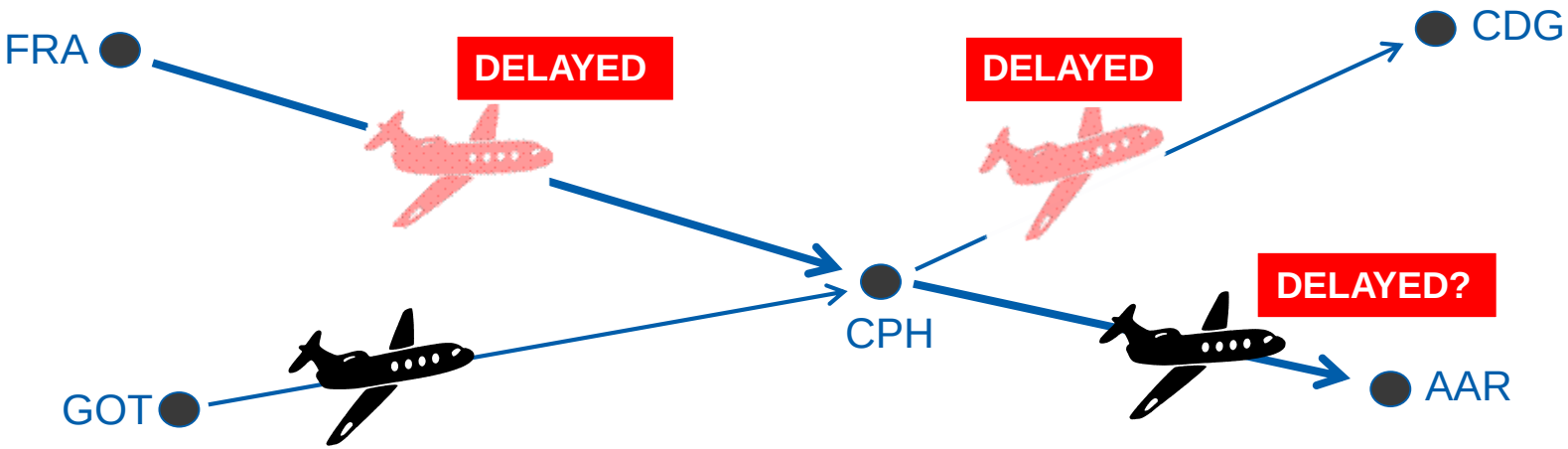
Timetables contain information on all own carrier flights in a given time period. Timetables are provided in SSIM* format

```
3 SK 23820801J01NOV9900XXX00123456  KRS09300930+0100  OSL10151015+0100  735CSMBGHVTLK XX DD
M 00388406

3 SK 23830801J18MAY9906AUG99123456  OSL10051005+0200  KRS10551055+0200  735CSMBGHVTLK XX DD
M 00388430

3 SK 23830901J09AUG9930OCT99123456  OSL10051005+0200  KRS10551055+0200  735CSMBGHVTLK XX DD
M 00388433

3 SK 23840801J18APR9924OCT99      7 KRS11301130+0200  OSL12151215+0200  735CSMBGHVTLK XX DD
M 00388460

3 SK 23840901J18MAY9906AUG99123456  KRS11301130+0200  OSL12151215+0200  735CSMBGHVTLK XX DD
M 00388463

3 SK 23841001J09AUG9930OCT99123456  KRS11301130+0200  OSL12151215+0200  735CSMBGHVTLK XX DD
M 00388466

3 SK 23850101J25MAR9926MAR9912345   OSL14051405+0100  KRS14551455+0100  735CSMBGHVTLK XX DD
M 00388472
```

*SSIM = Standard Schedules Information Manual

Input – rotations

In short-haul and medium-haul, you wish to avoid too many aircraft changes in trips. Every aircraft change potentially transfers delays to other flights.



Rotations may be defined in the SSIM file, delivered in a separate file or built heuristically by the system (we can guess) if not known at the time of planning.

Input – OAG file

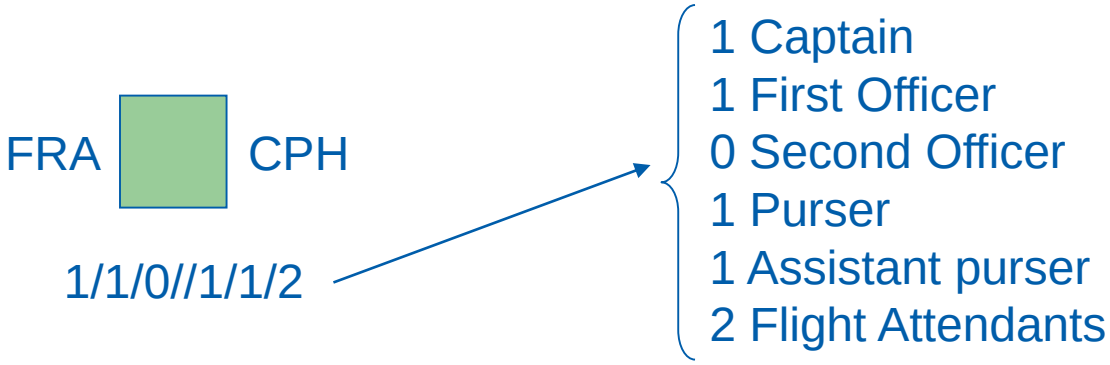
If deadheading with other carriers is needed, timetable information for these carriers is normally provided in an OAG file.

OAG files are provided in SSIM format.

* OAG = Official Airline Guide

Input – crew need

For every flight leg, a crew need must be defined. The crew need describes how many crew members of each crew category the leg requires.



The crew need is defined in the rule set. The crew need may be fixed or dependent on the duty in which it occurs (augmented crew)

Input – legal and union rules

Legal and union rules are fixed by law or agreed with the union. They typically limit duty time, ensure minimum rest between duties etc.

Examples:

- duty time must not exceed 12:00 hours
- block time in duty must not exceed 10:00 hours

Legal and union rules are implemented in Rave and can be turned on/off by the planner.

Input – quality and stability aspects

Quality and stability aspects aim at increasing solution stability or crew quality of life.

Examples:

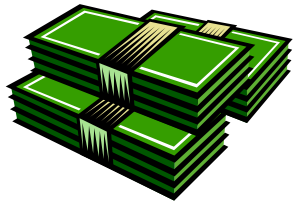
- more than one aircraft change per duty should be avoided
- more than two early consecutive departures should be avoided

Quality and stability aspects are implemented in Rave as rules or penalties or both.
(the planner can turn them on/off or change their value)

Input – cost definitions

Includes real costs and penalties (quality/stability aspects).

Real costs include:
duty-day costs, hotel costs, per diem,
deadhead costs, overtime costs etc.



Penalties include:
short connection penalty,
excessive aircraft change penalty etc.

Costs and penalties are implemented in Rave and their value can be modified by the planner.

Input – rostering aspects

The created trips must be possible to use for rostering.

Examples of demands to consider:

- the number of available crew members per base
- the qualifications of the crew members (per base)
- not too many long trips

Rostering aspects are normally implemented as vertical rules (global constraints) in Rave.

Output – trips

According to the steps in the planning process, trips are:

Daily (all trips begin in one day)

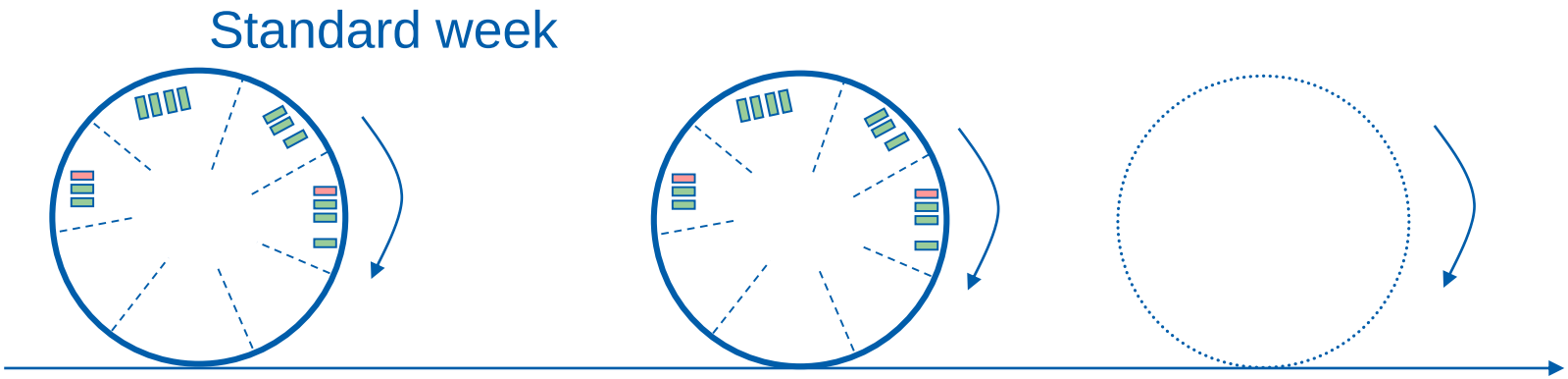
Weekly (all trips begin in one week)

Dated (all trips begin within a given time span, for example a month)

Planning process

Create standard solution

Create dated solution

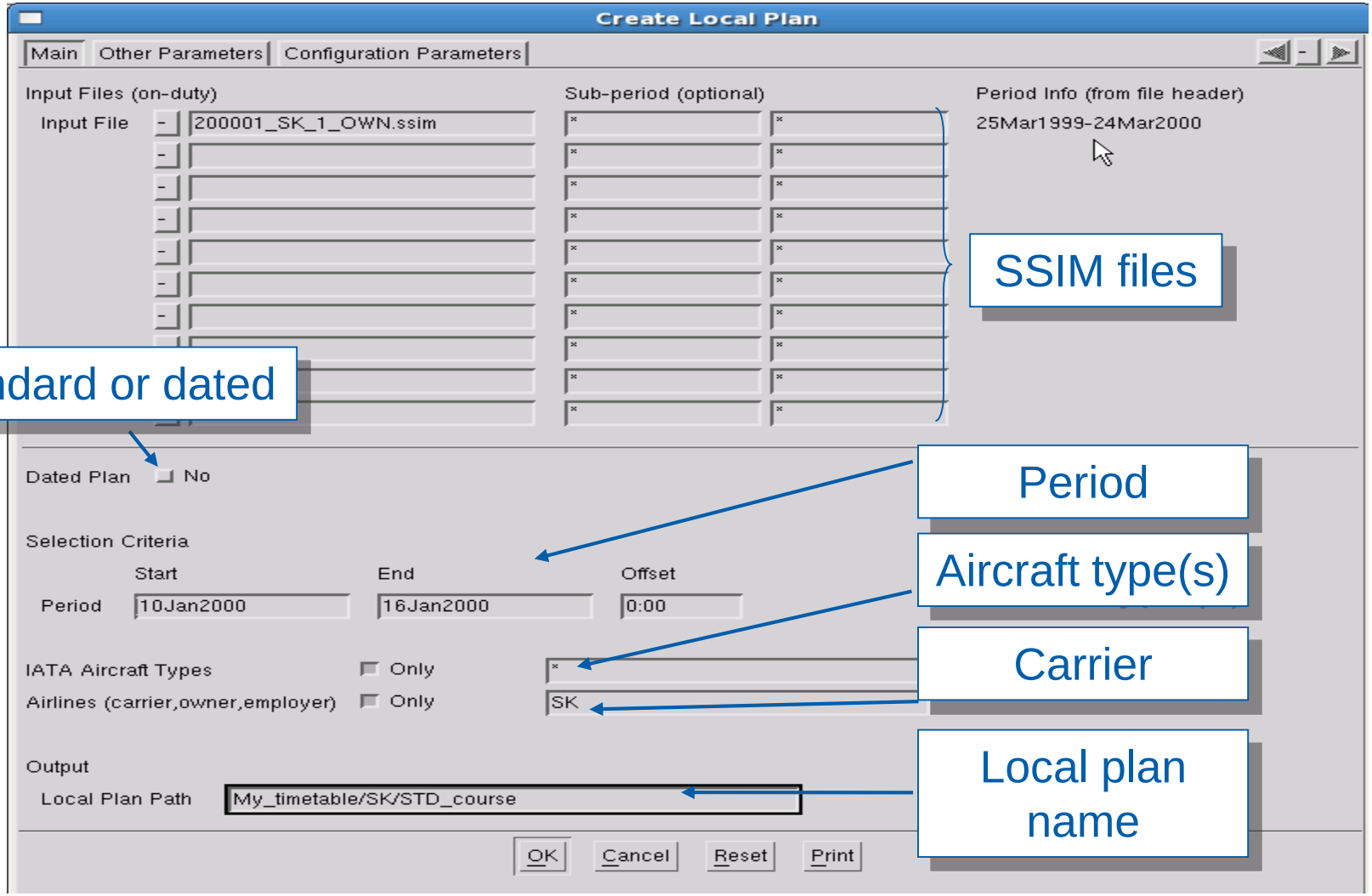


Create a standard solution

- Create standard local plan (from SSIM file)
- Build rotations
- Load rules and parameters
- Create sub-plan by adding legs
- Prepare Optimization job
  (set parameters/apply hard locks)
- Run the Crew Pairing optimizer
- Verify solution

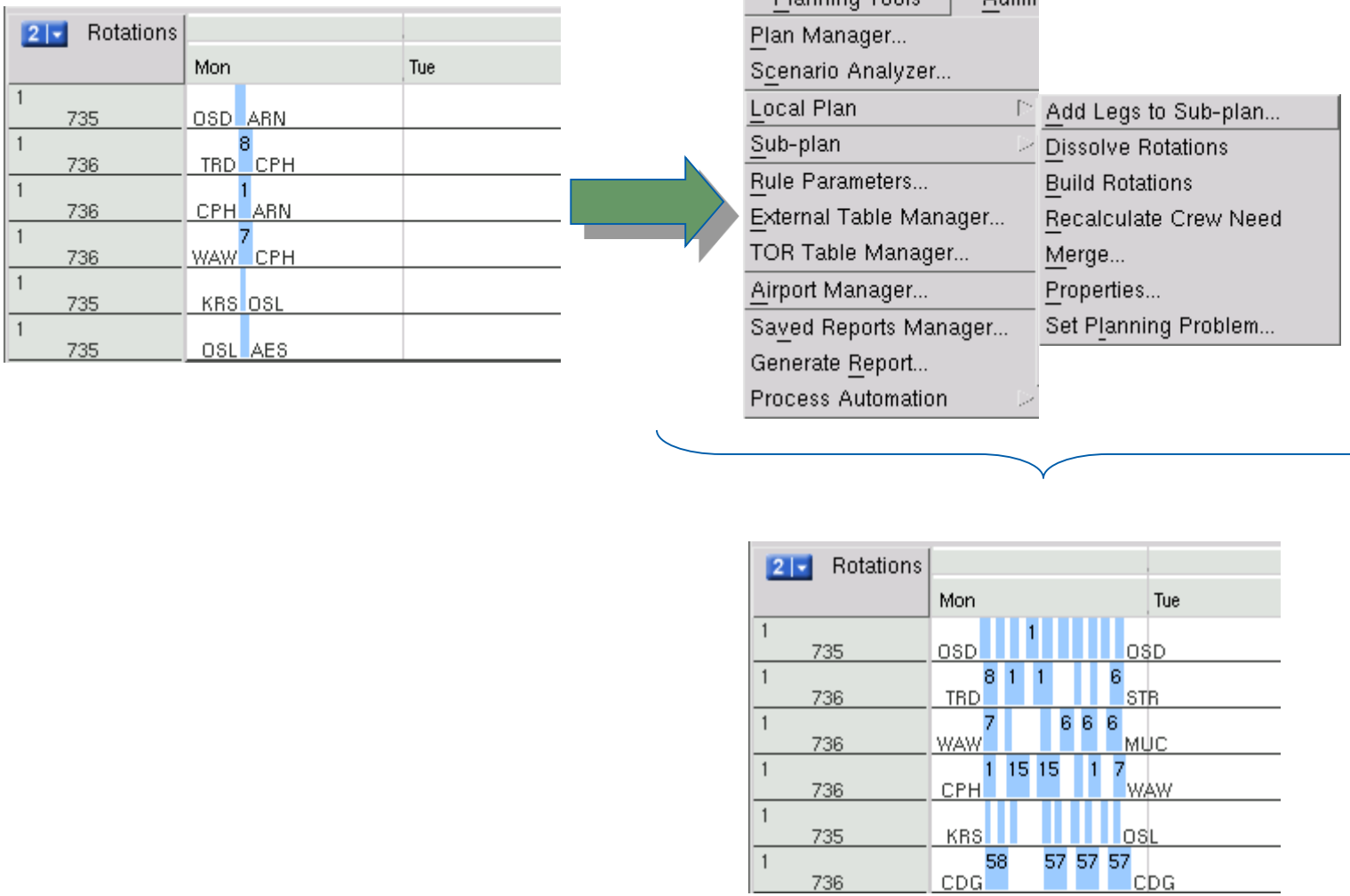
Create a standard solution

Create a standard local plan



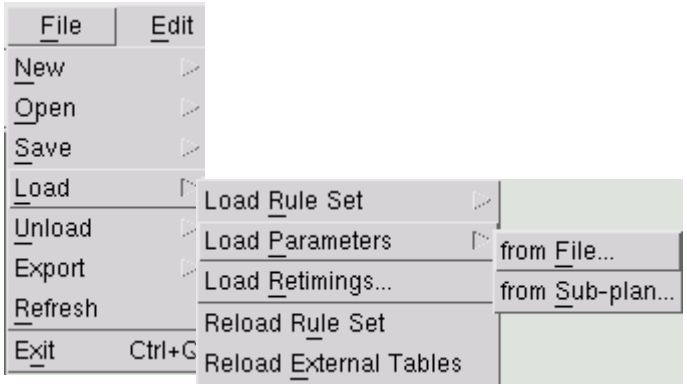
Create a standard solution

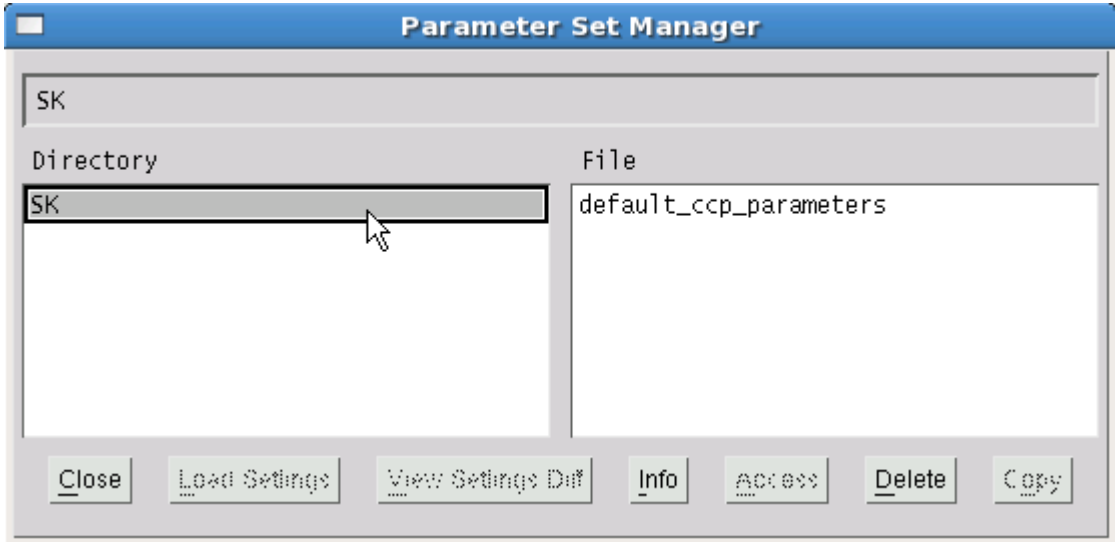
Build rotations



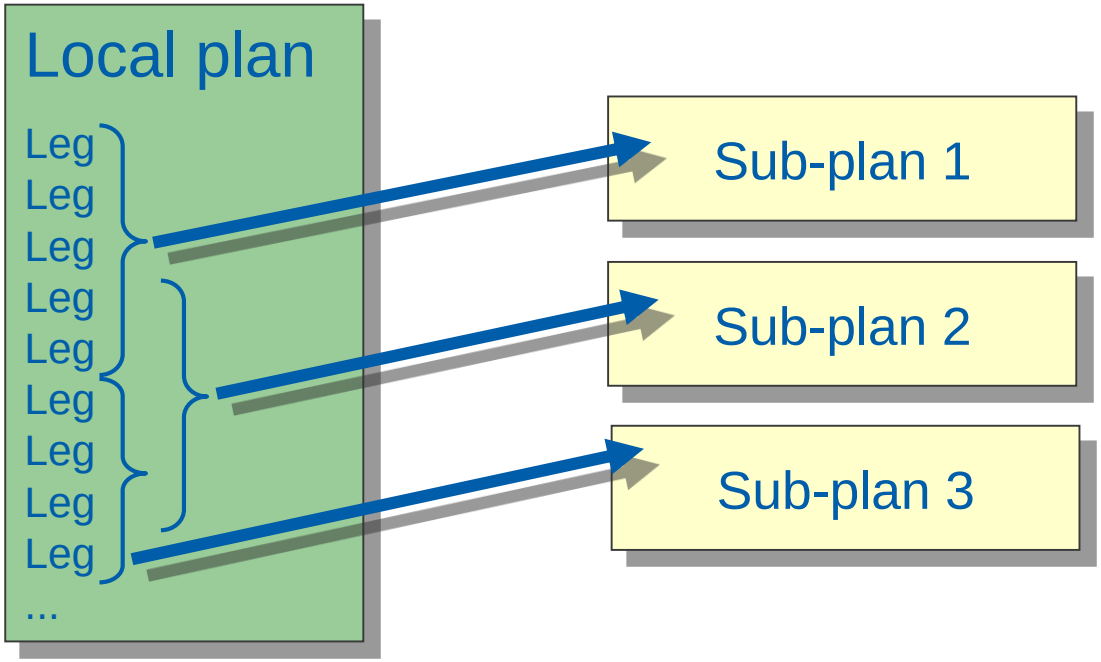
Create a standard solution

Load parameters





Add legs from local plan



- Legs are added from local plan
- One leg can be part of several sub-plans
- Content of sub-plan reflects the planning problem

