Logistics optimization of slab relocation problem in the steel industry

Reporter: Peixin Ge
Supervisor: Lixin Tang

The Logistics Institute, Northeastern University, Shenyang, China
http://tli.neu.edu.cn
1. Slab relocation problem (SRP)

2. Model for SRP

3. Classification of stacks and relocations

4. Two tree search methods for the SRP

5. Computational experiments

6. Conclusions
Slab relocation problem

Background:

With the development of logistics technology in recent years, the efficiency of logistics operations becomes an effective means to reduce the logistics cost in the steel industry. SRP is happening in slab warehouse of the steel industry.
A slab warehouse serves as a storage buffer between the continuous casting stage and the reheat furnace. Slabs from continuous casting are stacked in the warehouse to await rolling. When implementing the rolling production schedule, slabs need to be picked up from the slab warehouse one by one according to the scheduled rolling sequence, heated and then rolled.
What is the SRP?

SRP can be formulated as follows: we are given $S$ stacks and each stack has $H$ slots. A distribution of $N$ slabs of $G$ groups to $S$ stacks is called an initial layout. The slabs are usually divided into groups in accordance with the steel grade and width of slabs. To make the disorganized layout regular, slabs of the same group should be piled up in the least stacks. The operation that relocated slab is called a relocation.

What is “regular”?

1) The “unification” demand:
If slabs of a stack are all the same group, we say that the stack satisfies the “unification”
2) The “concentration” demand:
But the slab group may be still not regular, for example the group has 33 slabs in 5 stacks and a stack contains 15 slabs at most. For saving two stacks to store other slabs, we should put the slabs of this group in three stacks. We call the demand of slabs “concentration”.
The regular layout has three advantages for practical production:

1) It makes the import operation easier. The crane transports the import slab of group y to the suitable stack which only contains slabs of the same group. If not, it’ll make the layout more disorganized.

2) It reduces the cost of export operation. If a slab of group g is to be taken out of warehouse, all the slabs above it have first to be transferred to other stacks. If the layout is regular, the stack will only contain the slabs of group g, we can just use the top slab without relocation.

3) It can ensure the safety of the import and export operations. The stack which contains slabs of different widths is easier to collapsing during some operations or to other stacks. If the layout is regular, the stack will only contain the slabs of group g, we can just use the top slab without relocation.
Slab relocation problem

Review the relevant literature.

See Tang et al. (2002) for a slab shuffle problem in a steel slab warehouse.

Watanabe (1991) suggested an accessibility index method to estimate the number of relocations in container terminals. Kim (1997) also proposed a formula to estimate the number of relocations for import containers and showed that his method is better than Watanabe's (1991) method in the accuracy.

To minimize the number of relocations, Kim et al. (2000) suggested a model and a dynamic programming technique for locating export containers. They assumed that the containers are classified into three groups, according to their weight. Thus, when containers arrive at the yard, placing heavier containers onto higher tiers will reduce the expected number of relocations. Kim et al. suggested decision trees for locating export containers, the case in their study was a special one which had three groups of blocks. The concept of group is similar to the group of slabs. And we refer to the idea of decision trees and propose the tree search methods for SRP.
The **container pre-marshalling problem (CPMP)** is fairly similar to SRP. The CPMP is to premarshal the containers according to the pickup sequence, resulting in a layout that would allow the containers to be removed without any further relocation. Algorithms for the CPMP were developed by Bortfeldt (2004), Lee and Hsu (2007), Lee and Chao (2009), Caserta and Voß (2009) and recently by Bortfeldt and Forster (2012).
Slab relocation problem

Model for SRP:

Variables:

L: A layout may be represented by a matrix $L$ with $H$ rows and $S$ columns. 

$(i, j)$: layer $i = 1, \ldots, H$; stack $j = 1, \ldots, S$.

G: The matrix $L$ assigns a group index $g$ ($1 \leq g \leq G$) to each slot.

N: The initial layout contains $N$ slabs, labeled $1, \ldots, N$.

T: The time horizon $t = 1, \ldots, T$, where each time period $t$ is defined by a single move. 

\hspace{1cm} T is a constant got by a greedy heuristic

$h_{ti}$: The height of stack $i$ in time period $t$.

