Technical report bds:98-03

Designing optimal timing and sequencing strategies for a continuous steel foundry[*]

B. De Schutter

Control Systems Engineering
Faculty of Information Technology and Systems
Delft University of Technology
Delft, The Netherlands
Current URL: `https://www.dcsc.tudelft.nl`

[*]This report can also be downloaded via `https://pub.deschutter.info/abs/98_03.html`

DESIGNING OPTIMAL TIMING AND SEQUENCING STRATEGIES FOR A CONTINUOUS STEEL FOUNDRY

B. De Schutter

Control Laboratory, Fac. of Information Technology and Systems
Delft University of Technology, P.O. Box 5031
2600 GA Delft, The Netherlands
fax: +31-15-278.66.79, email: b.deschutter@its.tudelft.nl

Keywords: optimal timing, optimal sequencing, genetic algorithms, tabu search

Abstract

In this paper we consider a typical case study of a continuous steel making process. More specifically, we consider the part of a steel plant consisting of the furnace, the secondary steel making units and the continuous caster. First we write down a model for the evolution of the system. Then we develop some techniques to design (sub)optimal timing and sequencing schemes for this system. The methods we use include a mixture of linear programming, genetic algorithms, tabu search and heuristics.

1 The set-up of the system

Consider the part of a continuous steel production process that consists of an electric arc furnace (EAF), a convertor, two vacuum oxygen decarburization units (VODs) that operate in parallel, a waiting ladle (WL) and a continuous casting unit (CC) (see Figure 1). In the next paragraph we shall describe the different steps in this process. A description of steel production processes from a control engineering point of view can be found in [9]. Note that the techniques developed in this paper can also be applied to basic-oxygen-furnace-based processes.

The system works in batches[1]. Let us now describe the path followed by the kth batch. All times will be expressed in minutes. The raw material of the kth batch (scrap or pig iron) is inserted into the EAF at time $t_{\mathrm{EAF}}(k)$ and stays there during $d_{\mathrm{EAF}}(k)$ minutes. Then it is transported to the convertor (with a waiting stage if necessary). This takes $d_1(k)$ minutes. The kth batch enters the convertor at time $t_{\mathrm{CNV}}(k)$ and stays there for $d_{\mathrm{CNV}}(k)$ minutes. Then it is transferred to the VOD that is currently empty, which takes $d_2(k)$ minutes. The VODs work in parallel. So if the kth batch goes to the first VOD, then the $(k+1)$-st batch goes to the second VOD, and vice versa. The kth batch stays in the VOD for $d_{\mathrm{VOD}}(k)$ minutes. Then it is transferred to the waiting ladle. At time $t_{\mathrm{CC}}(k)$ the liquid steel is then fed to the continuous casting plant. The difference between the time instant at which the kth batch leaves the VOD and the time instant at which it is fed to the CC is denoted by $d_{\mathrm{WL}}(k)$. The CC setup time for the kth batch is $d_3(k)$, and the casting time for the kth batch is $d_{\mathrm{CC}}(k)$.

The times $d_{1,\min}(k)$, $d_{1,\max}(k)$, $d_{2,\min}(k)$, $d_{2,\max}(k)$, $d_{\mathrm{WL},\min}(k)$, $d_{\mathrm{WL},\max}(k)$ are hard lower and upper bounds for $d_1(k)$, $d_2(k)$, $d_{\mathrm{WL}}(k)$ respectively that are imposed by technological and process constraints. Let $w(k)$ be the desired width of the kth finished product that leaves the CC and let $h(k)$ be its height. The CC setup time $d_3(k)$ has to satisfy the following constraints:

$$d_3(k) \geqslant d_{\mathrm{wc},\min} \quad \text{if } w(k-1) \neq w(k) \tag{1}$$

$$d_3(k) \geqslant d_{\mathrm{hc},\min} \quad \text{if } h(k-1) \neq h(k) \tag{2}$$

$$\left.\begin{array}{r} \text{either } d_3(k) = 0 \\ \text{or } d_3(k) \geqslant d_{\mathrm{break},\min} \end{array}\right\} \text{ if } \left\{\begin{array}{c} w(k-1) = w(k) \\ \text{and} \\ h(k-1) = h(k)\,, \end{array}\right. \tag{3}$$

where $d_{\mathrm{wc},\min}$ is the minimal setup time after a width change, $d_{\mathrm{hc},\min}$ is the minimal setup time after a height change, and $d_{\mathrm{break},\min}$ is the minimal setup time after a break in between two consecutive batches.

