Genetic Process Optimization

John Morrow

1 Introduction

Determining optimal control settings for an industrial process can be challenging. For example, when there are interactions between the effects of the controls, adjusting one setting can require readjusting other settings. A previous article addressed this challenge using deep reinforcement learning to determine the settings for a multi-zoned reflow oven used for soldering electronic components to a circuit board (Figure 1 & Figure 2). This article addresses the same problem using genetic optimization.

Figure 1: **Reflow oven**
(image via Adobe under license to author)

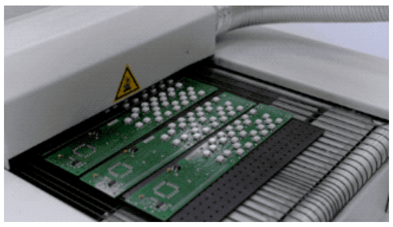

Figure 2: **Circuit boards on oven belt**
(image via Adobe under license to author)

2 The problem

The oven comprises a moving belt that transports the circuit board (product) through eight heating zones (Figure 3), each with a control for setting the temperature of the zone's heater. An additional control determines the speed of the belt.

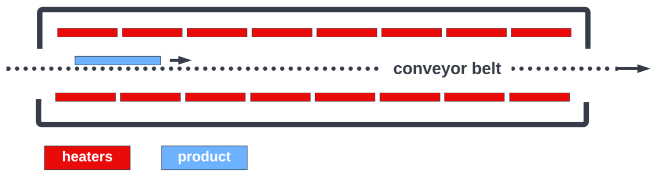

Figure 3: **Reflow oven schematic diagram**

The reflow process heats the product according to a target temperature-time profile (Figure 4) required to ensure reliable solder connections. The temperature of the product at each point along the path is determined by the heat transferred to the product from the heaters. Sensors report the temperature of the product as it travels through the oven.

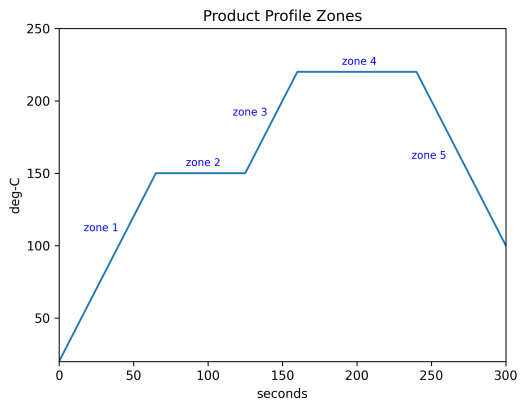

Figure 4: **Target temperature-time profile**

Before making a production run of circuit boards, an operator typically takes the following steps to determine the proper oven settings to produce the target profile:

- run one pass of the product through the oven
- observe the temperature-time profile from the sensor readings
- adjust the heater settings and belt speed to improve the profile
- wait for the oven to stabilize to the new settings
- repeat these steps until the profile from the sensor readings is acceptably close to the target profile

3 Genetic optimization

The genetic optimization solution described in this article follows a similar procedure. The algorithm models the oven's controls as a chromosome that comprises nine genes representing the eight heater controls and the belt speed (Figure 5). The genes hold the heaters' temperature settings and the belt speed.

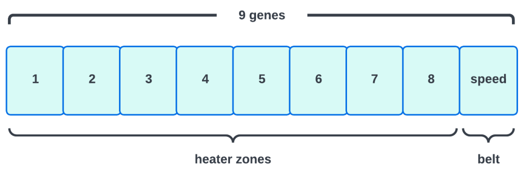

Figure 5: **Reflow oven control gene**

The algorithm begins by generating a population of oven chromosomes, each with randomly generated gene values. For each member of the population, the oven is programmed with the gene values from that member's chromosome, and the product is then passed through the oven. The resulting sensor temperature readings are compared to the target temperature-time profile to determine the fitness of that chromosome. The closer the match, the greater the fitness value.

> Note: Since considerable time is required to stabilize an oven's temperature after changing the heater settings (up to 40 minutes) and passing the product through the oven (5 minutes), an oven simulator is used to speed up the genetic learning process. The simulator emulates a single pass of the product through the oven in a few seconds compared to the minutes required by a physical oven. A detailed description of the simulator is available here.

For the next step, the algorithm starts a new generation by creating a new population based on the fitness values of the previous generation and the operations of elitism, cross-over, and mutation. Elitism adds the highest fitness-valued chromosome from the last generation to the new population. Cross-over creates new chromosomes by mixing the genes from high-fitness parent chromosomes from the previous generation. Figure 6 shows two possible child chromosomes formed from two parent chromosomes.