$g_{n}$: The group number of the slab $n$

$R_{g}$: The number of regular stacks which only contain slabs of group $g$ and satisfy the demand of concentration.

M: A big enough constant.
Slab relocation problem

Decision variables:

\[ x_{tijn} = \begin{cases} 1, & \text{if slab } n \text{ is in slot } (i,j) \text{ in time period } t, \\ 0, & \text{otherwise}; \end{cases} \]

The first one aims at defining feasible configurations.

\[ z_{tijkln} = \begin{cases} 1, & \text{if slab } n \text{ is relocated from slot } (i,j) \text{ to slot } (k,l) \text{ in time period } t \text{ and } i \neq k, \\ 0, & \text{otherwise}; \end{cases} \]

The second is used to define feasible moves.

\[ y_{ti} = \begin{cases} 1, & \text{if stack } i \text{ is disorganized in time period } t, \\ 0, & \text{otherwise}; \end{cases} \]

We figure out how many stacks are disorganized in time period by the third variable.

\[ w_{tig} = \begin{cases} 1, & \text{if stack } i \text{ is the stack which only contains the slabs of group } g \text{ in time period } t, \\ 0, & \text{otherwise}; \end{cases} \]

The fourth variable ensures that the regular stacks satisfy the demand of concentration.
Slab relocation problem

The penalty for the number of relocations

\[ \text{Min: } \lambda_1 \sum_{t=1}^{T} \sum_{i=1}^{S} \sum_{j=1}^{H} \sum_{k=1}^{S} \sum_{l=1}^{H} \sum_{n=1}^{N} z_{tijkln} + \lambda_2 \sum_{t=1}^{T} \sum_{i=1}^{S} y_{ti} + \lambda_3 \sum_{t=1}^{T} \sum_{g=1}^{G} \sum_{i=1}^{S} w_{tig} - R_g \]

\[ s.t. \]
\[ \sum_{i=1}^{S} \sum_{j=1}^{H} x_{tijn} = 1, \quad t = 1, 2, ..., T; \quad n = 1, ..., N \]
\[ \sum_{n=1}^{N} x_{tijn} \leq 1, \quad t = 1, 2, ..., T; \quad i = 1, 2, ..., S \]
\[ \sum_{n=1}^{N} x_{tij+1,n} - \sum_{n=1}^{N} x_{tijn} \leq 0, \quad t = 1, 2, ..., T; \quad i = 1, ..., S; \quad j = 1, ..., H \]
\[ \sum_{i=1}^{S} \sum_{j=1}^{H} \sum_{k=1}^{S} \sum_{l=1}^{H} z_{tijkln} \leq 1, \quad t = 1, 2, ..., T \]
\[ \sum_{i=1}^{S} \sum_{j=1}^{H} x_{t+1,ijn} + \sum_{i=1}^{S} \sum_{j=1}^{H} x_{tijn} \leq 2 - \sum_{i=1}^{S} \sum_{j=1}^{H} \sum_{k=1}^{S} \sum_{l=1}^{H} z_{tijkln} \]

- Ensure that each slab must be within one slot in each time period.
- Ensure that each slot (i, j) must be occupied by at most one slab in each time period.
- Ensure that no gaps are allowed within each stack.
- Ensure that at most one move is allowed in each time period.
- If slab n is relocated in time period t, ensure that the slab in time period t+1 is not in the former slot.
Slab relocation problem

\[ \sum_{j=1}^{H} x_{t+1,ijn} \geq \sum_{j=1}^{H} x_{tijn} - \sum_{j=1}^{H} \sum_{k=1}^{S} \sum_{l=1}^{H} z_{tijkln}, \quad t = 1, 2, \ldots, T; n = 1, \ldots, N; i = 1, \ldots, S \]  
(7)

\[ \sum_{j=1}^{H} x_{t+1,ijn} \leq \sum_{l=1}^{H} x_{tijn} + \sum_{j=1}^{H} \sum_{k=1}^{S} \sum_{l=1}^{H} z_{tijkln}, \quad t = 1, 2, \ldots, T; n \]

Ensure that slab n is still in the former slot in time period t+1.