Since as soon as a batch leaves an EAF the next batch is fed to the EAF, the evolution of the system can be described by the following system of equations:

$$t_{\mathrm{EAF}}(k) = t_{\mathrm{EAF}}(k-1) + d_{\mathrm{EAF}}(k-1) \tag{4}$$

$$\begin{aligned} t_{\mathrm{CNV}}(k) = \max\big(&t_{\mathrm{EAF}}(k) + d_{\mathrm{EAF}}(k) + d_{1,\min}(k), \\ &t_{\mathrm{CNV}}(k-1) + d_{\mathrm{CNV}}(k-1)\big) \end{aligned} \tag{5}$$

$$\begin{aligned} t_{\mathrm{VOD}}(k) = \max\big(&t_{\mathrm{CNV}}(k) + d_{\mathrm{CNV}}(k) + d_{2,\min}(k), \\ &t_{\mathrm{VOD}}(k-2) + d_{\mathrm{VOD}}(k-2)\big) \end{aligned} \tag{6}$$

$$\begin{aligned} t_{\mathrm{CC}}(k) = \max\big(&t_{\mathrm{VOD}}(k) + d_{\mathrm{VOD}}(k) + d_{\mathrm{WL},\min}(k), \\ &t_{\mathrm{CC}}(k-1) + d_{\mathrm{CC}}(k-1) + d_3(k)\big)\ . \end{aligned} \tag{7}$$

Note that the presence of two VODs that work in parallel implies the use of the index $k-2$ instead of $k-1$ on the right-

[1] A typical batch weighs around 150 tonnes.

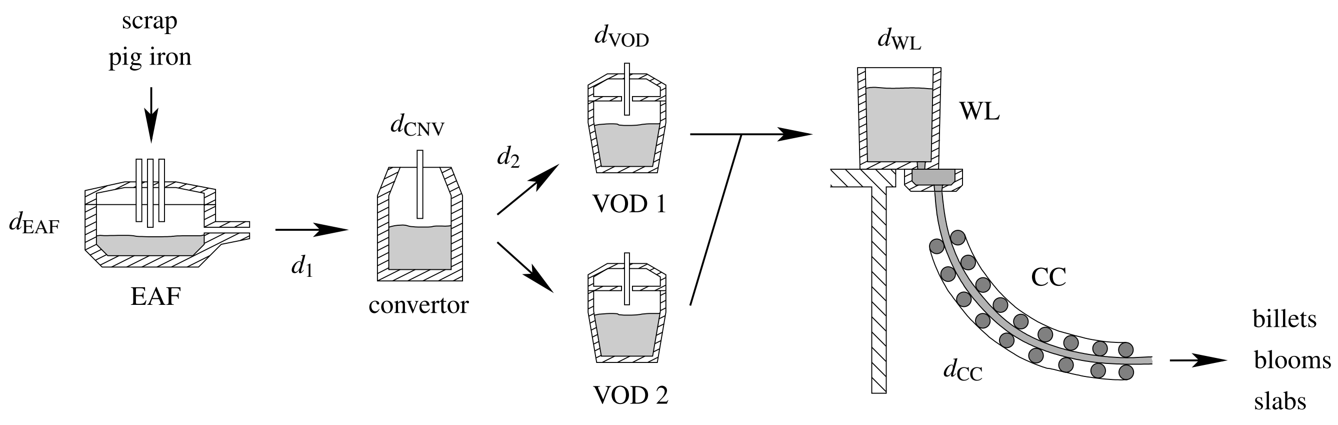


Figure 1: The part of a steel plant consisting of the furnace (EAF), the secondary steel making units (convertor and VODs), the waiting ladle (WL) and the continuous caster (CC).

hand side of (6). The effective transportation and waiting times for the kth batch are now given by:

$$\begin{aligned} d_1(k) &= t_{\mathrm{CNV}}(k) - t_{\mathrm{EAF}}(k) - d_{\mathrm{EAF}}(k) \\ d_2(k) &= t_{\mathrm{VOD}}(k) - t_{\mathrm{CNV}}(k) - d_{\mathrm{CNV}}(k) \\ d_{\mathrm{WL}}(k) &= t_{\mathrm{CC}}(k) - t_{\mathrm{VOD}}(k) - d_{\mathrm{VOD}}(k) \ . \end{aligned}$$

The casting time for the kth batch is given by the following formula:

$$d_{\mathrm{CC}}(k) = \frac{W(k)}{\rho\, w(k)\, h(k)\, v(k)} \ ,$$

where $W(k)$ is the weight of the batch, ρ is the density of the product and $v(k)$ is the prescribed casting velocity, which in our case is a piecewise linear function of the width and the height of the finished product.

