

# A Multi-Population Parallel Genetic Algorithm for Continuous Galvanizing Line Scheduling

Muzaffer Kapanoglu  
Department of Industrial Engineering  
Eskişehir Osmangazi University  
26030, Eskişehir, Turkey  
+90 222 2303972

muzaffer@ogu.edu.tr

Ilker Ozan Koc  
Department of Industrial Engineering  
Eskişehir Osmangazi University  
26030, Eskişehir, Turkey  
+90 222 2303972

ikoc@ogu.edu.tr

## ABSTRACT

The steelmaking process consists of two phases: primary steelmaking and finishing lines. The scheduling of the continuous galvanizing lines (CGL) is regarded as the most difficult process among the finishing lines due to its multi-objective and highly-constrained nature. In this paper, we present a multi-population parallel genetic algorithm (MPGA) with a new genetic representation called  $k^{\text{th}}$  nearest neighbor representation, and with a new communication operator for performing better communication between subpopulations in the scheduling of CGL. The developed MPGA consists of two phases. Phase one generates schedules from a primary work in process (WIP) inventory filtered according to the production campaign, campaign tonnage, priorities of planning department, and the due date information of each steel coil. If the final schedule includes the violations of some constraints, module two repairs these violations by using a secondary WIP inventory of steel coils. The developed scheduling system is currently being used in a steel making company with encouraging preliminary results.

## 1. INTRODUCTION

The steelmaking process (for steel sheet products) consists of two phases: primary steelmaking and finishing lines. In the primary steelmaking, slabs created by the upstream processes are transformed to hot coils at a hot strip mill. Although some papers have been published on production scheduling in primary steelmaking ([2], [4], [5], [7]), there could be found only one research related to the finishing line scheduling [6]. Okano, et al., [6] have worked on the problem of creating production campaigns and sequencing of coils within each campaign so that productivity and product quality are maximized and tardiness is minimized. They have proposed a construction heuristic and an improvement heuristic to generate schedules for finishing lines excluding CGL which is regarded as the most difficult process in the finishing lines in terms of sequencing [6].

Multi-population GAs are the most popular parallelization method with numerous publications. A detailed discussion of parallel GAs, including multi-population (or multiple-deme) GAs, can be found in [1]. Basically, multi-population GA work with more than one population called subpopulations, and facilitates some sort of communication between the subpopulations with a communication operator based on a topology of connections. In our previous study [3], a multi-population structure with a new genetic representation, called  $k^{\text{th}}$  nearest neighbor representation, was used to generate a number of local optimum solutions quickly by using a greedy GA over each subpopulation. However, in the

MPGA approach we present here, GAs are designed to maintain different levels of greediness over different subpopulations. The level of greediness supports the robustness and efficiency of GAs. We also developed (i) a new communication operator which works on a fully connected topology, and controls the knowledge transfer and the communication frequency by incorporating certain tabu search features, and (ii) a greedy mutation operator which preserves feasibility after mutation.

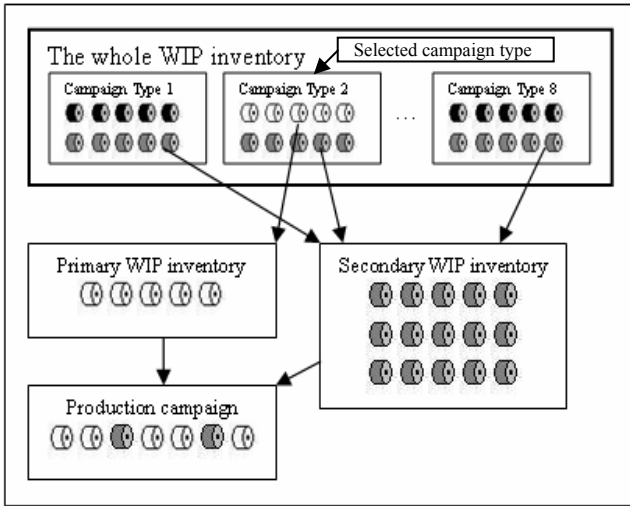
## 2. CONTINUOUS GALVANIZING LINE SCHEDULING

The considered problem is the scheduling of a continuous galvanizing line (CGL) in a flat steelmaking plant with the one-million-tonne annual production capacity. CGL scheduling is an extremely complex problem in steelmaking industries due to the following challenges. (i) Coil changeover requirements: Succeeding steel coils must comply with the changeover requirements of each equipment of CGL with respect to the preceding coils, (ii) Planning requirements: Further constraints such as the order due dates, and the priorities of the planning department, and (iii) Multi-objectivity: Schedules are subject to the optimization of objectives related to the product quality and the line productivity.