\[ \sum_{j=1}^{H} \sum_{k=1}^{S} \sum_{l=1}^{H} z_{tijkln} \leq \sum_{j=1}^{H} x_{tijn}, \quad t = 1, 2, \ldots, T; n = 1, \ldots, N; i = 1, \ldots, S \]  
(9)

\[ \sum_{i=1}^{S} \sum_{j=1}^{H} \sum_{l=1}^{H} z_{tijkln} \leq \sum_{l=1}^{H} x_{t+1,kl}, \quad t = 1, 2, \ldots, T; n = 1, \ldots, N; k = 1, \ldots, S \]  
(10)

\[ \sum_{j=1}^{H} \sum_{l=1}^{H} z_{tijkln} \geq \sum_{j=1}^{H} x_{tijn} + \sum_{l=1}^{H} x_{t+1,kl} - 1, \quad t = 1, 2, \ldots, T; n = \]

Define the decision variable \( z_{tijkln} \).

\[ \sum_{j=1}^{h_{ji}} \sum_{n=1}^{N} g_{n} \cdot x_{tijn} - \sum_{n=1}^{N} g_{n} \cdot x_{t+1,ln} \geq y_{ti}, \quad t = 1, 2, \ldots, T; i = 1, 2, \ldots, S \]  
(12)

\[ y_{ti} \geq \left| \sum_{j=1}^{h_{ji}} \sum_{n=1}^{N} g_{n} \cdot x_{tijn} - \sum_{n=1}^{N} g_{n} \cdot x_{t+1,ln} \right| / M, \quad t = 1, 2, \ldots, T; i = \]

Define the decision variable \( y_{ti} \).
Slab relocation problem

\[
\sum_{n=1}^{N} (g_n \cdot x_{t_{i1n}}) \cdot (1 - y_{ti}) - g \leq (1 - w_{tig}) \cdot M, \quad t = 1, 2, \ldots, T; i = 1, \ldots, S; g = 1, \ldots, G \quad (14)
\]

\[
1 - w_{tig} \leq \sum_{n=1}^{N} (g_n \cdot x_{t_{i1n}}) \cdot (1 - y_{ti}) - g \leq M, \quad t = 1, 2, \ldots, T; i = 1, \ldots, S; \quad \text{Define the decision variable } w_{tig}.
\]

\[
x_{1ijn} = x_{0ijn}, \quad i = 1, \ldots, S; j = 1, \ldots, H; n = 1, \ldots, N
\]

\[
x_{tijn} \in \{0, 1\}, \quad t = 1, 2, \ldots, T; i = 1, \ldots, S; j = 1, \ldots, H; n = 1, \ldots, N \quad (17)
\]

\[
y_{ti} \in \{0, 1\}, \quad t = 1, 2, \ldots, T; i = 1, \ldots, S \quad (18)
\]

\[
w_{tig} \in \{0, 1\}, \quad t = 1, 2, \ldots, T; i = 1, \ldots, S; g = 1, \ldots, G \quad (19)
\]

\[
z_{tijkln} \in \{0, 1\}, \quad t = 1, 2, \ldots, T; i = 1, \ldots, S; j = 1, \ldots, H; k = 1, \ldots, S; l = 1, \ldots, H; n = 1, \ldots, N \quad (20)
\]

Finally, the integrality conditions on the decision variables are specified by Eqs. 17-20.
Classification of stacks and relocations

1) Stacks:

- Final receiver stack (g)
- Prepared stack (g)
- Empty stack
- Empty slot
- Prepared stack (g)
- Surplus stack (g)

**Examples:**
- Slab num: 37
- Max height: 15
- Stored stacks: 4
- Need stacks: 3
- Surplus stacks: 1

The five stacks are regular, but in fact, the surplus stack is not. It's a stack of group g. The way we keep the receiving stack of group g is by keeping it as empty. At the end of this relocation process, the surplus stack will become the empty stack to store slabs of other group.

By using ILOG OPL 5.5, we are able to solve small instances of the Slab Relocation Problem. However, due to the exponential growth in computation time, this approach seems to be not applicable to most real-world scenarios. So we propose the tree search methods to solve the practical problem faster and more efficiently.

