SIMULATED ANNEALING AND GENETIC ALGORITHMS
APPLIED TO FINISHING MILL OPTIMISATION FOR HOT ROLLING OF WIDE STEEL STRIP

L. Nolle¹, D. A. Armstrong², A. A. Hopgood¹ and J. A. Ware³

¹Department of Computing and Mathematics, Nottingham Trent University
Burton Street, Nottingham, NG1 4BU, UK
Email: {lars.nolle|adrian.hopgood}@ntu.ac.uk

²Faculty of Technology, The Open University in Wales
24 Cathedral Road, Cardiff CF1 9SA, UK
Email: D.A.Armstrong@open.ac.uk

³School of Technology, University of Glamorgan
Treforest, Pontypridd, CF37 1DL, UK
Email: jaware@glam.ac.uk

Abstract: Genetic algorithms and simulated annealing have been evaluated for the optimisation of the set-up of the finishing train in a hot strip mill. Genetic algorithms proved unsuited to this optimisation problem, despite the development of a non-linear rank-based genetic algorithm that counters the effects of premature convergence and stalled evolution. The limitations of the genetic algorithm are believed to arise from epistasis, i.e. the interdependencies between input parameters. Simulated annealing, on the other hand, overcame this problem and was used for the final optimisation of the finishing train. The quality of the strip from a simulated mill was increased significantly as a result.

Keywords: Rolling, Wide steel strip, Optimisation, Genetic algorithm, Simulated annealing, Rank-based, Epistasis.

1. Introduction

There is a world-wide overcapacity for wide steel strip. In such a “buyers’ market”, producers need to offer a high quality product at a competitive price in order to retain existing customers and win new ones. Producers are under pressure to improve their productivity by automating as many tasks as possible and by optimising the process parameters to maximise efficiency and quality. One of the most critical processes is the hot rolling of the steel strip [1].

2. Problem Domain

In a rolling mill a steel slab is reduced in thickness by rolling between two driven work rolls in a mill stand (Figure 1). To a first approximation, the mass flow and the width can be treated as constant. The velocity of the outgoing strip depends on the amount of reduction. A typical hot rolling mill finishing train might have as many as 7 or 8 close-coupled stands.

2.1 Hot Mill Train

A hot-rolling mill transforms steel slabs into flat strip by reducing the input thickness $h_0$ of approx. 200 mm to an output thickness $h_7$ of approx. 2 mm.

Figure 1- Layout of a 4-high rolling mill stand.

Figure 2 shows a typical hot strip mill train, consisting of a roughing mill (stands R1-R2) and finishing stands (F1 - F7). The roughing mill usually comprises one or more stands which may operate as a reversing mill, i.e. the slabs are reduced in
thickness in several passes by going through the mill in both directions. When the slab or plate has reached the desired thickness of approximately 35 mm it is rolled by the “close-coupled” finishing stands in one pass.

Figure 2 - Hot strip mill train.

2.2 Factors Affecting the Strip Quality
The significant quality parameters of steel strip include: dimensions, profile and flatness. Strip profile is defined as variation in thickness across the width of the strip. It is usually quantified by a single value, the crown, defined as the difference in thickness between the centre line and a line at least 40 mm away from the edge of the strip (European Standard EN 10 051). Positive values represent convex strip profiles and negative values concave profiles. For satisfactory tracking during subsequent cold rolling a convex strip camber of about 0.5% – 2.5% of the final strip thickness is required [2]. Flatness – or the degree of planarity – is quantified in I-Units, smaller values of I-Units representing better flatness.

Figure 3 - Factors affecting roll gap.

(a) - unloaded rolls.

(b) - loaded rolls.

Both the flatness and the profile of the finished strip depend crucially on the geometry of the loaded gap between the top and bottom work rolls. As a consequence of the high forces employed, the work rolls bend during the rolling process, despite being supported by larger diameter back-up rolls [3].

Figure 3a shows a pair of cylindrical work rolls. In Figure 3b the effects of the loading can be seen. If the geometry of the roll gap does not match that of the in-going strip, the extra material has to be accommodated by an increase in the width of the strip, known as spread, as well as an increase in its length (Figure 4). When the thickness is less than about 8mm, spread cannot take place and there is non-uniform elongation across the width, which can result in a wavy surface [4] (Figure 5).

