Traveling Salesman Problem: A Case Study of a Scheduling Problem in a Steelmaking Plant

Kai Krämer[1][a], Ludger van Elst[1][b] and Asier Arteaga[2]
[1]*Deutsches Forschungszentrum für künstliche Intelligenz, Germany*
[2]*Sidenor, Spain*

Keywords: Scheduling, Traveling Salesman Problem, Optimization, Industry.

Abstract: To be competitive as a company today, it is important to have key competences such as flexibility and the ability to offer a wide range of products and minimize costs. In this article, we report on an steelmaking plant and its scheduling problem. We have interpreted the optimization problem as a travelling salesman problem and show how it can be modelled. To minimize the problem we chose the simulated annealing algorithm and see how the object function can be adapted to consider-factory based constraints and how to fasten the computation time with simple techniques.

1 INTRODUCTION

Nowadays to be competetive as a company it is important to be flexible and have a wide range of products. Depending on the field of application, especially in process industry safety for the workers has to be taken into account as well. Also, wide product ranges can lead to usage of many different machineries. All this makes planning very difficult for humans, as a lot of aspects and factors have to be taken into account. The subject of this work is an application example from the field of steel production and scheduling. The object of interest is a steel making plant which is specialized in refining and producing a large variety of steel semiproducts in different formats. The plant produces small quantities per individual steel grade, but has to cover a very wide range of steel grades. As a result, retooling is often necessary or high scrap costs are incurred based on the sequence in which their products are produced. In addition, there are specific safety or metallurgical motivated rules that should be respected if possible. Overall, a lot of costs and resources can be saved through sensible scheduling of orders. To this end, formats and steel grades must be taken into account in the sequence of production, which in general is a non-trivial task. This paper reports a prototypical solution for supporting production scheduling that aims at low scrap costs while taking into account various additonal domain-specific constraints. In general, it is not known whether for a specific set of orders a solutions exists for producing them completely without constraint violations. Thus, redusing constraint violations can be seen as a second optimization goal. Such scheduling problems are among the most studied problem classes in the field of flexible manufacturing control and are often called single machine jobshop problem (JSSP) or Traveling Sales Man Problem (TSP) (Ascheuer, 1996). The classical TSP is about finding the shortest route that visits all the cities in a list once. The orders have to be produced once and can be seen as the cities and the scrap costs incurred are the distance in between. The *Simulated Annealing* (SA) approach is typical solution procedure for the TSP and is applied here in the use case example.

[a] https://orcid.org/0000-0001-9792-2690
[b] https://orcid.org/0000-0001-5965-4250

2 USE CASE DESCRIPTION

The object of interest is a special steelmaking plant which, nevertheless, shares many characteristics with conditions and needs in other process industry environments. In the overall steel production process, the steelmaking plant deals with raw material melting, refining and solidification in order to produce the semiproduct that can than be further processed, for example by costumers. In general, there are two ways of steel production, either starting with ore as the main input material or having scrap metal as main material resource. This use case only deals with the latter. The first step in steelmaking is melting iron containing raw material, most frequently scrap, in an electric arc furnace. The steel needs around 1600°C

Krämer, K., van Elst, L. and Arteaga, A.
Traveling Salesman Problem: A Case Study of a Scheduling Problem in a Steelmaking Plant.
DOI: 10.5220/0011598000003329
In *Proceedings of the 3rd International Conference on Innovative Intelligent Industrial Production and Logistics (IN4PL 2022)*, pages 291-296
ISBN: 978-989-758-612-5; ISSN: 2184-9285

to melt. The melted content is tapped to a container called ladle which determines the size or weight of the main unit called *heat* and can be seen as the product. The secondary metallurgy starts in the ladle and involves a set of processes which aim at giving the heat a specific, desired chemical composition, defined as *steel grade*. After the desired composition is obtained, the steel will be solidified in a continuous casting machine to the required semiproduct size, the so-called *format*. The formats of long products can be divided into *billets* and *blooms*, billets are smaller and blooms are bigger; steel flat semiproducts use different names. The important fact about the casting machine is that it is best suited for continuous operation. Starting and ending a heat cast requires some time and resources for preparation. Therefore heats with equal formats and similar steel grades are grouped into so called *sequences*. The heats of one such group are cast one after another in the casting machine. If the casted heats are not of the same grade, this results in mixed steel in between two heats and the mixed steel in this part of the cast has to be scrapped. The amount of scrapped material depends on the difference in composition of the two heats and can at least aproximately be calculated. This scrap costs are the basis of the object function that is to be minimized in the planning system. Intuitively, one would group as many heats as possible of the same steel grade into a long sequence, but that approach would most often not fulfill the commercial demand of the steel producer. Furthermore, it is not possible to group an arbitrary number of heats in a sequence and the sequences can not be ordered completely arbitrarily, due to metallurgical rules and safety measures given by the plant. These form the constraints of the optimization problem, that should be satisfied if possible.