The five stacks are regular, but in fact, the surplus stack is not. It's a stack of group g. The number of group g is 37, there are four stacks storing slabs of group g, the maximum height of stack is 15. By computation, we just need 3 stacks to store these slabs. There is a surplus stack. It will become the empty stack to store the slabs of other group.

When the stacks of group g are regular, it will be the prepared stack or unrelated stack. The unrelated stack is full, we don't operate it any more. The prepared stack is prepared to receive the slabs of other group.

For example, the slabs of group h in the disorganized stack are relocated to the prepared stack. It will become the temporary stack to store the slabs temporarily. Until we get a new receiver stack of group h, then put these slabs to their receiver stack.

Of course the disorganized stack is the target to be relocated. We should relocate the slabs to the receiver stack, then the disorganized stack will become the temporary stack.
The Logistics Institute

Slab relocation problem

We proposed the cost of slab to measure the regularity of a layout according to the classification of stack.

The cost of slab of group g in:
- The final receiver stack: 1
- The surplus stack: 1
- The unrelated stack: 0
- The prepared stack: 0
- The temporary stack: if prepared 0 else 1
- The disorganized stack: 5

On the contrary, the slabs in the disorganized stack and the temporary stack have high cost.

The cost of slab of group h in the temporary stack:
If the slab have a final receiver stack: 10, If not: 15
The cost of slab of group h in the disorganized stack:
If the slab have a final receiver stack: 15 minus i, If not: 30 minus i

Number i is equal to the number of slabs of this stack minus the layer of the slab.

From this picture, we can know that the higher slabs in the disorganized stack, the more we need to make them regular.
According to the cost of slabs, we propose an easy method to measure the regularity of a stack or a layout: The cost of one stack which is represented to the regularity of stack is sum of cost of slabs in the stack. The cost of layout is sum of cost of slabs in the layout.

2) Relocations: A relocation can be written as pair (d,r) of different stacks.

<table>
<thead>
<tr>
<th>Donator stack</th>
<th>Receiver stack</th>
<th>Abbreviation</th>
<th>A serie of the relocations of group $g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surplus stack</td>
<td>Final receiver stack</td>
<td>SF</td>
<td>NS${}^{dF}_g$</td>
</tr>
<tr>
<td>Temporary stack</td>
<td>Final receiver stack</td>
<td>TF</td>
<td>NT${}^{dT}_g$</td>
</tr>
<tr>
<td></td>
<td>Empty stack</td>
<td>TE</td>
<td>NT${}^{dE}_g$</td>
</tr>
<tr>
<td></td>
<td>Prepared stack</td>
<td>TP</td>
<td>NT${}^{dP}_g$</td>
</tr>
<tr>
<td></td>
<td>Temporary stack</td>
<td>TT</td>
<td>NT${}^{dT}_g$</td>
</tr>
<tr>
<td>disorganized stack</td>
<td>Prepared stack</td>
<td>DP</td>
<td>ND${}^{dP}_g$</td>
</tr>
<tr>
<td></td>
<td>Empty stack</td>
<td>DE</td>
<td>ND${}^{dE}_g$</td>
</tr>
<tr>
<td></td>
<td>Final receiver stack</td>
<td>DF</td>
<td>ND${}^{dT}_g$</td>
</tr>
<tr>
<td></td>
<td>Temporary stack</td>
<td>DT</td>
<td>ND${}^{dT}_g$</td>
</tr>
<tr>
<td></td>
<td>disorganized stack</td>
<td>DD</td>
<td>ND${}^{dD}_g$</td>
</tr>
</tbody>
</table>

For example, $3S^1F^3_4$ means that there are three relocations of slabs of group 4 from the surplus stack 1 to the final receiver stack 3.
Slab relocation problem

This is an example to illustrate the role of classification:

Relocation: $2T^2E_4^5$, $1T^4F_3^1$, $1T^1F_4^1$, $2D^3F_2^1$, $1T^3F_4^1$, $2S^3F_4^5$

Cost of layout: 151 → 134 → 101 → 62 → 19 → 4 → 0
Two tree search methods for the SRP
A greedy algorithm to get the initial solution of a layout.

\[ \text{Find\_initial\_solution} \ \text{(in: L \{initial layout\},) } \]
\[ \text{out: s \{greedy solution\})} \]

1) \text{s := ø; \{initialise solution as empty list of relocations\}}
2) \text{while the cost of layout is not 0 and within the time limit do}

Sf, Tf, Df, Te, De, Tp, Tt, Dp, Dt, Dd are the sets of different relocations.
Priority of relocation: SF → TF → DF → TE → DE → TP → TT → DP → DT → DD

A relocation is selected from the first set which is not empty according to the priority, and satisfy the Greedy Criterion: to make the cost of the layout minimum.