Figure 4 - Mismatch of roll gap and strip geometry.

Figure 5 - Wavy strip surface.

3. Optimum Finisher Set-up
The challenge is to find the finisher set-up $h_1...h_7$, for a finishing mill train with seven mill stands, that will result in optimum strip quality, i.e. fitness. In this research, an existing computer model [5] has been used to simulate the finishing stands.

In this section, the experiments for the optimisation process, using genetic algorithms (GA) and simulated annealing (SA), are documented, and the results of the final optimisation are given.

3.1 The Optimisation Loop
Figure 6 shows the closed optimisation loop, containing the mill model and a suitable optimisation algorithm.
In a finishing train, the height $h_i$ of the rolling gap of stand $i$ must be less than the height $h_{i-1}$ of the previous stand $i-1$ (Figure 7a). Otherwise, the rolls in stand $i$ would not be in contact with the strip, and would leave stand $i$ with a thickness of $h_{i-1}$ (Figure 7b).

The overall reduction $r_{\text{overall}}$ is given by the desired strip thickness $h_7$ after the last mill stand and the thickness $h_0$ of the in-going plate:

$$r_{\text{overall}} = \frac{h_7}{h_0} \quad (2)$$

Each element of the output vector $\mathbf{x} = (x_1, x_2, \ldots, x_7)$ of the optimisation algorithm must be re-scaled by multiplication with a factor $c$ in order to match the desired overall reduction:

$$r_{\text{overall}} = \prod_{i=1}^{7} r_i = \prod_{i=1}^{7} cx_i \quad (3)$$

This factor $c$ must be re-computed for every new candidate solution of the optimisation algorithm by using equation 4:

$$c = \frac{r_{\text{overall}}}{\prod_{i=1}^{7} x_i} \quad (4)$$

It can be seen that each input variable affects the relative weight of all the other input variables, because the actual thickness $h_i$ of stand $i$ depends not only on the reduction $r_i$, but also on the reduction in each of the previous stands. This parameter interaction, known as *epistasis* [6], provides a challenging optimisation problem.

### 3.2 The Fitness Function

The fitness (objective function) used is a combination of crown and flatness values of the centre-line, the edge, and the quarter-line (equation 5). To avoid a division by zero, 1 has been added to the denominator:

$$f(x, \alpha) = \frac{1}{1 + \frac{1}{\alpha} \sum_{i=1}^{3} I_i(x) + |c_{\text{aim}} - c(x)|} \quad (5)$$

where:

- $f(x, \alpha)$ fitness of solution $x$, given parameter $\alpha$,
- $\alpha$: constant to determine the relative contribution of flatness and camber, chosen to be $5000$ for the experiments,
- $I_i(x)$ I-Units at line $i$ for solution $x$,
- $c_{\text{aim}}$ target crown,
- $c(x)$ achieved crown for solution $x$. 

$\cdash$
Because the flatness values are about 5000 times greater than the differences between the target crown and the achieved camber, the constant $\alpha$ has been chosen to be 5000 for the experiments. This is to ensure the same order of magnitude for the achieved flatness and the achieved strip camber, i.e. to accord both parameters the same importance.

The theoretical maximum value of this objective function is 1.0 but, due to the non-deterministic influence of the simulated measurement devices, the target maximum of this function is 0.994. The lowest acceptable value of 0.969 is determined by the tolerances defined by the client.

### 3.3 Evaluation of Optimisation Algorithms

Two experimental optimisation algorithms have been examined: simulated annealing (SA) and genetic algorithms (GA).

#### 3.3.1 Genetic Algorithms

As Darwin’s theory of natural selection articulates, nature is very effective at optimisation by means of the simple method of trial and error, e.g. enabling life-forms to survive in an unfriendly and changing environment. Genetic algorithms simulate this evolutionary mechanism by using heredity and mutation. They were first introduced in 1975 by Holland [7] who also provided a theoretical framework for Genetic Algorithms, the Schemata Theorem [8].

For a genetic algorithm, all input parameters to a system are encoded as a binary string, the so-called genotype. The individual represented by the genotype is called a phenotype. This phenotype needs a certain quality or fitness to survive. The search is now undertaken by a population of $N$ genotypes, i.e. the gene pool, rather than a single individual. Therefore, the search space is tested at $N$ points in parallel.