Production in the CGL is carried out in production campaigns constructed based on a selected campaign type for a given campaign tonnage. A campaign type is a cluster of coils of a particular type, with respect to quality and thickness. A production campaign is a production run with specific start and end times in which coils are processed continuously. A production campaign is constructed from the primary and secondary WIP inventories. The primary WIP inventory is the first  $N$  coils that fill the campaign tonnage when the coils of a campaign type are sorted according to the due dates and the predefined priorities. The remaining coils from the selected campaign type and the coils of different campaign types that can be used to improve the quality of the schedule are called the secondary WIP inventory. First, the coils of the primary WIP inventory are taken into the production campaign and scheduled. If the schedule violates any constraint(s), then the coils of the secondary WIP inventory are taken into the production campaign to fix the transition violations with a minimum increase in campaign tonnage. The relationship between the production types, primary WIP inventory, secondary WIP inventory, and the production campaign is shown in Figure 1. In Figure 1, the white coils are the coils of selected campaign type that fills the campaign tonnage. These white coils form the primary WIP

inventory. All the remaining coils in the selected campaign type are colored gray. Also the coils which belong to different campaign types but can be produced with the selected campaign type are colored gray. These gray coils form the secondary WIP inventory. Firstly, a production campaign is constructed with primary WIP inventory, i.e., with white coils. If there are constraint violations in production campaign, then these violations are fixed by using the secondary WIP inventory.

To make a schedule compatible with the last coil of the previous schedule and the first coil of the next schedule, it is needed to



**Figure 1. The relationship between the campaign type, primary WIP inventory, secondary WIP inventory and the production campaign**

define two coils as an input: starting coil and ending coil. The starting coil is the last coil of the previous schedule. Since the next schedule is not known beforehand, the ending coil is an artificial (dummy) coil that can be compatible with the schedule of the next campaign type. Now the scheduling process can be summarized in the following steps:

- Step 1. Determine primary and secondary WIP inventories.
- Step 2. Construct a production campaign compatible with the starting and ending coils by scheduling the primary WIP inventory.
- Step 3. If the schedule obtained is feasible, go to Step-5.
- Step 4. Repair the schedule by adding minimum number of coils from the secondary WIP inventory
- Step 5. Stop and report the final schedule of the current production campaign to the decision maker.

The CGL scheduling, as stated above, is an interesting problem since it includes many kinds of theoretical problems from the literature. In Step 2, if the starting and the ending coils are same, the problem is the shortest Hamiltonian cycle problem (i.e., well-known Traveling Salesman Problem). However, if the starting and the ending coils are not same, then the problem corresponds to the shortest Hamiltonian path problem. In Step 4, if the starting and

the ending coils are same, problem turns out to be a special case of Prize-Collecting Traveling Salesman Problem in which a traveler must visit minimum number of external nodes to find a legal tour. In Step 4, if the starting and the ending coils are different, then the problem can be defined as a "prize-collecting" Hamiltonian path problem in which a traveler must visit minimum number of nodes to find a legal path from initial node to the ending node.

The constraints of the CGL scheduling problem are presented below.

- i. The thickness difference between the two consecutive coils can be  $P_{thickness1}$ % of the thickness of thinner coil while getting thinner and  $P_{thickness2}$ % of the thinner coil while getting thicker at maximum.
- ii. The width difference between the two consecutive coils can be  $P_{width1}$  mm while getting narrower and  $P_{width2}$  mm while getting wider at maximum.
- iii. Coating thickness difference between the two consecutive coils can be  $P_{coating}$  gr per meter square at maximum.
- iv. The annealing cycle type of the two consecutive coils must correspond to a permitted transition in the annealing cycle transition matrix. The annealing cycle transition matrix shows the allowed and restricted transitions among annealing cycle types.
- v. The width enlargement in the schedule can only be made with the special coils, those with lower quality specifications or those that are non-skin-passed.
- vi. There must be at least one non-skin-passed coil between two coils requiring different skin-pass mills to operate.
- vii. The exiting inner diameter of a coil that will be side trimmed must be the same with the preceding and succeeding coils.