\text{endwhile;}
\text{end}
In the process of minimum cost heuristic, we need record the nodes for the tree search. If there is group g which will have its new final target stack after some relocations. Record the layout as a child node of the tree search after these relocations. From the initial layout, we can get many different child nodes according to the decisions on the new final target stacks which may be generated after relocations.

To keep the search effort within acceptable limits. The number of child nodes from a branch node is restricted. At most maxCN=3 different child nodes are applied alternatively to the branch nodes. So we first sequence the child nodes in the ascending order of the cost of child nodes. If the number of child nodes is greater than maxCN, we select the first maxCN child nodes.
Slab relocation problem

(in: \( s^* \{\text{current best solution} \}, L \{\text{current layout} \}, s'(L) \{\text{existed relocation sequence before } L \}, LC \{\text{child nodes set of } L \} \)

out: best solution)

if time limit exceeded then return; endif;

Find_initial_solution (in: \( L \); out: \( LC_n \{\text{child nodes set of } L_n, n=1 \ldots N+1 \} \),

\( s \{\text{greedy solution}, n(s) \text{ is the number of relocations by the greedy solution} \}, \)

\( L_n \{\text{the branch nodes in the } TB_1, n=1 \ldots N+1 \} \),

\( C(L') \{\text{the cost of the layout after relocations by the greedy solution} \}) \)

{If \( n(s+s'(L)) \neq 0 \) and \( C(L') \neq 0 \) abort the search path during the heuristic}

if \( n(s^*)=0 \) then \( n(s')=n(s+s'(L)) \) elseif \( n(s+s'(L)) < n(s^*) \) and \( C(L')=0 \) then \( s^* := s+s'(L) \); endif; endif; for \( n=1, n \leq N+1, n++ \) do

While \( LC_n \neq \emptyset \) do

Singleline_search_method (\( s^*, s'(L_n), L_n, LC_n \));

endfor

Singleline search method

The SSM makes recursive calls and only generates one tree branch in one iteration.

The Advantage of the search direction is to get more leaf nodes within the time limit. Look at \( Lf_1 \) and \( Lf_2 \), they have the same search path by the blue line. So when we want to get \( Lf_2 \), we can start from \( Lf_1 \), and we will get \( Lf_2 \) soon. Because we search the tree nodes from the leaf to the root, so the same search path will become shorter in the search process. This search direction is a double edged sword, the solutions are lacking diversity because of the same search path. 
Multi-line_search_method (in: s'(L){existed sequence from the initial layout}, s*{current best solution}, L{current layout}, D{search direction}, LC {child nodes set of L} out: best solution)

if time limit exceeded then return; endif; get the child nodes of L; the number of the child nodes is M; the relocations add to s'(L)

for m=1, m≤M, m++ do

Find_initial_solution (in: Lm {layout of m child node};
out: sm {greedy solution}, LC_{nm} { child nodes set of L_{nm} },
L_{nm} {branch nodes in the TBm; if D=downward, n=1...N_{m}, else upward, n=N_{m}...1; N_{m} is the number of the branch nodes in the TBm},
C(L'){the cost of the layout after relocations by sm})

