Scheduling Problem Solutions in Transport Enterprises

Andrey Orlov
Software Engineering School
National Research University Higher School of Economics
Moscow, Russia
lokaro.oa@gmail.com

Scientific Advisor: Prof. Sergey Avdoshin
Software Engineering School
National Research University Higher School of Economics
Moscow, Russia
savdoshin@hse.ru

***Abstract* – The classical transportation problem is a problem of optimal transportation plan of a homogeneous product of uniform items in the presence of homogeneous items of consumption on homogeneous vehicles with static data and the linear approach [1]. Development and application of optimal schemes of cargo flows can reduce the cost of transport and maximize profits. We should make a detailed transportation plan, which should take into account a number of limitations. Models for such tasks contain more 50,000 variables and constraints. The market hasn't practically any software, which satisfies all companies' needs. In this article we will consider the process of modeling in IBM ILOG CPLEX Optimizer. This software includes efficient algorithms for solving optimization problems.**

Keywords: optimization; modeling; scheduling; railway; CPLEX; OPL.

I. Introduction

The science of better decisions Operations Research (OR) is the discipline of applying advanced analytical methods to make better decisions [2]. By using techniques such as mathematical modeling to analyze complex situations, operations research gives executives the power to make more effective decisions and build more productive systems.

To understand how optimization touches many aspects of everyday life, consider a business trip. The price you pay for your ticket is determined by optimization. The flight crew is scheduled by optimization. Trucks you might see on the road are loaded up and routed by optimization. And when you get to your hotel room, and turn on the TV to relax, the scheduled ads are optimized to maximize the revenues for the network.

The IBM ILOG optimization products put the power of optimization in the hands of business decision makers. We will use these tools for modeling scheduling problem in transportation enterprise.

This work is a logical continuation of [3] and being performed within the scope of the research on the topic "Research and development of innovative unifying models of intelligent systems for the situational response and safety control on the Russian railways", state contract 07.514.11.4039 on September 26, 2011 at lot № 2011-1.4-514-045 "Development of algorithms and software systems for solving problems of exceedingly large scientific data sets storage and processing and data streams collection in real-time" as part of the federal target program activity "Research and development in Russian scientific-technological system 2007-2013 evolution priority directions".

II. Overview

We classify and give in detail the basic concepts [4]. All vehicles can be divided as follows:

- Aircraft (passenger, cargo);
- Train (passenger, freight, suburban, long-distance);
- Truck (Short-haul, long-haul, hazmat transportation);
- Ship (liner, costal traffic, ferry, tanker);
- Public transit (Bus, metro, suburban train);
- …

Thus, we can identify problems for each type. For example, passenger and public transport planning is contained in the scheduling of the day, while sending orders for truck or rail will be actually in planning for a month. Therefore, according to the duration, the planning can be divided into four types:

- Strategic (several years):
  - Network design and location of main facilities: airports, highways, subway lines;
  - The purchase of the fleet;
- Tactical (several weeks or months):
  - selection of a route bus lines;
  - location of the intermediate objects: stations, warehouses;
  - laying seasonal routes, transportation schedules;
  - prices;
- Operational (a few hours or days):
  - destination transport on the route;
  - transport of parcels;

- Real time (seconds or minutes):
    - Location and movement of ambulances or fire truck

Among these types we are interested only in tactical planning, as time is limited to one month. In the paper we discuss the problem of rail transportation. It can be divided into two classes - passenger and freight. Suburban and intercity trains are all the passenger type. Scheduling is characterized by a single goal for this class. In the case of force majeure on the railway, you can plan bypasses, but it is not as frequent. The special features of passenger transport are:

- Network design;
- Train scheduling;
- Blocking the way;
- Appointment of drivers;

The movement of freight trains is an important part of a fully functioning transportation system. Efficient movement of goods both within and across regions is necessary for industry, retail and international trade, and agriculture. There are airports, major terminals, shipyards in large cities, which are particularly affected by the issue of cargo transportation. The task of planning depends on a very large number of constraints. The peculiarities of this kind of traffic can be classified as follows:

- Fragmentation;
- Not so well organized and optimized, for example, in contrast to the aviation industry;
- The profit can be increased through better planning, use of the crew and pricing policies;

III. Technical details

IBM ILOG CPLEX Optimization Studio is one of the IBM ILOG optimization products [5]. These products include IBM ILOG Optimization Decision Manager (ODM) Enterprise and some packaged applications. We don’t consider other optimization software, because we need an enterprise application, which simply build with ODM. CPLEX Studio and ODM Enterprise are used to develop custom applications based on Mathematical Programming (MP) or Constraint Programming (CP).

