Int J Adv Manuf Technol (2009) 45:1215–1231
DOI 10.1007/s00170-009-2058-z

ORIGINAL ARTICLE

Guided local search algorithm for hot strip mill scheduling problem with considering hot charge rolling

Mohammad Reza Yadollahpour · Mehdi Bijari · Soheila Kavosh · Mehdi Mahnam

Received: 31 July 2008 / Accepted: 8 April 2009 / Published online: 7 May 2009

Abstract This study investigates the hot strip mill scheduling problem which is one of the most important planning problems in the steel industry. The problem is formulated using the prize collecting vehicle routing problem. The new proposed formulation considers more details and more realistic constraints than those used in previous studies. The hot charge technique leads to considerable savings in energy and other benefits in the process of steel production. In our proposed formulation, the necessary provisions required for obtaining an initial level of hot charge have been taken into consideration. A search algorithm has been developed that consists of three major phases including separation of slabs that can be scheduled, generation of an initial solution, and improvement of the solution. Generation of the initial solution is accomplished using a greedy constraint satisfaction algorithm and solution improvement through a guided local search. Proposed model and search algorithm have been tested on random and collected instances from practical production data in Mobarakeh Steel Complex. The experimental results show the high accuracy and efficiency of the proposed model and search algorithm.

Keywords Hot strip mill scheduling problem · Hot charge · Guided local search

M. R. Yadollahpour (✉) · M. Bijari · S. Kavosh · M. Mahnam
Department of Industrial and Systems Engineering,
Isfahan University of Technology,
Isfahan 84156-83111, Iran
e-mail: yadollahpour@in.iut.ac.ir

1 Introduction

The steel industry today is vital to most manufacturing and service industries including the auto-making, railroad, and bridge construction among others. Since the steelmaking industry is both capital- and energy-intensive, an effective tool can be employed that while saving on material and energy and improve upon machine productivity, production scheduling is a case in point. Steel manufacturing process mainly involves iron ore dressing, sponge iron production, steelmaking and continuous casting, hot rolling, cold rolling, and finishing processes. This paper concentrates on the hot strip mill scheduling problem (HSMSP) in Mobarakeh Steel Company (MSC), with particular consideration of the selection and sequencing of slabs to be hot rolled. In this stage, pig iron mixed with some additives is converted to molten iron in electric arc furnaces to be cast and cooled to form solid cube plates called slabs after certain supplementary processes. The slabs are then conveyed to the hot roll process where they are initially preheated. There are four parallel preheating furnaces receiving the slabs which lose in thickness by passing through the rollers of the roughing and finishing mills. After cooling, they are finally changed into hot coils. The hot coils can be either supplied to the market as final product or used as semi-finished product in subsequent processes. Figure 1 illustrates the material flow for casting and hot strip mill rolling.

In the hot charge technique, slabs are delivered for hot rolling after they exit the casting unit but before they lose all their heat. The advantages of this technique are as follows: (1) energy savings, (2) decreasing slab inventory and the slab yard space required, (3) reducing production cycle time, and (4) prevention of certain slab surface quality defects occurring during the cooling process.

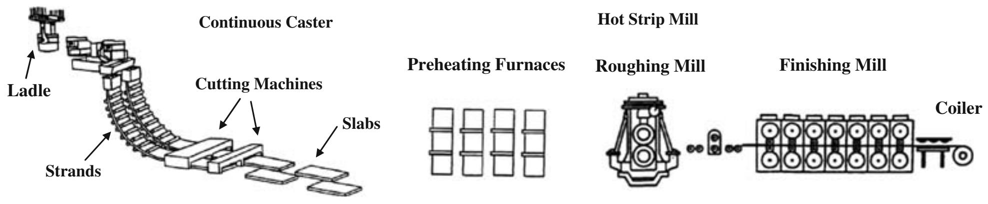


Fig. 1 Continuous casting and hot strip mill

Tang et al. [1] provide a description of various kinds of hot charges. In the initial level of the hot charge technique, slabs are placed in *pits* right after casting. Slabs are then charged to the preheating furnace depending on production demand where they will be preheated to be sent to the hot rolling unit. The preheating furnace is a tunnel-shaped space of about 40 m long equipped with gas torches on its walls. There are approximately 36 slabs in the furnace at a time. Slabs move slowly within the preheating furnace until they reach the desired temperature at the exit. Furnace temperature can be controlled by adjusting the output gas from the torches. In cases where slabs with different temperatures (ranging from ambient temperature to as high as 800°C) enter the preheating furnace, torches will be adjusted for those slabs with minimum temperature (i.e., the ones requiring maximum heat). If the slabs in the furnace have the same, or nearly the same, temperature, then the torches will be set in a manner to generate just the necessary amount of heat. Based on these considerations, the following two constraints must be added to HSMSP in order to help the initial hot charge stage: (1) Temperature differences of adjacent slabs in the program must be minimized, and (2) slabs with higher temperatures should be given priority for being rolled.

In this paper, a mathematical formulation based on prize collecting vehicle routing problem (PCVRP) is presented for the hot strip mill scheduling problem with regard to hot charge rolling. Moreover, according to the complexity of the problem and inefficiency of exact approaches, a two-phase meta-heuristic algorithm including a greedy constraint satisfaction algorithm and a guided local search (GLS) is proposed. The rest of this paper is organized as follows. The literature related to the hot strip mill scheduling problem is reviewed in Section 2. Section 3 presents a description and statement of the problem followed by a mathematical programming model formulated for this problem. The proposed heuristic and meta-heuristic algorithms are introduced in Section 4. Computational experiments and results are reported in Section 5. Finally, Section 6 provides conclusions and areas for future study.

2 Literature review

Due to the importance of the hot rolling process in the steel industry, the scheduling problem of this process has been widely investigated recently (see [1] for more details) and many kinds of modeling strategies have been proposed. Generally, these modeling strategies can be classified into two categories, serial and parallel methods [2, 3].

The first category is the traditional serial method, which can arrange only one program at a time. Using this method, Kosiba et al. [4] formulated the HSMSP as the traveling salesman problem and Lopez et al. [5] formulated it as the prize collecting traveling salesman problem (PCTSP). Each time the serial method generates a program by selecting coils from the unscheduled coils and when a program is generated, the coils in it are fixed and will no longer be considered. Then, another program is generated by selecting coils from the remaining unscheduled ones. This procedure continues until a sufficient number of programs have been generated. Therefore, the serial strategy is essentially a greedy procedure, which will inevitably make the programs following the first one poorer and poorer because the number of candidate unscheduled coils decreases as the number of programs generated increases. It suffers from the disadvantage of local optimization [3].

To avoid this disadvantage, the second category presents a parallel strategy in the modeling method which can simultaneously generate multiple programs at a time. Using this method, the HSMSP is often formulated as the multiple traveling salesmen problem [6] or the prize collecting vehicle routing problem [2, 3, 7, 8]. The prominent studies of HSMSP are listed in Table 1.

The studies mentioned above are mostly based on case studies and are totally different from the present one with respect to the model and the solution algorithm used. The model presented in this paper is based on PCVRP and belongs to the category of parallel strategy models. It considers more details and more realistic constraints than those used in previous literature, especially the study of Wang and Tang [3]. The following are the main differences:

1. Solution design: Each program in the proposed model contains three different routes (in VRP), each with its own specific features and constraints. This is in contrast to the programs reported in the literature in which each program is equal to one route and the constraints universally apply to all programs.
2. Constraints: Some of the constraints in the proposed model are identical in nature and type to those of the previous models, although they significantly differ in

Table 1 HSMSP Literature Review

References	Year	Type of algorithm	Model
Wright and Houck [9]	1985	Heuristic initial solution and hill climbing to improve it	–
Balas and Martin [10]	1991	Heuristic	PCTSP
Kosiba et al. [4]	1992	Heuristic	TSP
Lopez et al. [5]	1998	Heuristic initial solution and TS to improve it	PCTSP
Cowling [7]	1995	Initial solution with LS and TS to improve it	PCVRP
Tang et al. [6]	2000	Heuristic initial solution and GA to improve it	MTSP
Cowling [8]	2003	Propose a decision support system (DSS)	–
Tang and Wang [2]	2005	Heuristic initial solution and ILS to improve it	PCVRP
Wang and Tang [3]	2008	Heuristic initial solution and TS to improve it	PCVRP

design and detail. In addition to these, two campaign and next process constraints are provisioned which were not considered in previous models. The next process constraint plays an important role in balancing the following hot rolling processes.

3. Strict attention to hot charge: All studies reported in the literature except for Lopez et al. [5] investigated HSMSP under cold charge conditions. Lopez et al. designed their model based on TSP and serial strategy. Also, they did not consider such important realistic constraints as slab temperature decrease, differences in slab temperature in various programs, and desirability of rolling hot slabs.
4. Cost function: A comprehensive combination of transition costs and rolling utilization are considered. The cost function in the model developed here is different from those of the previous models, as it contains more realistic items.