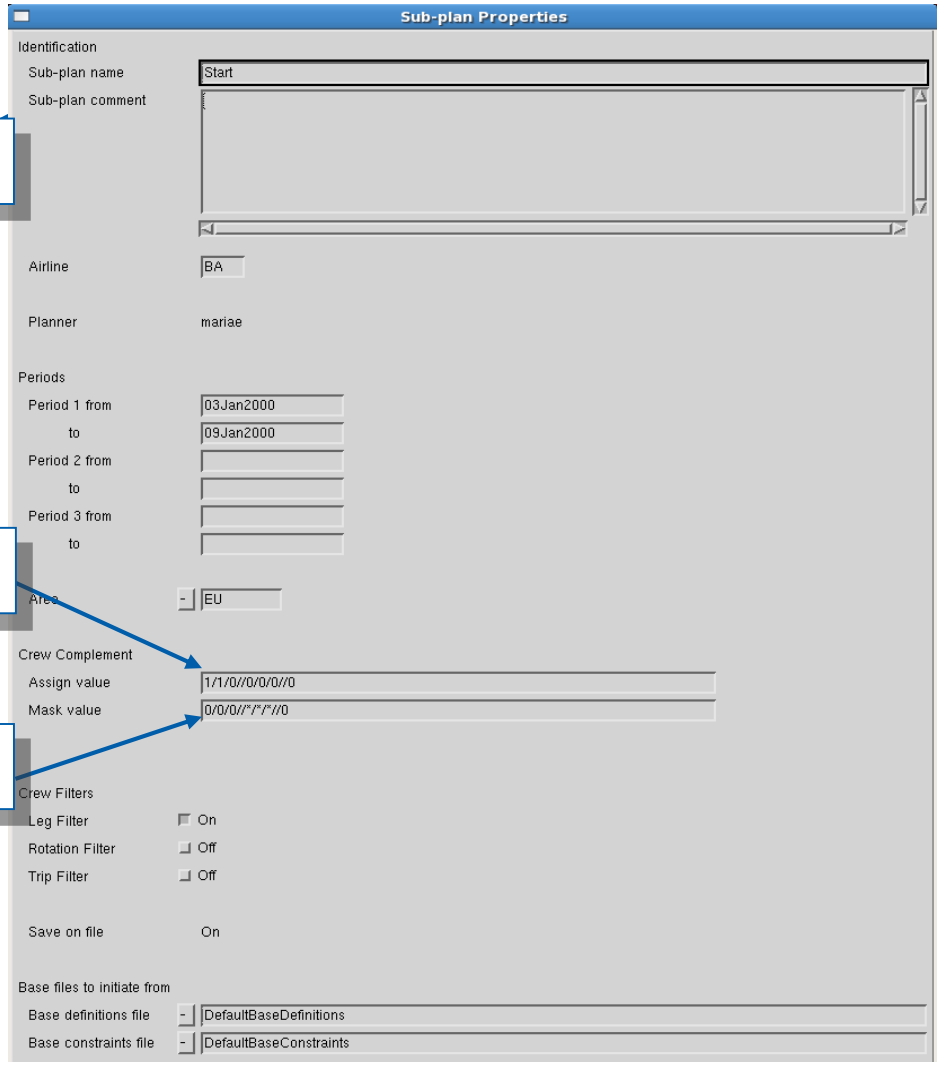
Create a standard solution

Create sub-plan

Sub-plan name

Default assign value

Mask value

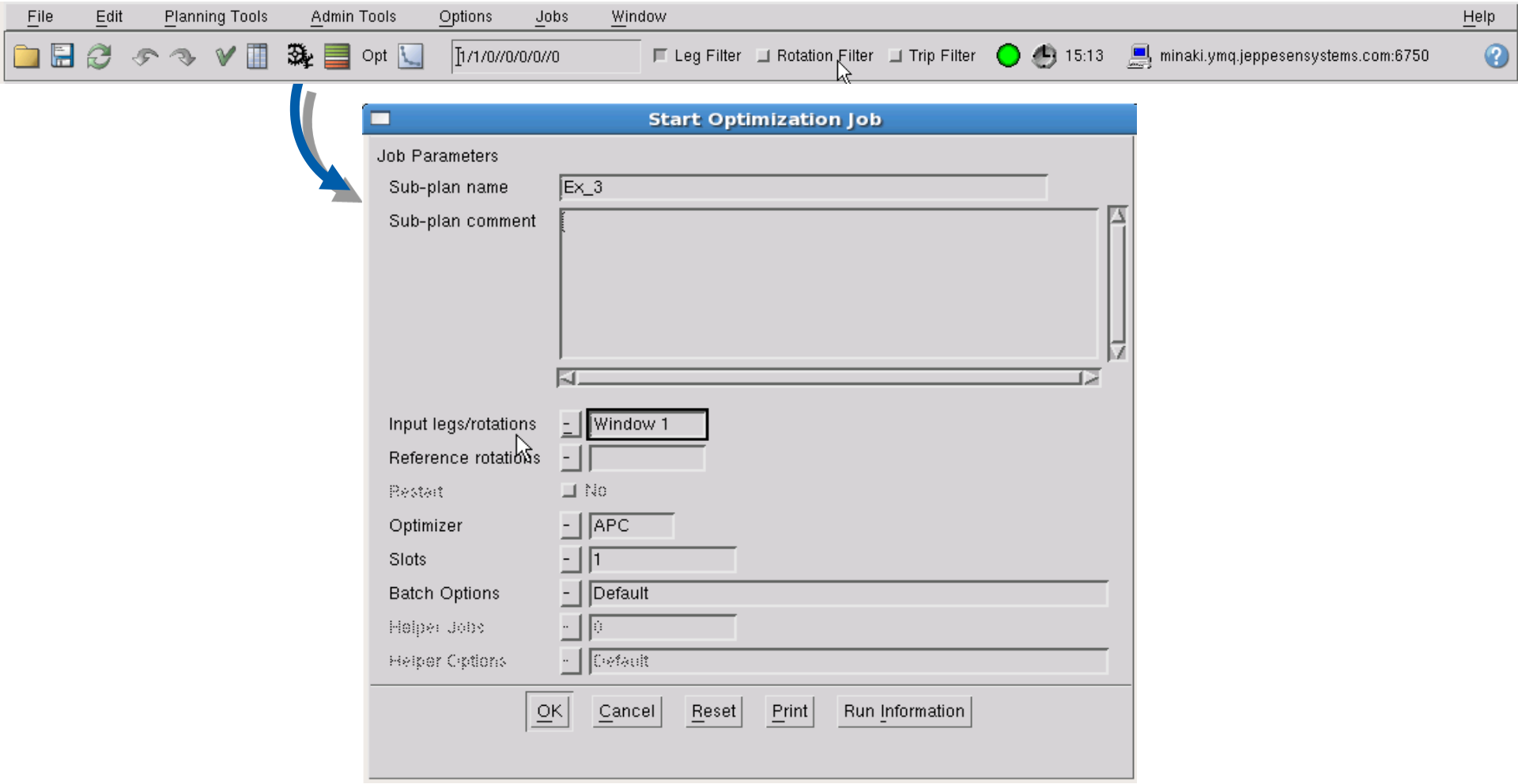


Using the optimizer

1. Start a job.
2. Monitor progress with
   - Batch Job Manager
   - Plan Manager.

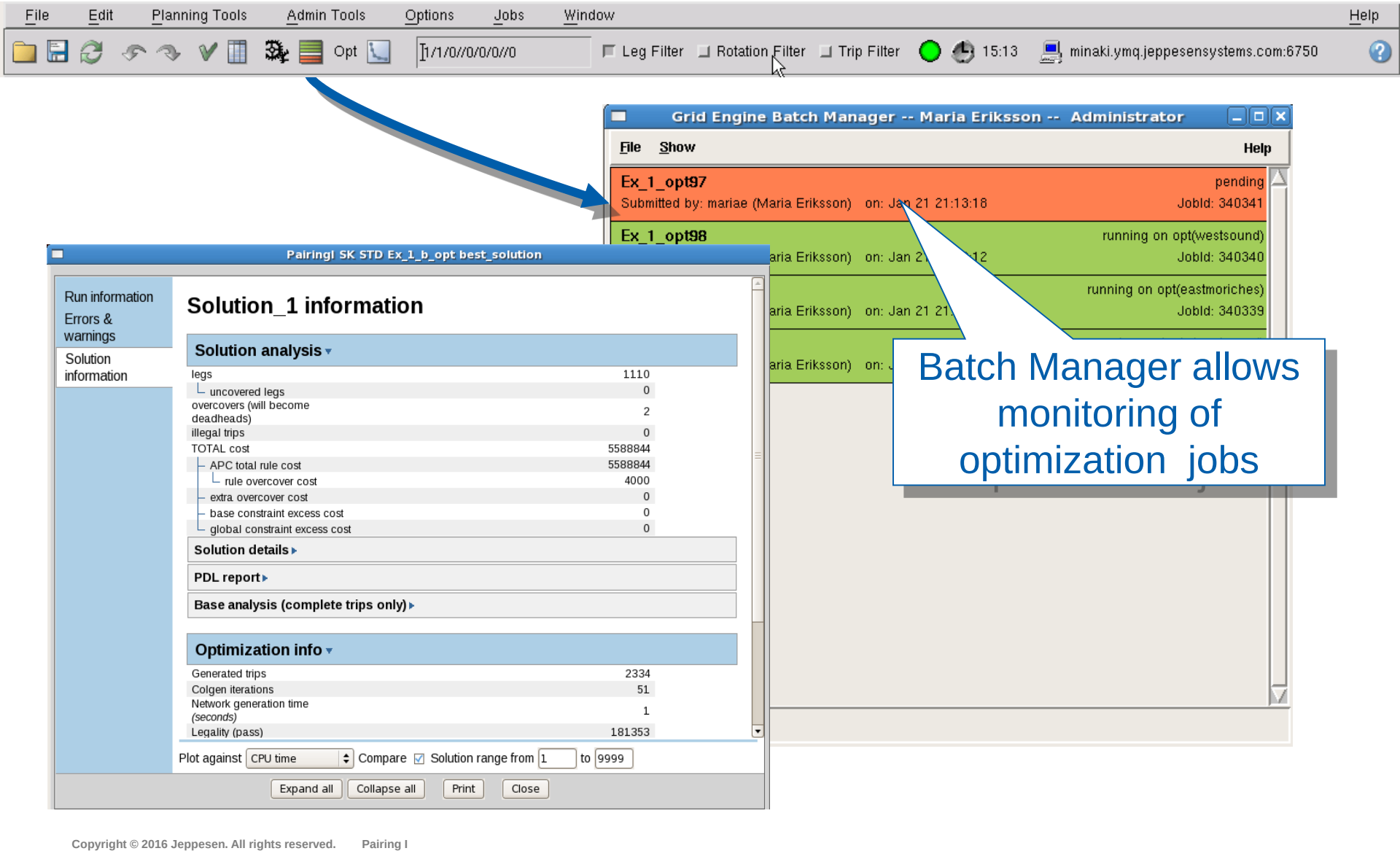
Create a standard solution

Using the optimizer



Create a standard solution

Using the optimizer

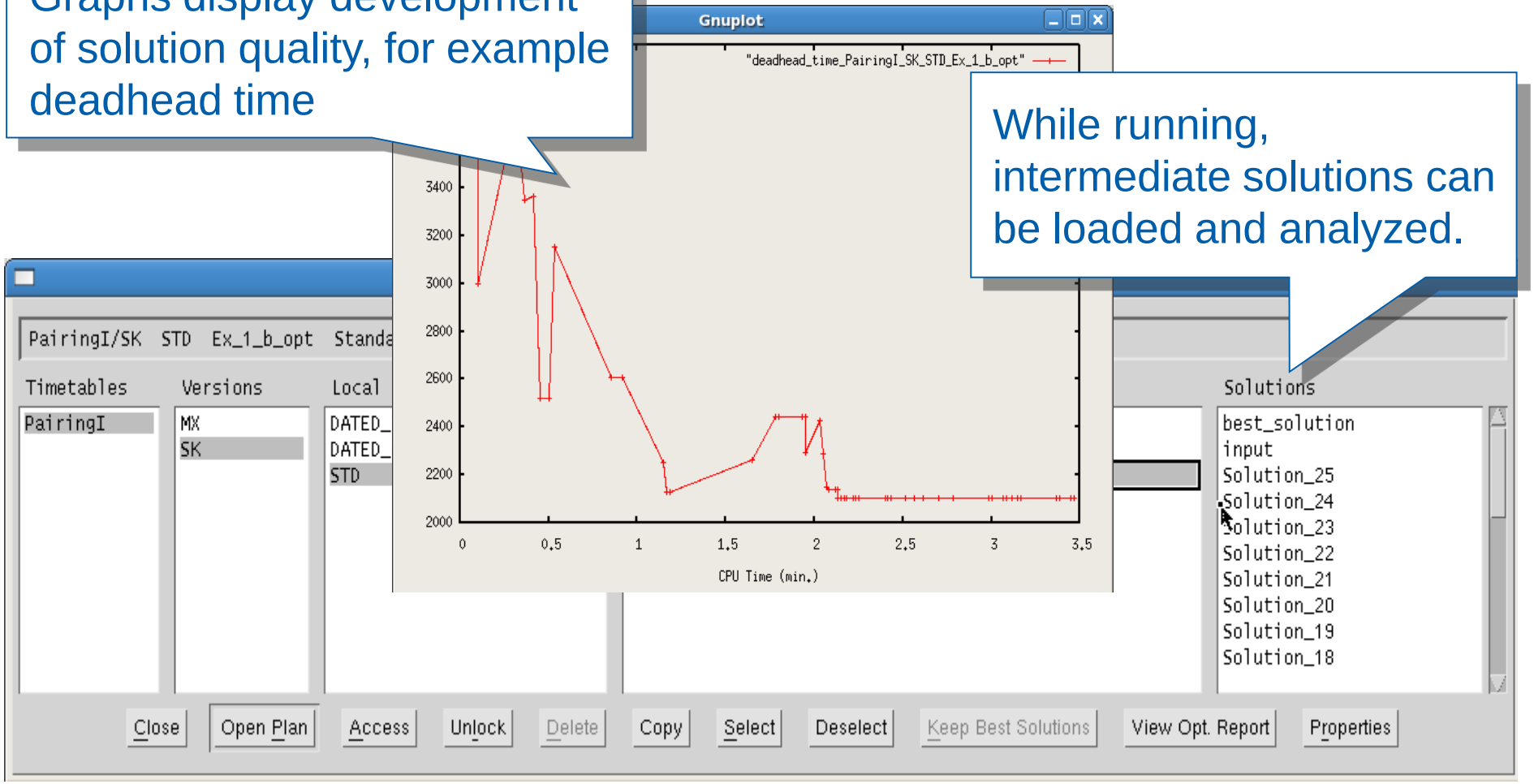


Create a standard solution

Using the optimizer

Graphs display development of solution quality, for example deadhead time

While running, intermediate solutions can be loaded and analyzed.



Create a standard solution

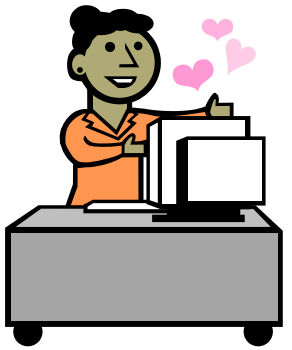
- Create standard local plan (from SSIM file)
- Build rotations
- Load rules and parameters
- Create sub-plan by adding legs
- Prepare optimization job
  (set parameters/apply hard locks)
- Run Optimization
- Verify solution

Exercise 3

Create standard solution

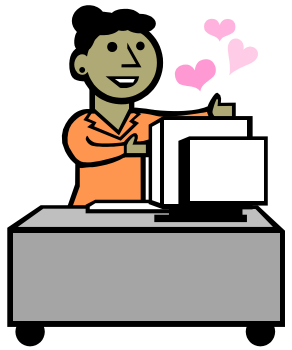
Purpose

To learn all steps involved in creating a standard solution



Exercise 3 – summary

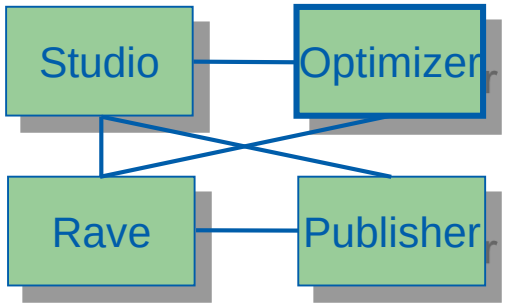
Summary of exercise 3



Optimizer

The Crew Pairing Optimizer solves Crew Pairing problems to optimality (or near optimality) using different optimal or heuristic methods designed for various problems:

- daily, weekly and dated problems
- short-haul and long-haul problems
- other carrier deadheading
- deadhead optimization.



All trips comply with rules as defined in Rave.

Optimization – definition

Merriam Webster:

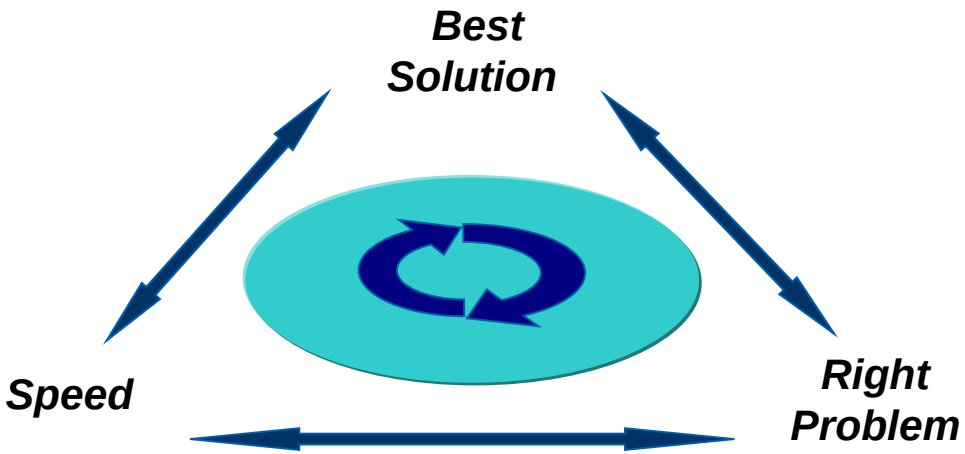
op-ti-mi-za-tion n (1857) : an act, process or methodology of making something (as a design, system, or decision) as fully perfect, functional, or effective as possible; specif: the mathematical procedures (as finding the maximum of a function) involved in this.

Optimization – definition

Our view

Optimization = Finding the best possible solution to the right problem as fast as possible.

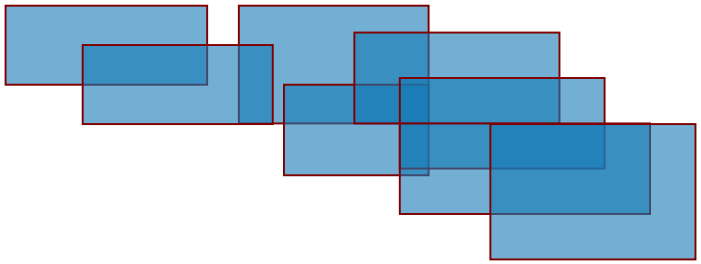
The three dimensions

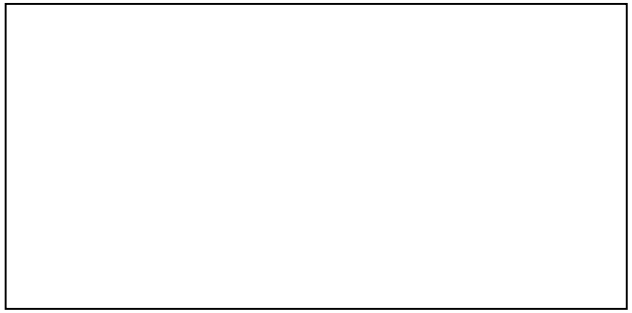


Optimization – examples

Packing

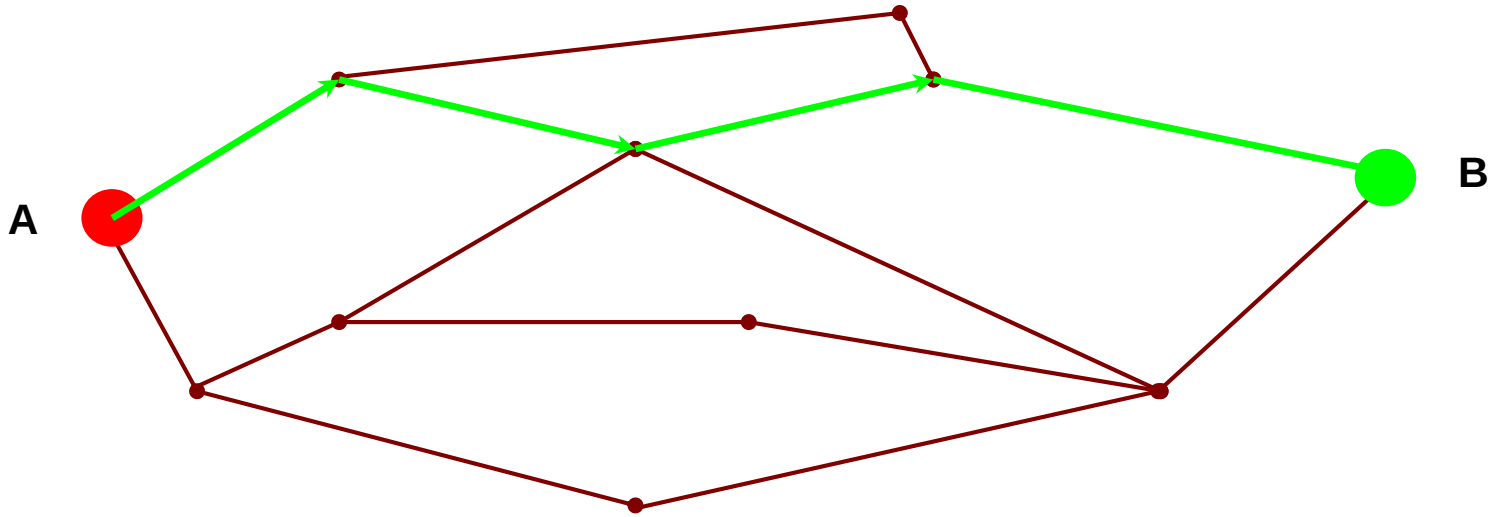
- Fit as much as you can in your bag





Optimization – examples

Shortest-Path

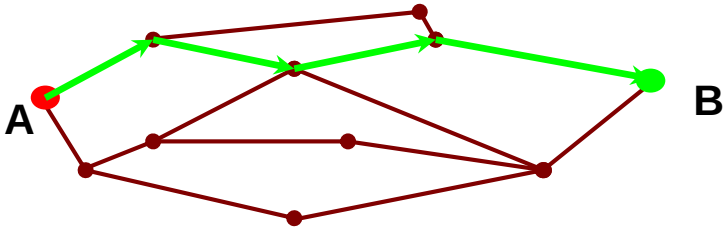


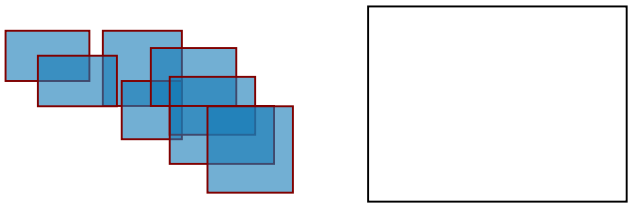
Find the shortest route in the network taking you from **A** to **B**.