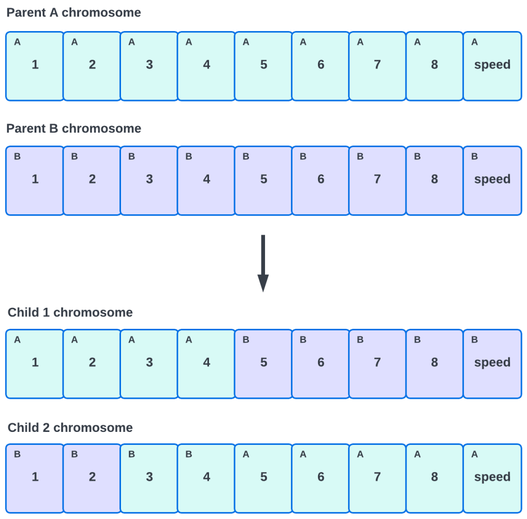
Parent A chromosome
A 1, A 2, A 3, A 4, A 5, A 6, A 7, A 8, A speed
Parent B chromosome
B 1, B 2, B 3, B 4, B 5, B 6, B 7, B 8, B speed
Child 1 chromosome
A 1, A 2, A 3, A 4, B 5, B 6, B 7, B 8, B speed
Child 2 chromosome
B 1, B 2, B 3, B 4, A 5, A 6, A 7, A 8, A speed

Figure 6: **Crossover**

Mutation randomly changes gene values in the new population (Figure 7).

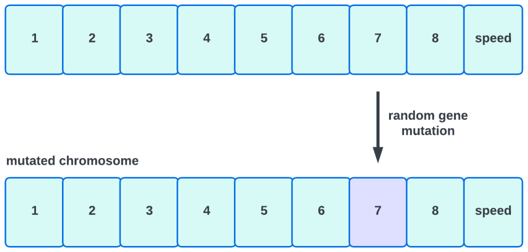

Figure 7: **Mutation**

4 Implementation

The complete Python code for this project is available on Github. The genetic algorithm is implemented with PyGAD, an extensive Python library for building genetic optimization applications.

5 Results

The results were produced with the following oven, target profile, and product parameters:

Oven Parameters	
heat transport coefficient (air)	50.0 W/degK-m**2
heater zones (1-8) lengths	0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225, 0.225 m
top heaters	on
bottom heaters	off

Target Profile Parameters

zone	start temp (degC)	slope (degC/sec)	duration (sec)
1	20	2	65
2	150	0	60
3	150	2	35
4	220	0	80
5	220	-2	60

Product Parameters	
material	FR4 circuit board
density	2000 kg/m3
specific heat	1300 J/kg-degC
thermal conductivity	0.8 w/m-degK
length	0.1 m
width	0.1 m
thickness	0.0016 m

PyGAD was instantiated with the parameters in Table 1.

Parameters	
population size	200
genes per chromosome	9
no. parents mating	40
elitism	1
crossover	single point
parent selection method	steady state selection

Table 1: Genetic algorithm parameters

Figure 8 shows the progression of the genetic algorithm from the initial population through generation 170. The best member in each successive generation brings the product temperature (red trace) in closer alignment with the target profile (blue trace). The red trace is the product temperature resulting from each generation's chromosome settings for the heaters and belt speed.