The population of individuals of the gene pool at a time $t_i$ is called a generation. A new generation for time $t_{i+1}$ can be generated by selecting fit individuals (with a fitness above the average fitness of the genepool) from the current population and recombinating them with a probability $p_c$ by using the crossover operator and applying mutation with the probability $p_m$. The offspring are inserted into the new generation gene pool. The search is usually carried out until either a fitness threshold or a limit on the number of iterations has been reached.

In these experiments, a variation of GA has been used, a rank-based genetic algorithm (RBGA) [9]. Figure 8 shows the flow chart of RBGA. It uses a non-linear selection function (equation 6), which was developed in earlier work [10]. It has been demonstrated that such a GA outperforms other GAs in this problem domain [5]. The non-linear selection function $n(x,c)$ scales the exponential function so that for given input values $x \in [0,1]$ the output is between 0 and $N$:

$$n(x,c) = \left\lfloor \frac{N}{1 - e^{c-x}} \right\rfloor$$

where:

![Flowchart of the basic RBGA](image-url)
In the original Rank Based GA published by Baker [9], the best individual becomes the highest rank in this case. The degree of non-linearity can be controlled by changing the constant $c$. This allows on-line adaptation of the algorithm. For example, $c$ could be adjusted to the ratio (on-line performance/off-line performance).

For the experiments, uniform crossover [11] and two different crossover probabilities $p_c$ have been used with the RBGA.

### 3.3.2 Simulated Annealing

This general optimization method was first introduced by Kirkpatrick et al. [12], based on the work of Metropolis et al. [13]. It simulates the softening process (“annealing”) of metal. The metal is heated-up to a temperature near its melting point and then slowly cooled-down. This allows the particles to move towards an optimum energy state, with a more uniform crystalline structure. The process therefore permits some control over the microstructure.

Simulated annealing (SA) is a variation of the hill-climbing algorithm. Both start off from a randomly selected point within the search space. The difference between them is that if the fitness of a new trial solution is less than the fitness of the current solution, the trial solution is not automatically rejected, as in hill climbing. Instead it becomes the current solution with a certain transition probability $p(T)$, which depends on the difference in fitness and the temperature (Equation 7). Here, “temperature” is an abstract control parameter for the algorithm rather than a real physical measure. The transition probability $p(T)$ for a given temperature and a given difference in fitness $\Delta F$ can be determined as follows:

$$ p(T) = e^{-\frac{\Delta F}{T}} $$

where:

- $p(T)$ transition probability in the interval $[0,1]$,
- $\Delta F$ reduction in fitness of the trial solution compared with the current solution,
- $T$ current temperature.

In Figure 9, the graphical representation of Equation 6 can be seen for expected values of $\Delta F$. In order to achieve an initial transition probability of 0.5, the start temperature $T_0$ has been chosen to be 0.2.

![Figure 9 - Boltzmann function.](image)

The algorithm starts with a high temperature, which is then reduced in steps:

$$ T_{i+1} = \alpha T_i $$

where:

- $T_{i+1}$ next temperature,
- $T_i$ current temperature,
- $\alpha$ cooling coefficient.

On each step, the temperature must be held constant for an appropriate period of time (i.e. for an appropriate number of iterations) in order to allow the algorithm to settle into a “thermal equilibrium”, i.e. a balanced state. If this time is too short, the algorithm is likely to converge to a local minimum. The combination of temperature steps and cooling times is known as the “annealing schedule”, which is usually selected empirically.

Figure 10 shows the flowchart of the basic SA algorithm. The coefficient $\alpha$ has been chosen to be 0.99 in order to approach a probability of about zero after approximately 4000 iterations. The number of iterations for one temperature step has been chosen to be 50. This number was determined by experiments: smaller numbers resulted in worse solutions, larger numbers only increased the computation time.
3.3.3 Comparison between algorithms

In order to compare the methods, each algorithm has been applied 50 times. To allow for a fair comparison of the different algorithms, the term `guess` has been used rather than `generation`: one guess equals one model calculation, e.g. a GA with a gene pool containing 50 individuals carries out 500 guesses in 10 generations. Each run has been allowed 4000 guesses.