{If n(sm+s'(Lm))=n(s*)≠0 and C(L')≠0, abort the search path during the heuristic}

if n(s*)=0 then n(s*)=n(sm+s');
else n(sm+s'(Lm)) < n(s*) and C(L')=0 then s* := sm+s'(Lm); endif;
endfor;
The multi-line search method promotes the diversity of solutions by generating many tree branches in one iteration. In the procedure of MLSM two steps are made to generate the solution for a given layout L. First, the initial solutions are calculated from the child nodes of the root node by the greedy heuristic. Then we regard the branch nodes as root nodes. Then use these root nodes as the input of MLSM by different orders. This method has two different search direction. First, we propose the downward search method to promote the diversity of the solutions further. For example, from the tree map of multi-line downward search method (MLDSM) which is shown in this picture, L11 which is the first branch node will generate many different solutions with the fewer same relocations than L21. From L11 to LN1 the same relocations become more and more, and the search time becomes less and less. Within the time limit the diversity of the solutions will be maximized, but the solutions become fewer than that by SSM. So we combine the advantages of branching strategy and the upward search direction to design the multi-line upward search method (MLUSM). The procedures of MLDSM and MLUSM are shown in this picture.
The mean computation time: $T$
mean number of relocations: $N_r$
mean number of solutions: $N_s$

$maxCN=3$
time limit=250s

The test block consists of 30 stacks

<table>
<thead>
<tr>
<th>No. of slabs</th>
<th>No. of groups</th>
<th>Cost of L</th>
<th>Heuristic</th>
<th>SSM</th>
<th>MLDSM</th>
<th>MLUSM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$N_s$ $T$ $N_r$</td>
<td>$N_s$ $T$ $N_r$</td>
<td>$N_s$ $T$ $N_r$</td>
<td>$N_s$ $T$ $N_r$</td>
</tr>
<tr>
<td>260</td>
<td>15</td>
<td>1073.3</td>
<td>1 2 186</td>
<td>86 203 163</td>
<td>81 192 163</td>
<td>83 197 163</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1238.5</td>
<td>1 2 211</td>
<td>91 212 184</td>
<td>85 201 184</td>
<td>90 210 184</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1474.3</td>
<td>1 2 249</td>
<td>95 227 211</td>
<td>90 215 211</td>
<td>93 222 211</td>
</tr>
<tr>
<td>280</td>
<td>15</td>
<td>1267.6</td>
<td>1 2 209</td>
<td>88 208 182</td>
<td>83 201 182</td>
<td>84 204 182</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1501.1</td>
<td>1 2 233</td>
<td>93 220 201</td>
<td>87 209 201</td>
<td>91 215 201</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1678.3</td>
<td>1 2 256</td>
<td>99 241 229</td>
<td>92 229 229</td>
<td>95 233 229</td>
</tr>
<tr>
<td>300</td>
<td>15</td>
<td>1395.3</td>
<td>1 2 248</td>
<td>90 209 218</td>
<td>86 204 218</td>
<td>88 211 218</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1562.7</td>
<td>1 2 271</td>
<td>97 228 244</td>
<td>91 212 244</td>
<td>96 225 244</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1707.2</td>
<td>1 3 290</td>
<td>107 220 259</td>
<td>99 203 259</td>
<td>102 211 259</td>
</tr>
<tr>
<td>320</td>
<td>15</td>
<td>1531.8</td>
<td>1 3 263</td>
<td>102 215 246</td>
<td>95 199 246</td>
<td>99 209 246</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1715</td>
<td>1 3 289</td>
<td>111 229 269</td>
<td>100 209 269</td>
<td>106 216 269</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1896.5</td>
<td>1 3 311</td>
<td>123 245 289</td>
<td>106 211 289</td>
<td>114 227 289</td>
</tr>
<tr>
<td>340</td>
<td>15</td>
<td>1694.9</td>
<td>1 3 293</td>
<td>119 241 277</td>
<td>101 213 277</td>
<td>109 230 277</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1821.1</td>
<td>1 3 313</td>
<td>121 250 281</td>
<td>109 229 281</td>
<td>113 237 281</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1982.4</td>
<td>1 3 333</td>
<td>119 250 309</td>
<td>111 250 295</td>
<td>117 250 289</td>
</tr>
<tr>
<td>360</td>
<td>15</td>
<td>1837.3</td>
<td>1 3 307</td>
<td>121 250 291</td>
<td>108 250 293</td>
<td>119 250 281</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>2011.6</td>
<td>1 3 331</td>
<td>122 250 312</td>
<td>110 250 304</td>
<td>117 250 291</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>2234.4</td>
<td>1 4 353</td>
<td>119 250 331</td>
<td>105 250 320</td>
<td>118 250 311</td>
</tr>
</tbody>
</table>
1) **The heuristic is greedy** and only generates one solution. So the computational time of the heuristic was less than 4s which is within the level that can be used in practical production. In contrast, for the same test case the computational time of the tree search methods exceeded 190s. But the methods can also be used during the maintenance period of the slab warehouse.
Slab relocation problem

Then look at the tree search methods.