3 Problem description and formulation

In the HSMS problem, there are two kinds of rollers in the finishing mill. Working rollers are in direct contact with the surface of the steel sheets under rolling. Backup rollers are parallel with the working rollers with no direct contact with steel sheet, their role being to provide support for the working rollers. Both working and backup rollers are subject to gradual wear due to high temperature and speed and must be replaced at regular intervals to ensure predefined product quality. The set of slabs being rolled in the period between two subsequent replacements of working rollers is called a *program*. The set of programs being rolled between two subsequent replacements of backup rollers is called a *campaign*. Usually, a program consists of about 150 slabs and a campaign contains 42 programs. HSMSP is simply producing one or more programs, which involves the selection of a sufficient number of slabs from the slab yard and sequencing them for the rolling line. A program consists of three sections designated as increase section, maximum width section, and decrease section. Slab width takes an ascending trend in the increase section, but shows very limited variations in the maximum width section, and finally in the decrease section, it assumes a descending trend. Therefore, slab width profile in one program is nearly coffin-shaped; hence, it is called the coffin constraint and shown in Fig. 2. In the increase section that involves only a small portion of the program, working rollers are warmed up. In order to avoid defects on coil edges, the slab width must have a descending trend for the greatest portion of the program (Fig. 3).

Initially, VRP and PCVRP will be introduced as the basic problems used in modeling HSMSP followed by a description of the proposed mathematical model of modified hot strip mill scheduling problem (MHSMSP). The VRP is defined on an undirected graph $G=(V, A)$ where $V=\{0, 1,\ldots,n\}$ is the vertex set and $A=\{(i, j): i,j \in V, i\neq j\}$ is the arc set. Vertex 0 represents a depot which are located at most m identical vehicles of capacity Q. With each customer $i \in V-\{0\}$ is associated with a non-negative demand $q_i \leq Q$. A cost matrix c_{ij} is defined on A. The problem consists of determining a set of m vehicle routes (1) starting and ending at the depot and such that (2) each customer is visited by exactly one vehicle, (3) the total demand of any route does not exceed Q, and (4) the total routing cost is minimized.

It is shown that vehicle routing problem (VRP) is NP-hard because it includes the traveling salesman problem (TSP) as a special case when $m=1$ and $Q=\infty$. For a more comprehensive review of VRP, see Laporte [11]. PCVRP is

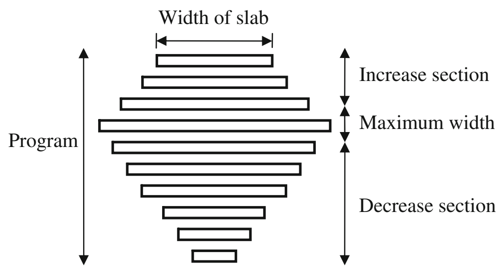


Fig. 2 Width changes pattern for slabs

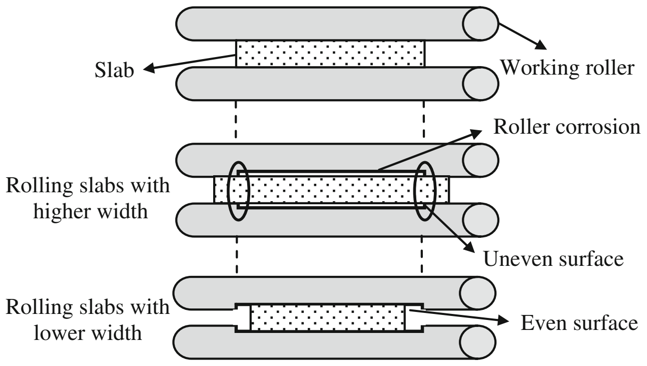

Slab
Working roller
Roller corrosion
Rolling slabs with higher width
Uneven surface
Rolling slabs with lower width
Even surface

Fig. 3 Coffin constraint

an extension of VRP, but contains two major assumptions that make it different from VRP. These assumptions are: (1) It is not necessary to visit all customers, and (2) visiting each customer has a prize. The prize amount can be considered in two different ways: (1) adding the prize value to the cost (objective) function with a negative sign or (2) adding a constraint to VRP: The total collected prize must be greater than a predefined value. We used the second strategy in modeling MHSMSP.

Each customer in PCVRP represents a slab in MHSMSP and its prize measures the utilization of processing that slab. An arc (i, j) represents the fact that slab j is scheduled immediately after slab i and its weight measures the desirability of scheduling slab j immediately after slab i. Table 2 summarizes the necessary parameters and variables.

3.1 Model constraints

The constraints in the MHSMSP model can be classified as follows:

1. Campaign constraint

This constraint considers all slabs and determines the ones available for planning (Avail), depending on which campaign program is to be constructed based on their width and thickness. Figure 4 shows the details of this constraint. Numbers show the minimum thickness in the program and correspond to the width range. For example, in programs 17 to 31 in the campaign and in the width range 1,541 to 1,600, the minimum thickness of coils produced is 2.5 mm. Therefore, the available slabs in each campaign will be simply obtained before solving the model using the corresponding table in Fig. 4.

2. Allocation constraints

These constraints guarantee the feasibility of assignments and also the program shape based on the coffin constraint:

Table 2 Parameters and variables

Parameters
$n(A)$: Number of members in set A
Total Avail: Total slabs in the slab yard
Avail=$\{0, 1, 2,...,n(\text{Avail}) - 1\}$: Slabs that can be rolled and slab 0
N: Number of programs in campaign
m: Number of programs to be generated
r: Program index
$R0$: Not scheduled slabs
Inc_Sec(r), r=1,...,m: Slabs in the increase section in program r
maxw_Sec(r), r=1,...,m: Slabs in the maximum width section in program r
Dec_Sec(r), r=1,...,m: Slabs in the decrease section in program r
q; (1: Inc_Sec, 2: maxw_Sec, 3: *Dec_Sec*): Each section of the program
Slabs(s,r): Set of available slabs in section q of program r
$n(q,r)$: Number of available slabs in section q of program r
$s(i,q,r)$: The ith slab in section q of program r
Cost(i,j,q,r) .i,j є Avail, q=1,2,3, r=1,...,m: Transition cost from slab i to slab j
Cost_func: Total cost function
util(i, r) , $i \in$ Avail, $r = 1, ..., m$: Utilization of rolling slab i in program r
km(i), $i \in$ Avail $-\{0\}$: Coil length i (km)
temp(i, r), $i \in$ Avail $-\{0\}$, $r = 1, ..., m$: Temperature of slab i in program r
env_temp: Ambient temperature
Cast_temp: Slab i temperature after casting
Curr_week: Current week
duedate(i), $i \in$ Avail $-\{0\}$: Due date for coil i
width(i), $i \in$ Avail $-\{0\}$: Width of coil i (mm)
ton(i), $i \in$ Avail $-\{0\}$: Weight of slab i (ton)
thick(i), $i \in$ Avail $-\{0\}$: Thickness of coil i (mm)
qual(i), $i \in$ Avail $-\{0\}$: Quality of slab i
Cost_qual(i,j), $i,j \in$ Avail: Quality transition cost for moving from slab i to slab j
Cost_temp(i,j,r), $i,j \in$ Avail: Temperature transition cost for moving
from slab i to slab j
Cost_thick(i,j), $i,j \in$ Avail: Thickness transition cost for moving from slab i to slab j
Cost_width(i,j,q), $i,j \in$ Avail: Width transition cost for moving from slab i to slab j
util_km (i), $i \in$ Avail: Length utilization for rolling slab i
util_temp (i, r), $i \in$ Avail, $r = 1, ..., m$: Temperature utilization for rolling slab i
util_tardiness(i), $i \in$ Avail: Tardiness utilization for rolling slab i
util_earliness(i), $i \in$ Avail: Earliness utilization for rolling slab i
max_km(q), q=1,2,3: Maximum length for program's sections (km)
max_ton(q), q=1,2,3: Maximum weight for program's sections (ton)
min_km(q), q=1,2,3: Minimum length for program's sections (km)
min_ton(q), q=1,2,3: Minimum weight for program's sections (ton)
max_km_total: Maximum length for program
min_km_total: Minimum length for program
max_ton_total: Maximum weight for program

Table 2 (continued)

min_ton_total: Minimum weight for program

min_width: Minimum width in decrease section

Start_width: Width of the first slab in program is greater than or equal to this amount

max_width: Width of all slabs in program is less than or equal to max_width+ 20

Qual_limit(q), q=1,2,3: Upper limit for quality number (grade) of slabs in section q

M: Very big value

min_thick(q), q=1,2,3: Minimum thick of slabs in section q

max_thick(q), q=1,2,3: Maximum thick of slabs in section q

num_process: Number of next processes that coils may be sent to them

$\text{next\_process}(i,p), i \in \text{Avail} - \{0\}, p = 1,2,...,\text{num\_process}$: 1, if next process of slab i is p and otherwise, 0

$\alpha(p)$, p=1,2,..., num_process: Upper bound for percent of slabs in program that next process of them is p

$\text{near\_width}(i,j), i,j \in \text{Avail} - \{0\}$: 1, if deferent between width of slabs i and j is less than 50 (mm) and otherwise 0

near_width_km(q): Maximum accumulative length of sequential near-width slabs in section q

near_width_ton(q): Maximum accumulative weight of sequential near-width slabs in section q

max_diff_width(q), q=1,2,3: Maximum width different between adjacent slabs in section q