The objectives of the CGL scheduling problem are stated as follows:

- i. Minimize the total number of the width-increase chains where a width-increase chain is defined as a sub-schedule in which the widths of the coils continuously increase.
- ii. Minimize the total length of the width-increase chains where the length of a width-increase chain is the number of coils in a chain.
- iii. Minimize the total number of the exiting inner diameter changes.
- iv. Minimize the total number of the passivation type change.
- v. Minimize the total deviations of thickness.
- vi. Minimize the total number of oil type changes.
- vii. Minimize the total number of coils used from secondary WIP inventory.

All the constraints and the objectives are unified into a single objective function by adding up the penalized constraint violations and the weighted objective values for minimization. The objective function with user-supplied penalties and weights can be represented as follows:

$$\min z = \sum_{i=1}^7 p_i C_i + \sum_{j=1}^7 w_j O_j$$

where  $C_i$  and  $p_i$  represent the violation amount of the constraint  $i$ , and its associated penalty; and  $O_j$  and  $w_j$  represent the value of the objective  $j$  and its assigned weight respectively. This approach allows constraint violations proportional to their relative penalties. If preemptive priorities are preferred for constraints and objectives, then penalties and the weights must be chosen accordingly.

### 3. DEVELOPED MPGA

We have developed a multi-population parallel genetic algorithm using the  $k^{\text{th}}$  nearest neighbor representation [3], for preparing schedules for the CGL. This representation has previously been presented in [3] for the Euclidean traveling salesman problem with high performance. In this paper, we introduce a new approach with some extensions and modifications for a highly constrained multi-objective real CGL scheduling problem.

The multi-population GAs are the most popular parallel method among the parallel GAs with numerous publications [1]. The two important characteristics of multi-population parallel GAs are a number of subpopulations and a communication operator with a topology that defines the connections between the subpopulations. Probably the most important part of the multi-population parallel GAs is the communication operator. If there is no communication among subpopulations, then the multi-population GA exhibits an equivalent performance to running a number of individual GAs in parallel, or running a GA in multiple times sequentially. In our MPGA we have developed a new communication operator which:

- uses a fully connected topology for communication,
- exchanges some parts of two individuals selected from different subpopulations,
- utilizes a communication length parameter (*comlength*) which controls the amount of knowledge transfer among individuals,
- utilizes a tabu list to control and restrict the communication among the subpopulations which have recently communicated,
- preserves the transferred knowledge against the evolution process for a number of generations.

The main framework of MPGA is given in Figure 2. A fully connected topology is used for the communication operator and communication takes place among randomly selected two subpopulations.

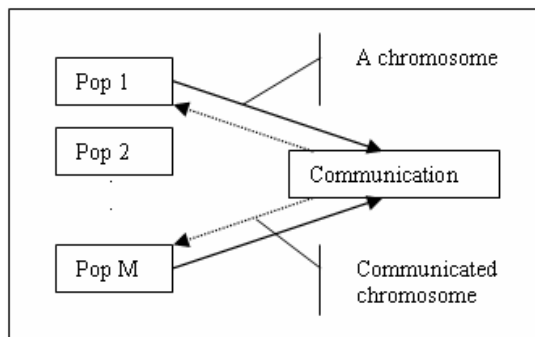


Figure 2. The framework of MPGA.

In [3], multi-population structure is used to generate a number of local optimum solutions quickly by using a greedy GA over each subpopulation. In our MPGA approach, we have used GAs that have different levels of greediness over different subpopulations. Different levels of greediness for GAs are obtained by using a different probability distribution for each subpopulation which is utilized by the  $k^{\text{th}}$  nearest neighbor representation. By this way, a GA with a lower level of greediness becomes more robust than a GA with a higher level of greediness. As the level of greediness decreases, the exploration capability of GA increases. While the level of greediness increases, the exploitation capability of the GA increases. Therefore, the communication among subpopulations enables GAs to operate on the imported knowledge with different levels of greediness. We also developed a greedy mutation operator which performs an improvement heuristic to preserve feasibility after mutation. In the following subsections we have described the genetic representation and the operators used in MPGA.