<table>
<thead>
<tr>
<th>No. of slabs</th>
<th>No. of groups</th>
<th>Cost of L</th>
<th>Heuristic</th>
<th>SSM</th>
<th>MLDSM</th>
<th>MLUSM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>N_s</td>
<td>T</td>
<td>N_r</td>
<td>N_s</td>
</tr>
<tr>
<td>260</td>
<td>15</td>
<td>1073.3</td>
<td>1</td>
<td>2</td>
<td>186</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1238.5</td>
<td>1</td>
<td>2</td>
<td>211</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1474.3</td>
<td>1</td>
<td>2</td>
<td>249</td>
<td>95</td>
</tr>
<tr>
<td>280</td>
<td>15</td>
<td>1267.6</td>
<td>1</td>
<td>2</td>
<td>209</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1501.1</td>
<td>1</td>
<td>2</td>
<td>233</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1678.3</td>
<td>1</td>
<td>2</td>
<td>256</td>
<td>99</td>
</tr>
<tr>
<td>300</td>
<td>15</td>
<td>1395.3</td>
<td>1</td>
<td>2</td>
<td>248</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1562.7</td>
<td>1</td>
<td>2</td>
<td>271</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1707.2</td>
<td>1</td>
<td>3</td>
<td>290</td>
<td>107</td>
</tr>
<tr>
<td>320</td>
<td>15</td>
<td>1531.8</td>
<td>1</td>
<td>3</td>
<td>263</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1715</td>
<td>1</td>
<td>3</td>
<td>289</td>
<td>111</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1896.5</td>
<td>1</td>
<td>3</td>
<td>311</td>
<td>123</td>
</tr>
<tr>
<td>340</td>
<td>15</td>
<td>1694.9</td>
<td>1</td>
<td>3</td>
<td>293</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1821.1</td>
<td>1</td>
<td>3</td>
<td>313</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1982.4</td>
<td>1</td>
<td>3</td>
<td>333</td>
<td>119</td>
</tr>
<tr>
<td>360</td>
<td>15</td>
<td>1837.3</td>
<td>1</td>
<td>3</td>
<td>307</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>2011.6</td>
<td>1</td>
<td>3</td>
<td>331</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>2234.4</td>
<td>1</td>
<td>4</td>
<td>353</td>
<td>119</td>
</tr>
</tbody>
</table>
1) The heuristic is greedy and only generates one solution. So the computational time of the heuristic was less than 2s which is within the level that can be used in practical production. In contrast, for the same test case the computational time of the tree search methods exceeded 190s. But the methods can also be used during the maintenance period of steel slab warehouse.

2) **MLDSM** can abort the non-optimal search path earlier to save the search time because it get the better solution earlier from the diversified solutions. In the applications of slab relocation system (SRS), compared with MLUSM and SSM, the operator often runs this program to get a better solution than the heuristic.
The mean computation time: $T$  
mean number of relocations: $N_r$  
mean number of solutions: $N_s$  
maxCN=3  
time limit=250s  
The test block consists of 30 stacks