$\max\_diff\_thick\_des(i,j), i,j \in \text{Avail} - \{0\}$: If $thick(i) \geq thick(j)$, maximum thick decrease in moving from slab i to slab j and otherwise$+\infty$

$\max\_diff\_thick\_asc(i,j), i,j \in \text{Avail} - \{0\}$: If $\text{thick}(i) < \text{thick}(j)$, maximum thick increase in moving from slab i to slab j and otherwise$+\infty$

Variables

$x(i,j,q,r),\ i,j \in \text{Avail} - \{0\}, q = 1,2,3\ , r = 1,...m$: 1 if slab j is inserted after slab i and otherwise 0 ($i \neq j$)

$y(i,q,r),\ i \in \text{Avail} - \{0\}, q = 1,2,3\ , r = 1,...m$: 1 if slab i is inserted in program r in section q and otherwise 0

$$\sum_{r=1}^{m}\sum_{q=1}^{3} y(i,q,r) \leq 1, i \in \text{Avail} - \{0\} \tag{1}$$

Equation 1 ensures that each slab would be assigned just one time in the planning horizon.

$$\sum_{r=1}^{m}\sum_{q=1}^{3} y(0,q,r) = 3m \tag{2}$$

Equation 2 assigns virtual slab 0 to each section of the program.

$$\sum_{i=1}^{n(\text{Avail})-1} x(0,i,q,r) = \sum_{i=1}^{n(\text{Avail})-1} x(i,0,q,r) = 1, \qquad q = 1,2,3, r = 1,...,m \tag{3}$$

Equation 3 makes sure that there is at least one slab before and after virtual slab 0 in each section of the program.

$$\sum_{u=0}^{n(\text{Avail})-1} x(u,i,q,r) = \sum_{v=0}^{n(\text{Avail})-1} x(i,v,q,r) = y(i,q,r), \qquad i \in \text{Avail} - \{0\}, q = 1,2,3\ , r = 1,..., \tag{4}$$

Equation 4 ensures that there is inevitably one slab before and one slab after slab i assigned to the program.

$$\sum_{i\in B}\sum_{j\in B} x(i,j,q,r) \leq n(B) - 1, \forall B \subset \text{Avail} - \{0\}, \quad 2 \leq n(B) \leq n(\text{Avail}) - 1, \quad q = 1,2,3, r = 1,...,m \tag{5}$$

Finally, Eq. 5 guarantees that there is no sub-tour in the program.

3. Constraint of total weight and length of the program

These constraints ensure that the total program weight and length of coils after rolling are less than their predetermined maximum values.

$$\min\_ton\_total \leq \sum_{q=1}^{3}\sum_{i=1}^{n(\text{Avail})-1} y(i,q,r)\ \text{ton}(i) \leq \max\_ton\_total \quad , r = 1,...,m \tag{6}$$

$$\min\_km\_total \leq \sum_{q=1}^{3}\sum_{i=1}^{n(\text{Avail})-1} y(i,q,r)\ \text{km}(i) \leq \max\_km\_total \quad , r = 1,...,m \tag{7}$$

4. Weight and length constraints in each section of the program

In addition to the weight and length constraints in the total program, there are distinct constraints on weight and length of coils for increase, maximum width, and decrease sections, separately, which are formulated as follows:

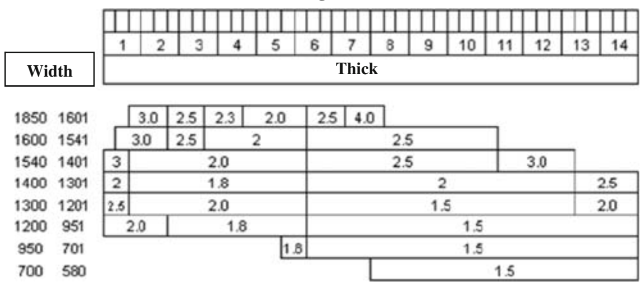


Fig. 4 Campaign constraint

$$\min\_\text{ton}(q) \leq \sum_{i=1}^{n(\text{Avail})-1} y(i,q,r)\ \text{ton}(i) \leq \max\_\text{ton}(q) \quad ,q=1,2,3, r=1,\ldots,m \tag{8}$$

$$\min\_\text{km}(q) \leq \sum_{i=1}^{n(\text{Avail})-1} y(i,q,r)\ \text{km}(i) \leq \max\_\text{km}(q) \quad ,q=1,2,3, r=1,\ldots,m \tag{9}$$

5. Width range constraint in each section

The slab width of each section is within a specific range. Therefore, constraints 10, 11, and 12 ensure that the width of slabs in increase, max-width, and decrease sections are in the specified range, respectively. Regarding the “big *M*” coefficient, the inequalities would be considered if slab *i* is allocated at least to one of the sections.

$$\text{start\_width} - M(1-y(i,1,r)) \leq \text{width}(i) \leq \max\_\text{width} - 21 + M(1-y(i,1,r)) \quad ,i \in \text{Avail}-\{0\}, r=1,\ldots,m \tag{10}$$

$$\max\_\text{width} - 20 - M(1-y(i,2,r)) \leq \text{width}(i) \leq \max\_\text{width} + 20 + M(1-y(i,2,r)) \quad ,i \in \text{Avail}-\{0\}, r=1,\ldots,m \tag{11}$$

$$\min\_\text{width} - M(1-y(i,3,r)) \leq \text{width}(i) \leq \max\_\text{width} - 21 + M(1-y(i,3,r)) \quad ,i \in \text{Avail}-\{0\}, r=1,\ldots,m \tag{12}$$

6. Quality and thickness range constraint in each section

In this study, slabs are classified into six categories based on their quality. Therefore, the higher the quality, the harder the slab will be. There is a quality upper bound in each program section which is considered in relation 13. For example, hard slabs should not be included in the increase section of the program.

$$\text{qual}(i) \leq \text{qual\_limit}(q) + M(1-y(i,q,r)), i \in \text{Avail} - \{0\}, r=1,\ldots,m\,,\ q=1,2,3. \tag{13}$$

Similarly, according to the roller efficiency, the thickness of coils produced in each section should be within a specified range. Thus, coils of very low thickness should not be used in the increase section.

$$\min\_\text{thick}(q) - M(1-y(i,q,r)) \leq \text{thick}(i) \leq \max\_\text{thick}(q) + M(1-y(i,q,r)) \quad ,i \in \text{Avail}-\{0\}, r=1,\ldots,m\,,\ q=1,2,3. \tag{14}$$

7. Next process constraint

After hot strip milling, coils will be dispatched to different processes depending on their product cycle. Each downstream process has its own capacity. Reasonable values must be assigned to total coil weight and length delivered to the downstream process so that oversupply and idleness are avoided as formulated below:

$$\sum_{i=1}^{n(\text{Avail})-1} \sum_{q=1}^{3} \text{km}(i)\ \text{next\_process}(i,p)\ y(i,q,r) \leq \alpha(p) \sum_{q=1}^{3} \sum_{i=1}^{n(\text{Avail})-1} y(i,q,r)\ \text{km}(i) \quad ,r=1,\ldots,m\,,\ p=1,..,\text{num\_process}. \tag{15}$$

8. Constraint of cumulative weight and length of sequential near-width slabs

To avoid accelerated roller wear, the cumulative weight and length of sequential slabs with nearly identical widths should not exceed distinct limits which are determined by Eqs. 16 and 17.

$$\sum_{j=1}^{n(\text{Avail})-1} y(j,q,r)\ \text{near\_width}(i,j)\ \text{km}(j) \leq \text{near\_width\_km(q)} + M(1-y(i,q,r)) \quad ,i \in \text{Avail}-\{0\}, r=1,\ldots,m\,,\ q=1,3 \tag{16}$$

$$\sum_{j=1}^{n(\text{Avail})-1} y(j,q,r)\ \text{near\_width}(i,j)\ \text{ton}(j) \leq \text{near\_width\_ton}(q) + M(1-y(i,q,r)) \quad ,i \in \text{Avail}-\{0\}, r=1,\ldots,m\,,\ q=1,3 \tag{17}$$

9. Width jump constraint

To prevent roller shocks and decrease the setup time between two adjacent slabs, width jump between adjacent slabs must be kept minimal. The adjacent slabs with non-acceptable jumps are formulated as constraints in relations 18, 19, and 20.

$$0 \leq \text{width}(j) - \text{width}(i) \leq \max\_\text{diff\_width}(1) + M(1-x(i,j,1,r)) \quad ,i,j \in \text{Avail}-\{0\}, r=1,\ldots,m \tag{18}$$

$$|\text{width}(i) - \text{width}(j)| \le \max\_\text{diff\_width}(2) + M(1 - x(i,j,2,r)) \quad ,i,j \in \text{Avail} - \{0\}, r = 1, \dots, m \tag{19}$$

$$0 \le \text{width}(i) - \text{width}(j) \le \max\_\text{diff\_width}(3) + M(1 - x(i,j,3,r)) \quad ,i,j \in \text{Avail} - \{0\}, r = 1, \dots, m \tag{20}$$

