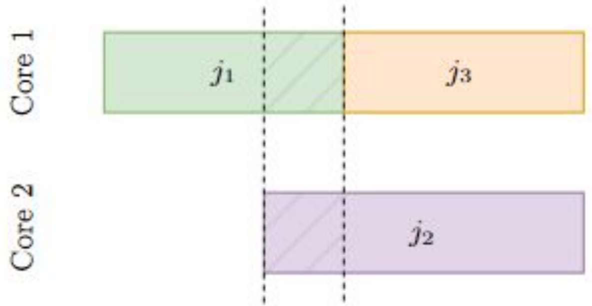
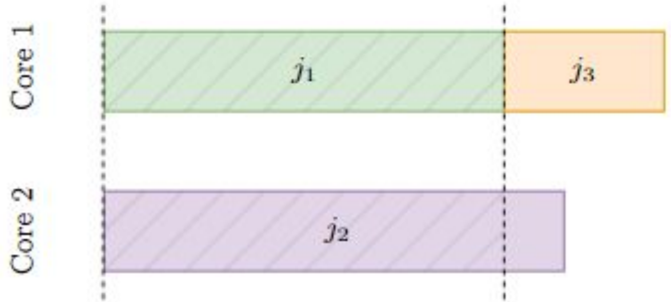
On Different Methods for Automated MILP Solver Configuration

Ustyugov V.N., Sobolev Institute of Mathematics, Siberian branch of Russian Academy of Sciences, Omsk

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

Base task





Base task

$$C_{\max} = \sum_{k\in K}\sum_{c\in C} t_{kc} \to \min, \tag{1}$$

$$t_{kc} \geqslant 0, \qquad k \in K,\ c \in C, \tag{2}$$

$$t_{kc} \leqslant x_{kc} T_{\max}, \qquad k \in K,\ c \in C, \tag{3}$$

$$\sum_{c\in C} x_{kc} = 1, \qquad k \in K, \tag{4}$$

$$\sum_{k\in K}\sum_{c\in C} t_{kc} v_{jc} = s_j, \qquad j \in J, \tag{5}$$

$$\sum_{c\in C} x_{kc} q_{jc} - \sum_{c\in C} x_{k-1,c} q_{jc} \leqslant y_{jk}, \qquad j \in J,\ k \in K, \tag{6}$$

$$\sum_{k\in K} y_{jk} = 1, \qquad j \in J, \tag{7}$$

$$\sum_{k\in K} (k+1) x_{kc_1} \leqslant (1 - \sum_{k\in K} x_{kc_1}) e + \sum_{k\in K} k x_{kc_2}, \quad c_1, c_2 \in C,\ a_{c_1 c_2} > 0, \tag{8}$$

$$x_{kc} \in \{0, 1\},\ y_{jk} \in \{0, 1\}, \qquad j \in J,\ k \in K,\ c \in C. \tag{9}$$

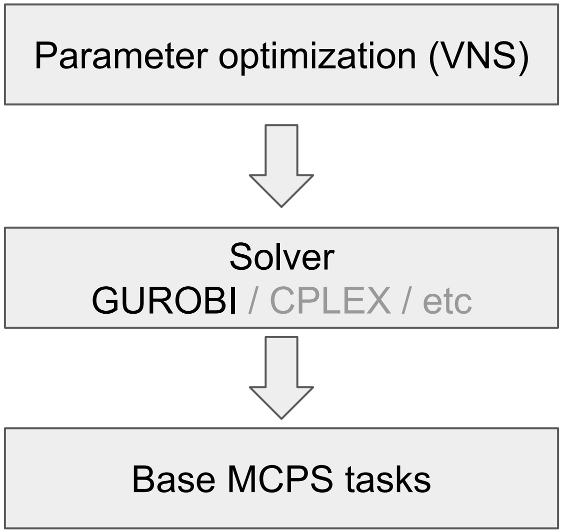
Base task

1. q_{jc} = 1 when and only when job j is included in configuration c, 0 else.
2. $a_{c_1c_2}$ = 1 when and only when configuration c_2 must be performed after configuration c_1, 0 else.
3. T_{max} — upper bound of configuration runtime in event point.