3 PRELIMINARIES

In this section, we formally define the structure of the travelling salesman problem (TSP) that is used to model the problem at hand and roughly describe the simulated annealing algorithm as a solution method for the TSP.

Travelling Salesman Problem: Let $V = \{1,\dots,n\}$ a set of vertices. The Travelling Salesman Problem (TSP) can be seen as an undirected graph $G = (V,E)$ or directed graph $G = (V,A)$ where $E = \{(i, j) : i, j \in V, i < j\}$ is a set of edges or $A = \{(i, j) : i, j \in V, i \neq j\}$ a set of arcs. The graphs are 2-regular and path-connected. A distance matrix $D = (d_{ij}) \in \mathbb{R}^{n\times n}$ is defined on E or on A where $d_{i,j}$ is the travelling distance between the vertices $i, j \in V$ (Matai et al., 2010). In our case we consider the asymmetric TSP. A very typical formulation is a linear integer programming formulation (LP) (Liu et al., 2008; Matai et al., 2010). Because of complexity of some real world problems, it can be very difficult to formulate their extra constraints as LP's. Therefore a metaheuristic approach can be more suitable alternative (Pasotti and Zavanella, 2007). A route can be described as permutation of V (Johnson and McGeoch, 1997):

$$\pi = (\pi(1), \pi(2), \dots, \pi(n)) \tag{1}$$

In most definitions the route has to be closed, but in our case we leave out the last edge between the start and end point. The objective of the TSP is to minimize the total distance

$$\min_{\pi} f(\pi) \tag{2}$$

with

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)}. \tag{3}$$

A typical solution method for this problem is the simulated annealing algorithm.

Simulated Annealing is a tour improvement method. After an initial generated route x, the algorithm tries to improve the quality of the route by generating a new route y with small changes like swap, inverse or insert operations (Zhan et al., 2016). Here, we call them shuffle functions. Key of the simulated annealing is that in principle moves can be accepted that make the route worse. The probability of accepting worse solutions decreases over the time and is depicted by the decreasing temperature parameter T and the acceptance function:

$$p(T) = \begin{cases} 1 & \text{, if } f(y) \leq f(x) \\ exp(\frac{f(x)-f(y)}{T}) & \text{, else} \end{cases} \tag{4}$$

That can help to avoid local minima. A pseudo code desciption of the SA algorithm can be seen in table 1.

4 MODELING OF THE PROBLEM

As explained in the previous section, the TSP is a combinatorial graph problem. The cities are the nodes and the connections with distances are the edges. Given a list of n cities, the TSP is to find the shortest route visiting each city just once.

4.1 Use Case TSP Connection

In the case of steel production, the input of the algorithm is a given order list with n heats that have be

Table 1: Simulated Annealing pseudo code.

```
T ← temperature
a ← cooling factor ∈ (0,1)
x ← initial generated route
cost_0 ← f(x)
while determine criterium of SA not met do:
    T ← a * T
    y ← shuffle(x)
    cost_1 ← f(y)
    if cost_1 < cost_0 do
        cost_0 ← cost_1
    else do
        r ← random(0,1)
        if r < exp((cost_0 − cost_1)/T) do
            cost_0 ← cost_1
```

produced. The switch from one heat to another generates scrap costs, based on properties of the heats. These costs can be seen as a distance and the goal is find a sequence of the production with the lowest costs. The steel plant produces a large number of steel grades with different formats, various bloom and billet formats. Due to factory based conditions, the billet and bloom formats cannot be freely selected and are subject to a set of constraining rules. Each time an *order list* $O = \{o_1, o_2, \cdots, o_n\}$ has to be produced with a finite number of *orders*. An order o_i is unique and has additional information like *steel grade* s_i and *format* f_i.

$$o_i = (s_i, f_i) \tag{5}$$

We call an order list with a series a *plan* P. The production time per product is independent from format and steel grade and can be assumed to be constant. Between two orders, *scrap costs* $sc(o_i, o_j)$ arise, depending on the steel grade and format. This cost function is asymmetric. This is the basis for an asymmetric TSP.