10. Thickness jump constraint

The ascending and descending ranges of coil thickness jumps should be within specific limits as determined by Eqs. 21 and 22.

$$\text{thick}(j) - \text{thick}(i) \le \max\_\text{diff\_thick\_asc}(i,j) + M(1 - x(i,j,q.r)), \quad i,j \in \text{Avail} - \{0\}, r = 1, \dots, m\,,\ q = 1,2,3 \tag{21}$$

$$\text{thick}(i) - \text{thick}(j) \le \max\_\text{diff\_thick\_des}(i,j) + M(1 - x(i,j,q.r)), \quad i,j \in \text{Avail} - \{0\}, r = 1, \dots, m\,,\ q = 1,2,3 \tag{22}$$

3.2 Model objective function

The cost function consists of transition cost and rolling utilization. The transition cost includes quality, temperature, thickness, and width costs. The following notations are used for defining these parameters in the model for the adjacent slabs i and j:

$$\text{Cost\_qual}(i,j), i,j \in \text{Avail} \tag{23}$$

$$\text{Cost\_temp}(i,j,r), i,j \in \text{Avail} \tag{24}$$

$$\text{Cost\_thick}(i,j), i,j \in \text{Avail} \tag{25}$$

$$\text{Cost\_width}(i,j,q), i,j \in \text{Avail} \tag{26}$$

Furthermore, utility function consists of length, temperature, earliness, and tardiness utilities which are obtained from Eqs. 27, 28, 29, and 30 as follows:

$$\text{util\_km}(i) = \text{km}(i), i \in \text{Avail} \tag{27}$$

$$\text{util\_temp}(i,r) = \frac{\text{temp}(i,r) - \text{env\_temp}}{\text{cast\_temp} - \text{env\_temp}}, i \in \text{Avail}, r = 1, \dots, m \tag{28}$$

$$\text{util\_tardiness}(i) = \max\{0, \text{curr\_week} - \text{duedate}\{i\}\}, i \in \text{Avail} \tag{29}$$

$$\text{util\_earliness}(i) = \max\{0, \text{duedate}(i) - \text{curr\_week}\}, i \in \text{Avail} \tag{30}$$

The earliness and tardiness utilities are consistent with the just-in-time production philosophy which has received considerable attention in recent decades [12]. In this production system, slabs should be completed as close to their due dates as possible. Therefore, the slabs with due or late deliveries must be immediately scheduled, while rolling must be delayed for coils that are yet early for delivery. The slabs with greater tardiness must enter the program sooner. Therefore, tardiness has a negative sign in the total cost function.

In this study, transition costs and utilities are combined into a single objective. To avoid convergence to a special objective because of different ranges, it is necessary to normalize all objectives over the range [0, 1] based on the following equation:

$$\widehat{x} = ((x - x_{\min})/(x_{\max} - x_{\min})) \tag{31}$$

where $\widehat{x}$ is the normalized measure and $x_{\min}$ and $x_{\max}$ are the minimum and maximum values, respectively. The lower and upper bounds of the objective costs and utilities are given by the experts of Mobarakeh Steel Complex.

Consequently, Eqs. 32, 33, and 34 are those required for calculating the cost function.

$$\begin{aligned}\text{Cost}(i,j,q,r) = {} & v_1\text{Cost\_width}(i,j,q) \\ & + v_2\text{Cost\_qual}(i,j) + v_3\text{Cost\_thick}(i,j) \\ & + v_4\text{Cost\_temp}(i,j,r), i,j \in \text{Avail}, \\ & q = 1,2,3, r = 1, \dots, m\end{aligned} \tag{32}$$

$$\begin{aligned}\text{util}(i,r) = {} & -\{\theta_1\,\text{util\_km}(i) + \theta_2\,\text{util\_temp}(i,r) \\ & + \theta_3\,\text{util\_tardiness}(i) - \theta_4\,\text{util\_earliness}(i)\}, i \in \text{Avail}, \\ & r = 1, \dots, m\end{aligned} \tag{33}$$

$$\begin{aligned}\text{Cost\_func} = {} & w_1 \sum_{r=1}^{m} \sum_{q=1}^{3} \sum_{i=0}^{n(\text{Avail})-1} \sum_{j=0}^{n(\text{Avail})-1} \text{Cost}(i,j,q,r) \times x(i,j,q,r) \\ & + w_2 \sum_{r=1}^{m} \sum_{q=1}^{3} \sum_{i=0}^{n(\text{Avail})-1} \text{util}(i,r) \times y(i,q,r)\end{aligned} \tag{34}$$

wherein the coefficients υ_1 to υ_4 in Eq. 32, θ_1 to θ_4 in Eq. 33, and w_1 and w_2 in Eq. 34 play special roles in determining the priorities of objectives which are given by the company experts.

4 The proposed algorithm

Since MHSMSP is an extension of VRP, it is, therefore, a NP-hard problem and exact algorithms like branch and bound and dynamic programming cannot be used to solve it as they are time-consuming. Therefore, a heuristic method should be applied for its large-scale solution. First, a set of slabs capable of being rolled regarding the campaign constraint is separated and the proposed algorithm is employed subsequently.

In this algorithm, constraints 7 and 8 [next process and cumulative weight (length) of sequential near-width constraints] are incorporated into the objective as penalty functions in order to overlap the difficulty of the model to find feasible solutions. The relaxed model objective function is expressed below:

$$\begin{aligned}\text{Cost\_func} = {} & w_1 \sum_{r=1}^{m}\sum_{q=1}^{3}\sum_{i=0}^{n(\text{Avail})-1}\sum_{j=0}^{n(\text{Avail})-1} (v_1 \text{Cost\_width}(i,j,q) + v_2 \text{Cost\_qual}(i,j) + v_3 \text{Cost\_thick}(i,j) \\ & + v_4 \text{Cost\_temp}(i,j,r) + v_5 \text{Cost\_near\_width}(i,j,r)) x(i,j,q,r) + w_2 \sum_{r=1}^{m}\sum_{q=1}^{3}\sum_{i=0}^{n(\text{Avail})-1} (-\{\theta_1 \text{ util\_km}(i) \\ & + \theta_2 \text{ util\_temp}(i,r) + \theta_3 \text{ util\_tardiness}(i) - \theta_4 \text{ util\_earliness}(i) - \theta_5 \text{ util\_next\_process}(i,r)\}) y(i,q,r)\end{aligned} \tag{35}$$

where variables util_next_process and cost_near_width represent violations of constraints 7 and 8, respectively. Consequently, when relation 15 is violated, the binary variable util_next_process will consider a value of 1 in the objective function, otherwise 0. Similarly, when relations 16 and 17 are violated, the binary variable cost_near_width will consider the value of 1 in the objective function, otherwise 0. Moreover, υ_5 and θ_5 are defined as big value coefficients.

This algorithm consists of two phases: (1) generating an initial solution and (2) improving the initial solution by guided local search.

4.1 Greedy constraint satisfaction algorithm

In the second phase, a feasible initial solution is obtained using the greedy constraint satisfaction algorithm (GCSM). This algorithm uses a set of sorts and sieves to put the best possible slab in the program. Below is a brief summary of the GCSM algorithm:

GCSM algorithm

```
For each program $r$
  For each section $q$
    Step 1. Separate the slabs feasible for section $q$ from the
            whole set of feasible slabs to be allocated to Slabs
            $(q,r)$.
    Step 2. A virtual slab should be considered as the first
            slab in section $q$.
    Repeat
      Step 3. Observing all constraints in the set Slabs$(q,r)$,
              place the set of feasible slabs in the set $S$.
      If $(S \neq \emptyset)$
        Step 4. Select the feasible slab with the best
                objective function in the set $S$ to be
                added to set $q$ and delete it from the set
                $R0$.
      Else
        Step 5. Recording one non-convergent case and
                assuming the big $M$ value, add the first
                slab feasible in terms of width to the set
                of slabs in section $q$ and delete it from the
                set $R0$.
      End IF
      Step 6. Update total length and weight of section $q$
              and program $r$.
    Until (Total length and weight of section $q$ and program $r <$
          maximum limit)
  End For
End For
```

Step 1 ensures that constraints 3, 4, and 5 are observed. In step 2, the first slab of the section will be allocated based on the last slab of the previous section. In steps 3 and 4, the problem constraints including constraints 8, 6, 1, 2, and 7 are controlled in this order. Under special conditions, no feasible slab may be available after the last one is placed in the section. Therefore, in step 5, the algorithm will place the first slab feasible in terms of width and includes a heavy penalty in the objective function. Finally, the algorithm finishes by violating length and weight constraints.

The process continues until all the capacity of the section is exhausted. Once a section is constructed, the algorithm will construct the next section of the program until an adequate number of programs is created.