Since the cost of renewing the refractory covering of the waiting ladle is one of the main factors in the overall production process, and since the speed of deterioration of the covering is proportional to the total waiting time, we want the minimize the following objective function:

$$J = \sum_{k=1}^{N} d_{\mathrm{WL}}(k) \ ,$$

where N is the number of batches to be scheduled.

2 Optimal timing schemes

In this section we assume that the order of the batches is fixed and we compute suboptimal timing schemes for the given sequence. In the next section we shall then discuss how suboptimal sequencing schemes can be computed.

We shall use $d_1(k)$ and $d_3(k)$ as the control variables. In our case study, we did not use $d_2(k)$ as an additional control variable, but we set it to the constant value d_2. If we then know the initial values $t_{\mathrm{EAF}}(k_0)$ and $t_{\mathrm{CC}}(k_0)$ then the evolution of the system can be computed using the equations given above.

Note that the either-or condition (3) introduces a combinatorial aspect in the problem. Therefore, we shall first approach the problem without taking this condition into account.

2.1 Optimal timing schemes without taking the condition (3) into account

If we do not consider condition (3) and if we make no extra assumptions then the system of equations (4) – (7) together with the upper and lower bound constraints can be recast as an Extended Linear Complementarity System (ELCP) [3]. In general the solution set of an ELCP consists of faces of a polyhedron. After computing this solution set using the algorithm of [3], we could then minimize the objective function J over each of the faces and then select the global optimum. This is the approach we have also used in [4] to compute optimal signal switching schemes for traffic signal controlled intersections. Since the general ELCP with rational data is NP-hard [2, 3], this approach is not feasible in practice, especially if the number of batches N is large.

However, if we take care that the following conditions are always satisfied (which will be the case in practice or can be effected by considering the conditions as extra constraints on the control variables $d_1(k)$ and $d_3(k)$):

$$t_{\mathrm{CNV}}(k) \geqslant t_{\mathrm{CNV}}(k-1) + d_{\mathrm{CNV}}(k-1) \tag{8}$$
$$t_{\mathrm{VOD}}(k) \geqslant t_{\mathrm{VOD}}(k-2) + d_{\mathrm{VOD}}(k-2) \tag{9}$$
$$t_{\mathrm{CC}}(k) \geqslant t_{\mathrm{VOD}}(k) + d_{\mathrm{VOD}}(k) + d_{\mathrm{WL,min}}(k) \tag{10}$$

then the system (4) – (7) can be rewritten as:

$$\begin{aligned} t_{\mathrm{EAF}}(k) &= t_{\mathrm{EAF}}(k-1) + d_{\mathrm{EAF}}(k-1) \\ t_{\mathrm{CNV}}(k) &= t_{\mathrm{EAF}}(k) + d_{\mathrm{EAF}}(k) + d_1(k) \\ t_{\mathrm{VOD}}(k) &= t_{\mathrm{CNV}}(k) + d_{\mathrm{CNV}}(k) + d_2 \\ t_{\mathrm{CC}}(k) &= t_{\mathrm{CC}}(k-1) + d_{\mathrm{CC}}(k-1) + d_3(k) \ . \end{aligned}$$

It is easy to verify that the problem of minimizing the objective function J subject to these evolution equations and subject to

(1) – (2), (8) – (10), the upper and lower bound constraints and the condition $d_3(k) \geqslant 0$, can be recast as a linear program (LP) which can be solved efficiently using (variants of) the simplex method or using an interior point method [8].

2.2 Optimal timing schemes with condition (3)

Let $\mathscr{K} = \{k_1, k_2, \ldots, k_K\}$ be the set of indices k for which $w(k-1) = w(k)$ and $h(k-1) = h(k)$.