Optimization – Objective function

What is the best solution?

- Quickest?
- Shortest?
- Cheapest?
- Nicest?





- Weight?
- Volume?
- Value?

Optimization

Two main steps:

- **Model** the problem:
  Define the objective function and the rules/constraints
- **Solve** the problem:
  Apply the best algorithm(s) to the problem

For step 2 we want to implement a **computer program** doing it automatically.

Algorithm = description in mechanical steps of how to solve a problem.

Optimization

A simple way of finding the optimal solution to the pairing problem:

generate all possible solutions (sets of trips)

select the best solution.

Is this possible?

The beggar made a deal with the King

Doubling rice grains

1, 2, 4…

8, 16, 32, 64, 128

A week later, he brought a teaspoon for 128 grains

256, 512, 1024, 2048, 4096, 8192, 16384, 32768

In two weeks it was a non-negligible amount of half a kilo

65536, 131072, 262144, 524288, 1048576,

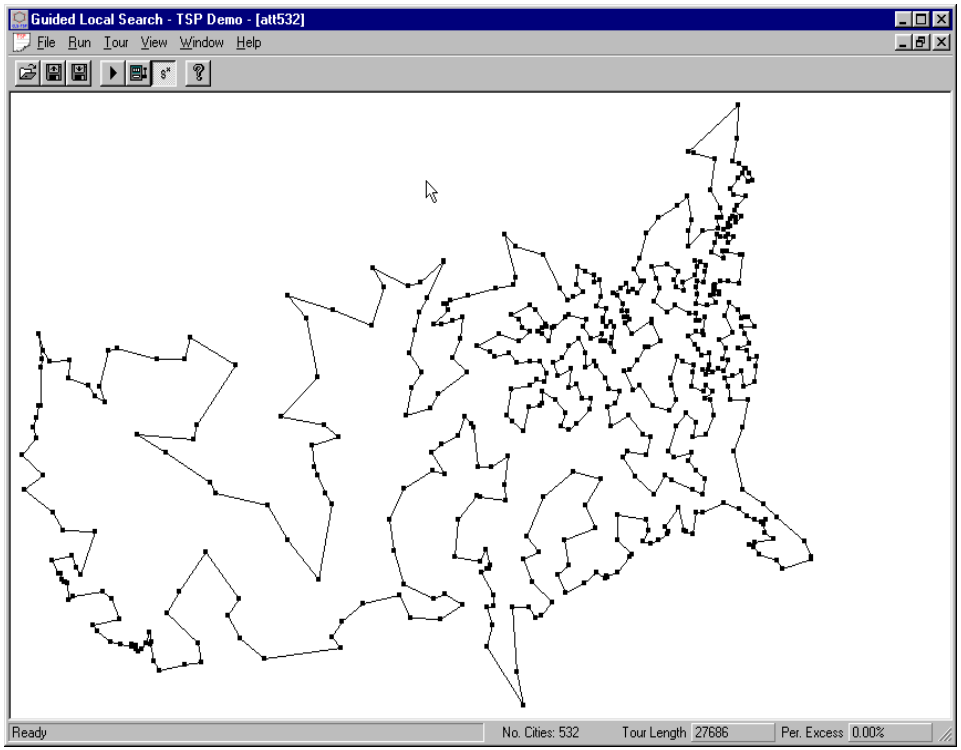
…, 1073741824, 2147483648

At the end of the month it had grown to a whopping 35 ton

A few days later the king had to declare bankruptcy.

Optimization – Travelling salesman problem

Given a set of cities, find the shortest way to visit each city exactly once.



Combinatorial Explosion

for example a 41 city problem has

41! =334,525,266,131,638,071,081,700,620,534,407,516,651,520,000,000

possible **cyclic tours**.

Optimization – TSP – Sweden

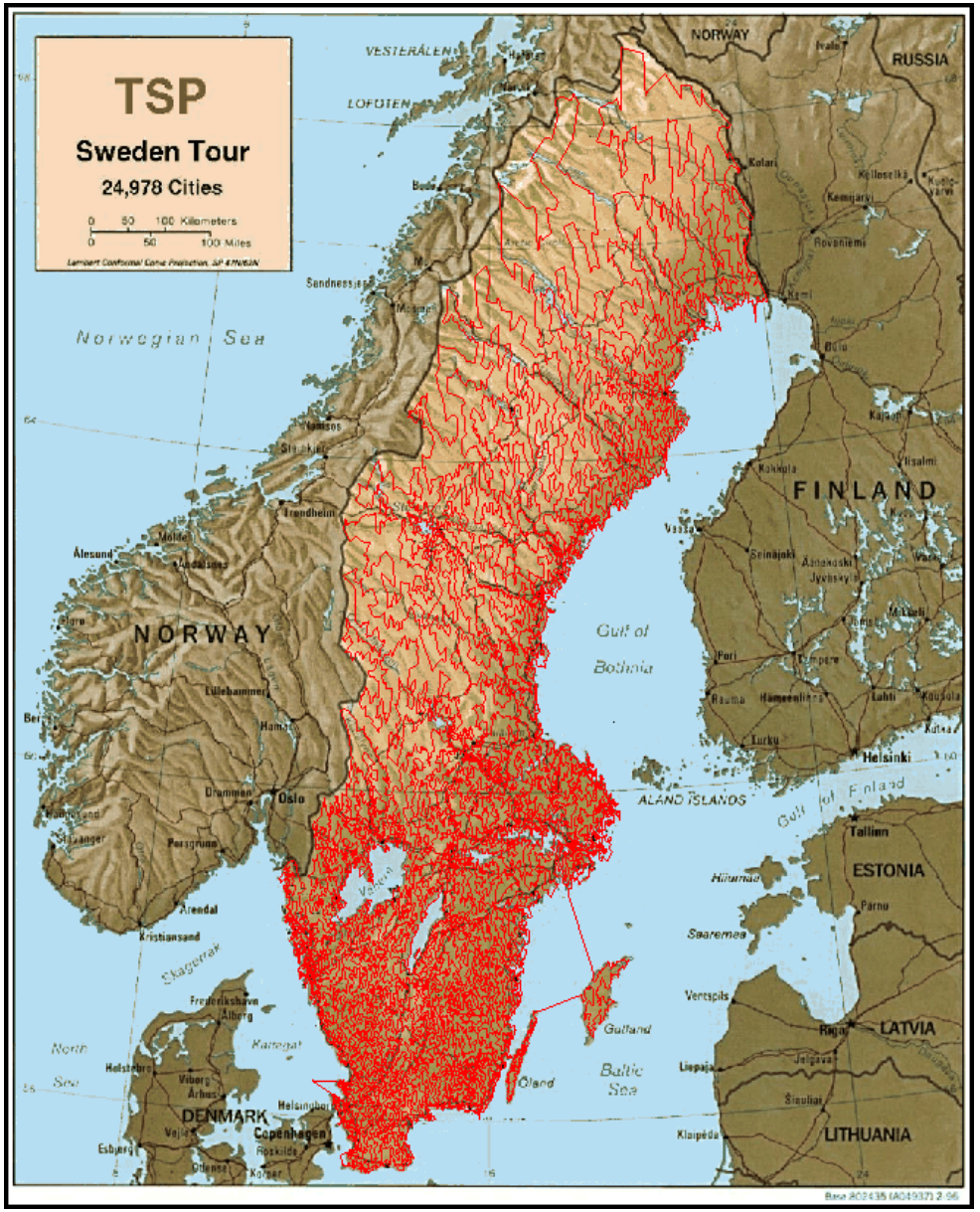
24,978 cities

Solved by Applegate, Bixby, Chvatal, Cook, Helsgaun in May 2004

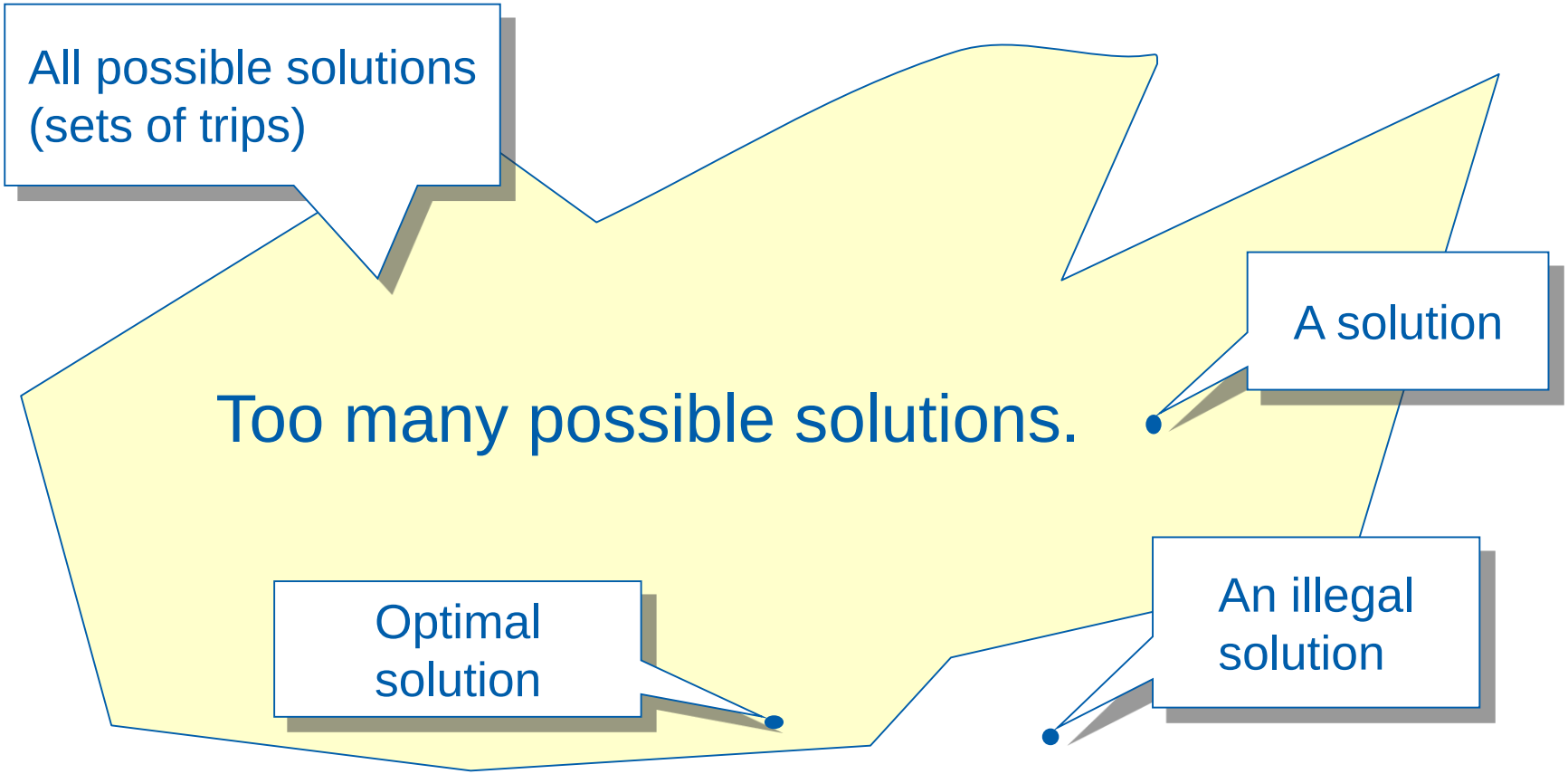
Length about 72,500 km

Applied latest optimization algorithms

Total CPU time 91.9 years (on one CPU)



Optimization

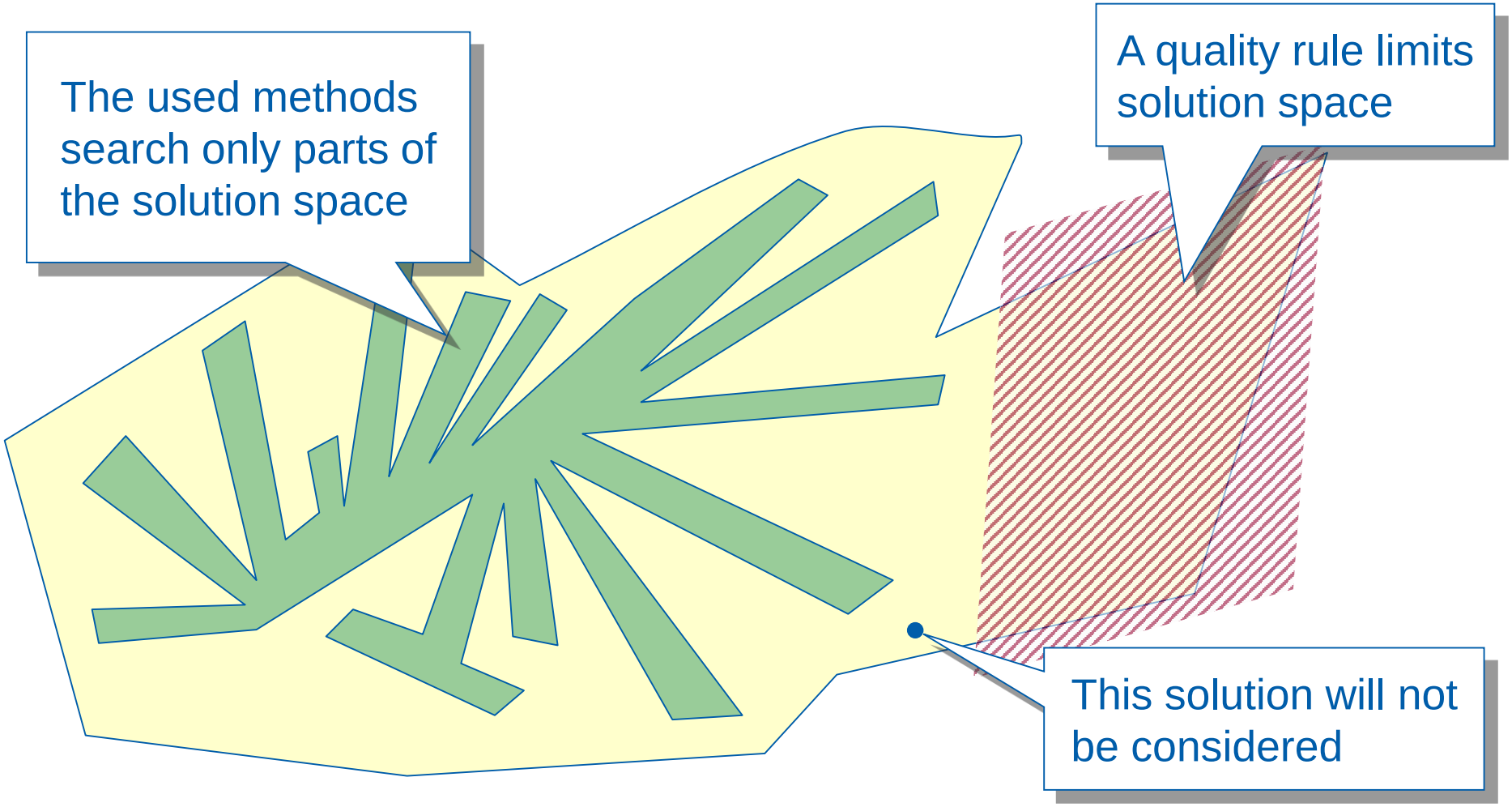


Optimization

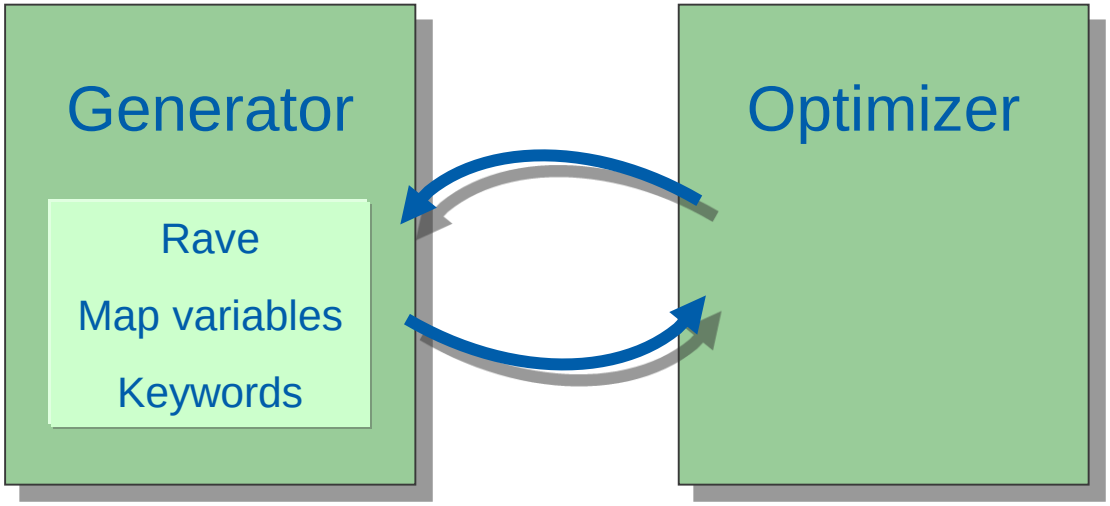
We need to limit the solution space and/or only search in parts of it:

- quality rules limit the solution space
- optimization methods limit the solution space search

Optimization



Optimization



Generator generates legal subsets

Optimizer finds the cheapest combination

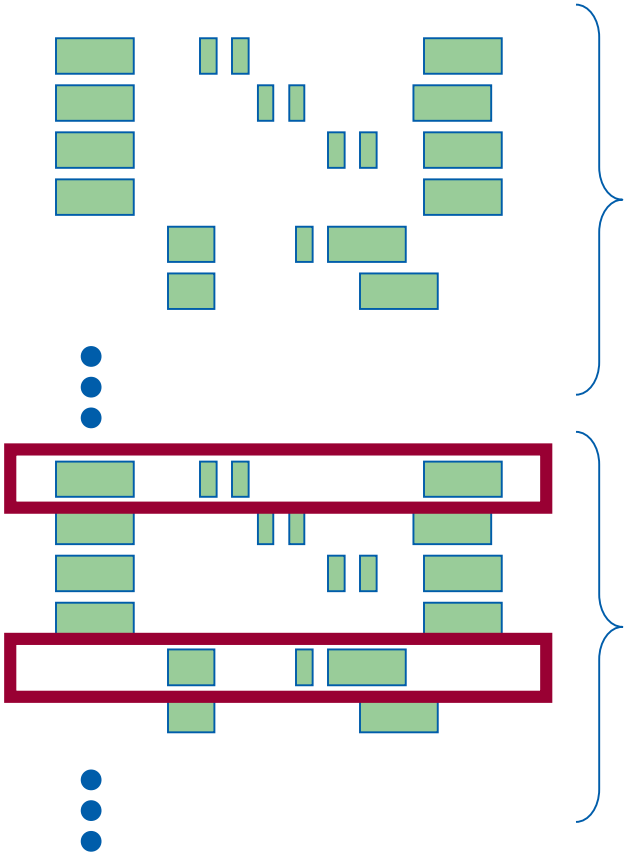
Optimization

The optimizer selects a subset of trips which:

- does not overcover any trips
- has the lowest total penalty

Penalty definitions typically include a high penalty for leaving legs uncovered.

Optimization



1000’s of trips

Optimal set of trips

Generation

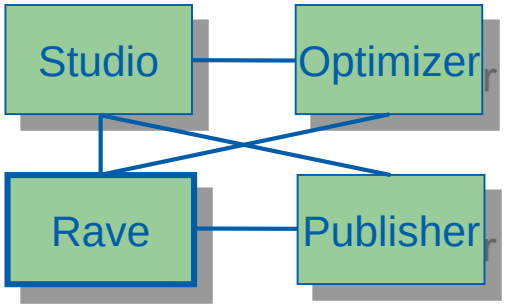
Optimization

Optimization

Keep in mind that the objective function depends on penalty parameter settings. Hereby the planner controls solutions produced by the optimizer.

Rules and objectives

Rules and objectives are defined in Rave



Objectives

Cover all flights

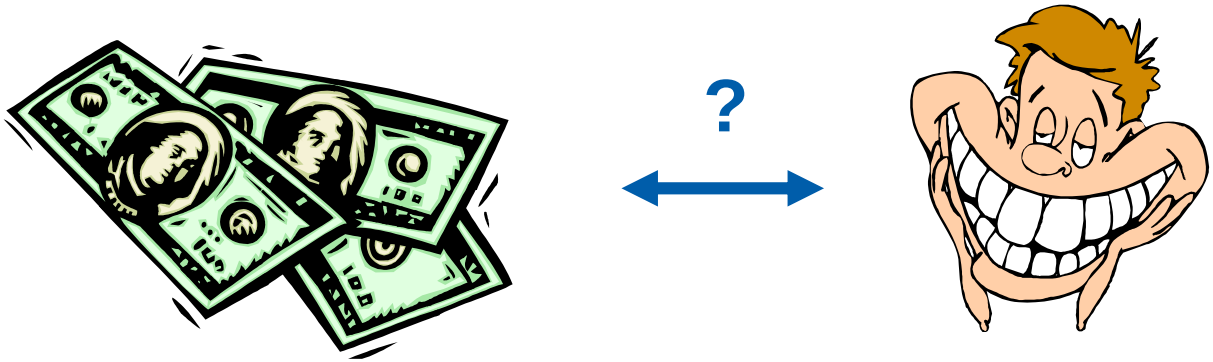
Create only trips complying with regulations, union rules

Minimize total cost

- Maximize productivity
- Maximize crew quality of life
- Maximize stability
- ...

Objectives

Some objectives are potential conflicts, for example costs and quality of life.



Planners usually solve these problems manually by experience, rules of thumb etc. However, this experience often differs among planners. It may be difficult to generalize.