#### 3.1 $k^{\text{th}}$ nearest neighbor representation

The  $k^{\text{th}}$  nearest neighbor representation extends the nearest neighbor heuristic to the  $k^{\text{th}}$  nearest neighbor for TSPs. A fully-connected Hamiltonian graph can be constructed where the coils are nodes, and the distances are the transition costs for scheduling CGL. The transition costs from one coil to another are computed based on penalized violations of the constraints and the weighted objectives. Therefore, each GA searches for a schedule of maximum fitness in which the next coil that will be taken into the schedule can be selected from out of the unscheduled nearest  $k$  coils. The parameter  $k$  can be considered as the maximum adjacency degree due to its restrictiveness over the neighborhood of a coil. Since a gene represents the  $g^{\text{th}}$  unvisited nearest neighbor of the current coil to schedule next where  $1 < g < k$ , the maximum values of genes must be determined depending on the total number of coils, the value of  $k$ , and the position of the gene as follows;

$$\max g_i = \begin{cases} k & \text{if } N - i + 1 \geq k \\ N - i + 1 & \text{if } N - i + 1 < k \end{cases}$$

where  $g_i$  represents the value of a gene in the  $i^{\text{th}}$  position of a chromosome, and  $N$  represents the total number of coils that will be scheduled. By using the  $k^{\text{th}}$  nearest neighbor representation for  $k = 3$ , a chromosome for a 5 coil problem, with the maximum value limits for each gene, is illustrated as follows.

Maximum value limits	3	3	3	2	1
Chromosome	1	3	2	1	1

The starting coil is the last coil of the previous schedule. Since the last coil of the previous schedule (i.e., our starting coil) that is currently in production line, and the first coil of the current schedule that will be prepared must be compatible with each other, we will decide which coil to take the schedule first according to our starting coil. Since the allele of first gene is 1, we select the first nearest neighbor of the starting coil. Suppose that the first nearest neighbor of the starting coil is coil 5, now the

current schedule is {5}. Since the allele of the second gene is 3, we select the third unvisited nearest neighbor of coil 5. Suppose that this coil is coil 2, now the current schedule is {5, 2}. Since the allele of the third gene is 2, we select the second unvisited nearest neighbor of coil 2, namely coil 3. Therefore, the current schedule is {5, 2, 3}. As the subsequent chromosome is decoded in the same manner, the schedule {5, 2, 3, 1, 4} is obtained.

Since the frequency of visiting the nearest neighbor, and the  $k^{\text{th}}$  nearest neighbor in the optimal solution can not be the same, the  $k^{\text{th}}$  nearest neighbor representation utilizes a probability distribution for the degree of neighborhood. This probability distribution is used in the initialization of the subpopulation phase, and in the mutation operator to determine the new allele of a gene. Therefore, these probability distributions, each one for a subpopulation, also describe the greediness of GAs. We have defined these probability distributions according to the ratio of probabilities of adjacent closeness degrees. For example, a ratio of R indicates that the probability of visiting the first nearest neighbor is R times more than the probability of visiting the second nearest neighbor and  $R^2$  times more than visiting the third nearest neighbor and so on. These ratios are given in Table 1 for each subpopulation.

**Table 1. The ratios used to generate probability distributions for  $k^{\text{th}}$  nearest neighbor representation.**

Subpopulations	Ratio
1	1.5
2	1.4
3	1.3
4	1.2
5	1.1

Since we have many constraints related to the adjacency of coils in CGL scheduling, a feasible transition from one coil to another is often limited to a degree of closeness. By restricting the available connections from a coil to at most its  $k^{\text{th}}$  nearest neighbors, we reduce the size of the search space, and eliminate most of the infeasible transitions among coils.

### 3.2 The communication operator

We have proposed a new communication operator which acts like a crossover operator to perform knowledge exchange among individuals. The proposed operator also utilizes a *tabu list* and a *tabutenure* parameter to store the recently communicated subpopulations and to restrict them to not to re-communicate for a number of generations, respectively. By this way, subpopulations are prevented against the imported knowledge influx. Also the communicated individuals are preserved against the evolution process if their subpopulations are still in the *tabu list* (i.e., for *tabutenure* generations). The waiting times in the *tabu list* are updated by GAs. Each GA checks the index of its subpopulation at the *tabu list* after every generation and updates its waiting time if its subpopulation index exists in the list. If the index of its subpopulation doesn't exist in the *tabu list*, it sends a communication request to the communication operator with a probability of communication (*pcom*). The communication operator is activated when a request is received from a subpopulation. The operator randomly selects another

subpopulation that is not in the *tabu list*, then performs communication among two individuals randomly selected from corresponding subpopulations, and finally updates the *tabu list*.

