ON LOCAL SEARCH FOR MILP SOLVER PARAMETERS OPTIMIZATION

A.V.Eremeev, V.N.Ustyugov, L.A.Zaozerskaya
OD of IM SB RAS

The research was supported by Russian Science Foundation grant N 22-71-10015, https://rscf.ru/en/project/22-71-10015/.

Motivation

Many high-performance algorithms as well as commercial solvers have many parameters whose settings control important aspects of their behavior. Those parameters can be set by user to improve solver's or algorithm's performance. The task of finding best parameter configuration is complicated and requires considerable effort from researchers. Usually, this was done for a specific problem classes manually, with high time consumption.

Our main goal is to find a way to automatize this task for modern MIP solvers.

Approaches

There are different approaches to this problem:

- Search methods, applied over parameter space of the solver. Namely, local search, Variable neighborhood descent (VND), ParamILS (Hutter et al, 2009);
- Machine learning methods, such as reinforcement learning and its subtypes;
- Built-in tools, such as Gurobi Tuning tool or CPLEX Tuning Tool.

Local Search Methods under Consideration

1. Local search performed on each problem instance independently, then a certain configuration is chosen by voting between present configurations for each instance.
2. ParamILS is an algorithm for tuning parameters of solvers (Hutter et al, 2009), it is a black-box technique. The idea is to perform a local search in parameter space, with a chance for perturbation - move to random point to leave current possible local minimum.
3. Variable Neighborhood Descent (VND) with the same objective as the latter.

Vehicle Routing Problem Statement

The model was proposed in (Borisovsky et al., MOTOR-2021) and is analogous to the model, proposed in (Kulachenko, Kononova, MOTOR-2020).

This model is based on MIP model for VRP.

The task is to find optimal route for rigs to drill all wells on all drilling sites.

Vehicle Routing Problem Statement

- Suppose the set of sites $\boldsymbol{I} = \{\boldsymbol{i_1}, \ldots, \boldsymbol{i_{|I|}}\}$ should be served by vehicles from a set $\boldsymbol{U} = \{\boldsymbol{u_1}, \ldots, \boldsymbol{u_{|U|}}\}$.
- For each site $\boldsymbol{i} \in \boldsymbol{I}$, a time interval $(\boldsymbol{a_i}, \boldsymbol{b_i}]$ is given for its service.
- Each vehicle $\boldsymbol{u}$ is located initially in its depot $\boldsymbol{id_u} \in \boldsymbol{I}$.
- The travelling time between sites $\boldsymbol{i}, \boldsymbol{j}$ for vehicle $\boldsymbol{u}$ is $\boldsymbol{s_{uij}}$.

Let us introduce the variables:

- $\boldsymbol{x_{uij}} \in \{0,\ 1\}$ indicates visiting site $\boldsymbol{i}$ by vehicle $\boldsymbol{u}$ and proceeding to site $\boldsymbol{j}$;
- $\boldsymbol{t^s_{ui}} \in \mathbb{R}^+$ is the time of visiting site $\boldsymbol{i}$ by vehicle $\boldsymbol{u}$.

Minimization of the total travelling time:

$$\sum_{u \in U} \sum_{i \in I} \sum_{j \in I} \boldsymbol{s_{uij} x_{uij}} \rightarrow \mathbf{min} \qquad (1)$$

Vehicle Routing Problem Statement

Balance constraint for each site:

$$\sum_{j \in I} x_{uij} = \sum_{j \in I} x_{uji}, \; u \in U, \; i \in I \setminus \{id_u\}, \qquad (2)$$

Each site is visited by one vehicle:

$$\sum_{u \in U} \sum_{i \in I} x_{uij} = 1, \; j \in I, \qquad (3)$$

Travelling time estimate:

$$t^s_{ui} + s_{uij} \leq t^s_{uj} + b_{\max}(1 - x_{uij}), \;\; i \neq j \in I, \; u \in U, \qquad (4)$$

Time windows constraint:

$$\sum_{j \in I} a_i x_{uji} \leq t^s_{ui} \leq \sum_{j \in I} b_i x_{uji}, \; i \in I, \; u \in U, \qquad (5)$$

Route start for each vehicle:

$$\sum_{i \in I} x_{u,id_u,i} = 1, \; u \in U. \qquad (6)$$

Computational experiment. Setup.

We’ve considered the tasks that have 30 or 50 sites. The experiment was set up on the server of OD of IM SB RAS, powered by two AMD EPYC processors (2x64 threads, 512 GB RAM).