Rules

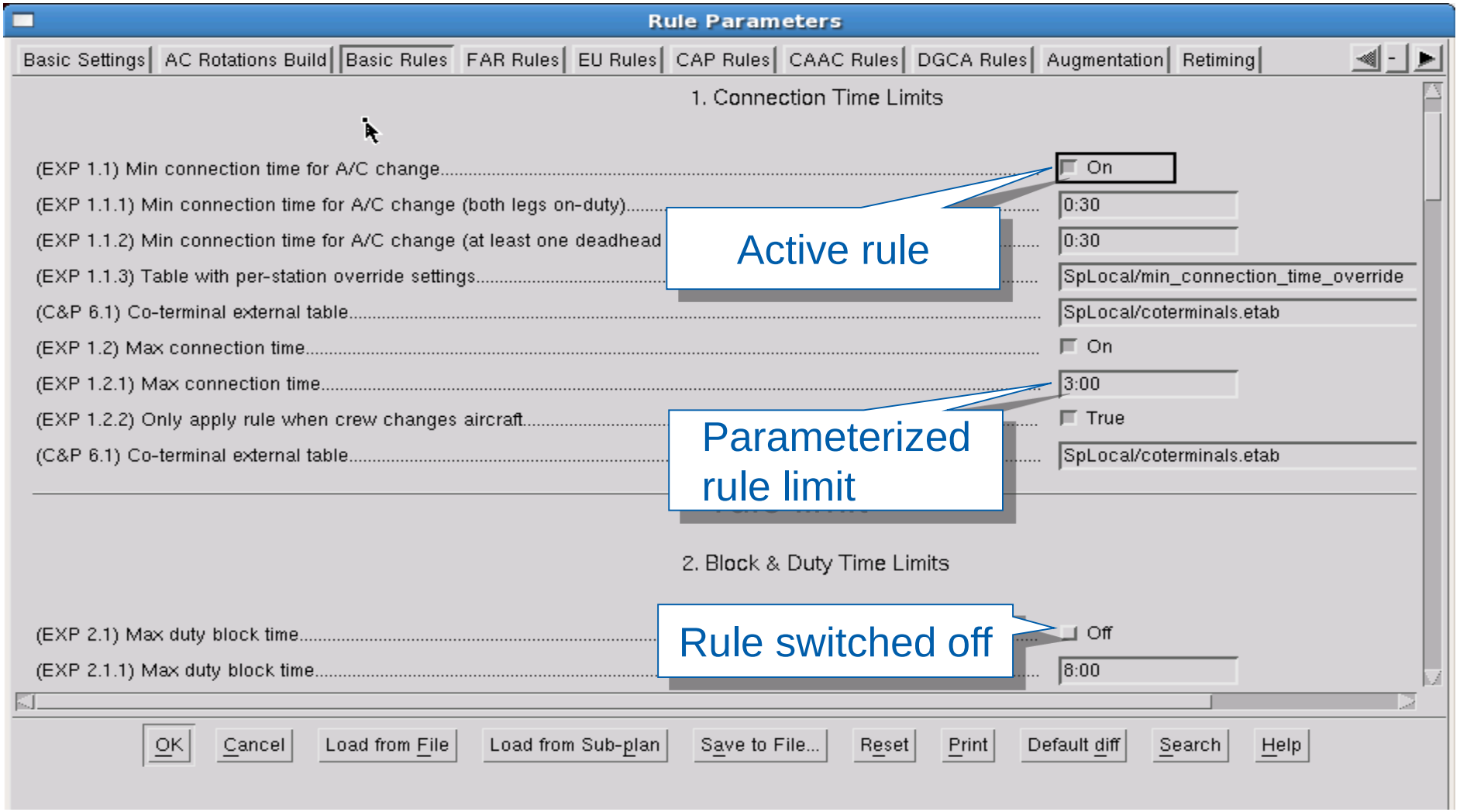
Trips must be created according to rules.

There are different kind of rules:

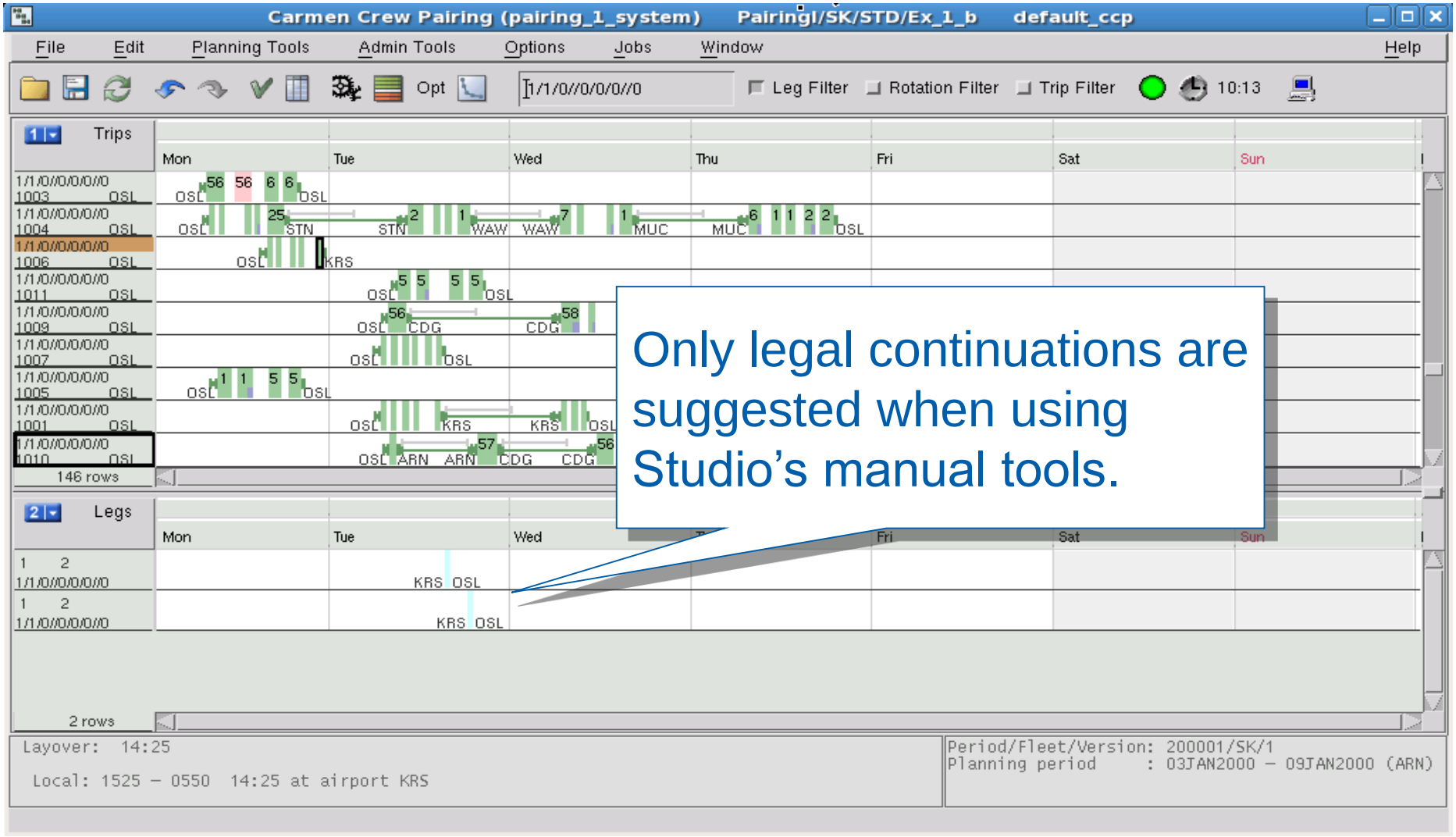
- legal
- union
- quality

In the system, rules are **always** enforced strictly.
Trips not complying with rules are called **illegal trips**.

Rules



Rules



Only legal continuations are suggested when using Studio's manual tools.

Quality

Quality are characteristics of a trip that the vast majority of crew members consider as good.

Examples

“Not more than two early starts in a trip”

“No repetitive duties in a trip”

“No long connection times”

Stability

Stability means that the trip is not too sensitive to delays and other changes. Good stability simplifies maintenance of the trips.

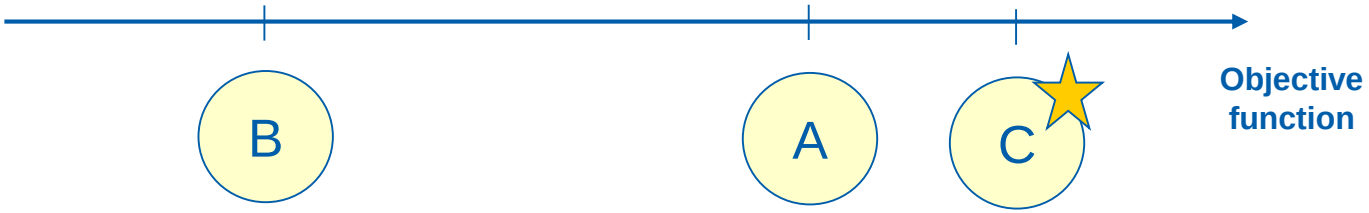
Examples

“Connection time between legs not too short”

“No aircraft change within a duty”

Objective function

When solving problems automatically, all Crew Pairing solutions should be uniquely ordered. This is the only way to identify the best solution.



Solution A is better than solution B, but solution C is better than both and therefore the best.

Objective function

The objective function:

- guides the optimizer when generating solutions
- is modelled in Rave
- returns a value for every solution.

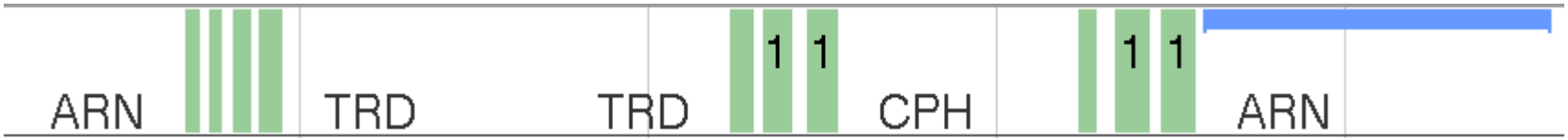
Objective function

The objective function contains two parts:

- real costs (e.g. hotel costs)
- penalties or costs for poor quality/stability
  (for example penalties for unpopular trips)

The **best solution** is uniquely identified as the one with **lowest value** of the objective function.

Objective function – example



Given a certain objective function, the value of the objective function (often called **total cost**) for the trip above is:

Real costs	18,400 Jeppesen$
Penalties	4,210 Jeppesen$
Total costs	22,610 Jeppesen$

Quantifying quality

Defining penalties (and hereby quantifying quality/stability) is always difficult, but often a very rewarding process.

It forces planning departments to:

- prioritize quality issues
- decide how much quality is worth

Defining a Scenario

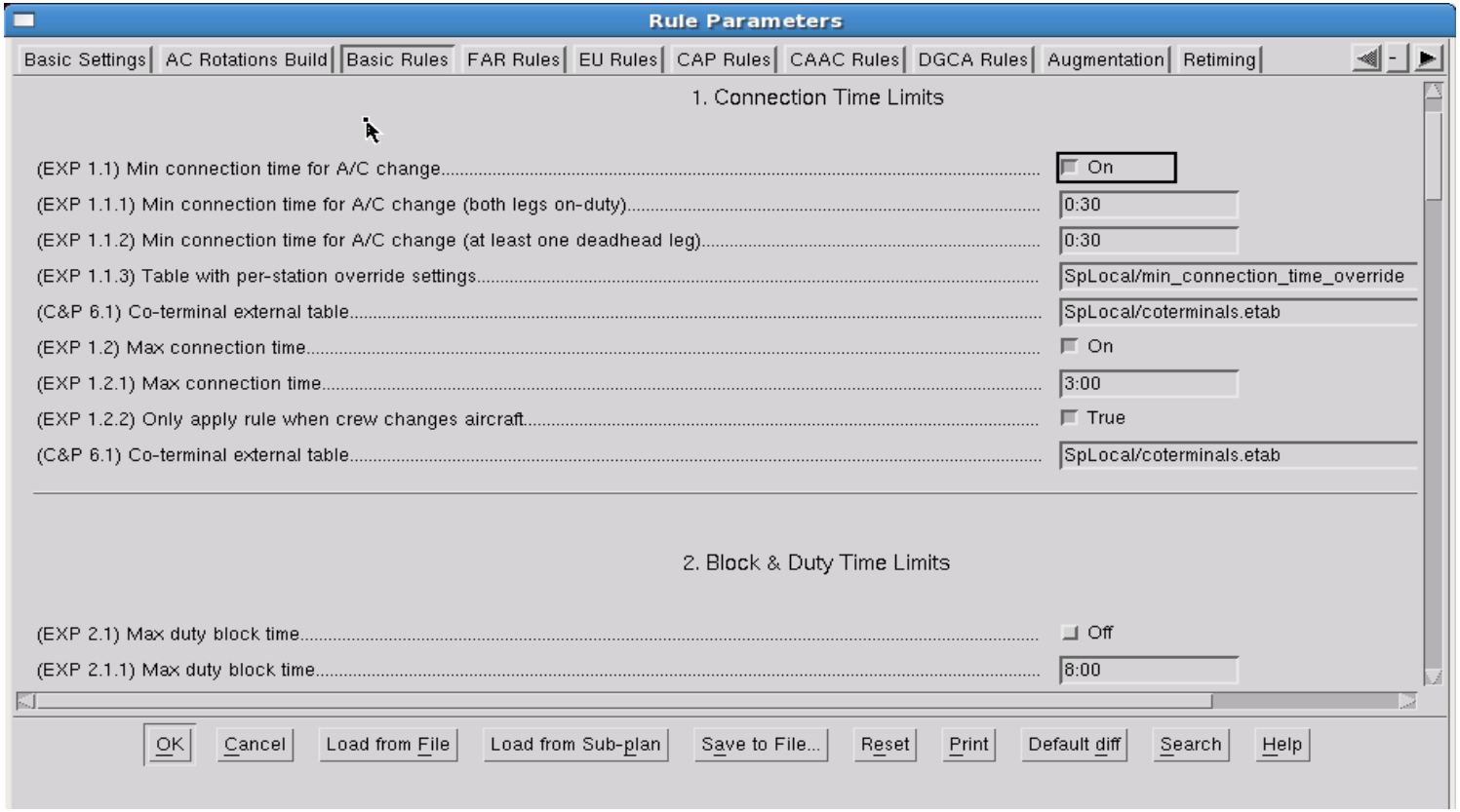
A Scenario is defined by a Sub-plan

Within the Sub-plan the planner can change

- Parameter settings
- E-table settings

Defining a Scenario

Defining Parameter Settings



Defining a Scenario

Defining e-table settings

Differences in Sub-plan e-tables

Name:	200001/SK/1:	default SK:	default:	Status:	Edit in Plan	Import to Plan S
ac_attributes.etab		x		OK	Edit	
ac_families.etab		x		OK	Edit	
augmentation.etab			x	OK	Edit	
augmentation_configuration.etab			x	OK	Edit	
base_constraints_credit.etab			x	OK	Edit	
base_constraints_duty_days.etab			x	OK	Edit	
briefing_override.etab			x	OK	Edit	
connection_banks.etab			x	OK	Edit	
coterminals.etab			x	OK	Edit	
credit_costs.etab			x	OK	Edit	
crew_need_ac.etab		x		OK	Edit	
crew_need_flight.etab			x	OK	Edit	
debriefing_override.etab			x	OK	Edit	
duty_day_costs.etab			x	OK	Edit	
exterior_passenger_flow.etab			x	OK	Edit	
ground_transport.etab			x	OK	Edit	
hotel.etab			!	Missing in Sub-plan		Import
illegal_deadhead_flights.etab			x	OK	Edit	

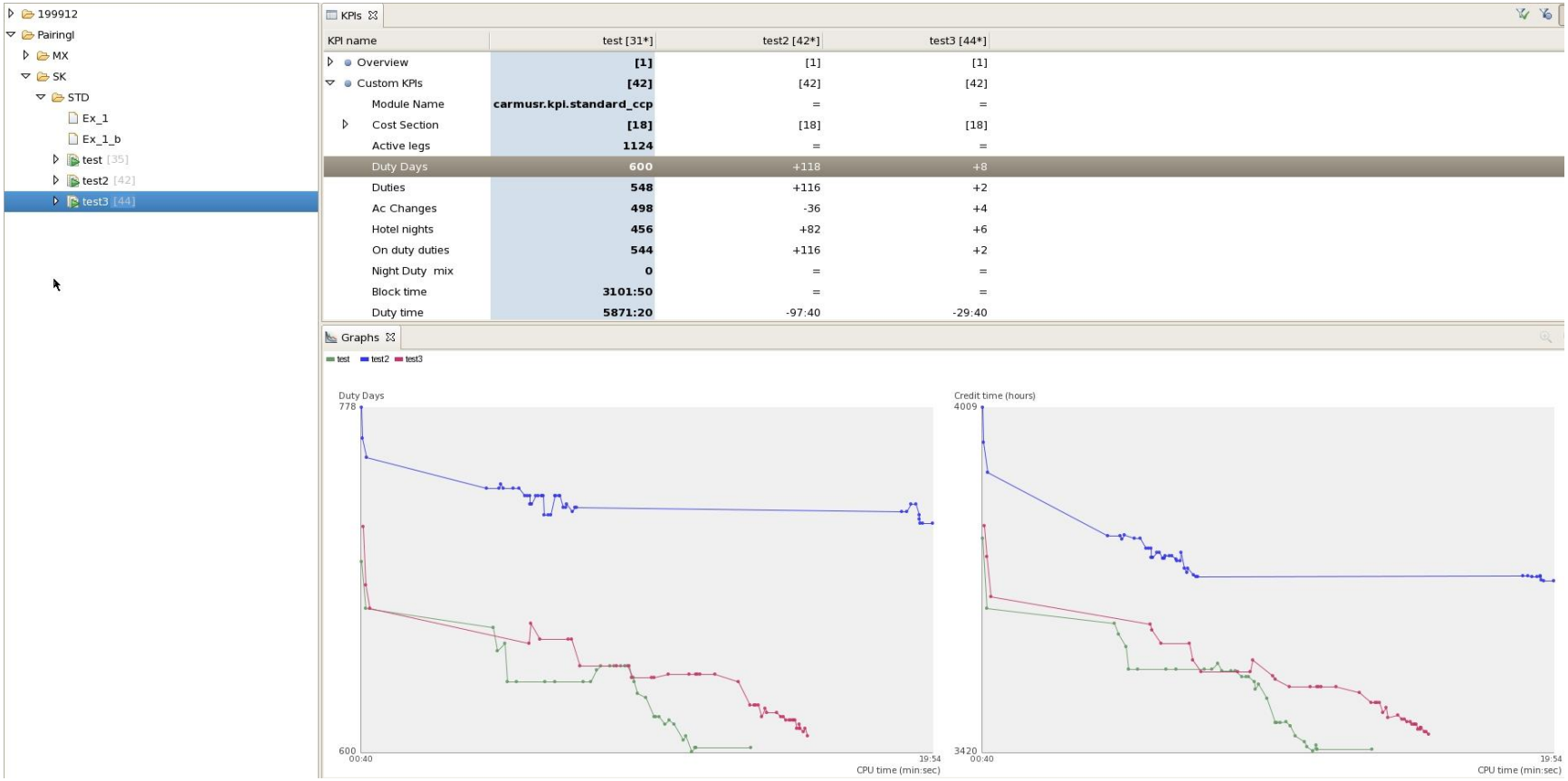
- Sub-plan e-tables in sync with source
- Sub-plan e-table content differs (scenario)
- Sub-plan e-table missing (error)

Analyzing the Solution

Four major ways to analyze the solution:

- Scenario Analyzer
- Optimization Report
- Customer Specific Reports
- Gannt View (manual inspection)

Analyzing the Solution

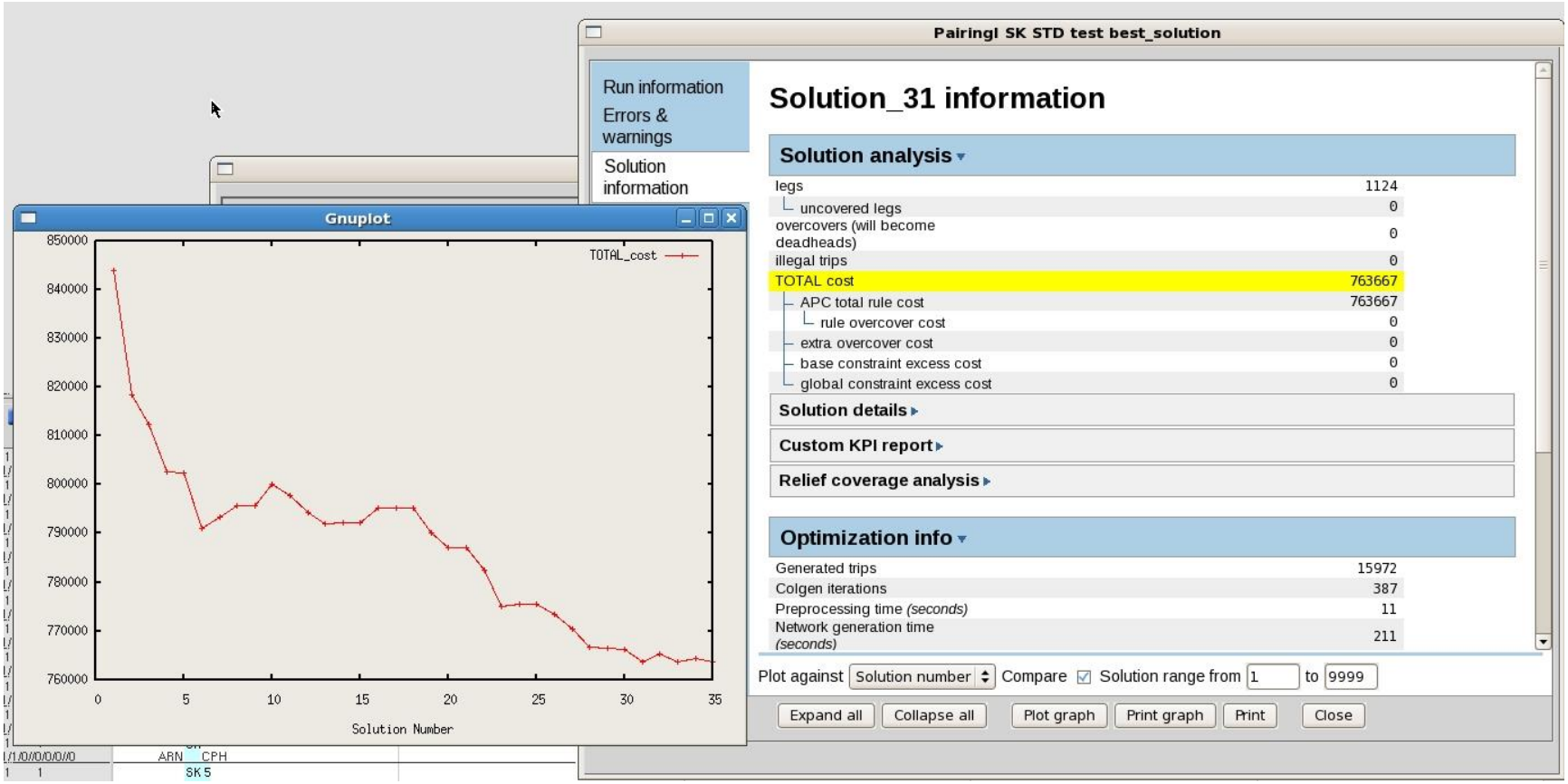


Analyzing the Solution

KPIs from Scenario Analyzer can be exported to for example Excel:

1. Select KPIs to export with Ctrl or Shift
2. Click Crtl+C to copy the selected data
3. Paste the data in e.g. Excel using Paste or Ctrl+V

Analyzing the Solution



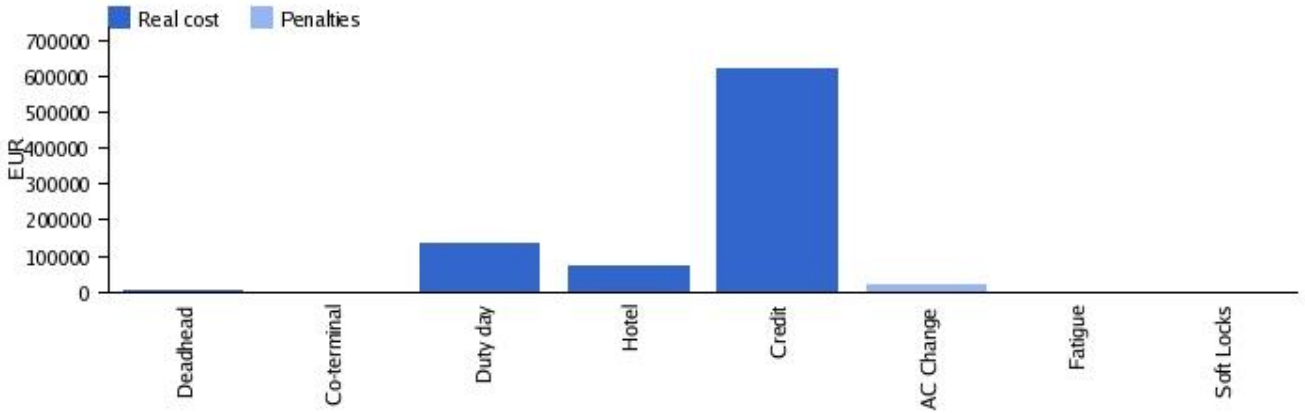
Analyzing the Solution

Trip Statistics

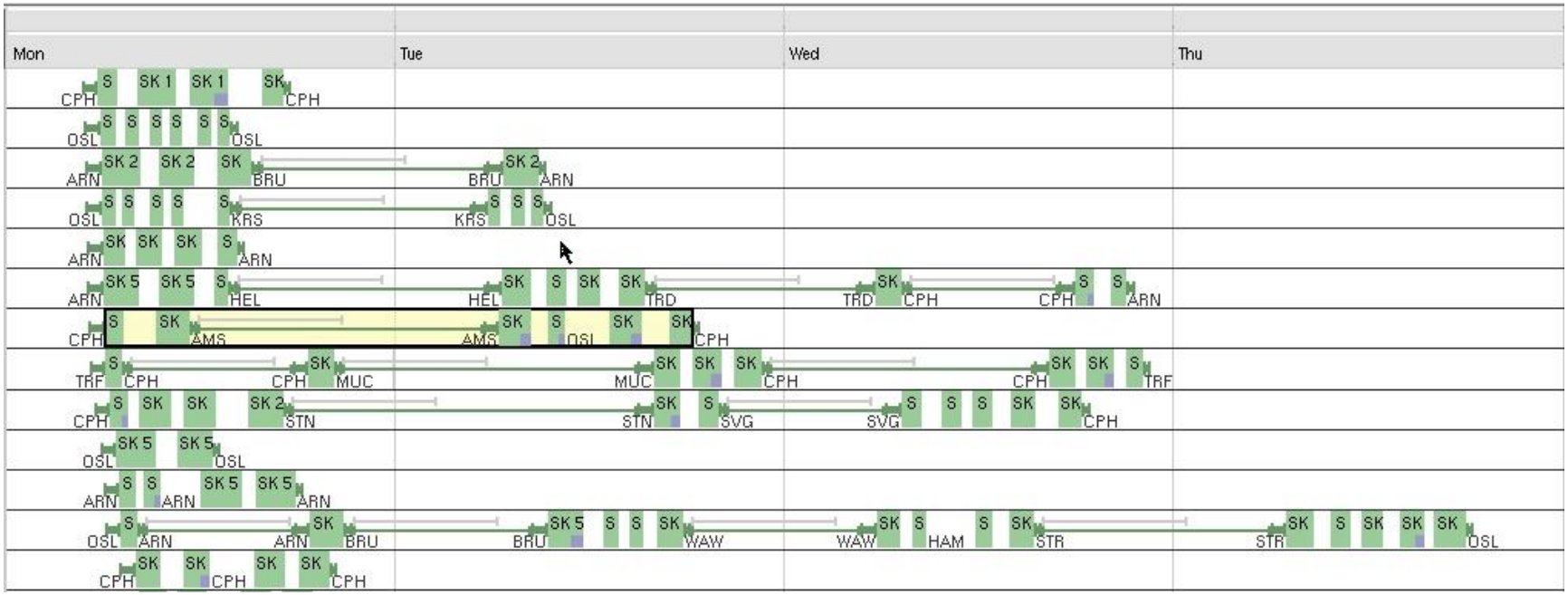
Planning Period: 03JAN2000-09JAN2000
Rule Set Name: default_ccp ()
Plan: PairingI / SK / STD / Ex_1

Cost Table

Element	Cost
Real Cost	**830303**
Deadhead	6166
Co-terminal	0
Duty day	135200
Hotel	70840
Credit	618097
Penalties	**21000**
AC Change	21000
Fatigue	0
Soft Locks	0



Analyzing the Solution

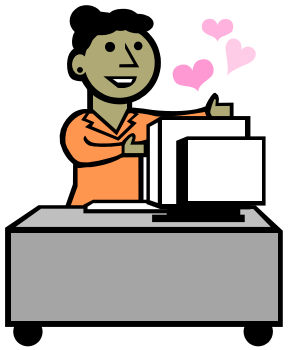


Exercise 4

Influence the solution

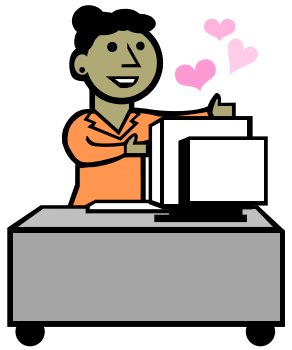
Purpose

Learn to change the priorities of the cost function

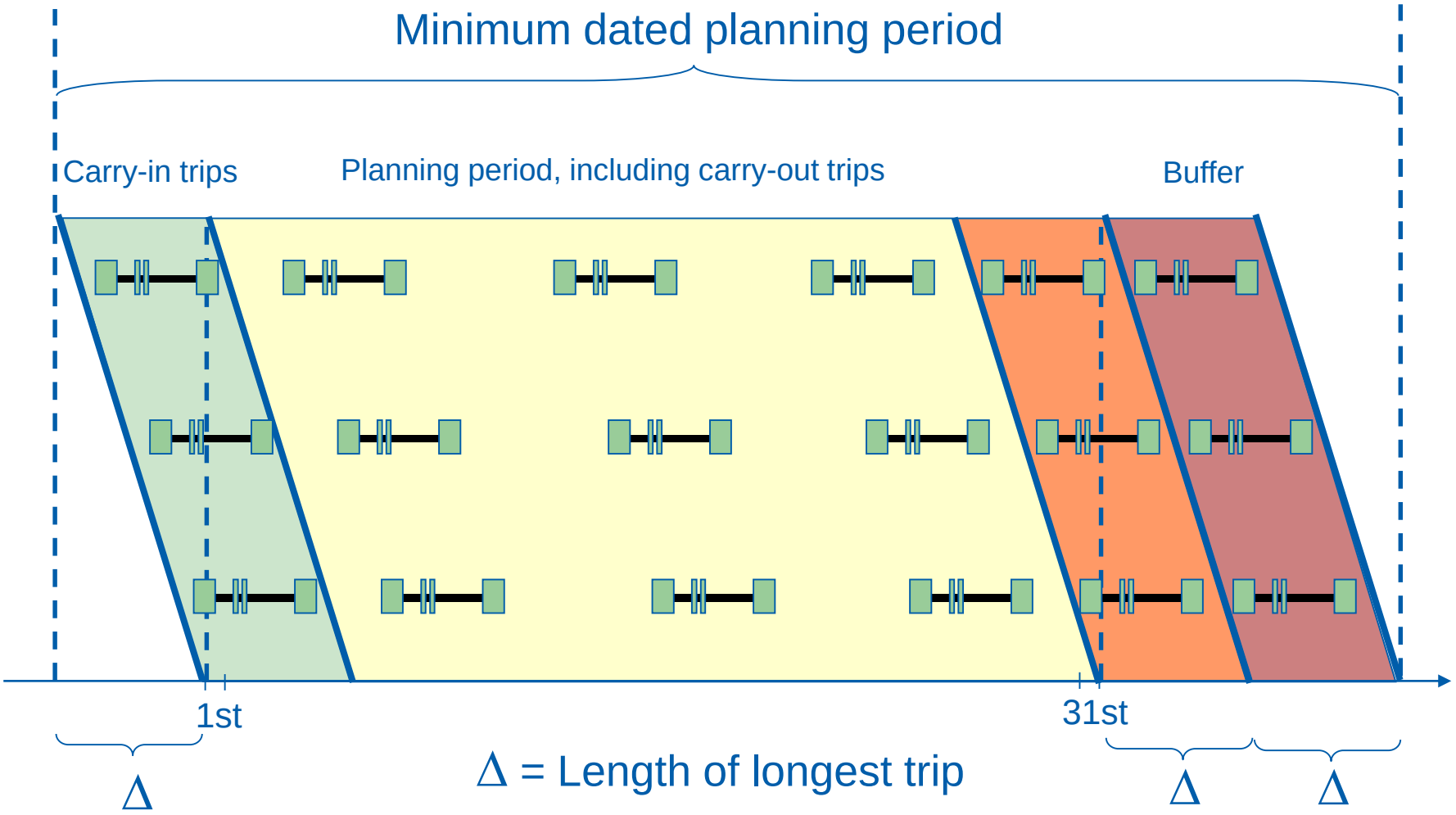


Exercise 4 – summary

Summary of exercise 4

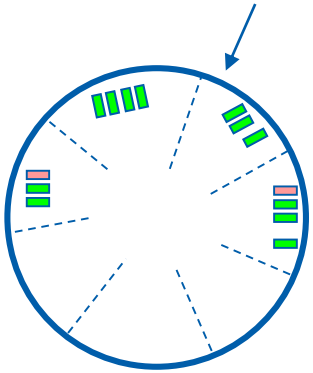


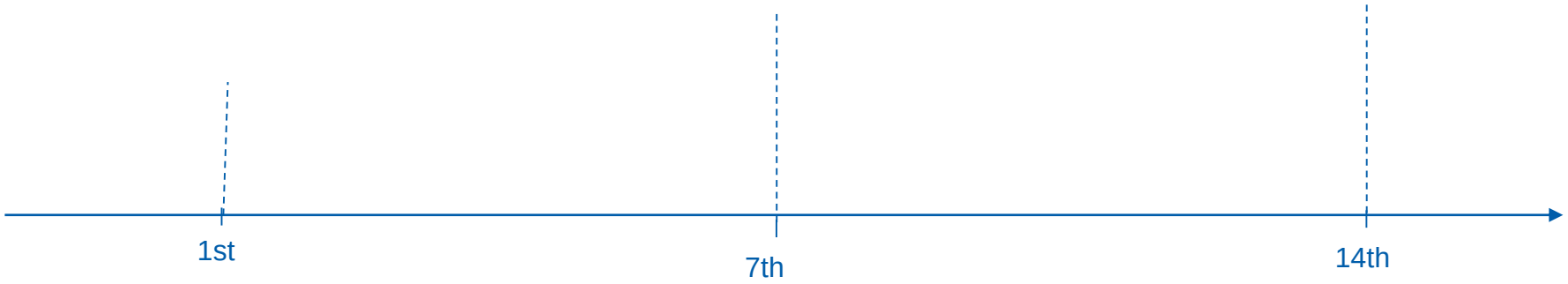
A dated local plan



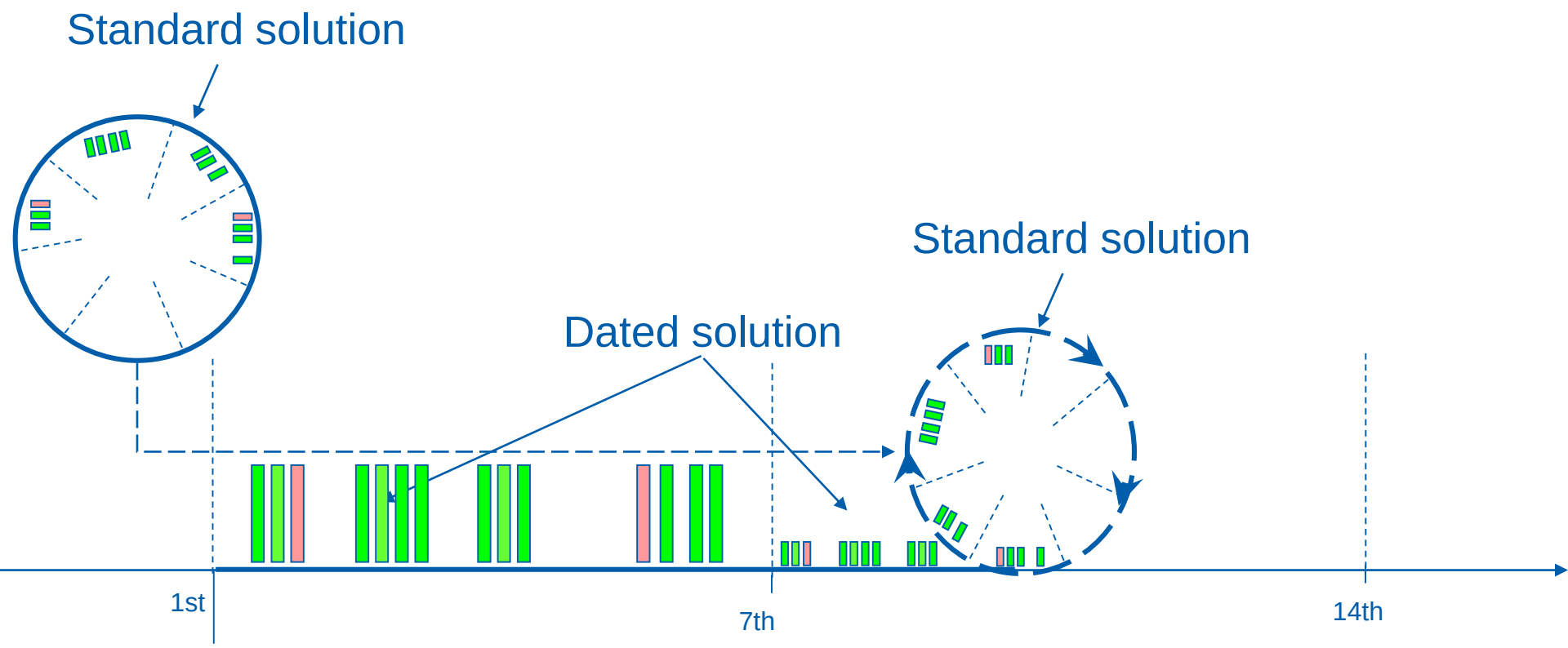
Create a dated solution – the idea

Standard solution





Create a dated solution – the idea

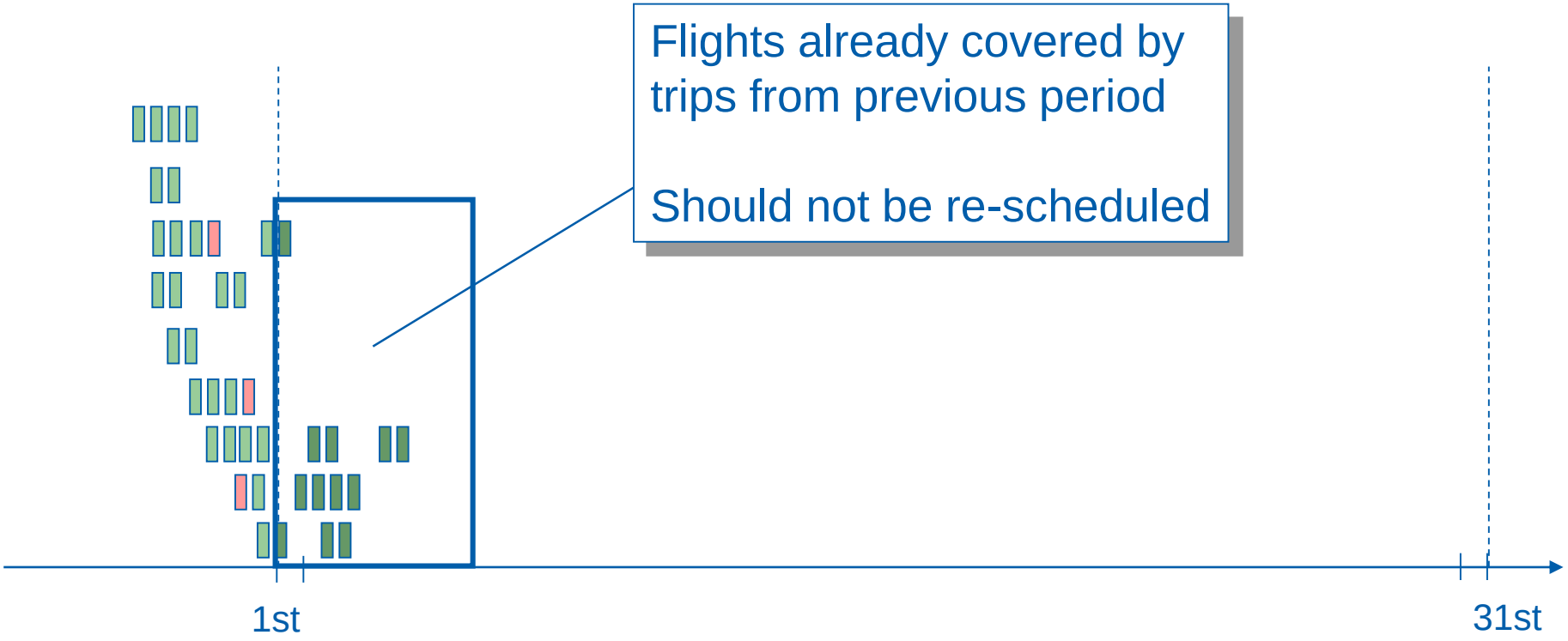


Standard to dated issues

Issues in brief:

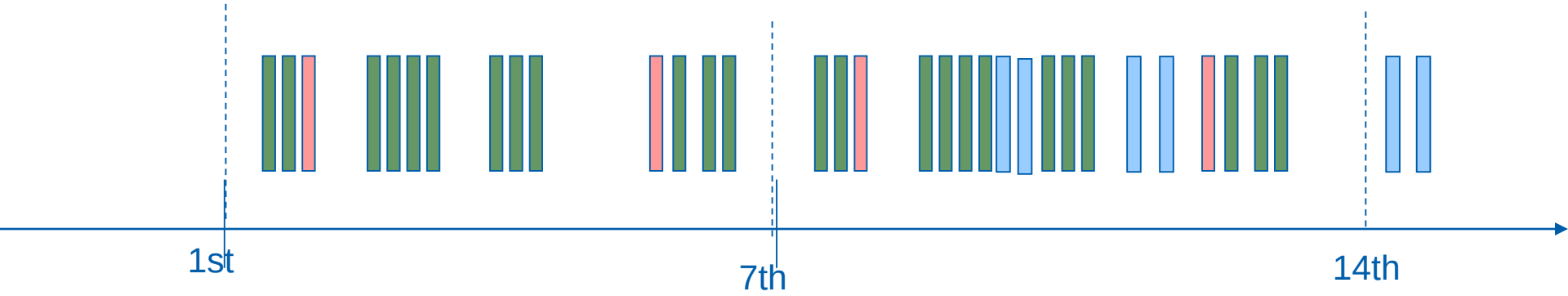
- trips from previous planning period cover flights in the beginning of this planning period
- new legs outside the standard week
- legs only existing in the standard week
- legs changing departure/arrival time
- legs changing aircraft type

Carry-in trips

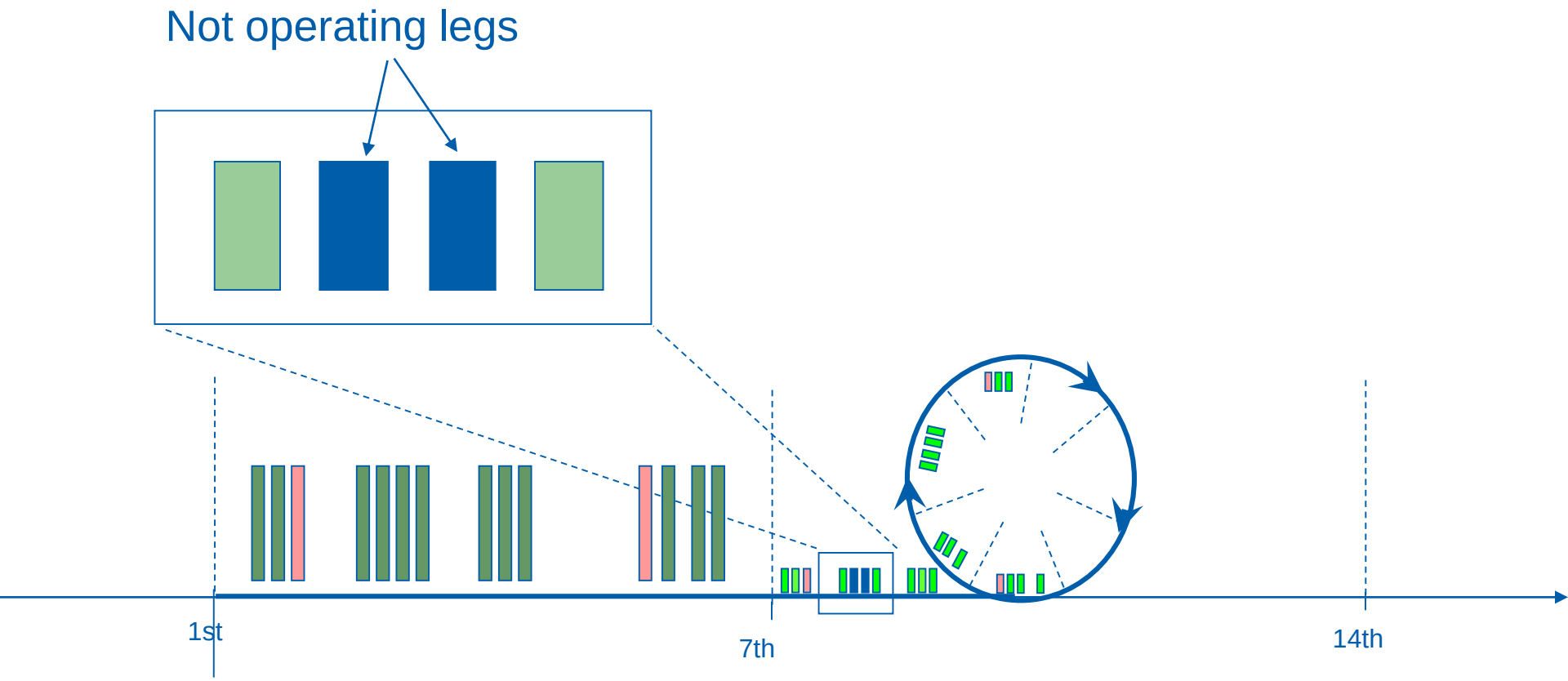


New legs

Legs you are supposed to plan that do not exist in the standard week must be **added** to the problem.