4.2 Guided local search algorithm

The second phase of the algorithm is designed according to the guided local search (GLS) method first developed by Voudouris and Tsang [13]. This meta-heuristic algorithm has been shown to be successful in solving VRP on large-scale problems [11, 14]. However, GLS has not been used before for solving the MHSMS problem. This method skips local minimum by penalizing unwanted solution features and the cost function changing technique. In this phase of the proposed algorithm, modified guided local search (MGLS), the arcs among slabs are considered as solution features. If a solution has a feature, then the *indicator function* of that feature for the solution in question will be equal to 1 as follows:

$$I_{(i,j)}(\text{solution}^*) = \begin{cases} 1 & \exists q, r \ni x(i,j,q,r) = 1 \\ 0, & \text{o.w.} \end{cases} \quad ; \ i,j \in \text{Avail}, i \neq j \tag{36}$$

where solution* is a local optima. Therefore, the cost function of the indicator function $I(i,j)$ can be obtained by Eq. 37.

$$c_{(i,j)}(\text{solution}^*) = \sum_{r=1}^{m} \sum_{q=1}^{3} \mathrm{I}_{(i,j)}(\text{solution}^*)[\text{Cost}(i,j,q,r) + \text{util}(i,r)] \ ; \ i,j \in \text{Avail}, i \neq j \tag{37}$$

The indicator functions of features are directly incorporated into the problem's cost function. When the local search is trapped in a local minimum, the solution features whose utilization values (Eq. 38) are maximum are penalized by the GLS. Then, an augmented cost function (Eq. 39) is developed by adding penalty terms to the main cost function.

$$\text{utilization}_{(i,j)}(\text{solution}^*) = \mathrm{I}_{(i,j)}(\text{solution}^*) \frac{\lambda p(i,j) + c_{(i,j)}(\text{solution}^*)}{p(i,j) + 1} \tag{38}$$

$$\text{aug\_cost\_func}(\text{solution}^*) = \text{cost\_func}(\text{solution}^*) + \lambda \sum_{i=0}^{n(\text{Avail})-1} \sum_{j=0}^{n(\text{Avail})-1} I_{(i,j)}(\text{solution}^*)\, p(i,j); \ i,j \in \text{Avail}\ , i \neq j \tag{39}$$

The local search continues with the augmented cost function in the next iteration. When a feature is penalized, the penalty parameter ($p(i,j)$) of the feature increases by one unit. So if a penalized feature is not deleted, it will be penalized with less probability in the next iteration. The pseudo-code for the MGLS algorithm is shown in Fig. 5.

In the MGLS structure, the four operators *deletion*, *exchange*, *insertion*, and *relocation* have been used to generate neighborhoods in the local search. The deletion operator deletes those slabs from the program whose deletion considerably reduces the cost function. In the exchange operation, the slabs allocated to the different sections of the program are exchanged for those in the warehouse (*R0*). The insertion operator adds new slabs from the warehouse (*R0*), while the relocation operator relocates the slabs in one section.

In these operations, the best move that maintains solution feasibility is selected and applied each time. Figures 6, 7, 8, and 9 show how these sub-algorithms operate.

5 Numerical results

Numerical examples have been presented in three parts. First, parameter λ is analyzed and the best value is selected. Then, five real examples are solved and the results obtained from the proposed algorithm are compared with those from programs developed by the operator of the Production Planning and Control Division at MSC. In the third part, 15 instances have been randomly generated and a constrained random search that will be executed with a large number of iterations is used for comparing results.

5.1 Sensitivity analysis of parameter λ

The parameter λ has the main role in correctly conducting the search path in GLS. Figure 10 shows the impact of different values of λ on the trend of solution improvement in a problem with 3,000 slabs and three programs. For better exposition of the trends, and considering the presence of negative values, a large constant value was added to all objective function values and cost functions were exponentially represented.

In fact, when the λ equals 0, the impact of the GLS algorithm is neutralized so that the search algorithm takes

```
MGLS algorithm

Input: solution obtained by GCSM as the best solution
While  number of solutions is less than a specified amount
    For  $r$=1 to $m$
        For  $q$=1 to 3
            For  $i$=1 to $n(Avail)$    / $Avail(1)=0$ /
                For  $j$=1 to $n(Avail)$,  $p(Avail(i), Avail(j))=0$, End  /For $j$/
            End  /For $i$/
            $sum\_penalty=0$
            $threshold=inf$
            $not\_imp=0$
            While  $\lambda \,.\, sum\_penalty <= threshold$  and  $not\_imp<=limit\_not\_imp$
                with augmented cost function do
                    Deletion
                    Exchange
                    Insertion
                    Relocation
                If there is no improvement, $not\_imp=not\_imp+1$ otherwise $not\_imp=0$;
                    $threshold = \lambda \max_{i=2 \ to \ n(q,r)} \{cost(s(i-1,q,r),s(i,q,r),q,r)+util(s(i,q,r),r)\};$
                For  $i=2$ to $n(q, r)$
                    $utilization\ (s(i\text{-}1, q, r), s(i, q, r))=$
                    $\dfrac{\lambda\,.\,p(s(i-1,q,r),s(i,q,r))+\{cost(s(i-1,q,r),s(i,q,r),q,r)+util(s(i,q,r),r)\}}{p(s(i-1,q,r),s(i,q,r))+1}$
                End  /For $i$/
                For  $i$=2 to $n(q, r)$
                    If  $utilization\ (s(i\text{-}1,q,r), s(i,q, r))= \max_{j=2 \ to \ n(q,r)} \{utilization(s(j-1,q,r),s(j,q,r))\}$
                        $p(s(i\text{-}1,q,r), s(i,q,r))= p(s(i\text{-}1,q,r), s(i,q,r))+1;$
                        $sum\_penalty=sum\_penalty+1;$
                    End  /If/
                End  /For $i$/
            End  /While/
        End  /For $q$/
    End  /For $r$/
End  /While/
```

Fig. 5 MGLS pseudo-code

on a slow trend of improvement and stops at a local optimum solution before generating 30 million solutions. The values 5, 10, and 20 for λ cause a slow and steady search so the improvement trend continues until final iterations. The fastest improvement occurred for λ = 50 and λ = 100; however, the algorithm stops before generating 60 million solutions. The speed of the search process slightly decreases and the improvement trend continues up to higher iterations when λ is increased to 200 or 300. In fact, very large values of λ create some kinds of diversification in the search process. Finally, it seems

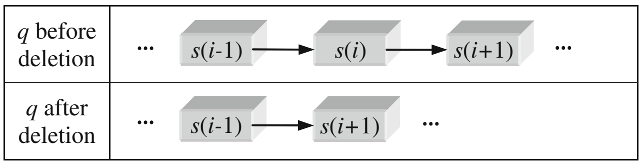


Fig. 6 Deletion operator

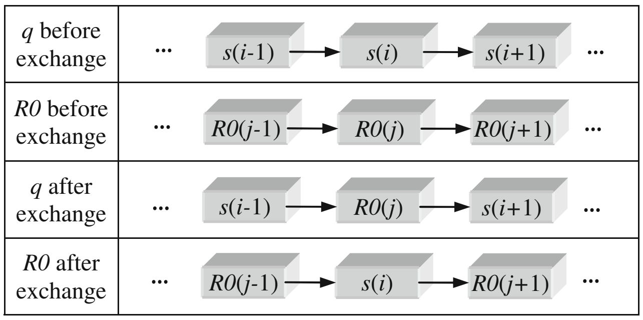

q before exchange
... s(i-1) → s(i) → s(i+1) ...
R0 before exchange
... R0(j-1) → R0(j) → R0(j+1) ...
q after exchange
... s(i-1) → R0(j) → s(i+1) ...
R0 after exchange
... R0(j-1) → s(i) → R0(j+1) ...

Fig. 7 Exchange operator

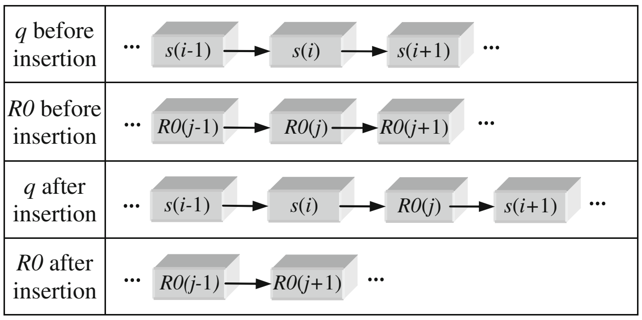

q before insertion
... s(i-1) → s(i) → s(i+1) ...
R0 before insertion
... R0(j-1) → R0(j) → R0(j+1) ...
q after insertion
... s(i-1) → s(i) → R0(j) → s(i+1) ...
R0 after insertion
... R0(j-1) → R0(j+1) ...