While tuning, each instance was given 1000 seconds to solve. Overall tuning time was limited to 20 hours for ParamILS and VND methods. One local search iteration took 5400s, 2 to 4 iteration per tuning.

Upon completion of parameters optimization, we ran the solver on all instances with 3-hour time limit per instance. The results are in the table below.

Local search results on each instance and voting result

Instance number	threads	presolve	gomorypasses	method	mipfocus
1	8	-1	1	-1	0
2	8	1	0	-1	0
3	8	-1	2	2	0
4	8	1	0	-1	0
5	8	-1	5	-1	0
6	8	-1	1	2	0
7	12	-1	5	2	1
8	12	2	0	0	2
9	8	1	5	1	0
10	4	-1	0	-1	0
Final voting	8	-1	2	-1	0

Computational experiment. Results.

average	Default	GTT	LS	ParamILS	VND
f_{LP}	27,582	27,582	27,582	27,582	27,582
f_{best}	42.8	42.8	**42.6**	43.2	42.8
LB	40.83	41.18	**41.36**	39.596	40.052
relative gap	0.046	0.038	**0.031**	0.083	0.063
time, s	5203	3746	3779	8537	7585

Conclusions

- Local Search with parameters voting has shown the best performance in terms of relative gap.
- ParamILS (Hutter et al, 2009) is inferior to VND on our testing data due to unnecessary random moves, which distract it on the way to the local optimum.
- Capping technique (Hutter et al, 2009), i.e. early stopping of configuration testing, might boost the performance of tuning procedure (further research).
- Solver performance prediction might be a good guide in local search, because it might save the CPU time significantly (further research).

Thank you for your attention!

The Local Search

K is the number of parameters to be set; *P = {1,...,K}*;

{1,…,T_k} is a set of values for parameter *k*;

f is the objective function value found by a solver

Bound is a lower bound found by solver (assuming minimization problem)

F_{iter}, B_{iter}, the best objective function value and lower bound for set parameter order

F_{min}, *BestBound* are the best found values respectively

Value(k) - current value for parameter *k*

BestValueIter(k), BestValue(k)

The Local Search

```
Fmin = 1e6, BestBound = 0, BestVaue(k)=Default for each k
Repeat:
  generate vector PN = (PN(1),...,PN(k)) - parameter assignment order
  Value(k) = default, BestValueIter = default for each k
  F_iter = 1e6, B_iter=0
  for k = 1, …, K:
      for t = 1, …, T_PN(k):
        Value(PN(k)) = t
        f, Bound ← solve model and obtain objective function value and bound
        if f < F_iter or (f = F_iter and Bound > Biter):
              F_iter := f
              B_iter := Bound
              BestValue_Iter(PN(k)) = t
      Value(PN(k)): = BestValue_Iter(PN(k))
  update F_min, BestBound and BestValue(k) according to obtained results
Repeat until (F_iter = F_min and B_iter < BestBound)
```

ParamILS

Input : Initial configuration $\theta_0 \in \Theta$, algorithm parameters r, $p_{restart}$, and s.

Output : Best parameter configuration θ found.

for $i = 1, \ldots, r$ **do**

  $\theta \leftarrow$ random $\theta \in \Theta$;

  if better(θ, θ_0) **then** $\theta_0 \leftarrow \theta$;

$\theta_{ils} \leftarrow$ *IterativeFirstImprovement* (θ_0);

while not *TerminationCriterion()* **do**

  $\theta \leftarrow \theta_{ils}$;

  // ===== Perturbation

  for $i = 1, \ldots, s$ **do** $\theta \leftarrow$ random $\theta' \in Nbh(\theta)$;

  // ===== Basic local search

  $\theta \leftarrow$ *IterativeFirstImprovement* (θ);

  // ===== AcceptanceCriterion

  if better(θ, θ_{ils}) **then** $\theta_{ils} \leftarrow \theta$;

  with probability $p_{restart}$ **do** $\theta_{ils} \leftarrow$ random $\theta \in \Theta$;

return overall best θ_{inc} found;

Procedure *IterativeFirstImprovement* (θ)

repeat

  $\theta' \leftarrow \theta$;

  foreach $\theta'' \in Nbh(\theta')$ *in randomized order* **do**

    if better(θ'', θ') **then** $\theta \leftarrow \theta''$; **break;**

until $\theta' = \theta$;

return θ;

VND

Variable Neighbourghood Descent Algorithm (VND)

```
x is the initial solution for VND
While (no final condition) do
        u = 1
        While (u ≤ u_max) do
                x′ is the best solution in N_u(x)
                If f(x′) < f(x) then
                        x = x′ and u = 1
                else
                        u=u+1
                end if
        end while
end while
Return best found solution
```