- the CPLEX engine for mathematical programming is used by default when you run your project if your model does not start with the statement using CP;
- the CP Optimizer engine for constraint programming is called if your model starts with the statement using CP;

IBM ILOG CPLEX CP Optimizer is specially adapted to solving detailed scheduling problems over fine grained time [6]. There are, for example, keywords particularly designed to represent typical scheduling model elements, such as tasks and temporal constraints.

With CP Optimizer, you can address the issues inherent in detailed scheduling problems from manufacturing, construction, driver scheduling, and more.

In a detailed scheduling problem, the most basic activity is assigning start and end times to intervals. Scheduling problems also require the management of minimal or maximal capacity constraints for resources over time, and of alternative modes to perform a task.

A. *Interval*

An interval variable represents an interval of time during which something happens (for example, a task occurs, an activity is carried out) and whose position in time is an unknown of the scheduling problem. An interval is characterized by a start value, an end value, a size and intensity. The length of an interval is its end time minus its start time.

An important additional feature of interval variables is the fact that they can be optional; that is, one can decide not to consider them in the solution schedule.

```
dvar interval <taskName>
       [optional[(IsOptional)]]
       [in StartMin..EndMax]
       [size SZ | in SZMin .. SZMax]
       [intensity F];
```

For example, the following variable:

```
dvar interval garden optional in 20..32 size 5;
```

means, that the task "garden", if it will run takes 5 time units and must begin after 20 units of time and before the end 32 units of time.

B. *Cumulative function*

In scheduling problems involving cumulative resources (also known as renewable resources), the cumulated usage of the resource by the activities is usually represented by a function of time. An activity usually increases the cumulated resource usage function at its start time and decreases it when it releases the resource at its end time (pulse function). For resources that can be produced and consumed by activities (for instance the content of an inventory or a tank), the resource level can also be described as a function of time; production activities will increase the resource level whereas consuming activities will decrease it. In these types of problems, the cumulated contribution of activities on the resource can be represented by a function of time and constraints can be modeled on this function, for instance a maximal or a safety level.

```
cumulFunction <functionName> =
       <elementary_function_expression>;
```

where <elementary_function_expression> is a cumulative function expression. This expression includes:

- step;
- pulse;
- stepAtStart;

- stepAtEnd.

Let us consider each of them.

1) Pulse

Pulse represents the contribution to the cumulative function of an individual interval variable or fixed interval of time. Pulse covers the usage of a cumulative or renewable resource when an activity increases the resource usage function at its start and decreases usage when it releases the resource at its end time.

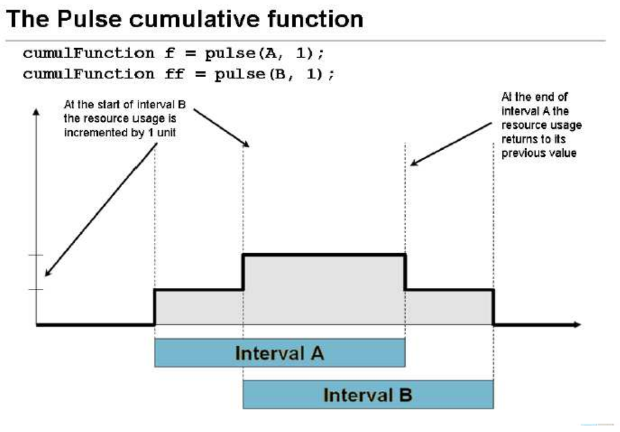
```
cumulFunction f = pulse(u, v, h);
cumulFunction f = pulse(a, h);
cumulFunction f = pulse(a, hmin, hmax);
```

The pulse function interval is represented by a, or by the start point u and end point v. The height of the function is represented by h, or bounded by hmin and hmax.