Fig. 8 Insertion operator

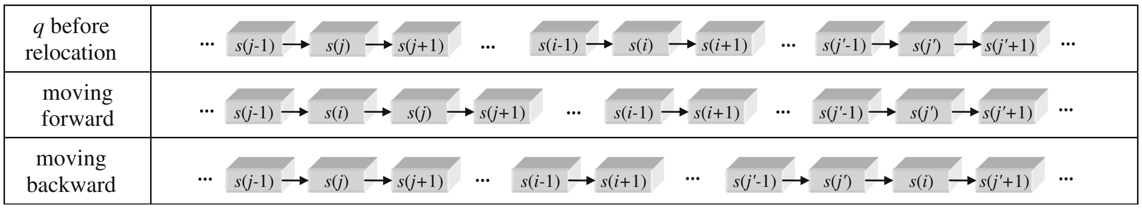

q before relocation
... $s(j\text{-}1) \rightarrow s(j) \rightarrow s(j+1)$ ... $s(i\text{-}1) \rightarrow s(i) \rightarrow s(i+1)$ ... $s(j'\text{-}1) \rightarrow s(j') \rightarrow s(j'+1)$ ...
moving forward
... $s(j\text{-}1) \rightarrow s(i) \rightarrow s(j) \rightarrow s(j+1)$ ... $s(i\text{-}1) \rightarrow s(i+1)$ ... $s(j'\text{-}1) \rightarrow s(j') \rightarrow s(j'+1)$ ...
moving backward
... $s(j\text{-}1) \rightarrow s(j) \rightarrow s(j+1)$ ... $s(i\text{-}1) \rightarrow s(i+1)$ ... $s(j'\text{-}1) \rightarrow s(j') \rightarrow s(i) \rightarrow s(j'+1)$ ...

Fig. 9 Relocation operator

that λ is better to be set to 50 without risking stopping the algorithm prematurely.

5.2 Comparison with operator's programs

The efficiencies of the GCSM and MGLS algorithms are compared with those from the programs developed by the operator of the Production Planning and Control Division. The parameters used are presented in Tables 3, 4, 5, 6, 7, 8, 9, and 10 and the results are reported in Table 11. The computational time of MGLS algorithm is shown in Table 11, which is so less than required time (about 1,800 s) by operators to make a program. Comparing with manual method, which will take about 0.5 to 1 h, the speed performance of the proposed algorithms is significant. GCSM results are relatively better than those of the operator's programs; significant differences appear, however, after the GCMS solution is improved by MGLS. Also, it is observed that operators could not consider next process and near-width constrains (Eqs. 15, 16, and 17) appropriately.

5.3 Comparison with CRS results

The CRS algorithm uses the same four techniques as used by MGLS for generating neighborhoods. CRS accepts the solutions better than the best found solution with a probability of 1 and the one better than the current solution with a probability of β in order to avoid sudden termination in local solutions. Values for parameters of MGLS algorithm are the same as those in the previous section, except θ_2 and υ_4 which are related to hot charge rolling specifications and are set to 1. Also, the parameter β in CRS algorithm is analyzed in primary experiments, and it is revealed that the algorithm is more efficient with a value of 0.2. Table 12 depicts the results of solving randomly generated problems 6–20. The problems have been generated based on information obtained from MSC. For each problem, the initial solution is achieved using GCSM and then improved by CRS and MGLS algorithms. The stopping criterion for both CRS and MGLS algorithms is equal computation time.

Fig. 10 Impact of different values of λ on solution improvement trend

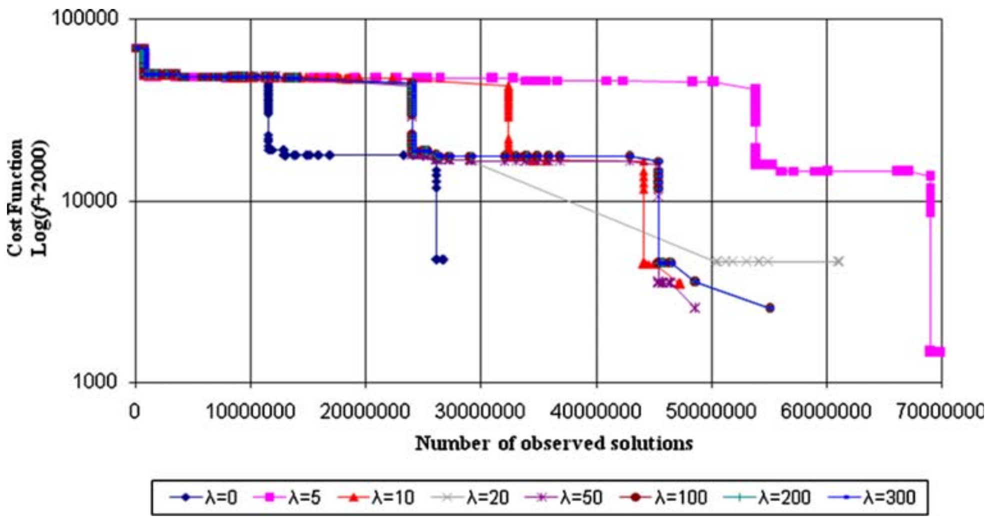


Table 3 Value of parameters

Notation	Value	Notation	Value	Notation	Value
m	1	max_km(1)	12 or 17[a]	max_diff_width(1)	+150
w_1	1	max_ton(1)	450 or 600[b]	max_diff_width(2)	±20
w_2	1	max_ton(1)	800 or 1,000[c]	max_diff_width(3)	−150
v_1	15	min_km(1)	8 or 13[d]	Inc_qual_limit	3
v_2	1	max_km_total	100	Maxw_qual_limit	6
v_3	1	min_km_total	80	Dec_qual_limit	6
v_4	0	limit_not_imp	5	min_thick_Inc	2.5
v_5	M	min_width	580	max_thick_Inc	5
θ_1	5	start_width	950	min_thick_maxw	1.5
θ_2	0	max_width	1,500	max_thick_maxw	16
θ_3	5	near_width_km_Inc	5	min_thick_Dec	1.5
θ_4	1	near_width_ton_Inc	200	max_thick_Dec	16
θ_5	M	near_width_km_Dec	15	Num_process	16
λ	50	env_temp	20	cast_temp	800

[a] If max_width<1,500 mm,12 and otherwise 17

[b] If max_width<1,500, 450 and otherwise 600

[c] If max_width<1,650 mm, 800 and otherwise 1,000

[d] If max_width<1,500 mm, 8 and otherwise 13

Table 4 Penalties due to changes in width

Increase section		Decrease section		Max. width section	
Width jump	Cost_width(*i*,*j*,1)	Cost_width(*i*,*j*,2)	Width jump	Width jump	Cost_width(*i*,*j*,3)
0 to 10	2	0 to 10	3	0 to 10	10
10 to 25	5	10 to 30	10	10 to 20	30
25 to 50	10	>30	M	20 to 30	40
75 to 100	40			30 to 40	60
100 to 150	60			40 to 50	80
150 to 200	90			50 to 70	100
200 to 250	120			70 to 90	200
>250	M			90 to 110	300
				110 to 130	400
				130 to 150	500
				150 to 200	600
				200 to 250	800
				>250	M

Table 5 The percentage of slabs with next process p

p	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\alpha(p)$%	50	3	7	4	3	1	15	0	6	0	2	1	0	1	3	4

Table 6 Penalties due to changes in quality

		Quality of slab *j*				
		1	2	3	4	5
Quality of slab *j*	1	0	2	4	8	16
	2	0	0	2	4	8
	3	0	0	0	2	4
	4	0	0	0	0	2
	5	0	0	0	0	0

Table 7 Penalties due to changes in temperature

Temp. jump	Penalty
0 to 10	0
10 to 25	10
25 to 50	20
50 to 75	30
75 to 100	40
100 to 150	70
150 to 200	100
200 to 400	200
>400	500

Table 8 The campaign number (*N*) and number of available slabs (*n* (Avail)) for problems 1, 2, 3, 4, and 5

Problem	*n*(Avail)	*N*
(1)	3,117	39
(2)	2,995	40
(3)	3,233	41
(4)	2,818	36
(5)	2,968	35

Table 9 Penalties due to changes in thick

Thick jump	Penalty
Less than normal limit	0
Between normal and max. limit	10
More than max. limit	M

Given the stochastic nature of the CRS, this algorithm has been run five times for each problem and the results of the best, worst, and average runs are reported in the relevant table. Also, the length (km), weight (ton), cost function, computational time (s), and the percent improvements for the initial solution (gap %) are shown in Table 12. The results revealed that there is a great dispersion in the CRS results, which makes them unreliable. Also, the GCSM computation time is short enough to be neglected. Moreover, the results indicate the considerable superiority of MGLS over CRS in terms of both CPU time and solution quality parameters.

Figures 11 and 12 show the trend in solution improvement by MGLS and different CRS runs for problems 7 and 12. MGLS exhibits a faster trend that continues up to higher iterations.

We developed a software application with Visual C# for this problem. The interface of our developed software is shown in Fig. 13. When a program is obtained, the planner is able to view the details of width, thickness, and quality profile of slabs via a graphical interface. If the planner is not satisfied with the current program, he/she can modify it by inserting, deleting, or exchanging slabs.

6 Conclusion

This paper investigated the HSMSP which is one of the most important planning problems in the steel industry. The problem was formulated using PCVRP which is a NP-hard problem. In the proposed formulation, necessary provisions required for obtaining an initial level of hot charge were taken into consideration.