Computational experiment. Parameters

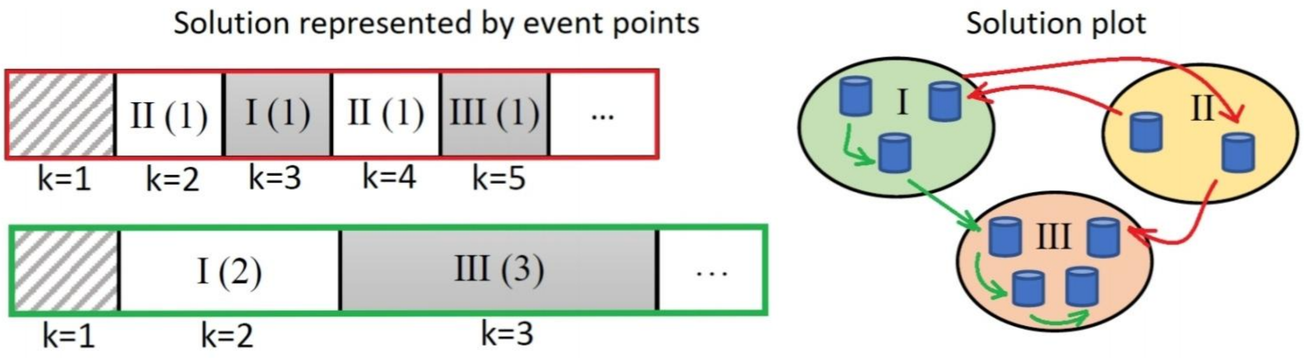
	NSK	NSK-DSM	4 threads	LS	ParamILS	VND
threads	8	8	4	8	8	12
presolve	2	2	-	-1	1	1
gomorypasses	0	0	-	0	2	5
method	0	1	-	-1	3	5
minrelnodes	10627	10627	-	-	0	0
mipfocus	-	-	-	0	0	0
improvestartime	8640	8640	-	-	-	-

Rig Routing Problem Statement

Event Points ↔ Positions in routes.

Structural variables:

- $\boldsymbol{x_{uik}} \in \{0, 1\}$ indicate that rig $\boldsymbol{u}$ visits site $\boldsymbol{i}$ in event point $\boldsymbol{k}$;
- $\boldsymbol{y_{uik}} \in \mathbb{Z}^+$ is the number of wells of site $\boldsymbol{i}$ drilled by rig $\boldsymbol{u}$ in event point $\boldsymbol{k}$.



Example of solution representation for a problem with sites I, II and III

Rig Routing Problem Statement

Assign event points to site visits

$$\sum_{i \in I_u} x_{uik} \leq 1,\ u \in U,\ k \in K, \tag{7}$$

$$\sum_{i \in I_u} x_{u,i,k-1} \geq \sum_{i \in I_u} x_{uik},\ u \in U, k \in K, k > 1, \tag{8}$$

$$1 \leq \sum_{u \in U_i} \sum_{k \in K} x_{uik} \leq m_i,\ i \in I. \tag{9}$$

Drill all wells of all sites

$$\sum_{u \in U_i} \sum_{k \in K} y_{uik} = n_i,\ i \in I, \tag{10}$$

$$y_{uik} \geq x_{uik}, \tag{11}$$

$$y_{uik} \leq n_i x_{uik},\ i \in I,\ u \in U_i,\ k \in K. \tag{12}$$

Rig Routing Problem Statement

Starting times (variables t^s_{uk}) of drilling:

$$t^s_{uk} \geq t^s_{u,k-1} + \sum_{i \in I_u} d_{ui} y_{ui,k-1} + t_{uk} - b_{\max} \left(1 - \sum_{i \in I_u} x_{uik} \right), \quad (13)$$

here variables z_{uk} give the travel time between event points $k-1$ and k.

$$z_{uk} \geq \sum_{i \in I_u} s_{uij} x_{u,i,k-1} - s_{\max} \left(1 - x_{ujk} \right). \quad (14)$$

Time windows

$$t^s_{uk} + \sum_{i \in I_u} d_{ui} y_{uik} \leq \sum_{i \in I_u} b_i x_{uik}, \quad (15)$$

$$t^s_{uk} \geq \sum_{i \in I_u} a_i x_{uik}, \; u \in U, \; k \in K. \quad (16)$$

Objective function

$$f = \sum_{u \in U} \sum_{k \in K, \; k > 1} z_{uk}. \quad (17)$$