IN SEARCH FOR AN EFFICIENT PARAMETER TUNING METHOD FOR STEEL CASTING

Tea Robič
Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia
tea.robic@ijs.si

Bogdan Filipič
Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia
bogdan.filipic@ijs.si

Abstract We deal with the optimization of process parameters in industrial continuous casting of steel. The process requires fine-tuning of numerous parameters with respect to the metallurgical cooling criteria to achieve the highest possible quality of the cast steel. We tackle the problem with various optimization methods: local optimization, conjugate gradient, downhill simplex and several types of evolutionary algorithms. They search the parameter space and evaluate candidate settings using a numerical simulator of the process and a cost function defined over the metallurgical criteria. We analyze the performance of the methods with respect to effectiveness and efficiency, and compare the optimized parameter settings with the manual setting used previously at the steel plant. The best results are achieved by local optimization and conjugate gradient what suggests that the applied cost function is not complex.

Keywords: Continuous casting of steel, Process simulator, Numerical optimization methods, Local optimization, Conjugate gradient, Downhill simplex, Evolutionary algorithms, Differential evolution, Comparative study

1. Introduction

Continuous casting of steel is broadly used at modern steel plants to produce steel semi-manufactures. The quality of the cast steel is subject to many process parameters, such as the casting temperature and speed, coolant temperatures and flows, etc. The number of possible parameter settings grows exponentially with the number of parameters considered. Consecutively, finding the parameter setting that will result in high-quality steel is a demanding task. Because of high cost and safety risks, the optimization cannot be carried out on a real caster and we therefore use a numerical simulator of the casting process. However, the evaluation of solutions on the simulator is a time-consuming process. For this reason, we need to find the best parameter setting in as few process simulations as possible. In other words, we search for an optimization method that is effective and efficient.

Over the last years, several advanced computer techniques have been used in attempts to enhance the process performance and product properties in continuous casting of steel. Cheung and Garcia [3] combined a numerical model of the process with a heuristic search technique to find parameter values that reduce the proportion of defects in steel production. Filipič and Šarler [4] optimized 14 process parameters of an industrial steel casting machine with an automated software environment consisting of a numerical process simulator and evolutionary algorithm (EA). Chakraborti and coworkers [1] found genetic algorithms to be the most suitable technique for optimizing the settings of the continuous casting mold. In a follow-up study [2] based on heat transfer modeling, they used genetic algorithms to determine the maximum casting speed.

An important question in tuning the casting process parameters is which optimization algorithm to use. In this paper, we extend the collection of applied methods from generational EA and steady-state EA, to differential evolution and other optimization methods (local optimization, conjugate gradient and downhill simplex). By investigating a variety of approaches, we hope to enhance the knowledge on suitability of the optimization methods for this problem.

In Sect. 2 we outline the process of continuous casting of steel, present the simulation-based optimization procedure, and give an example of the optimization problem. The applied methods are described in Sect. 3. The numerical experiments and results are presented in Sect. 4 and discussed in Sect. 5. The paper concludes with a summary of the work done and directions for further investigation.

2. Optimization of Continuous Casting of Steel

2.1 The Process

The process of continuous casting of steel (schematically shown in Fig. 1) is a complex metallurgical process where molten steel is cooled and shaped into semi-manufactures of desired dimensions. The main components of the casting system are the ladle, tundish, mold and cooling subsystems. The ladle is used to transfer batches of molten steel from a steel-making furnace into the tundish. The tundish holds steel while casting is carried out. It ensures the continuity of steel flow into the mold. The mold is the heart of the casting system. It extracts heat from the molten steel and initiates the formation of a solid shell on