4.2 Connection of Object Function and Constraints

In addition, a plan P is composed of partial *sequences*

$$P = (sq_1, sq_2, .., sq_m) \tag{6}$$

$$= (\underbrace{(o_1, o_2, .., o_{i-1})}_{sq_1}, \underbrace{(o_i, o_{i+1}, ..)}_{sq_2}, .., \underbrace{(.., o_{n-1}, o_n)}_{sq_m}) \tag{7}$$

where sq_i are disjoint sub-plans. The reason for this is that there is less preparation time within these sequences and saves scrap costs. Moreover, it exists a set of specific sequence rules *SQR* where all the rules have the following form:

Heat-to-Heat Rule: A sequence ends depending on the information of two consecutive heats. An trivial example would be that a sequence ends if the format changes.

Sequence Rule: A sequence ends depending on the information of a complete sequence. An example would be that the maximum allowed length per sequence is dependent on the single heats of the respective sequence.

For reasons of confidentiality, we can not provide additional detailed information about these rules. Note that the number of sequences sq for a plan is dependent on permutation of the plan. That makes it difficult to model the problem as a multi-route TSP. These sequences influence the scrap cost of a complete plan. Costs are either eliminated or added based on factory rules. Therefore the distance function is dependent on the order of the complete plan:

$$dist(P) = \sum_{i=1}^{n} sc_{SQR}((o_i, o_{i+1}), P), \tag{8}$$

where sc_{SQR} is the scrap cost function that takes into account the order of the plan P and the sequence rules. Moreover, it exists a set of constraints C that does not allow different combinations or sequences of products and must be avoided if possible. The constraints are modelled by a *heuristic cost function hdist*. (Dahal et al., 2000) proposed a sum of object function and penalty function for violations of the constraints. In our case, *hdist* is the scrap costs of the whole plan with addition of weighted penalties *pdist*.

$$hdist(P) = dist(P) + pdist(P) \tag{9}$$

with

$$pdist(P) = \sum_{c_i \in C} g_{c_i}(P) * \omega_{c_i} \tag{10}$$

where $g_{c_i}(P)$ counts the constraint violations of constraint c_i in the plan P and ω_{c_i} is the corresponding weighting coefficient. Depending on the weights ω_{c_i} in the *penalty distance pdist*, the focus is on the minimization of constraint violations or the scrap costs. More precisely each constraint can be weighted individually. It is a balancing act as to which is more important, scrap costs or constraints.

4.3 SA Simple Improvements

Note that each time *hdist* is calculated, the entire plan must be checked for the sequence rules and constraint violations, which takes a lot computation time. We used a standard SA approach to minimize the TSP with the often used shuffle functions swap, insert and inverse as noted in (Zhan et al., 2016). In each iteration the shuffle function was chosen by random among them.

Swap (1): switch element *i* and element *j*.

$$(..,o_{i-1},o_i,o_{i+1},..,o_{j-1},o_j,o_{j+1},..) \mapsto \quad (11)$$
$$(..,o_{i-1},o_j,o_{i+1},..,o_{j-1},o_i,o_{j+1},..) \quad (12)$$

Inverse (2): invert the order from element *i* to *j*

$$(..,o_i,o_{i+1}..,o_{j-1}o_j,..) \mapsto \quad (13)$$
$$(..,o_j,o_{j-1},..,o_{i+1},o_i,..) \quad (14)$$

Insert (3): element *i* insert into position *p*.

$$(..,o_{i-1},o_i,o_{i+1},..) \mapsto \quad (15)$$
$$(..,o_{i-1},o_{i+1},..,o_{p-1},o_i,o_p,..) \quad (16)$$

Based on expertise, we have added two more shuffle functions: shift cluster and shift steel grade cluster.