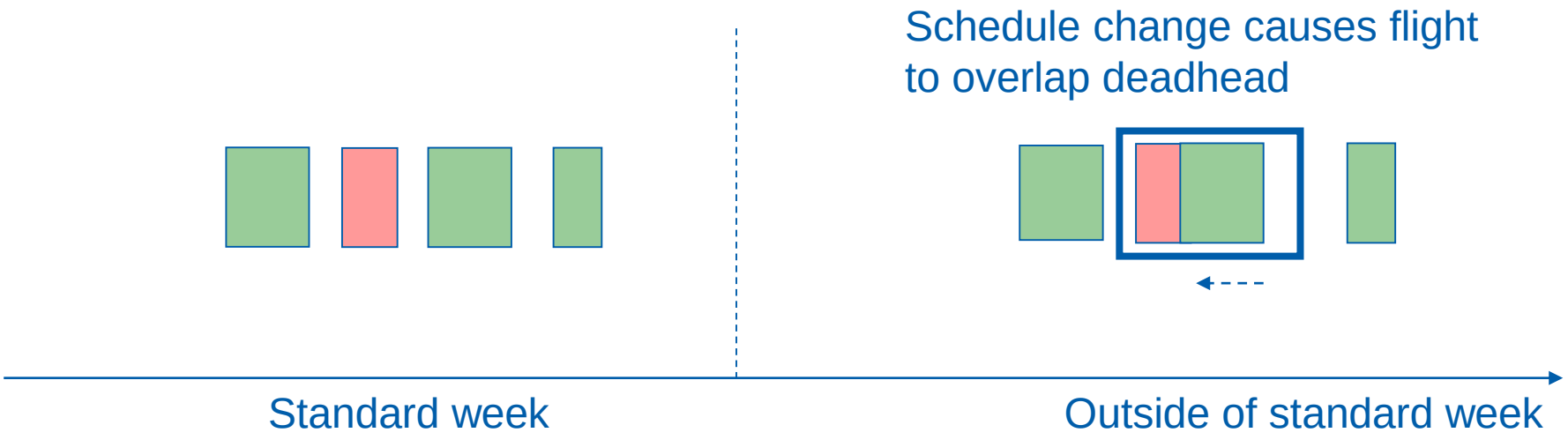
Flights existing only in standard week



Changes in departure/arrival time

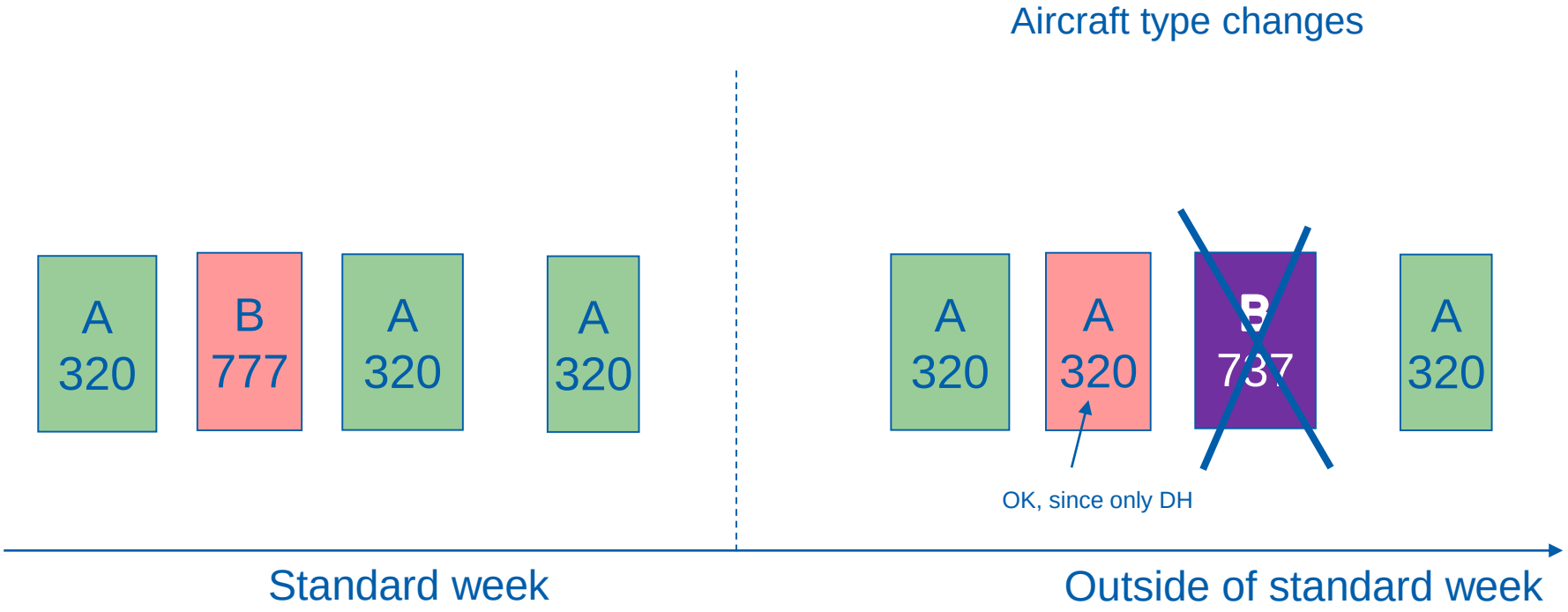
Smaller changes in departure/arrival times **may** not cause any problems.

Larger changes cause trips outside of the standard week to break.



Change of aircraft type

Aircraft type changes cause trips to break if crew is not qualified.



Create a dated solution

Main steps:

1. Create a dated local plan
2. Build rotations
3. Roll out the standard (weekly) solution
4. Add carry-in trips
   and new flights
5. Remove non-operating flights etc.
6. Repair the dated solution

Roll out to Dated

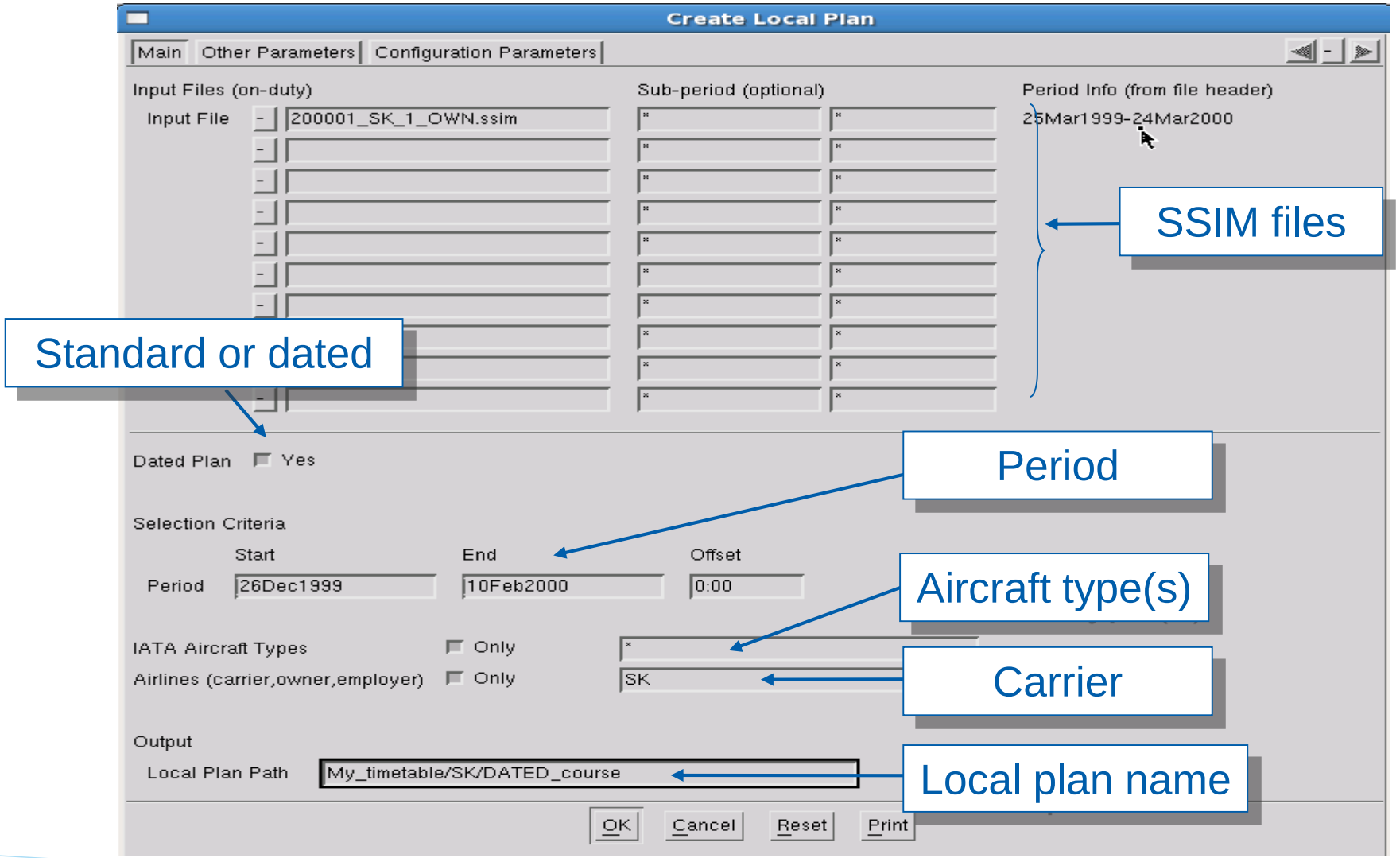
Roll out the standard solution

Open the standard (weekly) solution.

Roll out the standard solution to dated local plan.