To illustrate, consider cumulative function of using resources, which a volume is a measure. There are two intervals – A and B – which are time-limited. Each interval increases the value during duration.

```
cumulFunction f = pulse(A, 1);
cumulFunction ff = pulse(B, 1);
```

The functions are shown at Figure 1.



```
cumulFunction f = pulse(A, 1);
cumulFunction ff = pulse(B, 1);
```

Fig 1. Pulse function

2) Step

Step is an elementary cumulative function expression representing the contribution starting at a point in time. Step covers the production or consumption of a cumulative resource.

```
cumulFunction f = step(u, h);
```

where time u is the start of production or consumption and the height of the function is represented by h.

As another example, consider a function of flow measurement resources, similar to the budget:

The level of resources is equal to zero up to time 2, when the value increases to 4.

```
cumulFunction f = step(2, 4);
```

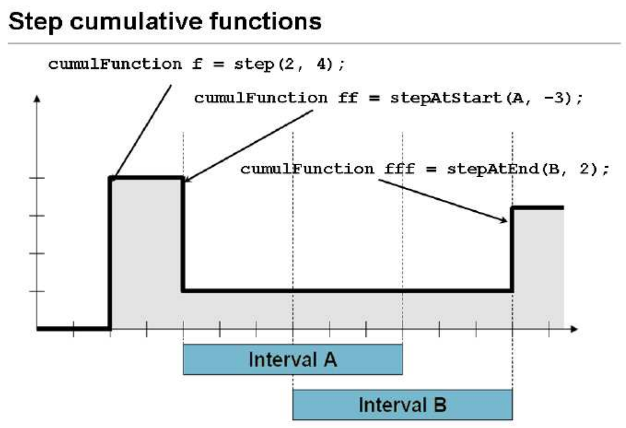
There are two intervals – A and B, which are fixed in time. Interval A reduced level of resources for 3 at the beginning of the interval.

```
cumulFunction ff = stepAtStart(A, -3);
```

Interval B increases resource level for 2 at the end of the interval.

```
 cumulFunction fff = stepAtEnd(B, 2);
```

The functions are shown at Figure 2.



```
cumulFunction f = step(2, 4);
cumulFunction ff = stepAtStart(A, -3);
cumulFunction fff = stepAtEnd(B, 2);
```

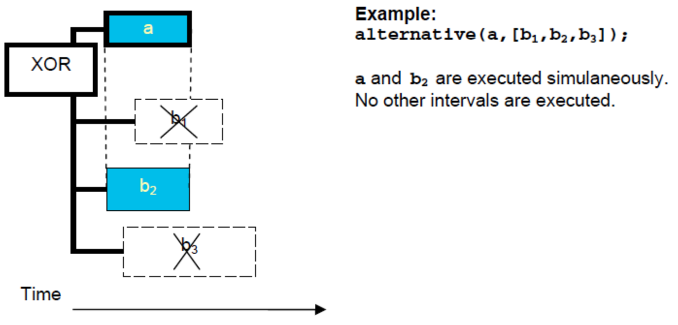
Fig 2. Step function

3) Alternative

The constraint alternative(a, {b1, .., bn}) models an exclusive alternative between {b1, .., bn}. If interval is present then exactly one of intervals {b1, .., bn} is present and a starts and ends together with this chosen one.

The constraint alternative(a, {b1, .., bn},c) models a selection of c intervals in the set {b1, .., bn}. If interval is present then exactly c intervals in {b1, .., bn} are present and a starts and ends together with these selected ones. If it is absent, then all b intervals are absent. This constraint is typically used to model the selection of 1 resource (or c resources) among a set of candidate ones. It can also be used in more complex cases to model alternative execution modes for activities or alternative time-windows for executing a task.

The array B must be a one-dimensional array; for greater complexity, use the keyword all.



Example:
`alternative(a,[b1,b2,b3]);`

a and b_2 are executed simulaneously.
No other intervals are executed.

Fig 3. Alternatives

IV. Experimental evaluation

The company has available near 100.000 wagons, which are evenly distributed over at the 1.000 stations. There are 10.000 applications for transportation of cargo from one station to another in the company. We should construct transportation plan for a month.

This condition is a real problem. For demonstration we reduce the scale of 100 times. Also part of constraints simplify to understanding of the problem: we aren't considered options such as progressive rate, rate group and so on.

For this problem, we build three optimization models and analyze them. Each of the following models will be an extension of the previous model. For all cases, we will have the same data.