Shift Cluster (4): put the cluster from element *i* to element *j* into position *p*.

$$(..o_{i-1},o_i,..,o_j,o_{j+1},..o_p..) \mapsto \quad (17)$$
$$(..,o_{i-1},o_{j+1},..,o_{p-1},o_i,..,o_j,o_p,o_{p+1},..) \quad (18)$$

Shift Steel Grade Cluster (5): A version of the shift cluster, but shift a cluster of elements that have similar steel grade into position *p*. Take element *i* and all elements with a similar steel grade next to it and put it into position *p*. Here is an example:

$$(..a,\underbrace{b,b_i,b,b}_{\text{steel cluster}},c,..o_p..) \mapsto \quad (19)$$
$$(..,a,c,..,o_{p-1},\underbrace{b,b_i,b,b}_{\text{steel cluster}},o_p,..) \quad (20)$$

The indices mark only the position and a,b,c mark only the steel grade types. The idea is to not destroy parts of a route which are known to be of good quality.

The shift cluster is a more general version of the insert. The shift steel grade cluster function is a special case of the shift cluster taking domain knowledge into account and can be seen as shifting 'good' subroutes. However, only using the sift steel grade cluster function will probably prune search space too much and we could miss unintuitive solutions. After testing, how often one of these shuffle functions generate an improved state of the plan *P*, we decided to only use function (4) and (5). This helped to reduce the number of iteration to run for similar results. As earlier mentioned, *hdist* is a slow function in our case; therefore a two step-approach was chosen, similar to (Pasotti and Zavanella, 2007)'s case. The difference is that we used the SA algorithm in both steps but with different heuristics. Step one used the much faster cost function *dist* and higher start Temperature to get an initial good solution in terms of scrap costs and sequences. In step two, *hdist* was used to improve the plan from step one with respect to the constraints. As mentioned, *hdist* is slow in our case and in order to not destroy too much of the plan from step 1, it makes sense to use a low starting temperature proposed as an possible speed-up technique by (Johnson and McGeoch, 1997) and (Dahal et al., 2000) or directly use a greedy up-hill climb.

5 RESULTS

We compared manually scheduled, historical plans with the standard one step SA approach and the two step approach. In case of the one step SA approach (SA1), see table 1, we used the classical shuffle functions swap(1), inverse(2), insert(3) randomly in each loop. The object function was the slow distance function *hdist* which takes into account the scrap costs and the constraints. For the two step SA approach (SA2) we used the customized shuffle functions shift cluster(3) and shift steel grade cluster(4). In the first step, we run the SA with the much faster distance function *dist* as object function. This allowed to run much more iterations in a shorter time. In the second step, we run the SA algorithm with a very low temperature and the plan generated by step one. The object function was *hdist*.

As test, we picked five example order lists $O_1,..,O_5$ with 77, 140, 85, 73 and 95 orders. Our goal was to generate good solutions in a reasonable amount of time and get similar or better results than the historical plans. The following table 2 shows the results of the historical plans. "Scrap" is in an unspecified unit of measure, "num sqe" is the number of sequences and "violations" gives information on the number of

Table 2: shows result information of historical plans.

	scrap	num seq	violations
O_1	422	25	2
O_2	572	43	2
O_3	445	27	1
O_4	225	24	0
O_5	338	28	1

constraints not respected. We run SA1 with 20000 iterations which took around 15 to 25 minutes dependent on the length of the order list. In SA2 the main time consumption is the second step, which we run with 5000 iterations. SA1 and the second step of SA2 have the same computational effort per iteration. Therefore the expected time for SA2 is a quarter of SA1 plus the short first step. In comparison, SA needed around 4 to 8 minutes which is about a third of the SA1 time and corresponds to the expectation. The following tables Table 3, Table 4 show the results.

Table 3: Shows the results of one step SA with 20000 iterations.

	scrap	num seq	violations	time
O_1	400	28	0	845s
O_2	620	58	2	1510s
O_3	409	32	0	902s
O_4	202	23	0	664s
O_5	286	33	2	915s

Table 4: Shows the results of two step SA with 5000 iterations in step 2 and 35000 iterations in step one.

	scrap	num seq	violations	time
O_1	373	27	0	232s
O_2	545	49	2	435s
O_3	409	29	0	260s
O_4	200	22	0	173s
O_5	270	31	2	238s

We can see that in terms of constraint violations, both SA1 and SA2 perform better than the historical plans with the exception of O_5. SA1 has its problems with the longest order list O_2 regarding scrap cost and has its limitations with only 20000 iterations. Furthermore we can see, that in our specific use case the SA2 outperforms SA1 in all important aspects scrap and computation time.