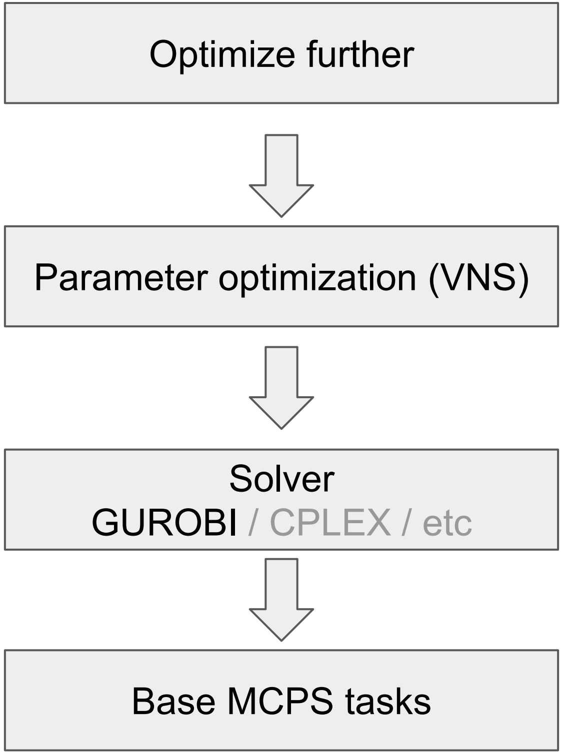
Let $K = \{0, 1, 2, . . . , e\}$ be a set of event points, where e is a maximum event point, so we add the following variables:

1. t_{kc} — runtime of configuration c event point k.
2. x_{kc} = 1 when and only when, configuration c is performed in event point k, 0 else. Let us assume that, zero configuration is performed in zero event point: $x_{00} = 1$ и $x_{0c} = 0$, $c \in C$.
3. y_{jk} = 1 when and only when job j starts in event point k, 0 else.