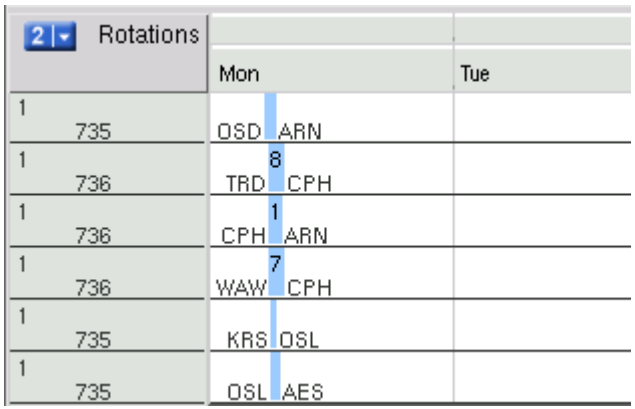
Create a dated solution

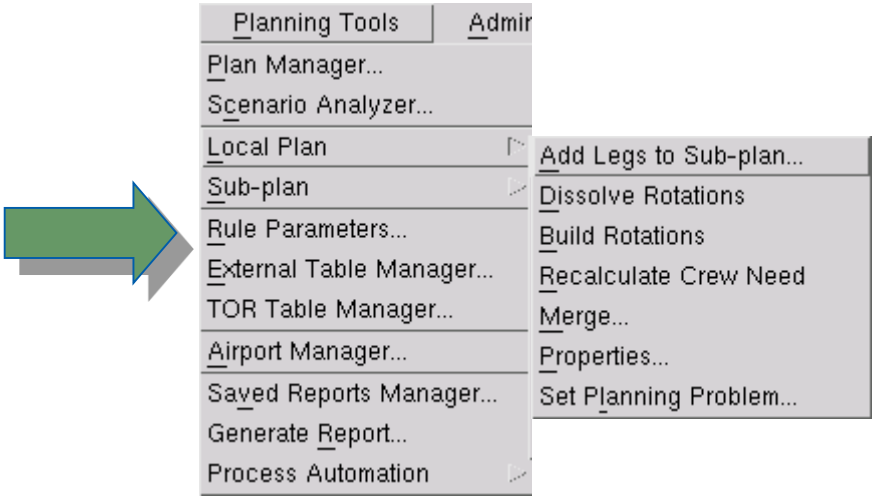
Create a dated local plan

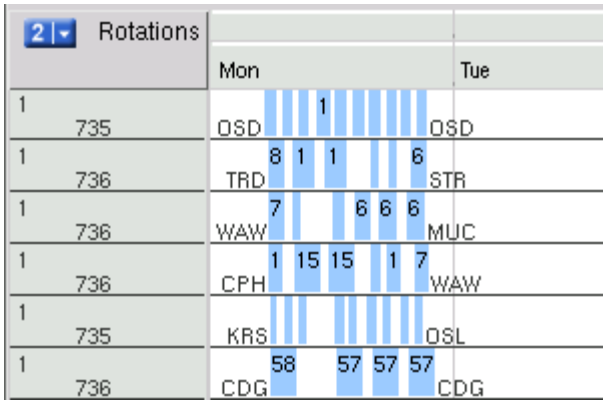


Create a dated solution

Build rotations

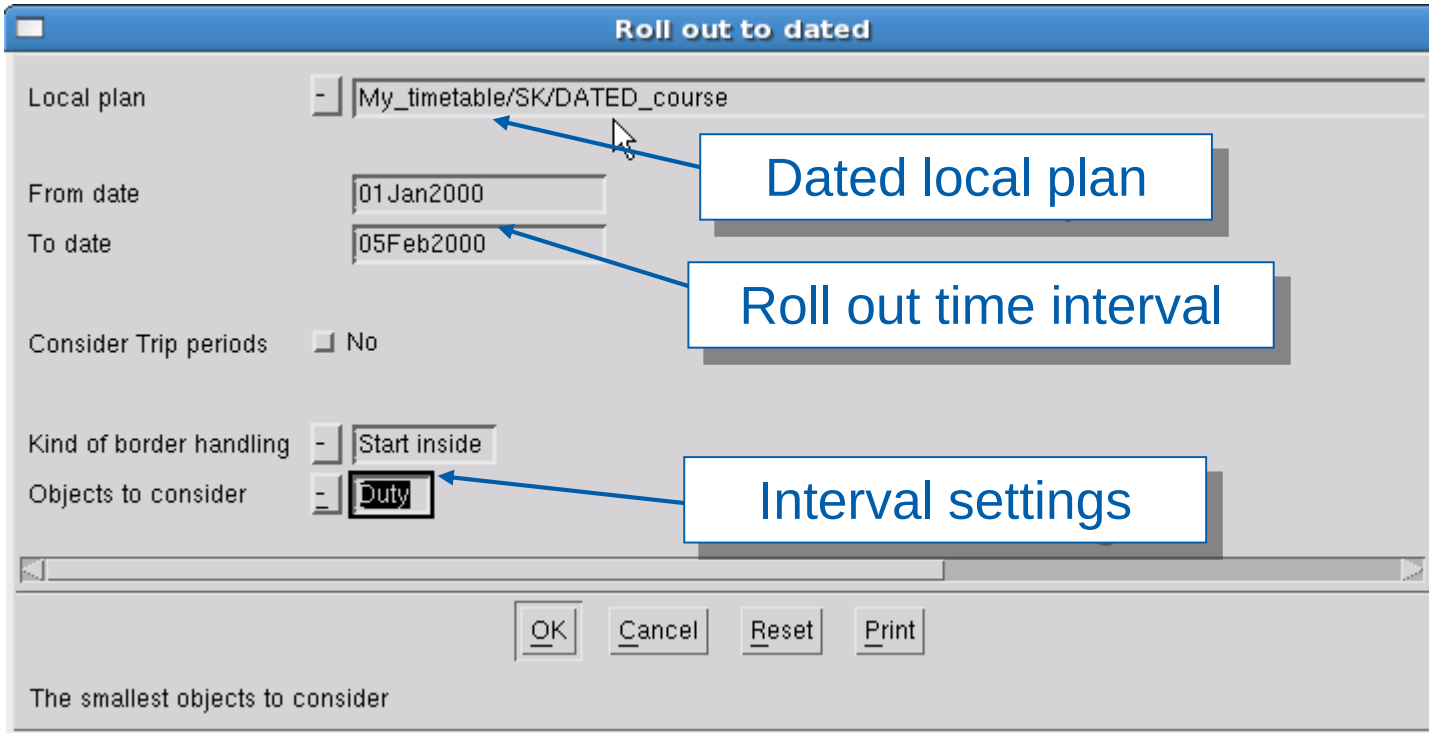






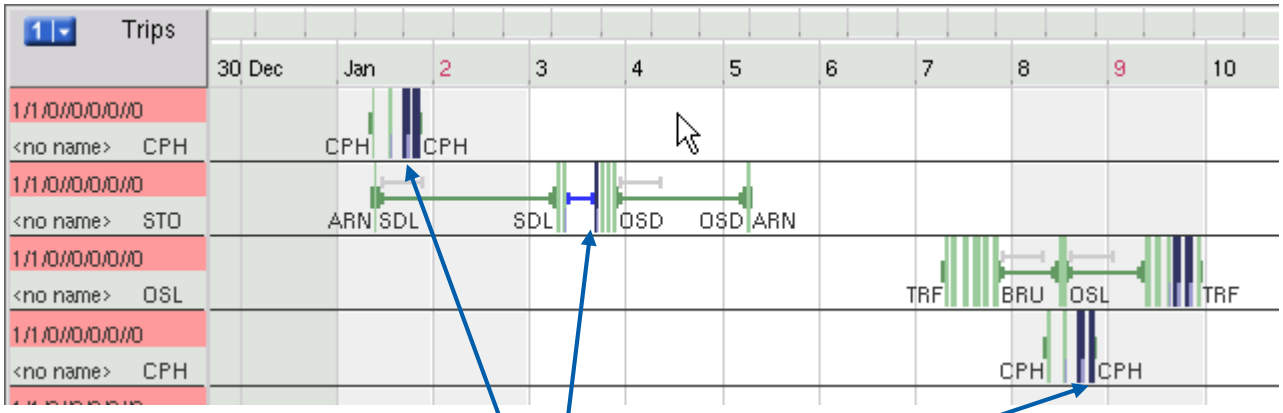
Create a dated solution

Roll out to dated



Create a dated solution

After roll-out



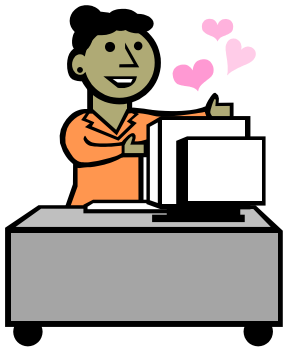
Not operating legs after roll-out

Exercise 5.A

Create a dated solution 5.1 to 5.4

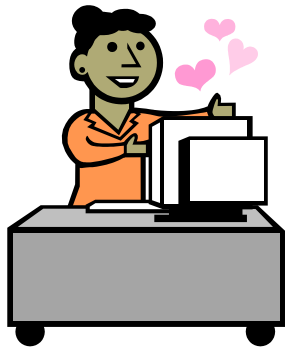
Purpose

To learn all steps for creating a dated solution



Exercise 5.A – summary

Summary of exercise



Add carry-ins and new flights

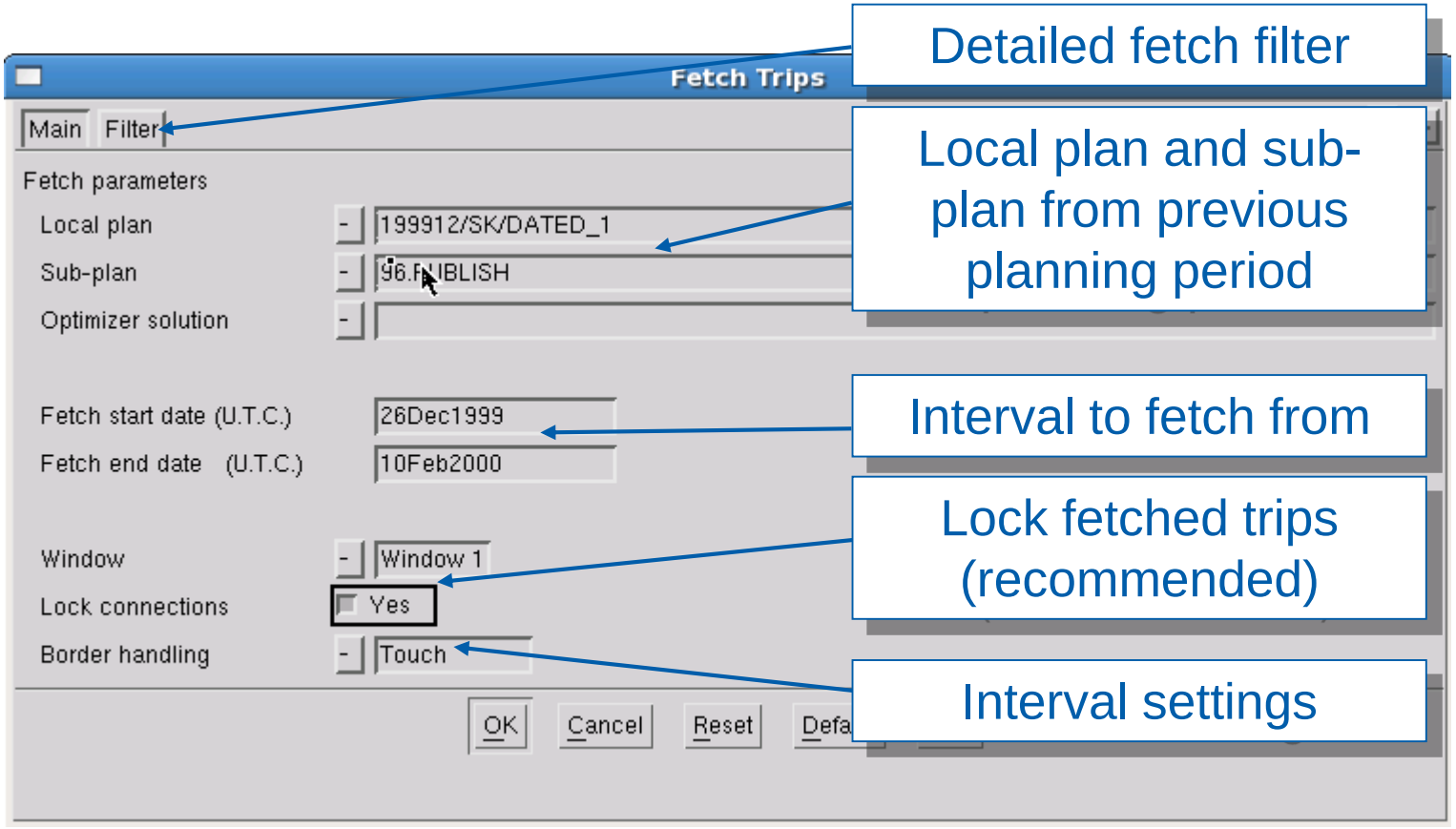
Add carry-in trips and new flights

Add carry-in trips from previous planning period

Add legs not existing in the standard week

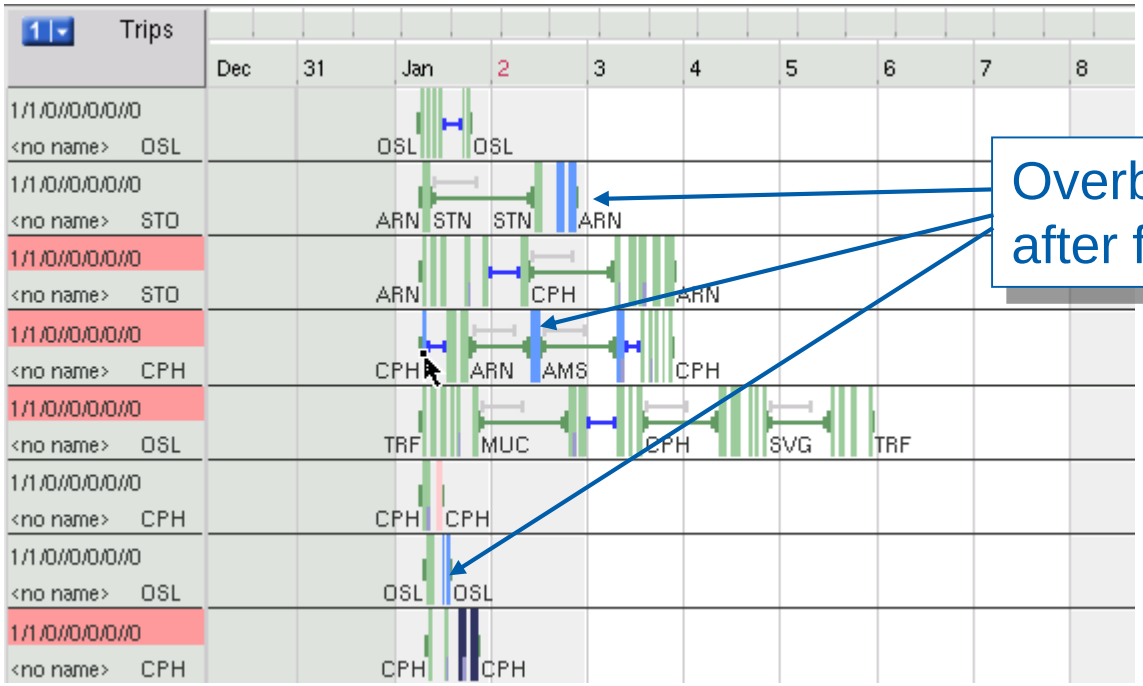
Create a dated solution

Fetch carry-in trips



Create a dated solution

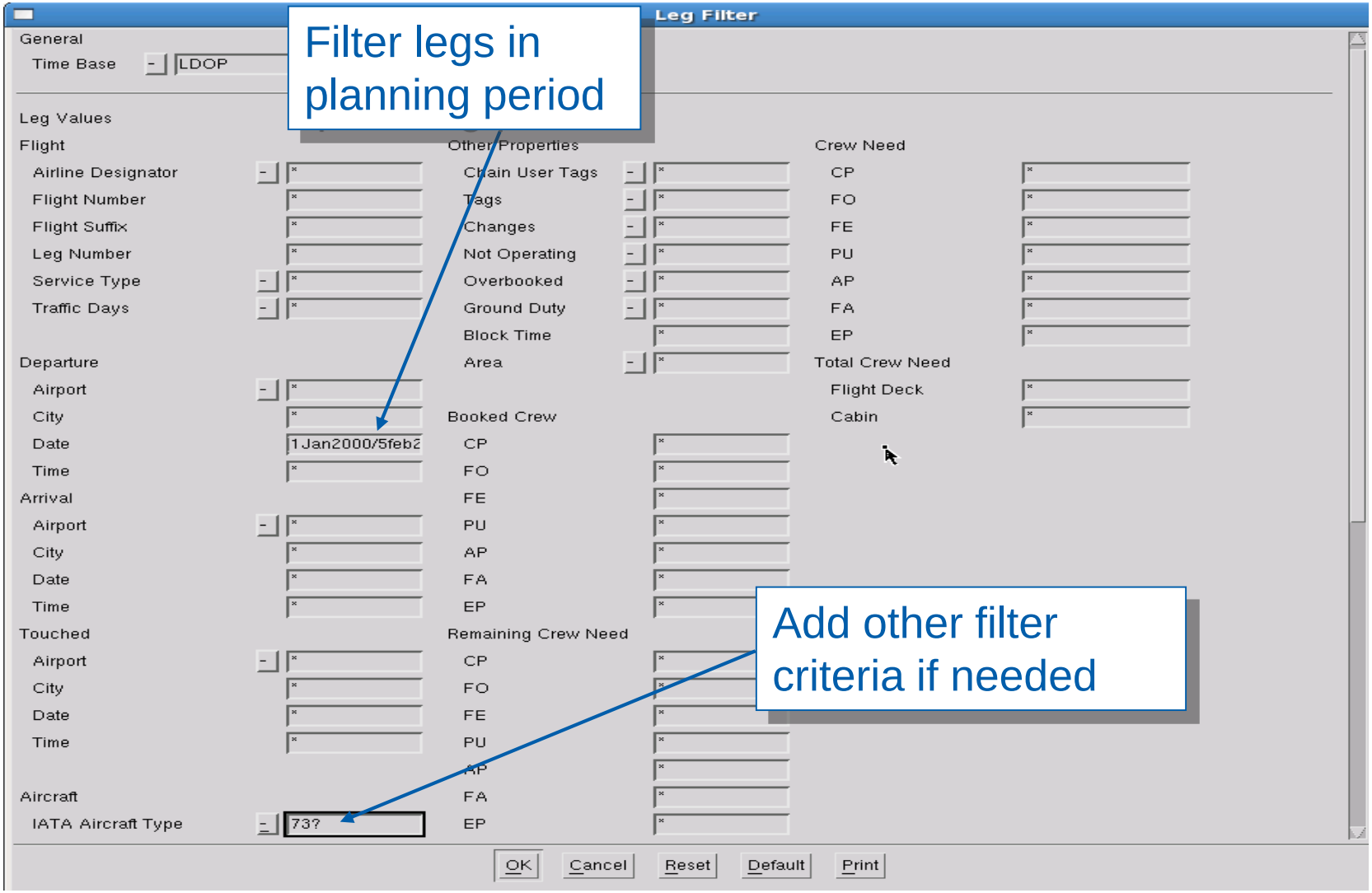
After adding carry-in trips



Overbooked legs after fetch

Create a dated solution

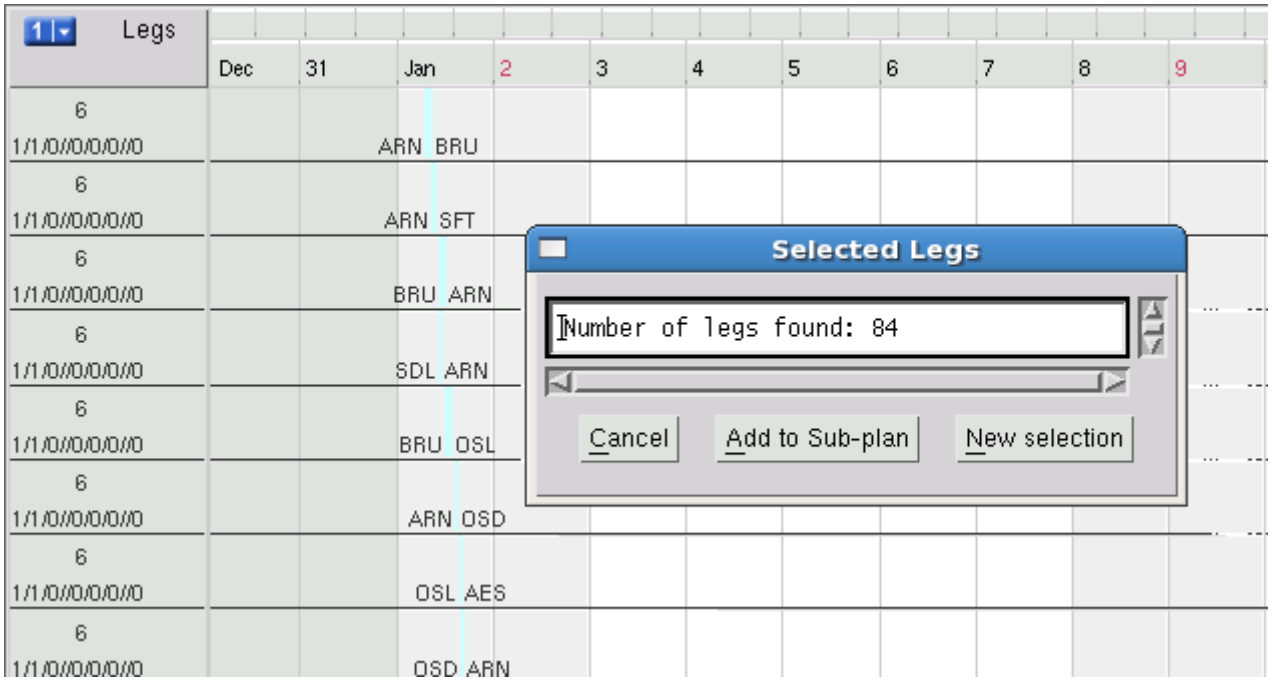
Add new leg



Create a dated solution

After adding new legs

Added legs should be converted to trips before starting the optimizer.

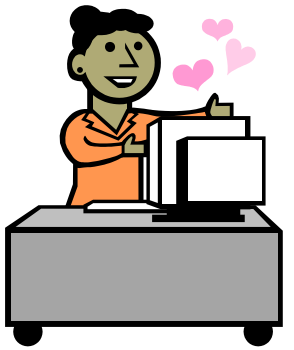


Exercise 5.B

Create a dated solution 5.5 to 5.6

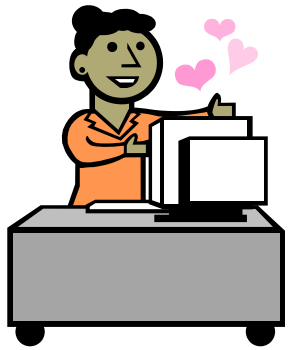
Purpose

To learn all steps for creating a dated solution



Exercise 5.B – summary

Summary of exercise



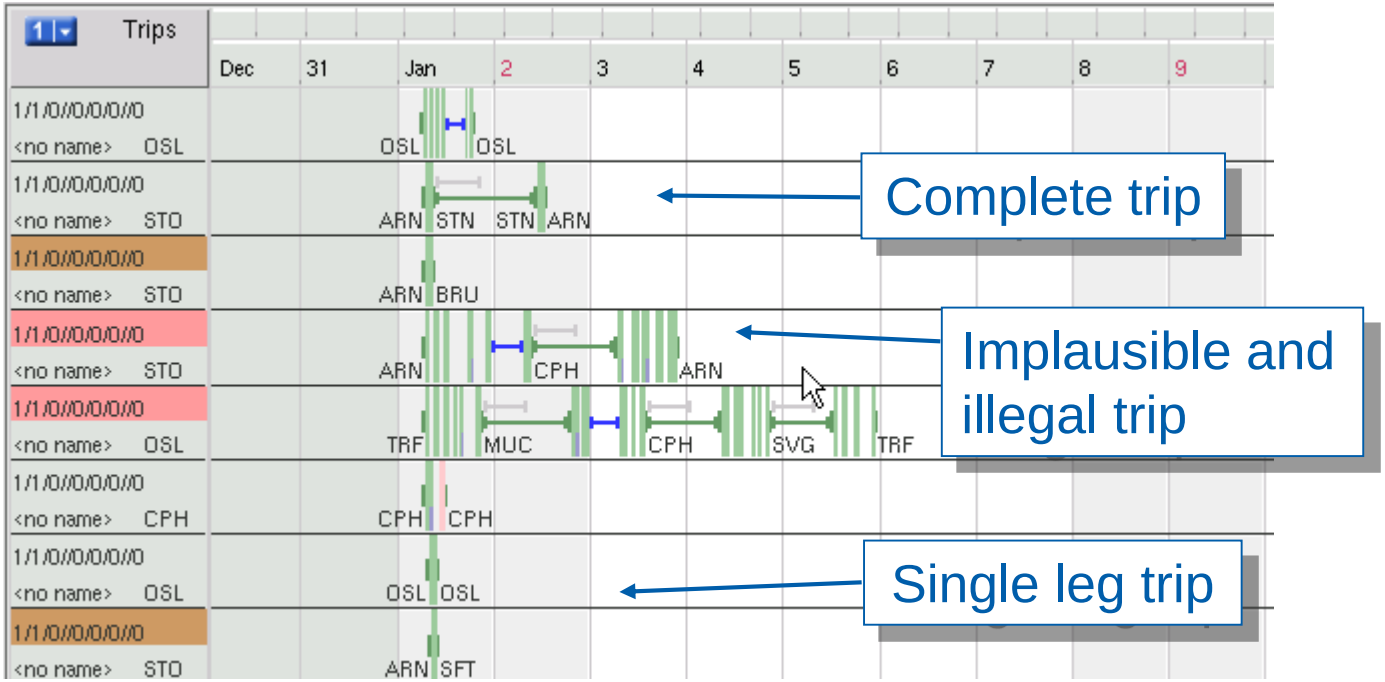
Remove and repair

Remove and repair

- Not operating legs
- Overbooked legs
- Other fleet on-duty
- Set parameters (from file and/or manually)
- Start Crew Pairing Optimizer

Create a dated solution

Show all trips and start the optimizer

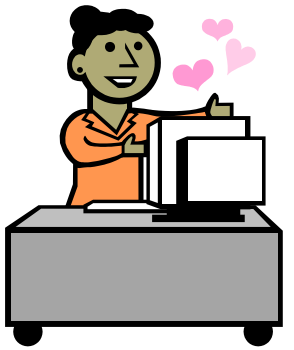


Exercise 5.C

Create a dated solution 5.7 to 5.9

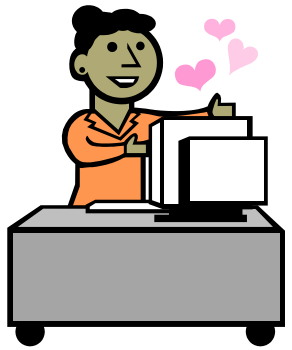
Purpose

To learn all steps for creating a dated solution



Exercise 5.C – summary

Summary of exercise



Variable crew need

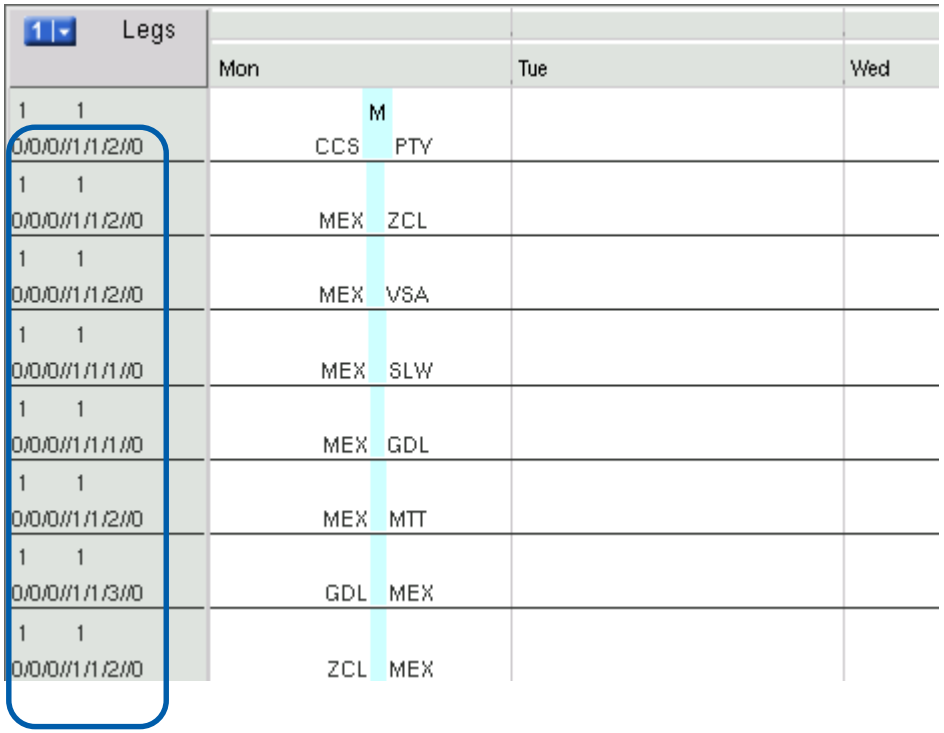
So far we have only been solving flight deck problems with a crew need of 1 CP and 1 FO (1/1/0).

How are cabin problems solved, where the crew need differs between the different legs?

This typically arises in cross-qualification cabin crew problems.

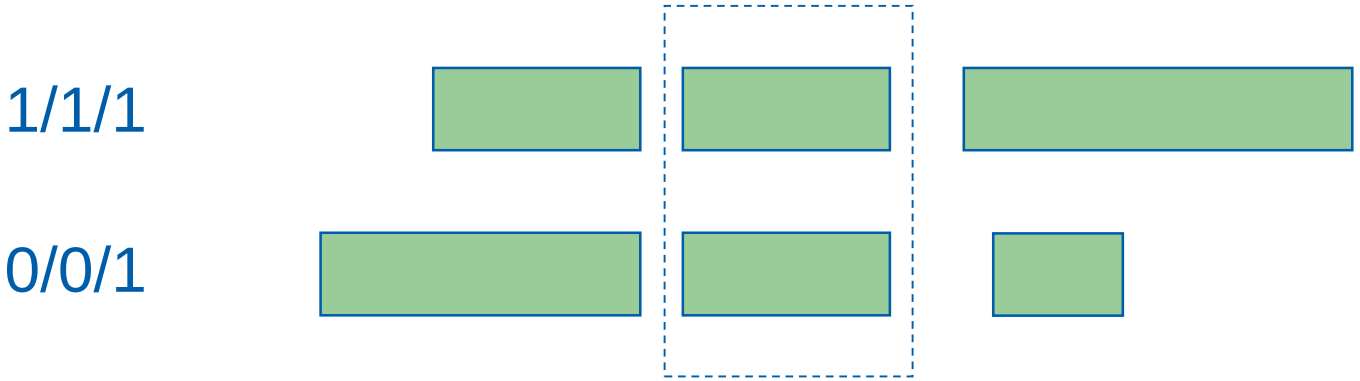
Variable crew need

Legs have different crew need (e.g. 1/1/1, 1/1/2, 1/1/3)



Variable crew need – crew change

Crew changes increase the sensitivity of the solution as a delay of one leg may have impact on several following legs.



Variable crew need

Variable crew need problems can be solved in a number of ways:

1. Sub-problems with identical crew need are solved separately (minimise crew changes).

2. Two different crew complements: One basic solution with trips for the maximum common crew need, covering all legs. One jumper solution with trips for one crew member.