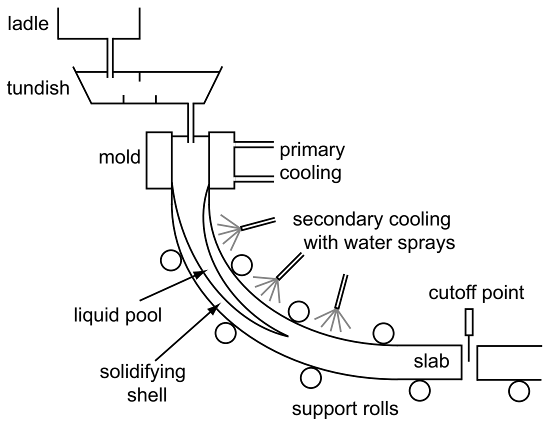

Figure 1. Schematic view of the continuous casting of steel.

the slab coming out of the mold. The mold oscillates to prevent the steel from sticking to the copper-alloy plates of the mold. Heat extraction is performed by coolant flowing through channels built in the mold. This represents the primary cooling subsystem of the caster. The heat extraction and solidification continue as the slab, led by support rolls, passes through the caster. Along the moving slab water sprays are located which form the secondary cooling subsystem. Cooling in this region results in complete solidification and the solidified slab is finally cut into pieces of the ordered length.

2.2 Simulation-Based Optimization Procedure

To tune the process parameters in continuous casting of steel, we have implemented an optimization environment consisting of the process simulator [12], a cost function and various optimization algorithms. Initially, given the process parameter values, the simulator computes the temperature field in the slab and extracts the metallurgical criteria of critical importance for the steel quality. Afterwards, the cost function value is calculated from the obtained criteria. The cost value is used by the applied optimization algorithm to generate new parameter setting and send it to the simulator. This represents one step in the simulation-based optimization procedure that operates in this manner until the stopping criterion is met.

In this study, five of the metallurgical criteria c_i provided by the simulator [12] are considered: maximum cooling and reheating rates on the slab surface in the secondary cooling zone, maximum surface temperature in the slab unbending point, and maximum negative and positive temperature deviations on the slab surface. They are taken into account in the cost function f as

$$f = \sum_{i=1}^{5} \frac{c_i - c_i^{\min}}{c_i^{\max} - c_i^{\min}}, \tag{1}$$

where $c_i^{\min}$ and $c_i^{\max}$ are the lower and upper bounds for the i-th criterion. The bounds have been determined empirically. The criteria are defined in such a way that a lower value indicates a more satisfied criterion. Thus the optimization task is to find a parameter setting that will result in the minimum value of the cost function f.

As the evaluation of parameter settings with the simulator is time-consuming, a database of the evaluated solutions is maintained. When a solution is to be evaluated, the optimization algorithm first checks for the presence of the solution in the database and activates the simulator only if the solution has not yet been evaluated.

2.3 An Example of the Optimization Problem

Let us consider an example of the optimization problem, where 12 spray coolant flows in the secondary cooling zone are subject to optimization. Table 1 shows the parameter search space for this problem. The total number of possible parameter settings equals to $5^{12} \approx 2.4 \cdot 10^8$. The task is to find the parameter setting $x^* = (x_1, \ldots, x_{12})$ that minimizes the cost function f.

Table 1. An example of the optimization problem: the search space.

Coolant flow number	*Min. value* [l/min]	*Max. value* [l/min]	*Discretization step* [l/min]	*Number of values*
1	120	160	10	5
2	65	85	5	5
3	200	280	20	5
4	190	270	20	5
5	160	240	20	5
6	150	230	20	5
7	120	160	10	5
8	140	180	10	5
9	120	160	10	5
10	120	160	10	5
11	130	170	10	5
12	120	160	10	5

3. Optimization Methods

For the optimization of continuous casting of steel, six optimization methods were tested. They include single-point iterative procedures and population-based algorithms, gradient and evolutionary methods, as well as stochastic and deterministic approaches. All methods use real vector representation of candidate solutions $x = (x_1, \dots, x_n)$, where n is the number of parameters to be optimized. The search space is discretized. The stopping criterion is the predefined number of the examined solutions. The methods are described in the following subsections.