Table 10 Thick jumping limits

Thick increase limits			Thick decrease limits		
Thick of the last slab	Thick jump		Thick of the last slab	Thick jump	
	Max.	Normal		Max.	Normal
1.5 to 2	+0.5	+1.0	1.5 to 2.49	−0.5	−0.5
2.01 to 3	+1.5	+2.0	2.5 to 3.99	−0.5	−1.0
3.01 to 5	+2.0	+3.0	4 to 5.99	−1.0	−1.5
5.01 to 8	+2.0	+4.0	6 to 7.99	−2.0	−2.5
8.01 to 12	+2.0	+4.0	8 to 11.99	−2.0	−3.0
12.01 to 16	+2.0	+4.0	12 to 16	−3.0	−4.0

Table 11 Comparison of GCSM, GCSM-MGLS, and operator

GCSM			GCSM+MGLS						Operator	
No.	Length (km)	Weight (ton)	Cost function	Length (km)	Weight (ton)	Cost function	CPU time (s)	Length (km)	Weight (ton)	Cost function
1	98.97	4,545.17	1,2064.50	98.86	4,946.13	−26.25	108.24	102.01	3,271.35	1,304.69
2	98.93	2,661.61	2,4546.13	99.98	2,994.56	−518.88	84.73	100.18	2,630.16	2,460.63
3	99.05	2,737.12	5,10.00	99.73	2,862.50	−486.25	77.47	100.17	3,147.75	5,366.31
4	98.86	3,207.44	1,363.75	99.91	3,403.83	302.13	79.19	93.00	2,707.23	2,481.56
5	98.88	2,638.47	2,9530.75	99.78	2,618.79	516.00	65.84	101.04	3,093.43	6,360.63

Table 12 Comparison of CRS and MGLS

No.	*n*	*m*	Algorithm		Length (km)	Weight (ton)	Cost function	CPU time (s)	Gap (%)
6	1,500	1	GCSM		99.09	3,600.13	190.12		
			CRS	Worst	99.82	3,634.75	116.62	67.58	38.66
				Mode	99.60	3,641.47	105.95	68.19	44.27
				Best	99.60	3,641.53	96.11	73.95	49.45
			GCSM-MGLS		99.98	3,760.57	−930.07	66.48	589.20
7	1,500	2	GCSM		183.66	6,966.68	1,1058.39		
			CRS	Worst	193.31	7,634.80	5,796.62	101.68	47.58
				Mode	192.17	7,736.74	4,819.51	102.76	56.42
				Best	199.76	8,127.93	4,646.05	100.83	57.99
			GCSM-MGLS		199.69	9,004.48	2,375.14	95.74	78.52
8	1,500	3	GCSM		293.50	11,179.13	8,4520.43		
			CRS	Worst	298.61	11,272.37	20,603.69	114.78	75.62
				Mode	299.51	11,349.03	8,506.97	102.39	89.94
				Best	299.71	11,770.53	5,364.13	95.43	93.65
			GCSM-MGLS		299.42	12,285.74	5,303.44	93.97	93.73
9	2,000	2	GCSM		186.84	7,557.40	9,292.53		
			CRS	Worst	189.61	7,771.97	3,117.19	135.82	66.45
				Mode	184.70	7,706.38	27.51	148.50	99.70
				Best	179.87	7,726.74	−204.56	134.79	102.20
			GCSM-MGLS		199.88	8,609.56	−1,028.98	129.50	111.07
10	2,000	3	GCSM		282.80	9,724.49	49,859.81		
			CRS	Worst	252.04	9,894.70	12,396.89	195.57	75.14
				Mode	279.43	12,300.85	3,336.64	202.15	93.31
				Best	265.51	9,800.88	1,349.87	187.13	97.29
			GCSM-MGLS		299.94	11,960.14	−654.92	183.67	101.31
11	2,000	4	GCSM		320.31	1,0946.81	211,192.04		
			CRS	Worst	323.53	11,353.05	27,111.75	330.25	87.16
				Mode	352.81	12,920.84	20,519.65	327.17	90.28
				Best	387.75	14,919.40	1,818.35	330.82	99.14
			GCSM-MGLS		398.50	17,678.56	−628.52	321.90	100.30
12	2,500	2	GCSM		198.71	7,574.37	26,021.82		
			CRS	Worst	197.36	7,676.56	5,973.41	172.34	77.04
				Mode	198.66	7,939.61	1,929.50	163.74	92.59
				Best	199.65	8,361.28	836.49	165.67	96.79
			GCSM-MGLS		199.91	10,713.93	−1,783.83	157.84	106.86
13	2,500	3	GCSM		261.15	10,290.88	5,856.37		
			CRS	Worst	269.07	10,810.45	1,710.70	480.80	70.79
				Mode	299.48	10,752.20	−334.88	485.47	105.72
				Best	298.35	11,599.77	−962.71	480.64	116.44
			GCSM-MGLS		261.15	10,290.88	−1,216.42	477.46	120.77

Table 12 (continued)

No.	*n*	*m*	Algorithm		Length (km)	Weight (ton)	Cost function	CPU time (s)	Gap (%)
14	2,500	4	GCSM		349.30	13,893.25	8,772.32		
			CRS	Worst	385.72	16,495.00	−78.63	182.54	100.90
				Mode	399.69	17,057.72	−303.54	283.69	103.46
				Best	388.29	17,423.75	−411.20	284.69	104.69
			GCSM-MGLS		399.60	18,174.72	−1,444.83	281.47	116.47
15	3000	3	GCSM		267.48	12,029.43	285,290.53		
			CRS	Worst	268.34	12,590.24	6,207.83	665.80	97.82
				Mode	297.34	14,572.94	1,523.80	633.34	99.47
				Best	297.24	14,739.26	1,346.70	666.50	99.53
			GCSM-MGLS		299.80	16,812.63	−2,951.83	631.35	101.03
16	3,000	4	GCSM		396.36	15,399.43	48,879.52		
			CRS	Worst	397.42	15,836.79	7,892.22	440.80	83.85
				Mode	399.59	16,203.48	774.43	408.99	98.42
				Best	399.50	16,607.74	−414.53	395.49	100.85
			GCSM-MGLS		396.36	15,399.43	231.67	393.74	99.53
17	3,000	5	GCSM		494.91	18,472.17	67,154.23		
			CRS	Worst	498.36	19,256.30	4,938.72	442.32	92.65
				Mode	499.38	19,308.31	1,926.51	424.49	97.13
				Best	498.72	18,945.05	16,101.11	464.55	76.02
			GCSM-MGLS		494.91	18,472.17	527.33	414.65	99.21
18	3,500	4	GCSM		396.51	15,499.08	32,949.81		
			CRS	Worst	397.44	17,493.48	16,663.71	603.19	49.43
				Mode	399.63	16,206.86	743.62	545.90	97.74
				Best	398.93	19,557.09	−1,195.86	560.99	103.63
			GCSM-MGLS		399.31	17,296.09	−1,489.30	537.43	104.52
19	3,500	5	GCSM		457.87	17,462.62	67,117.68		
			CRS	Worst	482.41	18,959.11	5,653.58	1053.71	91.58
				Mode	499.64	20,464.77	4,182.61	1040.71	93.77
				Best	499.11	20,707.49	3,937.06	1034.61	94.13
			GCSM-MGLS		499.40	23,486.56	2,437.39	1025.06	96.37
20	3,500	6	GCSM		581.37	23,258.62	189,208.71		
			CRS	Worst	590.06	21,722.08	5,808.29	999.38	96.93
				Mode	598.75	22,653.79	1,399.67	966.19	99.26
				Best	597.86	22,596.49	466.04	991.30	99.75
			GCSM-MGLS		599.84	28,014.93	−4,935.03	945.21	102.61

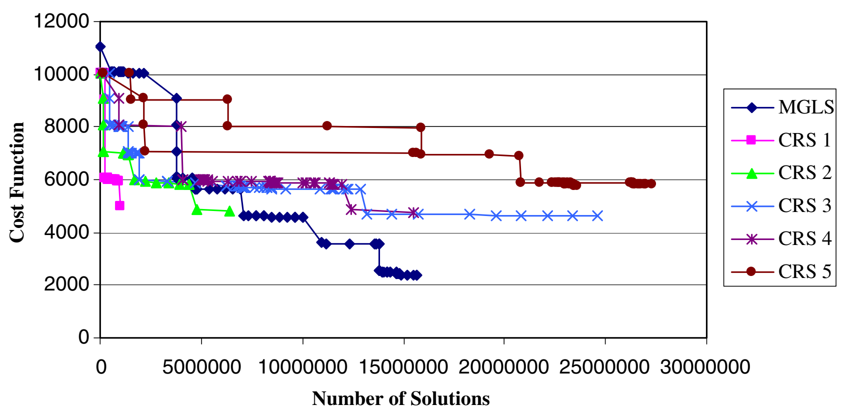


Fig. 11 Improvement trend by GCSM-MGLS and CRS (problem 7)