In the following figures we see the behaviour of the scrap cost per iteration. In figure 1 the scrap cost decreases slowly as expected of SA1.

In figure 2 the scrap cost behaviour of SA2 is shown. The sudden rise at the end indicates the second step. Step one generates a good basis plan regarding only scrap cost and the second step fixes the constraint violations which on the other hand, partially increased the scrap again.

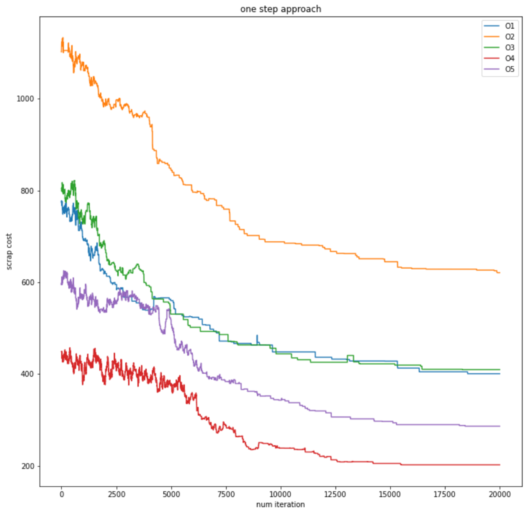

Figure 1: Shows the behaviour of the scrap cost over the iterations from SA1.

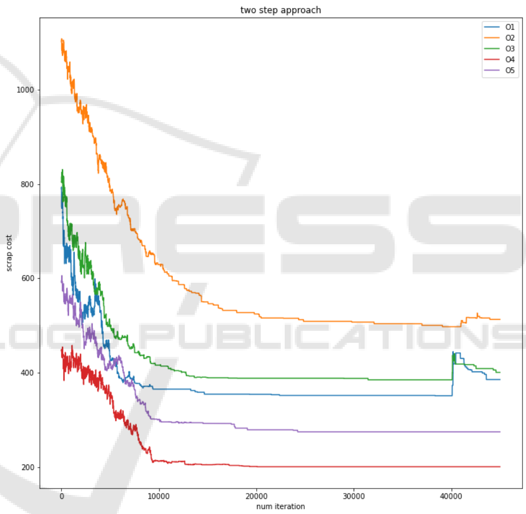

Figure 2: Shows the behaviour of the scrap cost over the iterations from SA2.

6 CONCLUSION

In this article, a planning problem from the steel industry was presented, together with some typical rules and constraints to be considered. We see that even today, long-established optimization methods still pay off. In this case the scheduling problem was modelled as a well-studied TSP and a SA algorithm was chosen to minimize the cost. Most of the time the difficulty is to formulate the specific problems of the factory and constraints into rules and constraints. If this succeeds, we have shown that the constraints and rules can be described by the object function and that small changes to the SA algorithm can improve the quality of the solutions and reduce the computation time. It is also important to observe how such optimization programs can eventually be integrated into the running processes of a factory.

The work described in this article has been funded by the European Union under Grant Agreement

869886. The authors would like to thank Nico Herbig for his important contributions in the initial problem definition and modeling phase.

REFERENCES

Ascheuer, N. (1996). *Hamiltonian path problems in the online optimization of flexible manufacturing systems*. PhD thesis.

Dahal, K., McDonald, J., and Burt, G. (2000). Modern heuristic techniques for scheduling generator maintenance in power systems. *Transactions of the Institute of Measurement and Control*, 22(2):179–194.

Johnson, D. S. and McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1(1):215–310.

Liu, S., Pinto, J. M., and Papageorgiou, L. G. (2008). A tsp-based milp model for medium-term planning of single-stage continuous multiproduct plants. *Industrial & Engineering Chemistry Research*, 47(20):7733–7743.

Matai, R., Singh, S. P., and Mittal, M. L. (2010). Traveling salesman problem: an overview of applications, formulations, and solution approaches. *Traveling salesman problem, theory and applications*, 1.

Pasotti, A. and Zavanella, L. (2007). Implementing advanced scheduling techniques in industrial environments: evidences and outcomes.

Zhan, S.-h., Lin, J., Zhang, Z.-j., and Zhong, Y.-w. (2016). List-based simulated annealing algorithm for traveling salesman problem. *Computational intelligence and neuroscience*, 2016.