SCHEDULING OF MATERIAL THROUGH A STEEL PLANT

M. M. Ali*, P. Kaelo* and J. Ackerman †

Abstract

This paper addresses the optimal scheduling of material in a steel plant. The genetic algorithm is adapted to handle various constraints in the processing mills.

1 Introduction

Scheduling problems arise frequently in industry. This paper is concerned with the scheduling of steel material in a steel plant. The problem arose at Columbus Stainless Steel, a well known steel company in South Africa (Ackerman and Waldeck, 2004). The problem can be stated as follows. Given the weekly demand for the steel products, what should be the daily scheduling of steel mills in the plant in order to satisfy the demand and constraints in the mills?

Coils of various types are processed through steel mills to produce steel of various outgauges. A stockpile of coils needs to be processed through a number of available mills per day. Mills are constrained by the types of coils in that certain coils can not be processed by certain mills. Given the constraints, a scheduling algorithm has to be developed that will balance the workloads of all mills while maximizing the output material in tons. We have used a modified genetic algorithm (Ali and Törn, 2004) for this purpose.

*School of Computational and Applied Mathematics, University of Witwatersrand, Private Bag 3, Wits 2050, South Africa. *e-mail: mali@cam.wits.ac.za, pkaelo@cam.wits.ac.za*

†Columbus Stainless Steel, Middlesburg, South Africa. *e-mail: ackerman.johan @columbus.co.za*

2 The problem

In this section we present the scheduling problem. Figure 1 shows the layout of the problem. The coils to be processed are stored (see the buffer on the left hand side in Figure 1). They have different widths and incoming gauges. They also belong to two processing groups, namely Groups 1 and 2. Group 1 consists of coils of widths 1100, 1300 and 1500 and Group 2 consists of coils of widths 1100 and 1300 only. The processing of the coils is done by three mills: mill 1, mill 2 and mill 3. Mill 1 processes coils of widths 1100, 1300 and 1500 while the other two mills process coils of widths 1100 and 1300 only. However, mill 1 does not process coils of processing Group 2.

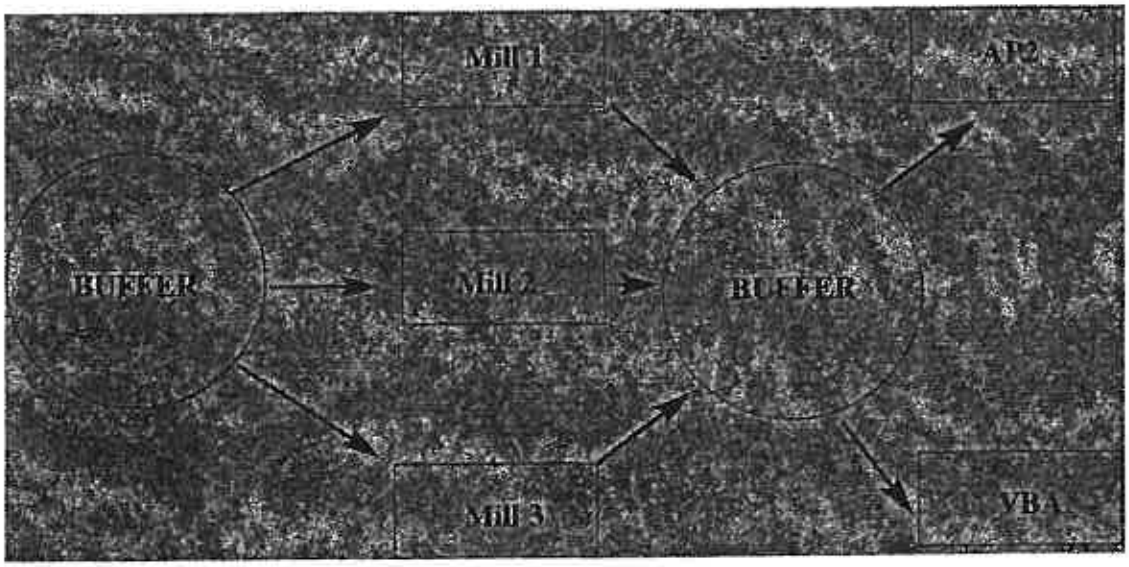

Figure 1: Problem layout.