3.1 Local Optimization

Local optimization (LO) is a simple optimization method that searches for an optimum by examining the neighborhood of the current solution. The points $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ are defined to be neighbors if, according to the given discretization, they differ by one step in exactly one dimension:

$$x_k = y_k \pm d_k \text{ for an arbitrary } k \in \{1, \dots, n\}, \text{ and } x_i = y_i \text{ for all } i \neq k. \quad (2)$$

Therefore, every point of the n-dimensional search space (except for the border points) has $2n$ neighbors. The LO procedure is presented in more detail in Algorithm 1.

Algorithm 1: Local optimization (LO)

```
randomly select an initial solution $x$
initiate direction $d := 0$
while stopping criterion not met do
  for $i := 1$ to num_neighbors do
    if $d = 0$ then
      randomly select neighbor $y$ from the neighborhood of $x$
    else
      get neighbor in the given direction $y := x + d$
    end if
    if $y$ is better than $x$ then
      remember direction $d := y - x$
      replace solution $x := y$
      break
    else
      reset direction $d := 0$
    end if
  end for
  if stuck in a local optimum then
    randomly select a new solution $x$
  end if
end while
```

3.2 Conjugate Gradient

The conjugate gradient method (CG) is used for minimizing functions f, for which the gradients f' and f'' can be computed. The method consists of iterative steps in which the search of the space is made in conjugate directions.

Two vectors (or directions) d_i and d_j are conjugate with regard to the symmetric positive definite matrix A if

$$d_i^T A\, d_j = 0. \qquad (3)$$

In an n-dimensional space, there are n conjugate directions. In CG the directions are conjugate with regard to the Hessian matrix f'' (that can be approximated by its diagonal). The CG method is outlined in Algorithm 2. Further details on the method can be found in [11].

Algorithm 2: Conjugate Gradient (CG)

```
randomly select a starting point $x_{(0)}$
calculate the preconditioner $M$ (the diagonal of the Hessian matrix $f''$)
compute the first search direction as $d_{(0)} = -M^{-1} f'(x_{(0)})$
**while** stopping criterion not met **do**
  **for** $i := 0$ **to** $n - 1$ **do**
    with the Newton-Raphson method find $\alpha_{(i)}$ that minimizes $f(x_{(i)} + \alpha_{(i)} d_{(i)})$
    the next point is $x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)}$
    the next search direction is $d_{(i+1)} = -M^{-1} f'(x_{(i+1)}) + \beta_{(i+1)} d_{(i)}$
    (the coefficient $\beta_{(i+1)}$ is calculated with the Polak-Ribiere method)
  **end for**
  restart from the best point found in the direction $d_{(0)} = -M^{-1} f'(x_{(n)})$
**end while**
```

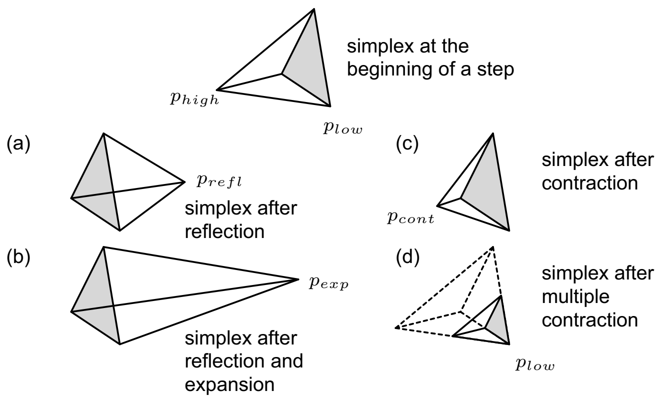
simplex at the beginning of a step
p_{high}
p_{low}
(a)
p_{refl}
simplex after reflection
(b)
p_{exp}
simplex after reflection and expansion
(c)
p_{cont}
simplex after contraction
(d)
simplex after multiple contraction
p_{low}

Figure 2. Simplex at the beginning of a step and possible outcomes for a step of DS [8].