The first heuristic approach to obtain suboptimal timing schemes is to first set $d_3(k) = 0$ for all $k \in \mathscr{K}$ and $d_1(k) = d_{1,\min}$ for all k. Then we select the lowest index $k \in \mathscr{K}$ for which the constraint $d_{\mathrm{WL,min}} \leqslant d_{\mathrm{WL}}(k+1) \leqslant d_{\mathrm{WL,max}}$ is violated and we determine $d_1(k)$ and/or $d_3(k) \geqslant d_{\mathrm{break,min}}$ such that the constraint holds. However, this approach does not always lead to a valid solution.

In the second heuristic approach we first solve the LP problem of the Section 2.1. Then we select a threshold τ and we consider a new LP consisting of the LP of Section 2.1 augmented with the following constraints:

$$\begin{aligned} d_3(k) &\geqslant d_{\mathrm{break,min}} && \text{if } d_{3,\mathrm{LP}}(k) > \tau \\ d_3(k) &= 0 && \text{if } d_{3,\mathrm{LP}}(k) \leqslant \tau \end{aligned}$$

for $k \in \mathscr{K}$ where $d_{3,\mathrm{LP}}$ is the LP solution. Solving this new LP then yields a valid, suboptimal timing scheme.

The best heuristic solution will be selected as an initial solution. Let us now consider some techniques to improve this solution.

Since the problem we want to solve is a combinatorial problem, we propose to use an approach that is based on genetic algorithms [1, 7]. Note that we can code each possible combination resulting from condition (3) by a binary string b consisting of K bits, where we add the following constraints to the LP of Section 2.1:

$$\begin{aligned} d_3(k_i) &= 0 && \text{if } b_i = 0 \\ d_3(k_i) &\geqslant d_{\mathrm{break,min}} && \text{if } b_i = 1\,. \end{aligned}$$

This will lead again to an LP, which can be solved efficiently. Now we can use genetic algorithms to gradually improve the timing sequence. We have implemented a genetic algorithm with elitist recombination, niching through crowding, restricted mating and adaptive crossover (see, e.g., [10]). During the evolution of the genetic algorithm we display the best solution obtained so far and we allow the human operator to stop the algorithm as soon as she thinks that sufficient improvement has been obtained.

Note that considering all possible combinations resulting from condition (3) would require solving 2^K LPs, which is obviously not feasible if K is large.

3 Optimal sequencing schemes

Let us now discuss some methods to compute optimal sequencing schemes. Since this is again a combinatorial problem and since enumerating and evaluating all possible combinations is not tractable, we now present to some methods to obtain suboptimal solutions.

The first heuristic approach we could use consists in sorting the batches lexicographically[2] according to their width and height.

The second heuristic approach is a "best fit" approach. We start with a given sequence and we compute the corresponding optimal value of J (using, e.g., the second heuristic approach of Section 2.2). Let this optimal value be J_1. Then we insert the second batch before the first batch and compute again the corresponding optimal value of J. Let this optimal value be equal to J_2. Next we restore the original order and insert the third batch before the first batch and we compute again the optimal value of J which we assign to J_3. We repeat this process for the other batches. This leads to the values J_3, J_4, …, J_N. Note that we can reduce the number of computations by taking into account that if $w(k-1) = w(k)$ and $h(k-1) = h(k)$ then we have $J_{k-1} = J_k$. Now we select the best batch k_{best}, i.e., the batch that leads to the largest decrease in J: $k_{\mathrm{best}} = \arg\min_k J_k$, and we insert batch k_{best} before batch 1. The position of this batch is then fixed, and we repeat the process for the remaining $N-1$ batches, and so on. It is easy to verify that the computational complexity of this approach is at most $O(N^2 C_{\mathrm{LP}})$ where C_{LP} is the computational complexity of the LP of Section 2.1.

We use again the best heuristic solution as an initial solution and then we gradually improve this solution. Currently we are also investigating tabu search methods [5, 6], genetic algorithms and decomposition methods[3] to determine (sub)optimal switching schemes.

Note that while determining the optimal sequence we can also use one or more of the methods of Section 2.2 to simultaneously obtain an optimal timing scheme. If we use a genetic algorithm based approach we could merge the binary coding of the sequence with the binary coding given in Section 2.2 and simultaneously optimize the timing and the sequence of the batches. We have also developed some local improvement methods that consider pairs or triplets of consecutive batches and that use very simple algebraic expressions to determine whether changing the position or timing of the batches within the pair or the triplet yields a lower value for the objective function J.