In our case, we have the following transportation network shown in Figure 4. The edges represent the distance (in days) between the two stations, and the figure at the vertex – the number of wagons the first day of planning.

We will answer a few questions about building a model:

- What is the purpose? – Maximization profit for cargo transportation.
- How is the solution? – The decision depends on the time of commencement of the application and the number of wagons.
- What are the limitations imposed on the model? – Number of wagons:
    - Do not exceed the total amount.
    - Should not be below a certain predetermined threshold at the station.
    - Should not be above a certain threshold at the station.

A. «With returns»

The first model is called «Order fulfillment with returning». Each order consists of two parts:

- Wagons are sent to the destination station.
- After that wagons will be sent to the original station.

Thus, each order is like a pendulum, has double length of time.

1) Decision variables

Decision variables of this model could be identified as two arrays of interval variables. First – Applications:

```
dvar interval app[a in OrderID] optional;
```

We point out the applications are not required. This is due to the fact that not all of them can be performed (total number of wagons in orders at one station may not exceed the number of wagons on this station). The following is a list of identification numbers of applications to find a solution among the set of alternative. As we pointed out above, we consider doubling time of the order:

```
dvar interval alt [o in allOrder] optional in
  o.order.StartData..(o.order.FinshData +
 o.place.Duration) size o.place.Duration * 2;
```

We include cumulative function to decision variables. It will take into account the number of wagons at the station in time:

```
cumulFunction count [l in Locations] = step(0,
 l.Units) - sum (ao in allOrder : l.Station ==
  ao.order.From) pulse (alt[ao], ao.count);
```

2) Objective function

Since the goal is to maximize profits for its implementation, we will summarize the number of wagons multiplied by travel time:

```
maximize sum (ao in allOrder) presenceOf(alt[ao])
        * ao.count * ao.place.Duration;
```

3) Constraints

This model is a classic example of vertical alternative:

```
forall (d in OrderID) alternative (app[d], all (ao
   in allOrder : ao.order.ID == d) alt[ao]);
```

Also, number of used wagons at any time shouldn't be less than zero:

```
forall (l in Locations) count[l] >= 0;
```

4) Result

This method finds the optimal solution if and only if all kinds of data are distributed evenly. If the number of applications for transport to a particular station is large enough, and the number of wagons on it is a little as we pointed out above, the decision will be far from optimal. But in the original sample data, the optimal solution was found quickly. We get a ready-formed schedule by day, which we could see at figure 5.

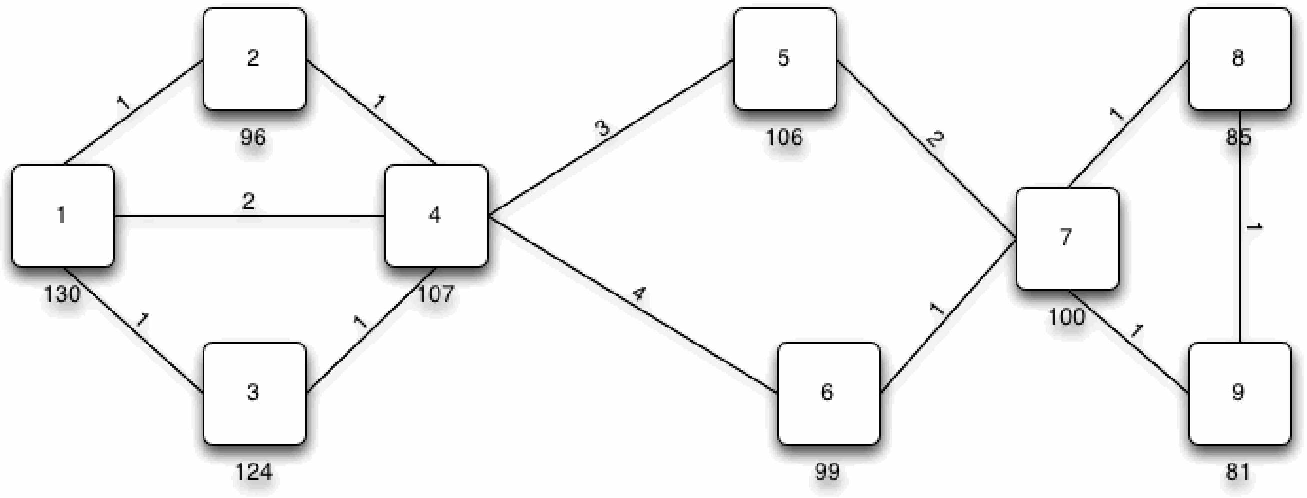
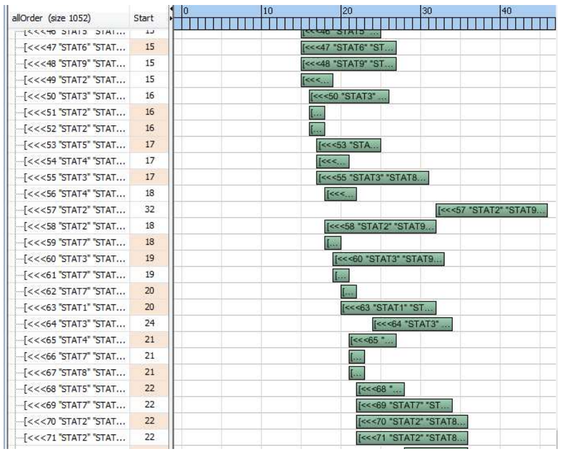