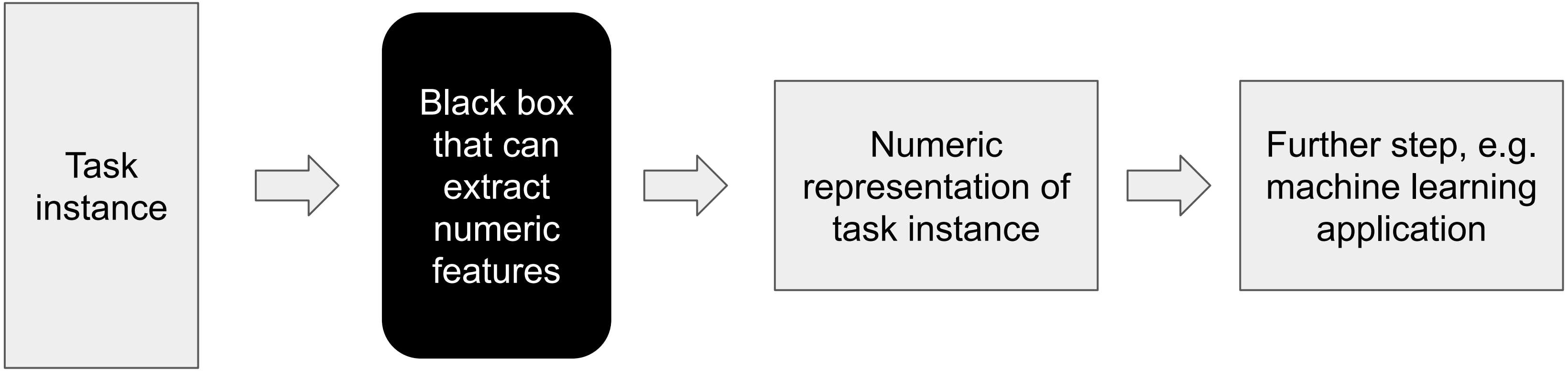
Optimization process



Optimization process



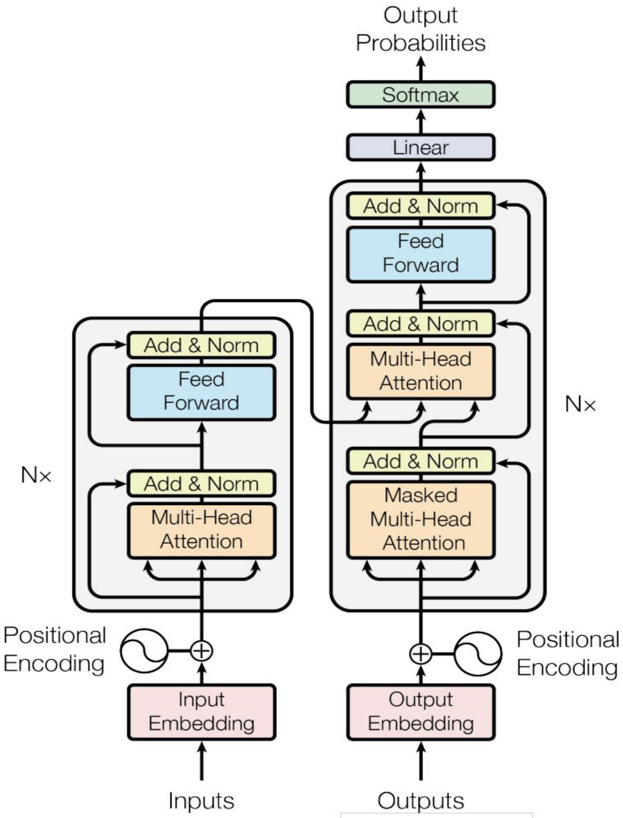
Optimization process



LLM (Large Language Models)