3.3 Downhill Simplex

The downhill simplex method (DS), also named the Nelder-Mead method after its authors [7], searches for a minimum of an n-dimensional function by making use of a simplex. A simplex is a geometrical figure consisting, in n dimensions, of $n + 1$ vertices and all their interconnecting line segments, polygonal faces, etc.

DS starts with a randomly chosen simplex and takes a series of steps (reflections, expansions and contractions) which transform the simplex and move it towards the minimum. At every step the point of the simplex with the highest function value (p_{high} – "highest point") is transformed into a lower point (see Fig. 2 and Algorithm 3).

Algorithm 3: Downhill Simplex (DS)

```
randomly choose the vertices of an initial simplex
while stopping criterion not met do
  if simplex too small then
    construct a new simplex with $p_{low}$ and $n$ random points
  end if
  reflect $p_{high}$ through the opposite face of the simplex (Fig. 2(a))
  if $p_{refl}$ is lower than $p_{low}$ then
    expand the simplex in the same direction (Fig. 2(b))
    if $p_{exp}$ is lower than $p_{refl}$ then
      replace $p_{high}$ with $p_{exp}$
    else
      replace $p_{high}$ with $p_{refl}$
    end if
  else if $p_{refl}$ is lower than $p_{high}$ then
    replace $p_{high}$ with $p_{refl}$
  else
    contract the simplex (Fig. 2(c))
    if $p_{cont}$ is lower than $p_{high}$ then
      replace $p_{high}$ with $p_{cont}$
    else
      contract the simplex around $p_{low}$ (Fig. 2(d))
    end if
  end if
end while
```

3.4 Generational Evolutionary Algorithm

The generational evolutionary algorithm (GEA) is an optimization method that imitates the principles of Darwinian theory of evolution. By applying selection, crossover and mutation to a population of solutions, it creates better and better offspring populations (Algorithm 4). This method was originally studied in [6] and made popular by [5].

Algorithm 4: Generational Evolutionary Algorithm (GEA)

```
fill up the initial population pop with random solutions
while stopping criterion not met do
  create an empty population new_pop
  repeat
    select two parents from pop
    create two offspring by crossing the parents
    mutate the offspring
    add the offspring into the new population new_pop
  until new_pop full
  copy new_pop into pop
end while
```

3.5 Steady-State Evolutionary Algorithm

The steady-state evolutionary algorithm (SSEA) is similar to GEA, with the exception of maintaining a single population of solutions. Like in GEA, at every step two offspring are created by applying the evolutionary operators. But instead of filling a new population, the offspring replace the worst two individuals in the current population (Algorithm 5).

Algorithm 5: Steady-State Evolutionary Algorithm (SSEA)

```
fill up the population pop with random solutions
while stopping criterion not met do
  select two parents from pop
  create two offspring by crossing the parents
  mutate the offspring
  replace the two worst individuals in pop with the offspring
end while
```

3.6 Differential Evolution

Differential evolution (DE) is a population-based algorithm for optimizing functions on totally ordered spaces. It was developed by Price and Storn [9] as a variant of an evolutionary algorithm. The basic idea of DE is outlined in Algorithm 6.

Algorithm 6: Differential Evolution (DE)

```
evaluate the initial population $P$ of random solutions
while stopping criterion not met do
  for $i := 1$ to pop_size do
    randomly select three different individuals $I_1, I_2, I_3 \in P$
    generate the candidate solution as $C := I_1+$ weight$\cdot(I_2 - I_3)$
    alter the candidate by binomial crossover with the $i$-th individual
    evaluate the candidate
    if the candidate is better than the $i$-th individual then
      replace in $P$ the $i$-th individual with the candidate
    end if
  end for
end while
```

4. Numerical Experiments and Results

The optimization methodology was experimentally applied to continuous casting of the construction steel AC-0113 at the Acroni steel plant in Jesenice, Slovenia. The computation was performed for a slab with the cross-section of 1.03 m $\times$ 0.20 m. Out of more than 20 influential process parameters, 12 spray coolant flows in the secondary cooling zone were subject to optimization (see Sect. 2.3). The task was to check whether the manual coolant flow setting used at the plant could be improved and which optimization method is the most suitable for this problem. The calculations were run on a 1.8 GHz Pentium IV computer where the execution time to evaluate a solution through numerical simulation was 2.5 minutes.