Fig 4. Example net



allOrder (size 1052)	Start
[<<<47 "STAT6" "STAT...	15
[<<<48 "STAT9" "STAT...	15
[<<<49 "STAT2" "STAT...	15
[<<<50 "STAT3" "STAT...	16
[<<<51 "STAT2" "STAT...	16
[<<<52 "STAT2" "STAT...	16
[<<<53 "STAT5" "STAT...	17
[<<<54 "STAT4" "STAT...	17
[<<<55 "STAT3" "STAT...	17
[<<<56 "STAT4" "STAT...	18
[<<<57 "STAT2" "STAT...	32
[<<<58 "STAT2" "STAT...	18
[<<<59 "STAT7" "STAT...	18
[<<<60 "STAT3" "STAT...	19
[<<<61 "STAT7" "STAT...	19
[<<<62 "STAT7" "STAT...	20
[<<<63 "STAT1" "STAT...	20
[<<<64 "STAT3" "STAT...	24
[<<<65 "STAT4" "STAT...	21
[<<<66 "STAT7" "STAT...	21
[<<<67 "STAT8" "STAT...	21
[<<<68 "STAT5" "STAT...	22
[<<<69 "STAT7" "STAT...	22
[<<<70 "STAT2" "STAT...	22
[<<<71 "STAT2" "STAT...	22

Fig 5. Scheduling

Although we found the optimal solution, this solution is not acceptable. We have to spend a certain amount on the empty wagons run to the station. Thus, the profit will be:

```
Profit = FullPrice – EmptyRun
```

This model, due to the rather large losses in empty run is not applicable in the real world; however, it is the launching pad for the next model.

B. *«Without returns»*

If in the previous model constructed after the wagons went back, it is logical to assume that these wagons can be left at the station of arrival. Thus, we will not spend their finances on the empty run.

1) *Decision variables*

Decision variables are changed in relation to the amended model. All alternatives will not have twice the range and will be similar to the input data:

```
dvar interval alt [o in allOrder] optional in
o.order.StartData..o.order.FinishData size
                o.place.Duration;
```

Also, the changes will affect the cumulative function of counting wagons at the station, as the wagons do not come back:

```
cumulFunction count [l in Locations] = step(0,
                    l.Units)
   - sum (ao1 in allOrder : ao1.order.From ==
l.Station) stepAtStart(alt[ao1], ao1.count)
    + sum (ao2 in allOrder : ao2.order.To ==
l.Station) stepAtEnd(alt[ao2], ao2.count);
```

We are using stepAtStart function considering that the part of wagons from current station run at the beginning of the application. It is a similar situation with stepAtEnd, when wagons come to station after fulfills the order.

The objective function and constraints in this case will not change as we change only the behavior of wagons during the execution of orders.