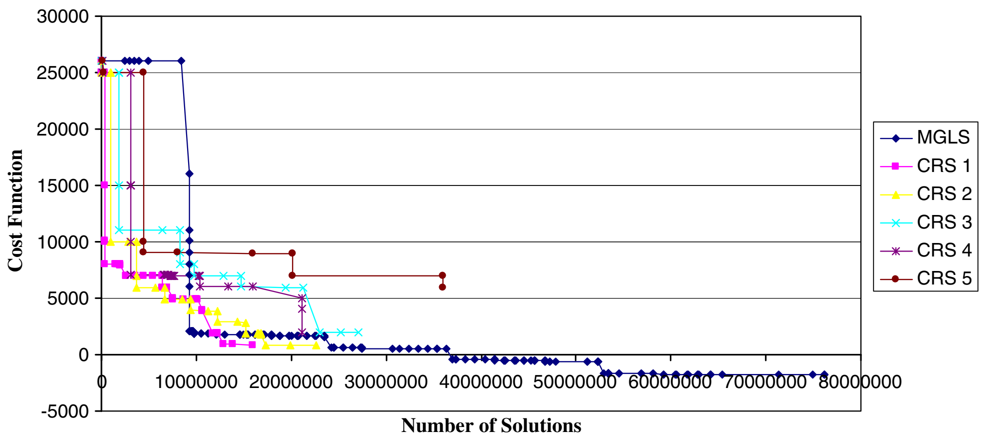
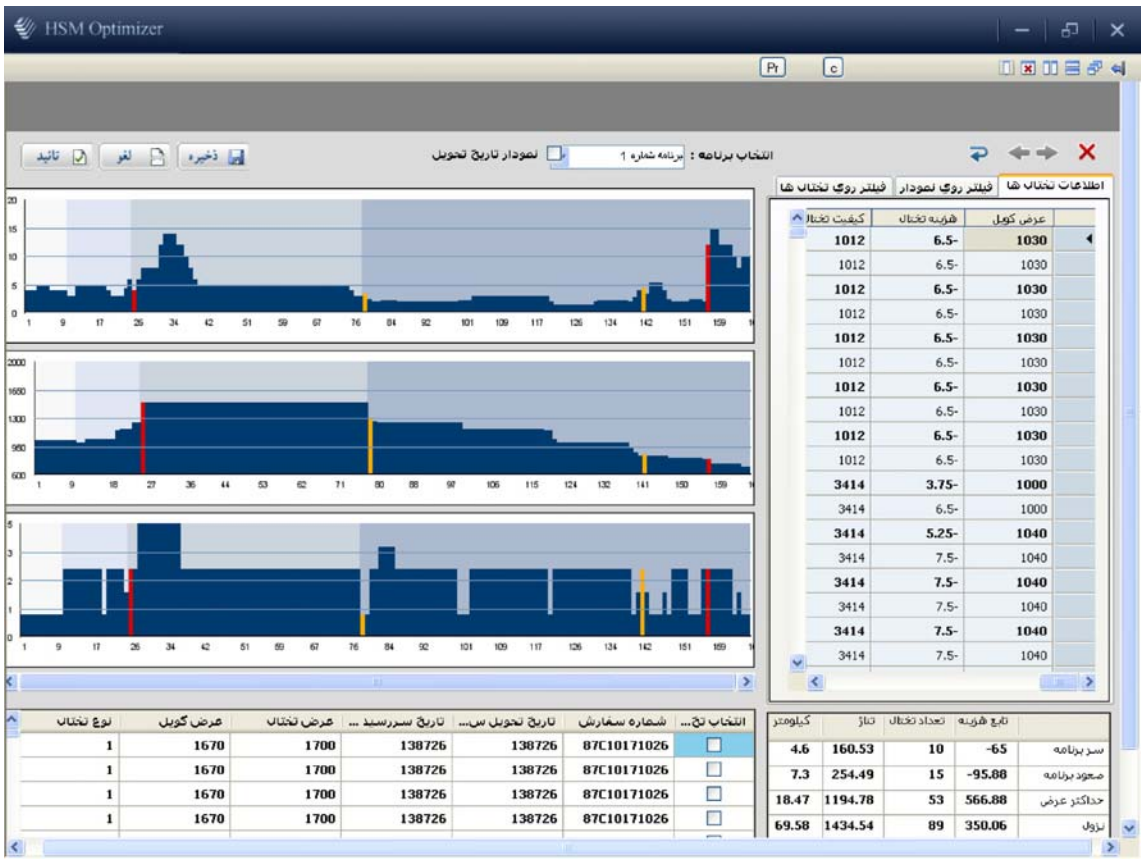


Fig. 12 Improvement trend by GCSM-MGLS and CRS (problem 12)

A search algorithm was developed that consists of three major phases: separation of slabs that can be scheduled, generation of the initial solution, and solution improvement. Generation of the initial solution and solution improvement were performed by a greedy constraint satisfaction and the GLS, respectively. The four methods of deletion, exchange, insertion, and relocation were applied to generate neighborhoods.

The efficiency of the proposed search algorithm was investigated using real and random examples. The algorithm showed a satisfactory efficiency for both problem categories. It is suggested that further study should be conducted for investigating other meta-heuristics in improving the phase of algorithm and for developing other methods of generating neighborhoods.



کیفیت تختا	هزینه تختال	عرض کویل
1012	6.5-	1030
1012	6.5-	1030
1012	6.5-	1030
1012	6.5-	1030
1012	6.5-	1030
1012	6.5-	1030
1012	6.5-	1030
1012	6.5-	1030
1012	6.5-	1030
1012	6.5-	1030
3414	3.75-	1000
3414	6.5-	1000
3414	5.25-	1040
3414	7.5-	1040
3414	7.5-	1040
3414	7.5-	1040
3414	7.5-	1040
3414	7.5-	1040

نوع تختال	عرض کویل	عرض تختال	تاریخ سررسید ...	تاریخ تحویل س...	شماره سفارش	انتخاب تخ...
1	1670	1700	138726	138726	87C10171026	☐
1	1670	1700	138726	138726	87C10171026	☐
1	1670	1700	138726	138726	87C10171026	☐
1	1670	1700	138726	138726	87C10171026	☐

کیلومتر	تناژ	تعداد تختال	تابع هزینه	
4.6	160.53	10	-65	سر برنامه
7.3	254.49	15	-95.88	صعود برنامه
18.47	1194.78	53	566.88	حداکثر عرض
69.58	1434.54	89	350.06	نزول

Fig. 13 An instance of the user interface

Acknowledgments The authors would like to thank the Mobarakeh Steel Co. and IRISA Co. for providing actual hot rolling production data and the corresponding manual scheduling results. Moreover, the authors would like to express their gratitude to Mr. Mohammad Talachian and Mrs. Narges Sereshti for their valuable comments on this paper. The authors are also grateful to anonymous referees for their suggestions regarding the improvement of this article.

References

1. Tang L, Liu J, Rong A, Yang Z (2001) A review of planning and scheduling systems and methods for integrated steel production. Eur J Oper Res 133:1–20. doi:10.1016/S0377-2217(00)00240-X
2. Tang LX, Wang XP (2005) Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. Int J Adv Manuf Technol 29:1246–1258. doi:10.1007/s00170-005-0014-0
3. Wang XP, Tang LX (2008) Integration of batching and scheduling for hot rolling production in the steel industry. Int J Adv Manuf Technol 36:431–441. doi:10.1007/s00170-006-0857-z
4. Kosiba ED, Wright JR, Cobbs AE (1992) Discrete event sequencing as a travelling salesman problem. Comput Ind 19:317–327. doi:10.1016/0166-3615(92)90069-Y
5. Lopez L, Carter MW, Gendreau M (1998) The hot strip mill production scheduling problem: a tabu search approach. Eur J Oper Res 106:317–335. doi:10.1016/S0377-2217(97)00277-4
6. Tang LX, Liu JY, Rong A, Yang Z (2000) Multiple travelling salesman problem model for hot scheduling in Shanghai Baoshan Iron & Steel Complex. Eur J Oper Res 124:267–282. doi:10.1016/S0377-2217(99)00380-X
7. Cowling P (1995) Optimization in steel hot rolling. Optimization in industry. Wiley, Chichester, England
8. Cowling P (2003) A flexible decision support system for steel hot rolling mill scheduling. Comput Ind 45:307–321. doi:10.1016/S0360-8352(03)00038-X
9. Wright JR, Houck MH (1985) An application of systems modeling in steel production scheduling. Engineering Software IV. Proceedings of the 4th International Conference. Kensington Exhibition Center, London, England, vol 15, pp 127–140
10. Balas E, Martin C (1991) Combinatorial optimization in steel rolling. In: The DIMACS/RUTCOR Workshop on Combinatorial Optimization in Science and Technology, Rutgers University, New Brunswick, NJ
11. Laporte G (2007) What you should know about the vehicle routing problem. Nav Res Logist 54:810–820
12. Monden Y (1998) Toyota production system, an integrated approach to just-in-time, 3rd edn. Engineering & Management Press, Norcross, GA
13. Voudouris C, Tsang E (1996) Partial constraint satisfaction problems and guided local search. Proceedings of Second International Conference on Practical Application of Constraint Technology (PACT'96), pp 337–356
14. Kytojoki J, Nuortio T, Braysy O, Gendreau M (2007) An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Comput Oper Res 34:2743–2757. doi:10.1016/j.cor.2005.10.010