All applied methods used real vector representation of candidate solutions. Every method was run 5 times and in each run 400 solutions were evaluated (calculated with the simulator or read from the database). LO had no additional parameters. CG was implemented as shown in Algorithm 2, i.e. with preconditioning, Newton-Raphson and Polak-Ribiere methods. DS used reflection factor 1, contraction factor 0.5 and expansion factor 2. The evolutionary methods (GEA, SSEA and DE) operated on populations of 20 individuals. Both GEA and SSEA used tournament selection with the size of tournament 2, crossover probability 0.8 and mutation probability 0.05. DE applied the strategy *DE/rand/1/bin* [10] with crossover probability 0.5 and multiplication factor 0.5.

The results of the applied optimization methods are presented statistically in Table 2, while Fig. 3 shows the improvement of the best solution cost during the optimization process. The plots represent averages over five algorithm runs and are compared with the cost of the manual setting. They are divided into two graphs to enable a better view of the results.

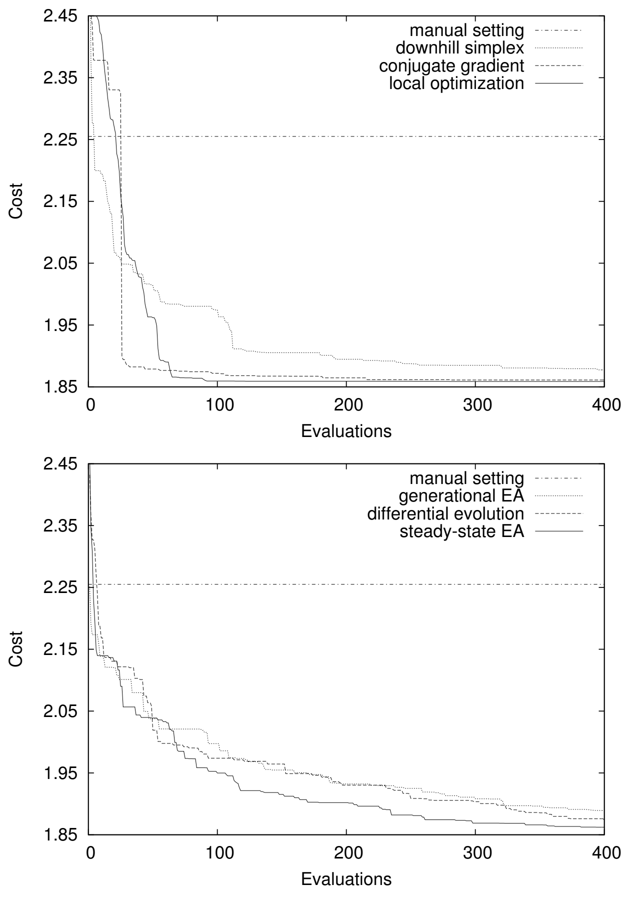

Figure 3. Performance of the optimization methods averaged over five runs.

Table 2. Results of the applied optimization methods in terms of cost given by Eq. (1).