The communication operation works on the phenotypes of the selected chromosomes. To illustrate the communication operation, assume that the solutions decoded from the selected chromosomes are as follows:

$$A = (1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9)$$

$$B = (2 \ 5 \ 4 \ 6 \ 9 \ 1 \ 3 \ 7 \ 8)$$

It randomly selects a communication site (*comsite*) from interval  $[0, N-comamount]$  where, *comamount* represents the communication amount (i.e., the length of substring that will be exchanged). Assume that *comsite* = 2 and *comamount*=4. Then, the substrings that will be exchanged are  $s_1 = (3, 4, 5, 6)$  and  $s_2 = (4, 6, 9, 1)$ . Now we will produce the second substring in parent A, and the first substring in parent B. While completing this operation, our purpose is to protect the relative positions of the sub-strings in the parents in which they will be produced. To accomplish this goal, all the elements, except the first one, in the second substring are deleted from parent A, and similarly all the elements, except the first one, in the first substring are deleted from parent B. After this operation, current schedules are reduced to (2 3 4 5 7 8) and (2 9 1 3 7 8). Adding the remaining elements of the substrings after their first element in the corresponding offspring yields the following two new schedules:

$$a = (2 \ 3 \ 4 \ 6 \ 9 \ 1 \ 5 \ 7 \ 8)$$

$$b = (2 \ 9 \ 1 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8)$$

Preserving the relative starting positions of the exchanged substrings is the main advantage of this communication operator.

### 3.3 Genetic operators and parameters

The operators and the parameters of the designed GA are described below.

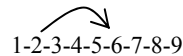
- **The maximum nearest neighborhood degree allowed:**  $k = 10$ .
- **Population size:**  $popsiz = 20$ .
- **Initial population:** Initial subpopulations are generated randomly according to the predefined probability distributions for each subpopulation, computed based on Table 1.
- **Selection operator:** We used the *tournament selection operator* with tournament size  $tsiz = 3$ . The tournament selection operator simply selects  $tsiz$  chromosomes from the current population and places the fittest one to the new population until the new population is filled.
- **Crossover operator:** We have used the single point crossover operator with probability  $pcross = 0.1$ .
- **Mutation operator:** We have proposed a new mutation operator. The mutation simply selects a gene, and mutates its

current value according to the probability distribution generated according to the ratios given in Table 1. The mutation operation is performed on the genotype. Since mutating a gene may result in a complete change in the phenotype succeeding the mutated position, we only consider the jump effect of the mutation on the phenotype. To illustrate this case, consider the following phenotype and assume that the gene at the third position is to be mutated.

1-2-3-4-5-6-7-8-9

Mutating the gene at the third position corresponds to a jump from the second position in the phenotype.

1-2-3-4-5-6-7-8-9



In this case the coils 3, 4 and 5 will be excluded from the schedule. To restore these coils to the schedule, we perform a cheapest insertion operation which inserts these coils to the cheapest available location while preserving the newly produced connection, i.e., the connection from coil 2 to 6. In cheapest insertion operation, we do not care about the “*k*” restriction over the neighborhood. Therefore, if needed, it is allowed to exceed “*k*”. The final schedule is then re-encoded into the chromosome.

- **Elitism:** Elitist strategy is used to preserve the best solution obtained against the selection, crossover and mutation operators. Elitism simply saves the best-so-far chromosome throughout generations, and replaces the worst chromosome with the elite after each generation.

#### 4. CONCLUSION

CGL scheduling in steelmaking is a challenging real-world problem incorporating multiple objectives, and multiple constraints into various types of TSP and Hamiltonian path problems. In this study, a multi-population parallel genetic algorithm with a new genetic representation and new operators is developed for this challenging problem. The developed approach produces a schedule in two phases: (i) schedule construction phase, and (ii) schedule improvement phase. Phase one schedules the primary WIP inventory which includes *N* coils selected according to the campaign type, campaign tonnage, priorities of the planning department and the due dates. Phase two is designed to repair violations by using a secondary WIP inventory for improving the quality of the schedules. Secondary WIP inventory includes the remaining coils from the selected campaign type, and some of the coils of other campaign types that are compatible with the ones in primary WIP inventory.