2) *Results*

Similarly to the previous model, the optimal solution will be found only when data have uniform distribution. For example, the solution was found, shown in Figure 6:

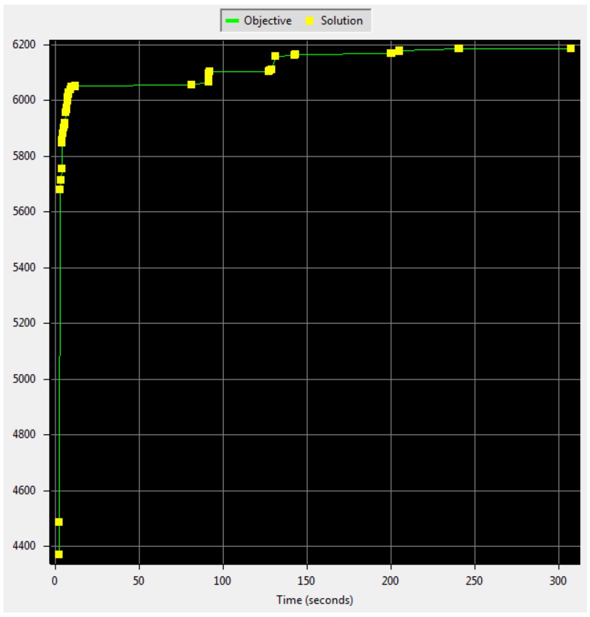


Fig 6. Optimization process

This solution is not optimal, but the ratio of the solution found to the absolute solution will be acceptable for cargo companies:

```
6185 / 6310 = 0,98019…
```

We will get 98% of the maximum possible profit. It can be considered an excellent option.

The drawback of the model is inability to drive wagons from nearby stations. If one station has a lot of applications, then the wagons don't come to the station. As a result, some orders will not be performed. Figure 7 shows the number of wagons at each station everyday. You may notice that the station 2, 5 and 7 have an application for a large number of departing wagons, so for these stations, the number of wagons is almost zero at the end of the month. In contrast, for example, for stations 1, 6 and 8, the number of wagons is increased by 1.5 or even 2 times.