Method Name	*Best*	*Average*	*Worst*	*St. dev.*
Generational EA	1.8638	1.8892	1.9137	0.0192
Downhill Simplex	1.8598	1.8775	1.8879	0.0137
Differential Evolution	1.8641	1.8741	1.8935	0.0116
Steady-state EA	1.8587	1.8622	1.8654	0.0028
Conjugate Gradient	1.8587	1.8612	1.8645	0.0027
Local Optimization	1.8587	1.8587	1.8587	0

All methods significantly improve the performance of the manual setting. LO outperforms all other methods by always reaching the best solution in less than

300 evaluations. The second best method is CG that makes a huge improvement in its first step (for one step the method needs 25 evaluations). SSEA is the best evolutionary method. The other methods (DE, GEA and DS) performed a little worse.

5. Discussion

The analysis of the solutions shows that the best result (cost value 1.8587) is always reached in the same point of the search space. This fact and superior performance of LO and CG over other methods indicate that the optimized function is probably unimodal. Although this outcome was not expected, it can be explained through the underlying physics. The 12 spray coolant flows subject to optimization are namely highly independent and in either monotonic or unimodal relationshis with the metallurgical criteria. Consequently, the resulting 12-dimensional cost function is also not very complex.

Although not as successful as LO, the applied evolutionary methods produce good results too, but they need more evaluations to converge (this is especially true for DE). They serve as a good comparison to LO and CG, since they are more robust and achieve good results also on more complex functions.

Our findings could be applied to similar optimization tasks in material processing. The physics behind such problems usually makes the search space simpler than expected. It is therefore a good idea to try methods like LO or CG in addition to EAs.

6. Conclusion

In the presented study, we have compared the performance of different optimization methods on process parameter tuning in continuous casting of steel. We have used an automatic optimization procedure based on a process simulator, compound cost function and various numerical optimization methods. All applied methods considerably improved the manual setting of process parameters. The best results were achieved by local optimization and the conjugate gradient method. These findings suggest that the cost function is of low complexity, most probably unimodal. However, our results are founded on certain assumptions, including the parameter intervals and discretization of the search space. With a finer discretization, the applied methods would probably perform differently. Testing the methods on different discretizations remains a task for further investigation. In addition, the improved coolant flow settings need to be practically evaluated at the plant.

Acknowledgment

The work presented in the paper was supported by the Slovenian Ministry of Education, Science and Sport under Research Programme P0-0541-0106

Intelligent Systems, and the European Commission under project COST 526: *Automatic Process Optimization in Materials Technology* (APOMAT).

References

[1] N. Chakraborti, R. Kumar, and D. Jain. A study of the continuous casting mold using a pareto-converging genetic algorithm. *Applied Mathematical Modelling*, 25:287–297, 2001.

[2] N. Chakraborti, R.S.P. Gupta, and T.K. Tiwari. Optimisation of continuous casting process using genetic algorithms: Studies of spray and radiation cooling regions. *Ironmaking and Steelmaking*, 30:273–278, 2003.

[3] N. Cheung and A. Garcia. The use of a heuristic search technique for the optimization of quality of steel billets produced by continuous casting. *Engineering Applications of Artificial Intelligence*, 14:229–238, 2001.

[4] B. Filipič and B. Šarler. Evolving parameter settings for continuous casting of steel. In: *Proceedings of the 6th European Conference on Intelligent Techniques and Soft Computing EUFIT'98*, Aachen, Germany, 1998, Vol. 1, pp. 444–449.

[5] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, 1989.

[6] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[7] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[8] W.H. Press. *Numerical Recipes in Pascal: The Art of Scientific Computing*. Cambridge University Press, New York, 1989.

[9] K.V. Price and R. Storn. Differential evolution – a simple evolution strategy for fast optimization. *Dr. Dobb's Journal*, 22(4):18–24, 1997.

[10] K.V. Price and R. Storn. Differential evolution homepage (http://www.icsi.berkeley.edu/˜storn/code.html).

[11] J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1994 (http://www.cs.cmu.edu/˜jrs/jrspapers.html).

[12] B. Šarler. Numerical procedure for calculating temperature field in continuous casting of steel. *Metals, Alloys, Technologies*, 30:217–223, 1996.