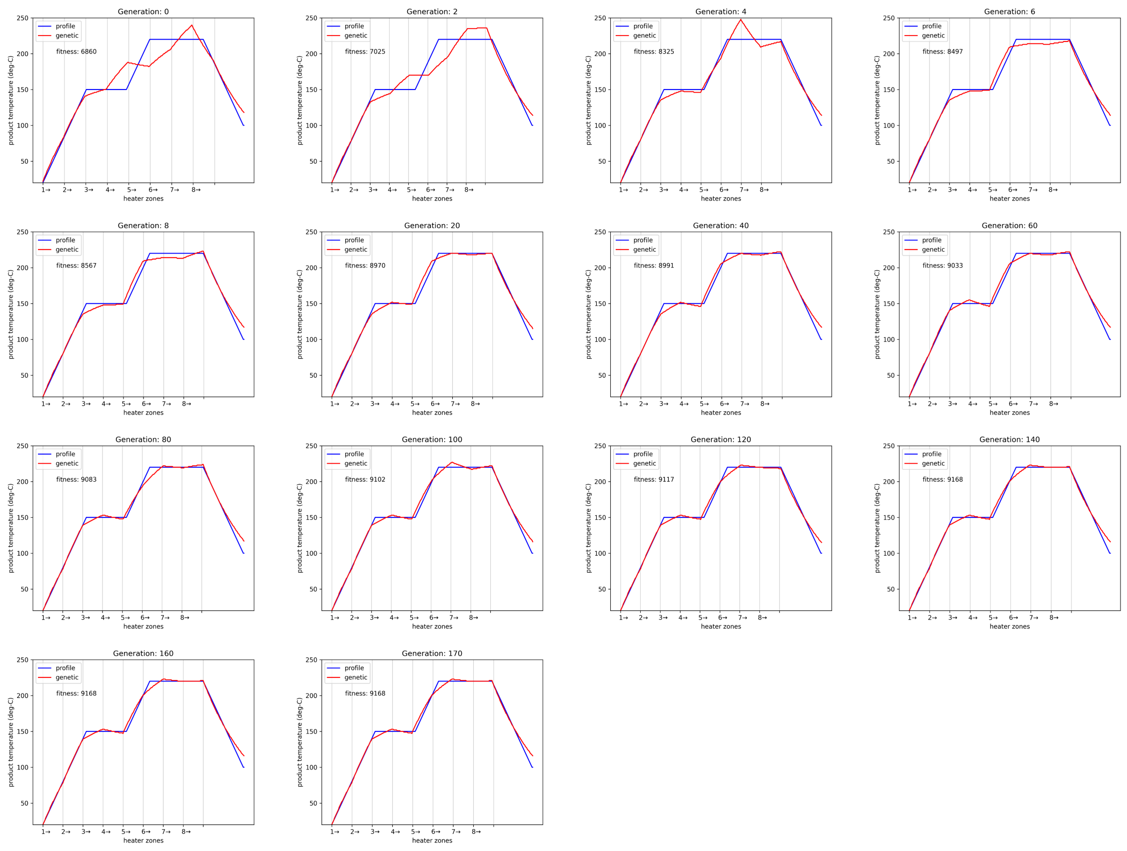

Figure 8: **Evolution sequence** (blue: target profile, red: product temperature)

Figure 9 shows the generation that first achieved the highest fitness value (with subsequent generations maintaining the same value). Even though the algorithm stabilizes at a high fitness value, it does not achieve perfect alignment between the actual product temperature and the target profile. This is partially a result of the physical constraints of the oven since there is no belt speed that can exactly match the timing of the heater zones with the target profile zones. Another constraint is that the relationship between temperature change vs. time is non-linear. Given these constraints, the chromosome settings of this generation represent the best alignment possible with the target profile. The resulting profile is acceptable for producing reliable solder connections.

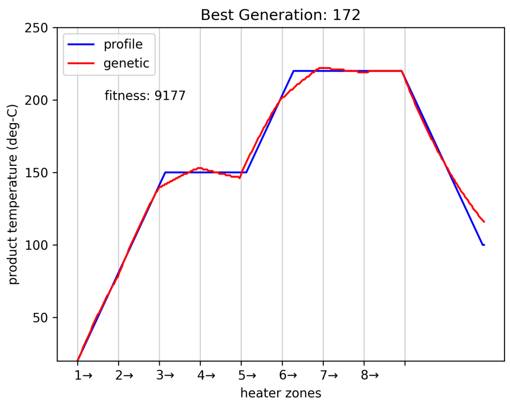

Figure 9: **Best generation**

Figure 10 is the chromosome of the best generation with gene values representing the optimal settings for the heaters and belt speed.

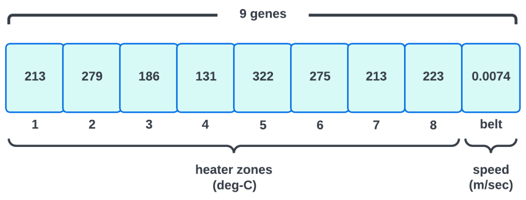

Figure 10: **Chromosome of best generation**

Figure 11 plots the improvement in fitness as the algorithm progresses through the generations.

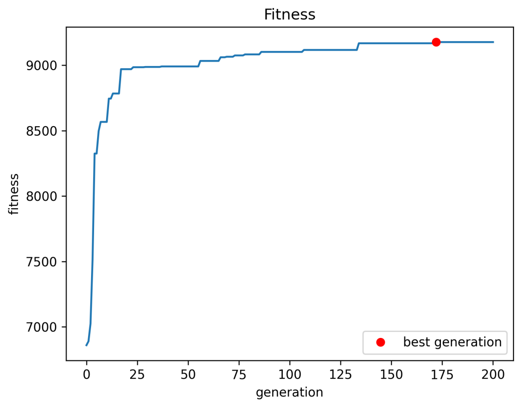

Figure 11: **Fitness**

6 Conclusion

This article presented a genetic algorithm to determine the optimal settings for a multi-stage reflow oven. The algorithm successfully determined the process settings necessary to produce a product temperature-time profile that is acceptably close to the desired target profile.