Table 1 gives the results for the average fitness found by each method, and the average number of guesses taken by the method to find the best result during one run.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Fitness</th>
<th>Standard Deviation of Fitness</th>
<th>Average Number of Guesses</th>
<th>Standard Deviation of Guesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBGA (pc=0.6)</td>
<td>0.962514</td>
<td>0.000825</td>
<td>3636.8</td>
<td>387.1</td>
</tr>
<tr>
<td>RBGA (pc=0.0)</td>
<td>0.962682</td>
<td>0.001078</td>
<td>3332.0</td>
<td>597.5</td>
</tr>
<tr>
<td>Simulated Annealing</td>
<td>0.980136</td>
<td>0.012214</td>
<td>3108.6</td>
<td>897.4</td>
</tr>
</tbody>
</table>

Table 1 - Experimental Results.

It can be observed that the RBGA has achieved an average fitness of about 0.962 in both cases, despite the very different crossover probabilities. Hence, the crossover operator has no effect on the search results.

The solutions found by the different methods have then been divided into two classes. The first class ("unsuitable solutions") contains all solutions performing below the lowest acceptable tolerance limits (fitness < 0.969, see Section 3.2). The second class ("good solutions") contains all suitable solutions, i.e. those with a fitness over the lower tolerance limits.

<table>
<thead>
<tr>
<th>Method</th>
<th>Unsuitable Solutions</th>
<th>Good Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-linear Rank Based GA (pc=0.6)</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>non-linear Rank Based GA (pc=0.0)</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Simulated Annealing</td>
<td>4%</td>
<td>96%</td>
</tr>
</tbody>
</table>

Table 2 - Class analysis.

It can be seen from Table 2 that 96% of the solutions found by SA are within the defined tolerances, i.e. are usable solutions. In contrast, none of the solutions found by GA reaches the required fitness. Hence, GA failed completely in this application. This is caused by the nature of the optimisation problem, i.e. the high degree of
epistasis present in the system. Figure 11 and Figure 12 show that only the convergence speed is affected by crossover. Due to the dependent coding of the input parameters, the GA has been degenerated into random search.

Figure 11 - Typical run of the RBGA with a crossover probability of 0.6.

Figure 12 - Typical run of the RBGA with a crossover probability 0.0.

Figure 13 shows a typical run of SA. Figure 15 shows that SA has found a solution in almost every run. The points represent the average achieved fitness, the length of the error bars the standard deviation.

Based on these results, SA has been chosen for the final optimisation.

4. Final Results

In these experiments, the finishing mill set-up has been optimised for five different slabs. The start temperature $T_0$ had been chosen to be 0.2. The annealing function was $T_{i+1} = 0.99T_i$ and the number of iterations per temperature was 50. The maximum number of guesses was 5000. Table 3 gives the details of the results.

It can be seen that in all cases considerable improvements in flatness have been achieved, although the solutions found by the algorithm did not differ greatly from the original settings. An average improvement of 66.14% has been accomplished.
<table>
<thead>
<tr>
<th>Slab No.</th>
<th>Original Flatness [I-Units]</th>
<th>Improved Flatness [I-Units]</th>
<th>Improvement [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>194.67</td>
<td>9.00</td>
<td>95.4</td>
</tr>
<tr>
<td>2</td>
<td>21.33</td>
<td>10.33</td>
<td>51.6</td>
</tr>
<tr>
<td>3</td>
<td>6.00</td>
<td>3.67</td>
<td>38.8</td>
</tr>
<tr>
<td>4</td>
<td>26.00</td>
<td>5.33</td>
<td>79.5</td>
</tr>
<tr>
<td>5</td>
<td>27.00</td>
<td>9.33</td>
<td>65.4</td>
</tr>
</tbody>
</table>

Table 3 - Comparison of original and improved flatness.

5. Conclusion

For this particular optimisation problem, two experimental optimisation algorithms have been compared: simulated annealing and a non-linear rank-based genetic algorithm.

The failure of genetic algorithms in this optimisation problem is due to epistasis, the interacting of the input parameters, i.e. one input parameter has a strong influence on the other input parameters. Simulated annealing performed demonstrably better and was therefore used to optimise the finisher set-up for five different steel slabs. The quality of the strip profile and flatness from the simulated mill was significantly increased. It has been demonstrated that simulated annealing is capable of solving optimisation problems of a physical system, even where a dependency exists between the inputs.

Acknowledgements

This work was funded by Metrology Systems Wales, Cowbridge, UK.

References