4 Implementation and results

We have implemented the methods presented above in a Matlab graphical user interface (GUI), that can be used to simulate the system, to display the results graphically and to compute the optimal timing and sequencing scheme. In Figure 4 we

[2] So batch k_i comes before batch k_j if $w(k_i) < w(k_j)$ or if $w(k_i) = w(k_j)$ and $h(k_i) < h(k_j)$.

[3] I.e., first dividing the batches in smaller groups that will be kept together and then determining the optimal sequence of the groups and the optimal sequence within the groups.

show a screen shot of this GUI. The GUI is a modular system and allows the user to change almost every parameter that is used in the simulations and in the optimization. Currently, both English and Dutch are supported; other languages can easily be incorporated. The user can also select the method and the level (timing, sequence, or timing and sequence) of optimization. During the optimization the best value of the objective function obtained so far is displayed so that the human operator can monitor the procedure and stop the optimization if sufficient improvement has been obtained.

In order to facilitate the acceptance of our tool in the actual manufacturing process and among the operators, we have developed the GUI so that it functions as a guide or an advisor rather than as a solution generator. We allow the human operator to change the resulting scheme according to his own insights. Furthermore, the built-in simulator enables the human operator to examine the effects of various changes he makes to the current timing and sequencing scheme.

In our case study we were able to reduce the total waiting ladle time J with up to 16 % for a typical production run. Since cost of renewing the refractory covering of the waiting ladle is one of the main factors in the overall cost of the production process, this means that substantial gains can be obtained by using our tool.

5 Conclusions and further research

In this paper we have developed methods to design timing and sequencing schemes for the part of a continuous steel making process consisting of the furnace, the secondary steel making units and the continuous caster. Since computing the globally optimal scheme is not feasible in practice, we have presented methods to compute suboptimal timing and sequencing schemes using heuristic methods, linear programming relaxations, genetic algorithms, tabu search and decomposition methods. We have implemented these techniques in a Matlab GUI. A practical example shows that using the techniques presented in this paper can lead to a significant reduction of the overall production cost in a continuous steel foundry.

In our future research we will focus on developing other efficient algorithms to compute suboptimal timing and sequencing schemes, on incorporating the knowledge now present in the human schedulers into our methods, and on further developing and extending our Matlab GUI tool.

References

[1] L. Davis, ed., *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[2] B. De Schutter, *Max-Algebraic System Theory for Discrete Event Systems*. PhD thesis, Faculty of Applied Sciences, K.U.Leuven, Leuven, Belgium, Feb. 1996.

[3] B. De Schutter and B. De Moor, “The extended linear complementarity problem,” *Mathematical Programming*, vol. 71, no. 3, pp. 289–325, Dec. 1995.

[4] B. De Schutter and B. De Moor, “Optimal traffic light control for a single intersection,” *European Journal of Control*, vol. 4, no. 3, pp. 260–276, 1998.

[5] F. Glover, “Tabu search: A tutorial,” *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.