<table>
<thead>
<tr>
<th>No. of slabs</th>
<th>No. of groups</th>
<th>Cost of L</th>
<th>Heuristic</th>
<th>SSM</th>
<th>MLDSM</th>
<th>MLUSM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$N_s$</td>
<td>$T$</td>
<td>$N_r$</td>
<td>$N_s$</td>
</tr>
<tr>
<td>260</td>
<td>15</td>
<td>1073.3</td>
<td>1</td>
<td>2</td>
<td>186</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1238.5</td>
<td>1</td>
<td>2</td>
<td>211</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1474.3</td>
<td>1</td>
<td>2</td>
<td>249</td>
<td>95</td>
</tr>
<tr>
<td>280</td>
<td>15</td>
<td>1267.6</td>
<td>1</td>
<td>2</td>
<td>209</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1501.1</td>
<td>1</td>
<td>2</td>
<td>233</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1678.3</td>
<td>1</td>
<td>2</td>
<td>256</td>
<td>99</td>
</tr>
<tr>
<td>300</td>
<td>15</td>
<td>1395.3</td>
<td>1</td>
<td>2</td>
<td>248</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1562.7</td>
<td>1</td>
<td>2</td>
<td>271</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1707.2</td>
<td>1</td>
<td>3</td>
<td>290</td>
<td>107</td>
</tr>
<tr>
<td>320</td>
<td>15</td>
<td>1531.8</td>
<td>1</td>
<td>3</td>
<td>263</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1715</td>
<td>1</td>
<td>3</td>
<td>289</td>
<td>111</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1896.5</td>
<td>1</td>
<td>3</td>
<td>311</td>
<td>123</td>
</tr>
<tr>
<td>340</td>
<td>15</td>
<td>1694.9</td>
<td>1</td>
<td>3</td>
<td>293</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1821.1</td>
<td>1</td>
<td>3</td>
<td>313</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>1982.4</td>
<td>1</td>
<td>3</td>
<td>333</td>
<td>119</td>
</tr>
<tr>
<td>360</td>
<td>15</td>
<td>1837.3</td>
<td>1</td>
<td>3</td>
<td>307</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>2011.6</td>
<td>1</td>
<td>3</td>
<td>331</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>2234.4</td>
<td>1</td>
<td>4</td>
<td>353</td>
<td>119</td>
</tr>
</tbody>
</table>
Slab relocation problem

1) The heuristic is greedy and only generates one solution. So the computational time of the heuristic was less than 2s which is within the level that can be used in practical production. In contrast, for the same test case the computational time of the tree search methods exceeded 190s. But the methods can also be used during the maintenance period of steel slab warehouse.

2) **MLDSM** can abort the non-optimal search path earlier to save the search time because it get the better solution earlier from the diversified solutions. In the applications of slab relocation system(SRS), compared with MLUSM and SSM, the operator often runs this program to get a better solution than the heuristic.

3) SSM has searched the solutions from only one tree branch that makes the solutions fall into the local optimum easily. But these solutions are more than those by MLSM within the time limit. Compared with SSM and MLDSM, **MLUSM** has the advantages of other tree search methods. So when the situation is complicated the operator can run MLUSM program of SRS to get a better near-optimal solution.
This is the operation interface of the slab relocation system.

Compared with initial layout, the final layout is more regular.

First select one area to operate.
Conclusions:

In this paper, we investigated the practical SRP in steel industry and proposed a model for SRP. The tree search methods that are based on a greedy heuristic and the suitable search strategies were designed to solve SRP. Combining the tree search methods with man-machine interactive method, a practical SR system has been developed. The computational results on the practical instances show the SR system effective.
Thank you for your listening!

The Logistics Institute, Northeastern University, Shenyang, China
http://tli.neu.edu.cn
1) If a NSdFrg is possible, the surplus stack may turn into an empty stack to store the slabs of other group.
2) A temporary stack may turn into a new final stack after relocation without using an empty stack.
3) The height of a disorganized stack will be reduced after relocations. And the final target stack will get more slabs of the same group.
4) A temporary stack and an empty stack may turn into two final target stacks. Because of wasting a empty stack, the operation has lower priority than above operations.
5) A disorganized stack may turn into a temporary stack and an empty stack turns into a final target stack.
6) At the moment there is no empty stacks or final target stacks to store the temporary slabs. So put the temporary slabs to the prepared stack, the temporary stack may turn into the final target stack which will receive the waiting slabs of the same group.
7) If the temporary slabs of the two temporary stacks are focused on one temporary stack, the other stack will be the final target stack.
8) If the waiting slabs of disorganized stack need to be the temporary slabs, a prepared stack is the best candidate to be temporary stack.
9) The temporary stack is the second candidate for waiting slabs.
10) If we wish a disorganized stack could be a final target stack, the worst case is to put the waiting slabs to other disorganized stack.