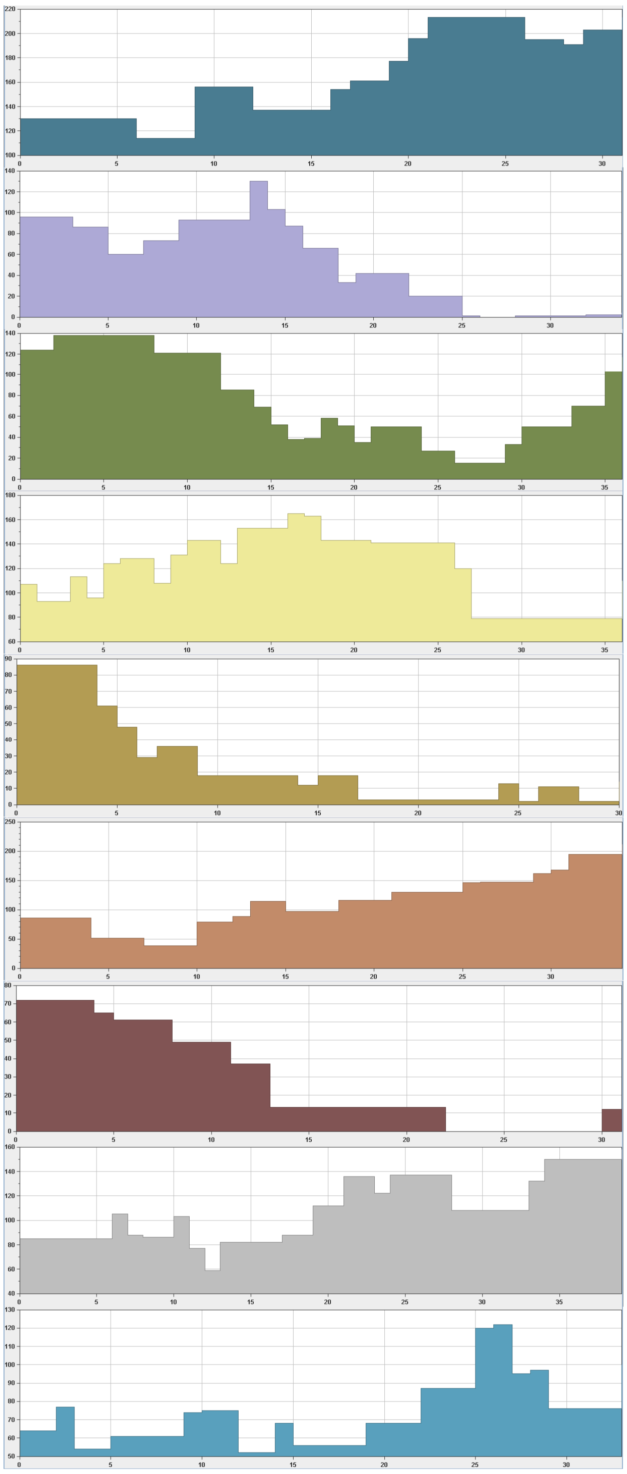


Fig 7. Number of wagons at stations after optimization

C. «*Additional empty run*»

If some station doesn't have enough wagons for the application, it is reasonable to drive the number of missing wagons from nearby stations. Maybe this will help even increase the objective function.

1) *Decision variables*

In this case, decision variables augmented with one more variable interval. It has the following type:

```
{EmptyRun} emptyRun = {<do, s, i> | do in detailOrder, s in Places, i in 0..do.order.Max : s.To == do.order.From && s.Time <= days};
```

Variables simulate all possible combinations of sending empty wagons from the neighboring stations. So we put the restriction that wagons can be sent from the stations, in which the stage is not greater than a special parameter.

```
dvar interval emptyRuns [o in emptyRun] optional in (o.order.StartDate - o.place.Duration >= 0 ? o.order.StartDate - o.place.Duration : 0).. (o.order.FinishDate - o.place.Duration) size o.place.Duration;
```

Cumulative function is changed to:

```
cumulFunction count [l in Locations] = step(0, l.Units)
    - sum (ao1 in allOrder : ao1.order.From == l.Station) stepAtStart(alt[ao1], ao1.count)
    + sum (ao2 in allOrder : ao2.order.To == l.Station) stepAtEnd(alt[ao2], ao2.count)
    - sum (er1 in emptyRun : er1.place.From == l.Station) stepAtStart(emptyRuns[er1], er1.count)
    + sum (er2 in emptyRun : er2.place.To == l.Station) stepAtEnd(emptyRuns[er2], er2.count);
```

We add a similar sum to account for incoming and outgoing wagons on the empty runs.

2) *Objective function*

All empty runs company pays for itself, so the empty run is subtracted from the profit:

```
maximize sum (ao in allOrder) presenceOf(alt[ao]) * ao.count * ao.place.Duration * FullPrice
    - sum (er in emptyRun) presenceOf(emptyRuns[er]) * er.count * er.place.Duration * EmptyPrice;
```

Add the following parameters: price for transportation cargo and price for empty run. Thus the objective function is an order of magnitude higher than the result of the previous examples.

3) *Constraints*

There are reserve wagons at the station, we make a new restriction. We establish that the number of cars can be in the range [20; 160] at the station at any time:

```
forall (l in Locations) {
    alwaysIn(count[l], 0, 60, 20, 160);
}
```

4) *Results*

This solution is closer to the optimal value, even with restrictions on the number of wagons:

```
311380 / 315500 = 0,98694…
```

This model does not depend on the location of wagons and we can just say that all orders will be fulfilled. It is as close to real use.

V. CONCLUSIONS

This article demonstrates the easy modeling of though simplified, but at the same time, the real problem. We considered several models and provides a comparative analysis. This is just the first step of constructing solutions of the system response and planning. This IBM's tool allows one to quickly build a model of a problem and get an optimal answer. In the future, these developments will be used for detailing that problem. In particular, several large companies are already interested in this. Of course IBM ILOG Cplex Optimization Studio is not designed for building large applications. The next step is to build an application based on the product IBM ILOG Optimization Decision Manager Enterprise, which is precisely designed for that purpose.

VI. REFERENCES

[1] Yudin, DB and Holstein, EG (1961). Objectives and methods of linear programming. Publ. “Soviet Radio”.

[2] The transportation planning process: key issues. A briefing book for transportation decisionmakers, officials and staff.

[3] Avdoshin S., Gorbatovskiy M., Chernov A., “The Concept of intellectual situational response system and railways safety of modern Russia”, Business Informatics №4 (18), 2011.

[4] Laporte, G. “An overview of transportation planning problem. HEC Montreal, Chaire de Recherché du Canada en Distributique“.

[5] Model development with IBM ILOG CPLEX Optimization Studio V12.2.

[6] Detailed scheduling in IBM ILOG OPL with IBM ILOG CP Optimizer.