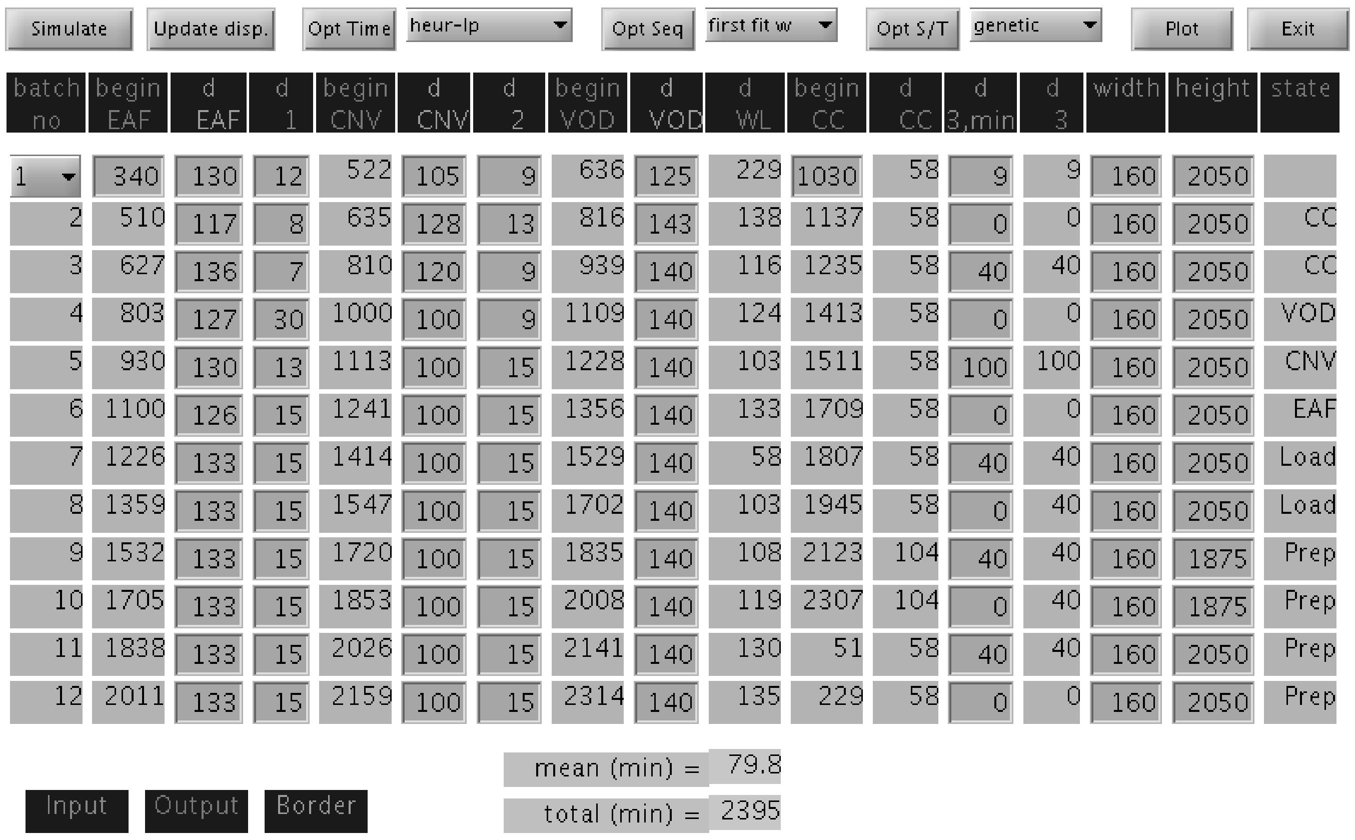
[6] F. Glover and M. Laguna, *Tabu Search*. Boston, Massachusetts: Kluwer Academic Publishers, 1997.

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

[8] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Philadelphia, Pennsylvania: SIAM, 1994.

[9] D. Popovic and V.P. Bhatkar, *Distributed Computer Control for Industrial Automation*. New York: Marcel Dekker, 1990.

[10] D. Thierens, *Analysis and Design of Genetic Algorithms*. PhD thesis, Faculty of Applied Sciences, K.U.Leuven, Leuven, Belgium, May 1995.



batch no	begin EAF	d EAF	d 1	begin CNV	d CNV	d 2	begin VOD	d VOD	d WL	begin CC	d CC	d 3,min	d 3	width	height	state
1	340	130	12	522	105	9	636	125	229	1030	58	9	9	160	2050	
2	510	117	8	635	128	13	816	143	138	1137	58	0	0	160	2050	CC
3	627	136	7	810	120	9	939	140	116	1235	58	40	40	160	2050	CC
4	803	127	30	1000	100	9	1109	140	124	1413	58	0	0	160	2050	VOD
5	930	130	13	1113	100	15	1228	140	103	1511	58	100	100	160	2050	CNV
6	1100	126	15	1241	100	15	1356	140	133	1709	58	0	0	160	2050	EAF
7	1226	133	15	1414	100	15	1529	140	58	1807	58	40	40	160	2050	Load
8	1359	133	15	1547	100	15	1702	140	103	1945	58	0	40	160	2050	Load
9	1532	133	15	1720	100	15	1835	140	108	2123	104	40	40	160	1875	Prep
10	1705	133	15	1853	100	15	2008	140	119	2307	104	0	40	160	1875	Prep
11	1838	133	15	2026	100	15	2141	140	130	51	58	40	40	160	2050	Prep
12	2011	133	15	2159	100	15	2314	140	135	229	58	0	0	160	2050	Prep

mean (min) = 79.8

total (min) = 2395

Figure 2: Screen shot of our Matlab GUI.