The developed MPGA has already been put to practice at a major steelmaking plant in Turkey. Although no detailed comparison has been completed yet to prove the contribution of the MPGA to the scheduling of CGL experimentally, the preliminary results are encouraging. High quality schedules have already been generated by using MPGA within very reasonable computational times (2-3 minutes for scheduling 150-200 coils in roughly 25-30

generations) when compared to those of the human scheduling experts.

Our intention to perform a comparison of our approach faces two drawbacks: (i) No prior research is found on CGL scheduling as highlighted in the first section. Therefore, a literature-based comparison of the performances of our MPGA and any other technique including standard GA is not available. (ii) The human scheduling experts have psychological reactions against the early successful results of MPGA which caused them to avoid many attempts in comparing their performances with MPGA’s. Although this is an on-going evaluation process that can take more than a year, a typical performance of MPGA versus human experts is presented in Table 2 for a smaller size sample case. For instance, MPGA was able to schedule all 66 coils of the primary inventory while only Expert#2 could schedule the same number of coils. Since the reason behind missing coils is to avoid some important violations, number of missing coils can also be counted as violations. Table 2 does not include all the constraints and the objectives as addressed in Section 2 due to the incapability of the human experts to evaluate more than nine criteria concurrently. For this case, the schedule obtained by using MPGA achieved 10 violations while the best expert resulted in 16 violations.

**Table 2 A sample case of 66 coils: MPGA vs. human scheduling experts**

Evaluation Criteria	MPGA	SE* #1	SE* #2	SE* #3
Number of coils	66	63	66	65
No. of violations on width differences	0	1	0	0
No. of violations on Thickness differences	0	1	1	0
No. of violations on widening	0	2	0	0
No. of violations on Annealing cycle type	1	0	1	1
No. of violations on skin-pass mill	0	0	0	0
No. of violations on inner diameter changes	8	8	10	10
No. of violations on inner diameter changes with side-trimmed coils	1	5	5	5
No. of violations on coating thickness	0	1	1	0
Total number of violations	10+0	18+3	18+0	16+1
*SE: Scheduling Expert				

According to our preliminary results, 60% to 75% improvement in the number of constraint violations, and 5-25% improvement in the objective values can be expected in a realistic realm.

#### 5. REFERENCES

- [1] Cantú-Paz, E. A survey of parallel genetic algorithms. IlliGAL Report 97003, University of Illinois, 1997.
- [2] Fang, H.-L. and Tsai, C.-H. A Genetic Algorithm Approach to Hot Strip Mill Rolling Scheduling Problems. In *Proceedings of the International Conference on Tools with*

*Artificial Intelligence*, IEEE, Piscataway, NJ, 1998, 264 – 271.

- [3] Kapanoglu, M., Koc, I.O., Kara, İ., Aktürk, M.S. Multi-population genetic algorithm using a new genetic representation for the Euclidean traveling salesman problem. In *Proceedings of the 35<sup>th</sup> International Conference on Computers & Industrial Engineering (Istanbul, Turkey, June 19-22, 2005)*. 2005, Vol. 1, 1047-1052.
- [4] Lee, H.-S., Murthy, S.S., W. Haider, S., and Morse, D. V. Primary Production Scheduling at Steelmaking Industries. *IBM J. Res. & Dev.*, 40 (1996), 231–252.
- [5] Lopez, L., Carter, M.W. and Gendreau, M. The Hot Strip Mill Production Scheduling Problem: A Tabu Search Approach, *Eur. J. Oper. Res.*, 106 (1998), 317–335.
- [6] Okano, H., Davenport, A.J., Trumbo, M., Reddy, C., Yoda, K. and Amano M. Finishing line scheduling in steel industry. *IBM J. Res. & Dev.*, 48, 5/6 (2004), 811-830.
- [7] Yasuda, H., Tokuyama, H., Tarui, K., Tanimoto, Y., and Nagano, M. Two-Stage Algorithm for Production Scheduling of Hot Strip Mill. *Operations Research*, 32 (1984), 695–707.