BERT

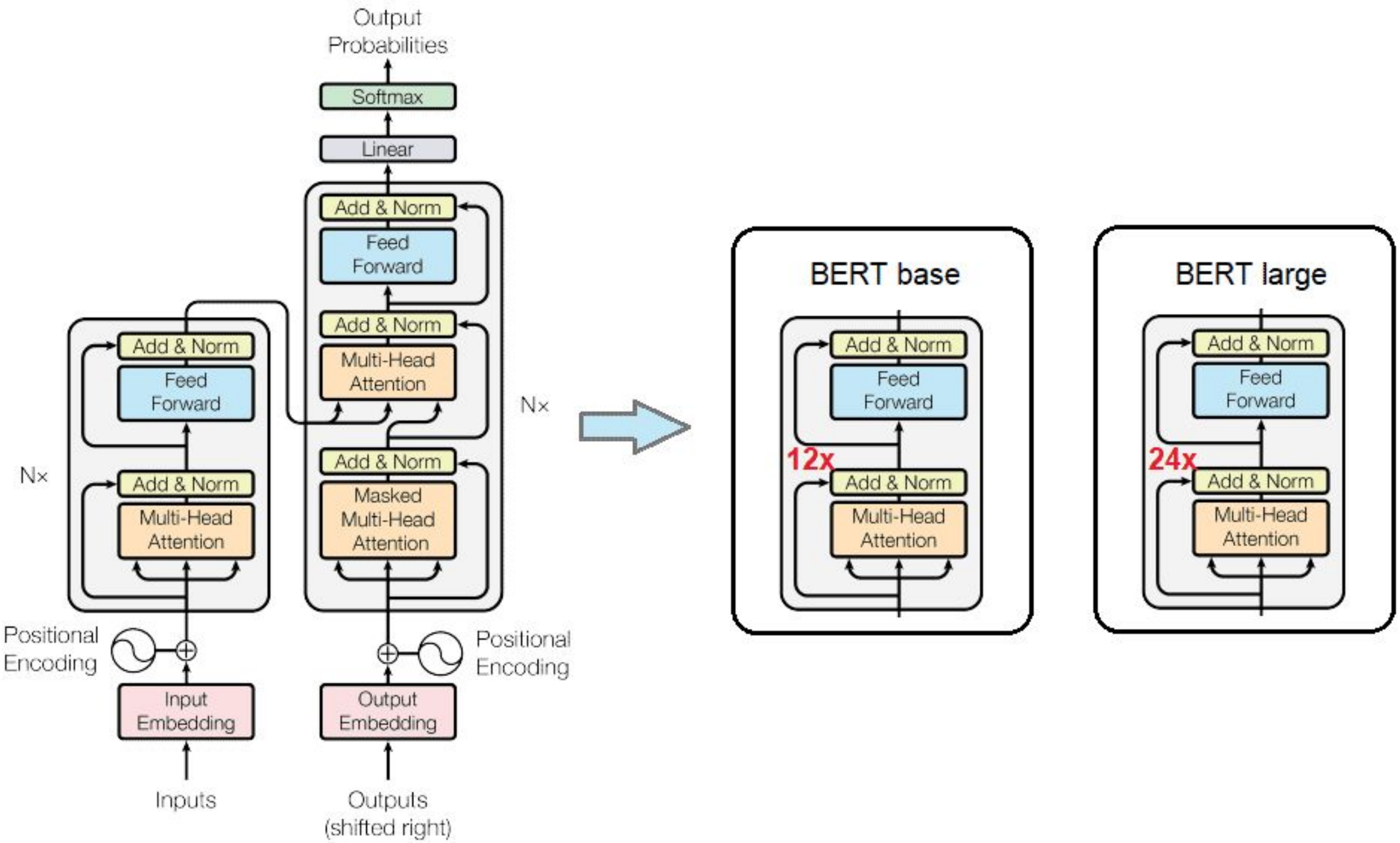
Encoder



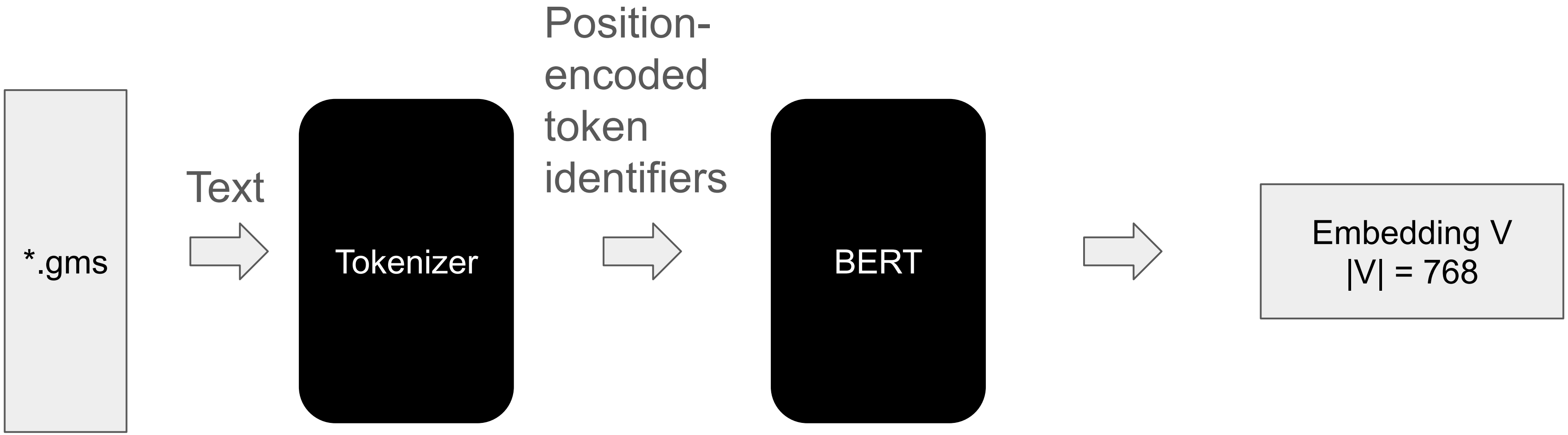
GPT

Decoder

BERT(Bidirectional Encoder Representations from Transformers)



BERT(Bidirectional Encoder Representations from Transformers)



Where do I pass embeddings?

$P = (p_1,...,p_k)$ - solver's parameter vector (*configuration*)

Regression task:

$\mathbf{P_i^*=V_iW}$, $i = 1,...,J$, where J is a number of different individual tasks

V_i = BERT(Tokenizer(I_i))

Experimental data

Dimensionality	Optimization type		
	No	VNS	LLM
4 jobs	0.0701 s	0.0698 s	0.0695 s
7 jobs	117,7 s	–	117,04 s

Table 2. Average runtime

Conclusions

- Since the process of obtaining embeddings and training linear regression was performed only for tasks with 4 jobs the unquestionable improvement was obtained only on this dimensionality;
- Bad generalization is seen on higher dimensionalities, probably because it’s interpolation so it might be inaccurate;
- Assumption: for the purpose of increasing prediction quality it’s best to “show” some amount of high-dimensional tasks, so the linear model can “extrapolate” the knowledge on in-between dimensionalities;
- Going for more parameters will make us face imminent curse of dimensionality, which is open problem in this case, since there are not so many task instances available;
- There is another way of obtaining vector representation, introduced in the paper about MIPLIB [3], those representations formed by looking inside of tasks during runtime and may be useful in our case.

Thanks for your attention!

References:

1. А.В.Еремеев, М.Ю.Сахно, *Построение расписания для многоядерного процессора с учетом взаимного влияния работ*
2. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
3. Gleixner, A., Hendel, G., Gamrath, G. et al. *MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library*.

vyacheslav.nikolayevich.ustyugov@gmail.com