3. Different methods combining the above.

Variable crew need

A simple example

An airline has four aircraft types with varying cabin crew need, everybody is qualified for all aircraft types:

F100 (1/1/1)

A320 (1/1/2)

B727 (1/1/2)

B757 (1/1/3)

Variable crew need – example

Sub-problem approach

Three independent sub-problems:

a) F100 flight legs (1/1/1)

b) A320 + B727 flight legs (1/1/2)

c) B757 flight legs (1/1/3)

GOOD: No crew changes
BAD: Often expensive

Variable crew need – example

Jumper approach

A two step procedure:

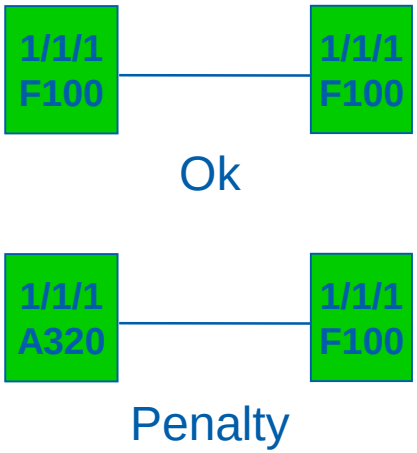
GOOD: Often cheapest
BAD: Lots of crew changes

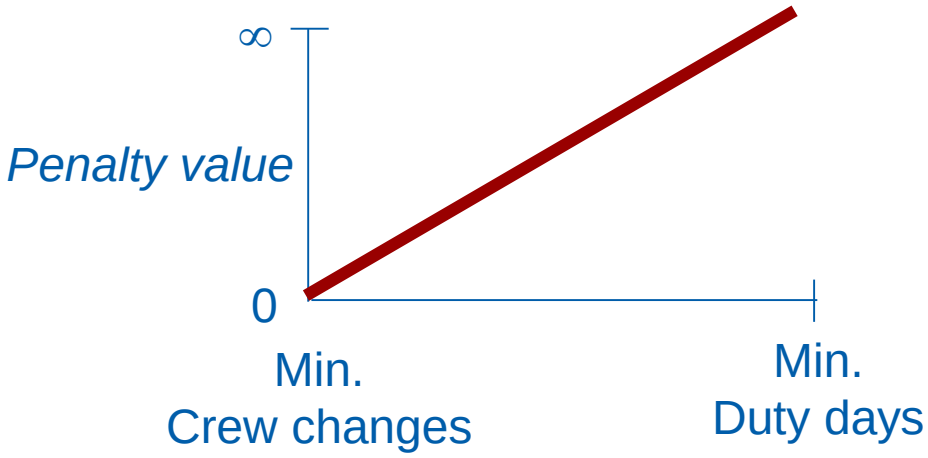
1. The Basic solution. All legs are included, creating trips with crew complement 1/1/1
2. The Jumper solution. Trips are created with a crew complement of 0/0/1. A320 + B727 flight legs are included once. B757 flight legs are included twice.

Variable crew need

Penalty approach:

Penalizing connections between flights of non-identical crew need (in the basic solution of jumper approach) will give results somewhere in between:





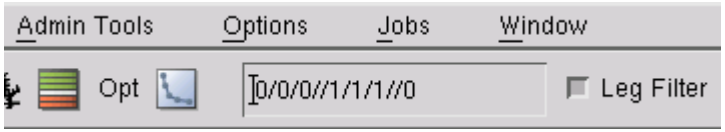
There is not any “correct” approach - every airline must decide on its own strategy.

Variable crew need – Jumpers

- Jumper pairings are usually not popular among crew - no one wants to fly “alone”.
- More jumper pairings can mean more instances of crew changes - and hereby less stable solutions. However, this can somewhat be controlled by forcing imitation between basic and jumper duties.
- Different strategies might be applied depending on flight hours in planning period, vacations, lack of resources etc.

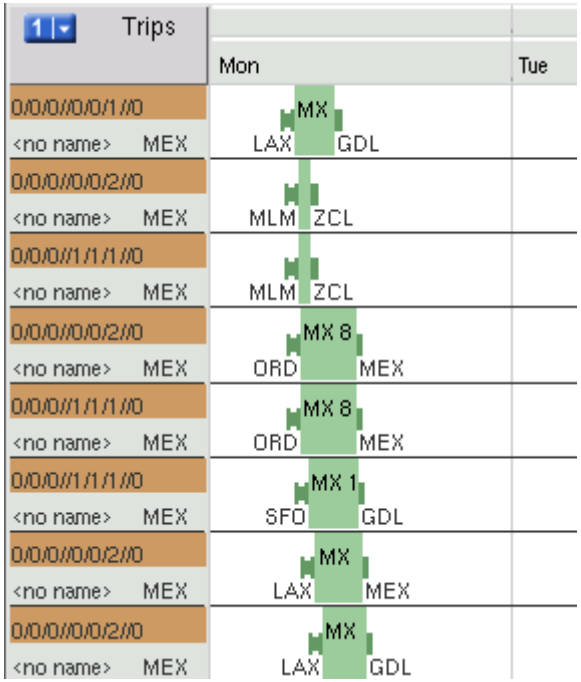
Variable crew need – Jumpers

When creating new trips from legs – either manually or using the optimizer – the crew complement of any new trip is given by the **assign value**.



Variable crew need – Jumpers

Several copies of the same leg can be planned at the same time. Hereby, several planning steps can be combined in one.

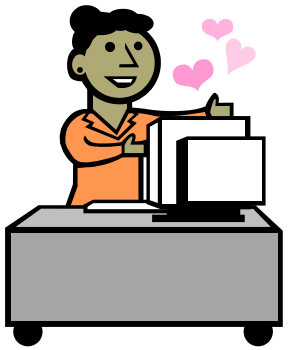


Exercise 6

Variable Crew Need

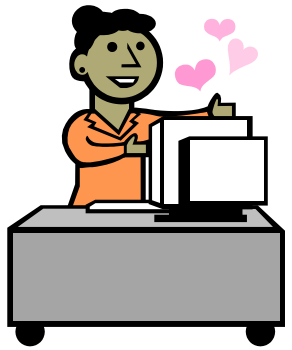
Purpose

To see differences by solving a variable crew need solution according to different approaches



Exercise 6 – summary

Summary of exercise



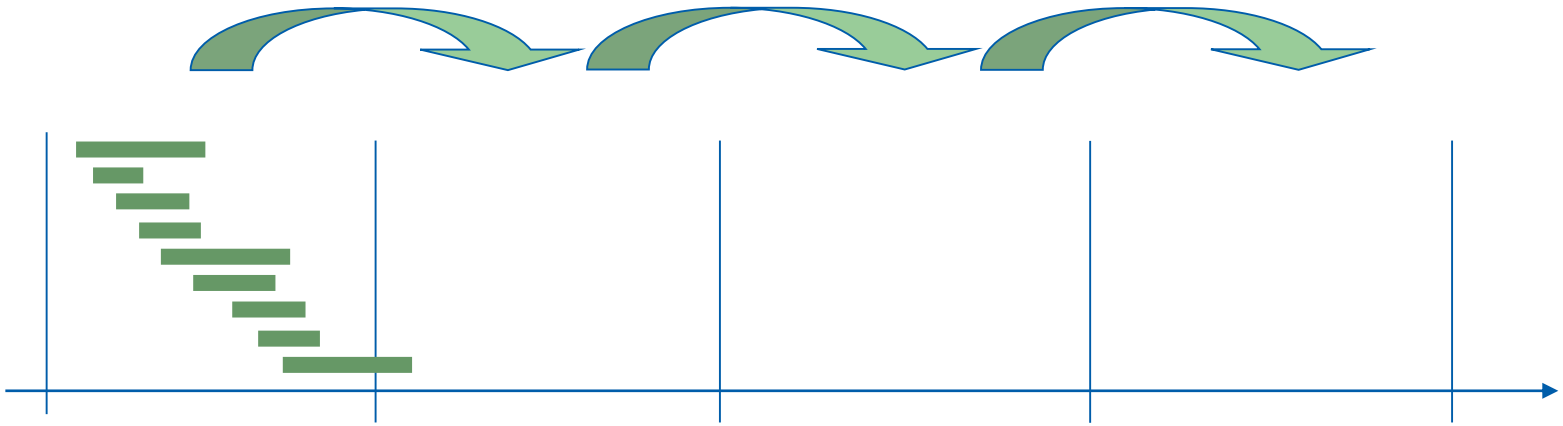
Regularity

Designing trips that are more or less identical from one period to another (typically from week to week:

- Union requirement
- Eases maintenance (day of operation)

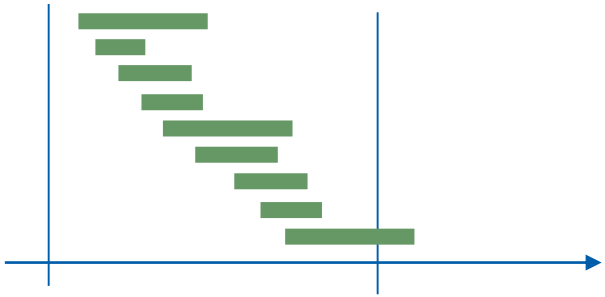
Regularity

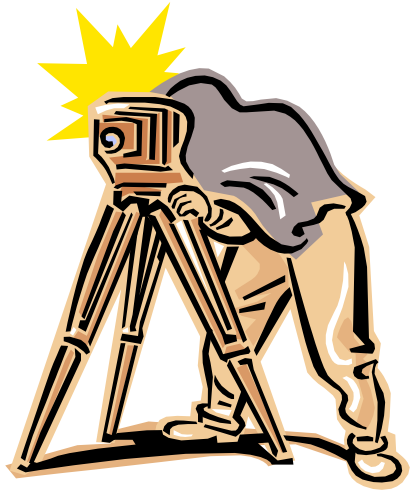
Pattern of the regular solution (daily or standard week solution) should be preserved in the dated solution.



Regularity

When solving problems arising in dated mode, regularity must be actively supported by rule code (rules and penalties), preserving the structure of the standard week solution.

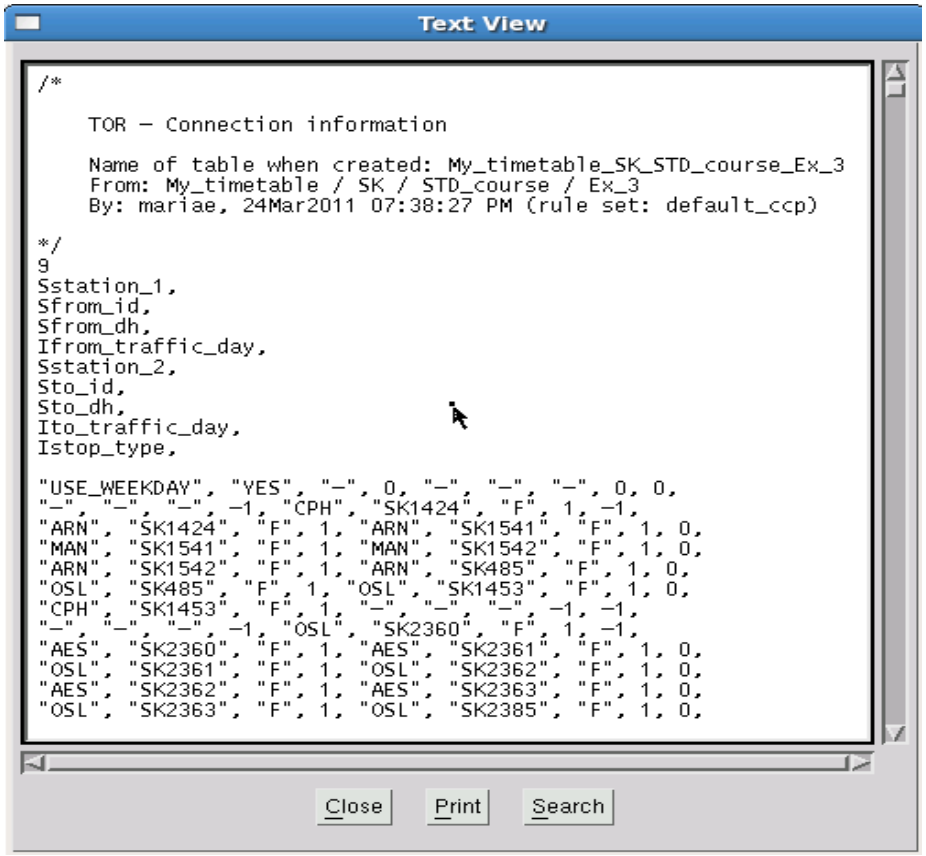




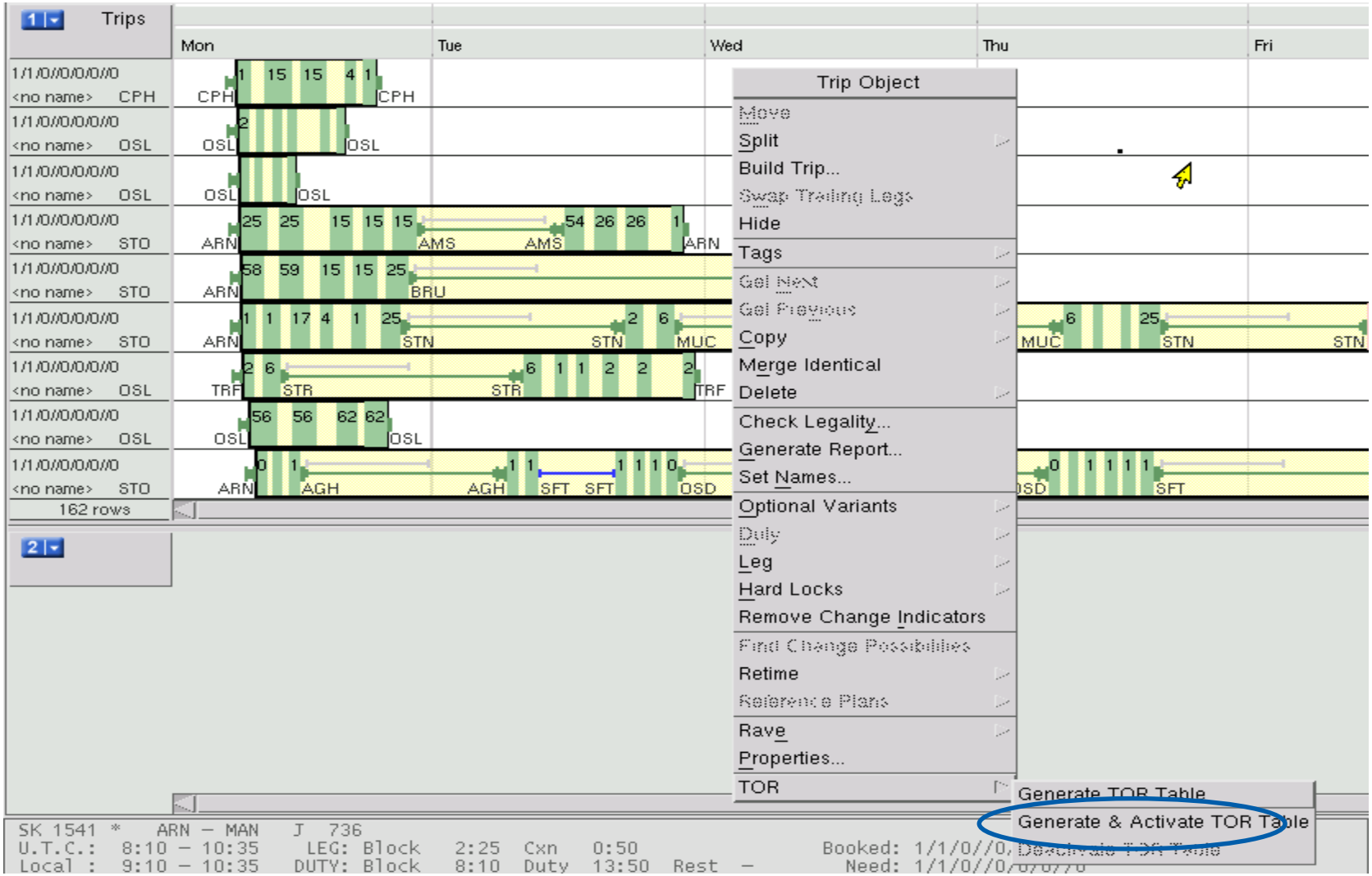
Regularity

Functionality in the Regularity add-in (also called TOR, Table Of Regularity):

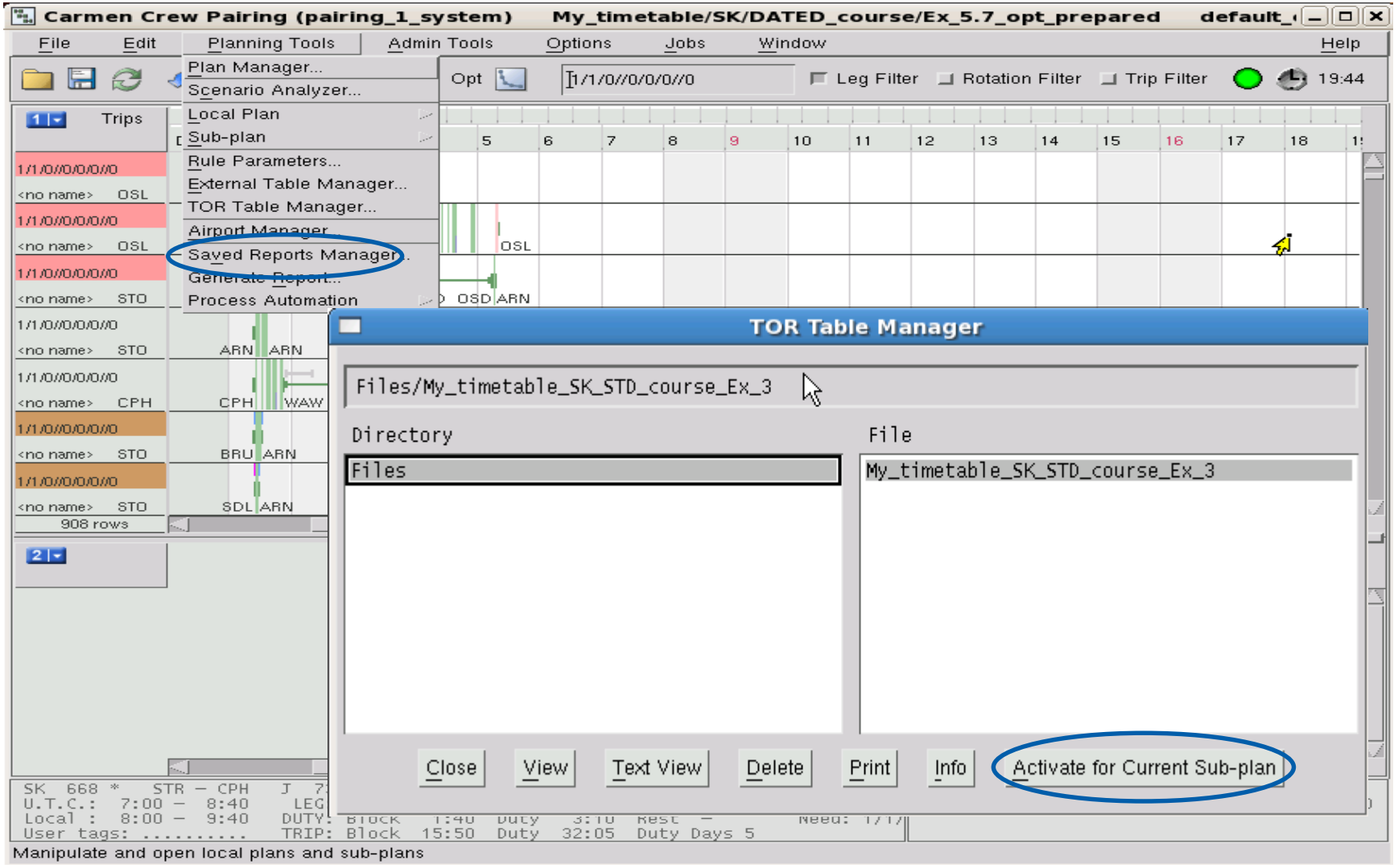
- takes a snapshot of a regular solution
- adds rules preventing anything but strict regularity
- adds penalties for deviations from strict regularity.



Generate TOR



Activate TOR table

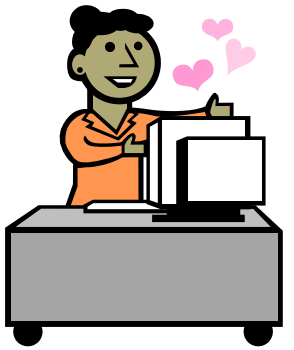


Exercise 7

Extra exercise – Imitation

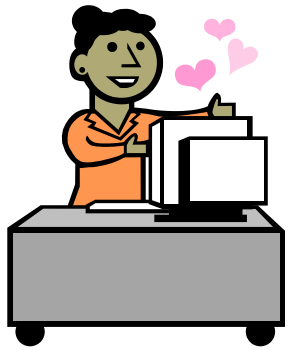
Purpose

To get a picture on how imitation effects the solution.



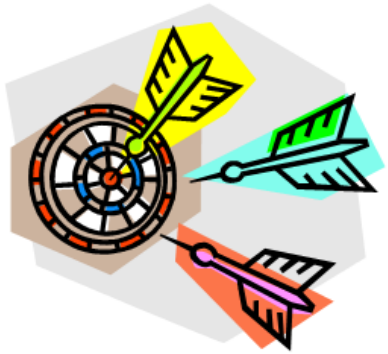
Exercise 7 – summary

Summary of exercise



Summary

- Four modules
- Windows and menus
- Manual planning functions
- Planning concepts
- Rules: legal, union and quality
- Standard week solution
- Influence the solution by penalties
- Dated solution
- Variable crew need
- Imitation

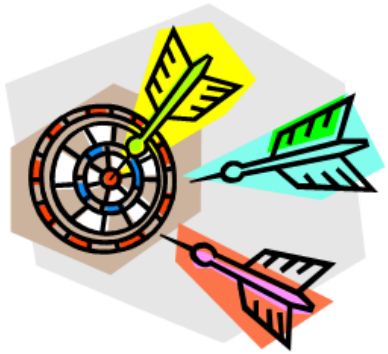


Summary

Planning process

- Create standard local plan
- Build rotations
- Load rules and parameters
- Create sub-plan by adding legs
- Run Crew Pairing Optimizer
- Verify solution

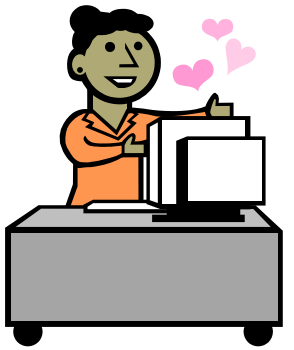
- Create dated local plan
- Build rotations
- Roll out standard solution to dated
- Load rules and parameters
- Add carry-in trips
- Remove NOPs, overbooked etc.
- Add new legs
- Run Crew Pairing Optimizer
- Verify solution



Exercise 8

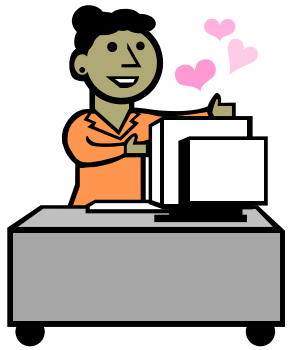
Entire Process Steps

Purpose
Repetition



Exercise 8 – summary

Summary of exercise



Course Evaluation

Please take a few minutes to complete the evaluation form, it will help us improve the courses for you and your colleagues:

- Login: 

- Start Explorer: 
- Fill in the course information
- …and your role (Internal for Jeppesen)

Are the exercise definitions too vague (too real-life), would you like them to be more exact and straight forward?

Would you like to have even more info on slides (for self studying) or would you be stressed about the time constraint?

The end

This was Pairing I.

Welcome back to Jeppesen Crew Academy!