The output from the three mills is then stored in the second buffer. The output consists of material of different outgauges. For coils of width 1500, the mill 1 produces outgauges of at least $1mm$ and for coils of widths 1100 and 1300 it produces a minimum outgauge of $0.9mm$. On the other hand, mill 2 produces a minimum outgauge of $0.4mm$ while mill 3 produces a minimum outgauge of $0.25mm$ for all coils that they process. The maximum outgauge is $6mm$. The output in the second buffer is then distributed to either Annealing and Pickling 2 (AP2) or Vertical Bright Annealing (VBA). VBA processes the output materials that have final outgauge of $0.3mm$ and materials from processing Group 2.

Each mill has a different processing time (or average tons per hour) for each coil type it processes. The problem is complex in the sense that the processing time of a coil by a mill is normally distributed with a given

mean and standard deviation for each coil. Given the constraints of the mills and data for coil types, the total processing time has to be minimized. Alternatively, the production rates of the mills (mill 1, mill 2 and mill 3) have to be maximized. We have adapted the genetic algorithm to suit the above optimization problem.

3 Genetic Algorithms (GA)

The genetic algorithm (GA) is a global optimization technique based on natural selection and the genetic reproduction mechanism (Michalewicz, 1996). GA maintains a set S, called the population set, of candidate solutions, where each solution is known as a chromosome. Central to GA is the natural evolution of the set S. At each generation of GA a new S evolves from the old S, i.e. each generation updates the set S. As the generation proceeds the set of solutions in S converges to the global minimum. In the basic GA, three steps are involved in the evolution from one generation to the next. These are as follows:

- evaluation of f at each new member of the current set S,
- stochastic selection of solutions (parents) from the current set S with a bias in the selection towards better solutions,
- reproduction of new points (children) from the selected points (parents) using the two genetic operations: crossover and mutation. The crossover operation is achieved by taking two selected solutions from S and exchanging their part(s). Mutation is achieved by simply flipping the element of chromosomes with some probability.

This cycle of evaluation, selection and reproduction, terminates after a desired number of generations.

The GA Algorithm

Step 1 **Determine the initial set** S. Generate N random solutions. Set $k = 0$.

Step 2 **Generate solutions to replace solutions in** S.

- Selection : select $m \leq N$ solutions from S as parents, with probability of each point (string) being selected proportional to its fitness.

- Crossover : pair the solutions (parents) and generate m new solutions (offspring), which replaces the m worst solutions (least fittest chromosomes) in S.
- Mutation : mutate an element of each solution (chromosome) with probability p_μ, say $p_\mu = 0.001$.

Step 3 **Update** S. Replace m bad parents in S with m new children, set $k := k + 1$ and go to Step 2.

4 Results

The standard genetic algorithm described in the previous section has been adapted to the scheduling problem considered in this paper. The adapted version differs from the standard version in that the crossover operation uses only one parent to create two children. We have also used $p_\mu = 0$. These changes have to be considered due to the special structure of the problem. The initial setting of S requires N random solutions; each random solution consists of three parts. The first part corresponds to the coils or material that are assigned to mill 1, the second part is assigned to mill 2 and the third part for mill 3. For instance a random solution (chromosome) in S has the following form:

$$x = (x_1^1, x_1^2, \cdots, x_1^{n_1}, x_2^1, x_2^2, \cdots, x_2^{n_2}, x_3^1, x_3^2, \cdots, x_3^{n_3}), \tag{1}$$

where x_i^j are random selections from the first buffer and $n_1+n_2+n_3 = M$ (M is the total number of coils to be processed). Note that the first n_1 variables represent the material that is processed by mill 1, the second batch of n_2 variables represent material processed by mill 2 and the last n_3 are processed by mill 3.

In each generation, GA calculates $2m$ solutions (children) using the existing solutions in S. It then replaces $2m$ bad parents in S with $2m$ new children. Two new solutions (children) are created by crossover using one parent. For example, if

$$x = (x_1^1, x_1^2, \cdots, x_1^{n_1}, x_2^1, x_2^2, \cdots, x_2^{n_2}, x_3^1, x_3^2, \cdots, x_3^{n_3}) \tag{2}$$

is a parent selected, then two children are produced either by crossing elements of part 1 of x (coils in mill 1) with the elements of part 2 or part 3 (coils in mill 2 or mill 3), or by crossing elements of part 2 (coils in mill 2) with elements of part 3 of x (coils in mill 3). However, when doing the

crossover, care must be taken to avoid having material that is processed by only one mill falling into the mill(s) that do not process them, so that the solutions are always feasible. Thus we do the crossover between the elements of say two mills if they can be processed by both mills. We used tournament selection (Ali and Törn, 2004) to select m parents from S, for crossover. We ran GA 10 times with different values for N and m. However, the results presented in Table 1 were obtained using $N = 50$ and $m = 3$.

In principle, n_1, n_2 and n_3 can be taken as variables. However, considering the availability of the mills per day as well as the capacity of each mill, we considered $n_1 = n_2 = n_3$ for the results presented here. We have considered three instances of the scheduling problem, namely the buffer consisting of 30 coils, 45 coils and 99 coils respectively. The full details of each group of coils can be found in Ackerman and Waldeck (2004). We only describe the optimal scheduling for the problem using 30 coils. For 150 generations, the optimal scheduling obtained by GA is presented in Table 1. The first ten rows of Table 1 are the coils that are processed by mill 1, the next ten rows are processed by mill 2 and the last ten rows are processed by mill 3. The first five columns of Table 1 show the properties of the coils, i.e. the types of the coils, group to which the coils belong, width of the coils, incoming gauges and the desired outcoming gauges respectively. Columns six to eight show the average tons per hour of each coil by mill 1, mill 2 and mill 3 respectively.

	Type	Group	Width	Ingauge	Outgauge	mill 1 tph	mill 2 tph	mill 3 tph
1	Type 1	1	1500	6	2.8	16.44		
2	Type 3	1	1500	6	6	34.91		
3	Type 2	1	1500	6	4.2	21.39		
4	Type 9	1	1500	3.2	1	18.59		
5	Type 1	1	1300	1.5	1	23.54		
6	Type 3	1	1300	1.5	1	24.95		
7	Type 2	1	1300	1.5	1	22.13		
8	Type 1	1	1100	4.5	1.8	16.26		
9	Type 1	1	1100	4.5	2.3	25.05		
10	Type 1	1	1100	5.5	2.8	25.26		
11	Type 1	2	1300	3.2	1		8.13	
12	Type 1	2	1300	3.2	1.8		18.66	
13	Type 1	2	1100	5.5	1		7.04	
14	Type 1	2	1100	4.5	1		8.03	
15	Type 2	1	1300	2.3	1		8.56	
16	Type 1	1	1100	3.2	2.3		20.14	
17	Type 5	2	1300	3.2	1		5.67	
18	Type 5	2	1300	4.5	1		9.11	
19	Type 5	2	1100	3.2	1		4.38	
20	Type 8	2	1100	1.5	1		7.32	
21	Type 2	2	1300	1.5	1			27.82
22	Type 1	2	1100	3.2	1			8.27
23	Type 3	2	1300	3.2	1			11.16
24	Type 6	2	1300	3.2	1			6.9
25	Type 1	2	1100	5.5	1.8			14.78
26	Type 7	2	1300	3.2	1			10.5
27	Type 3	1	1100	3.2	2.3			26
28	Type 1	1	1300	2.3	1			11.61
29	Type 4	1	1300	3.2	1			22.95
30	Type 1	1	1100	3.2	1.8			24.92

Table 1: Optimal solution found for 30 coils.

5 Conclusions

We adapted an algorithm for the scheduling problem based on real data. We have highlighted the complexity of the problem and modified the standard genetic algorithm for the optimal scheduling. The daily production capacity has been maximized (or daily scheduling time is minimized). The optimal scheduling is certainly more efficient than the current practice of manual scheduling. The quality of scheduling will result in savings in Columbus Stainless Steel. Further research is underway to consider large scale schedul-ing that can take a longer plaining horizon.

Acknowledgement

M.M. Ali acknowledges support of this work under the National Research Foundation of South Africa grant number 2053240.

References

J. Ackerman and G. Waldeck (2004). Columbus Stainless Steel, Middlesburg, South Africa, *Private Communication.*

M. M. Ali and A. Törn (2004). Population set based global optimization algorithms : some modifications and numerical studies, *Computers and Operations Research*, **31**, 1703-1725.

Z. Michalewicz (1996). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin.