Parallel Process Scheduling at Norma

K. Epner, B. Ghandhachi, C. Gheorghiu, B.K. Muite, ?

February 8, 2019

1 Abstract

Norma AS is a component manufacturing company based in Tallinn. They produce car safety parts out of stamped metal and have 3 heat treatment processing lines. They desire to know if their current production techniques are efficient. To determine the efficiency, a static scheduling model for the production process is created and evaluated on real world test data to determine its effectiveness.

Key words: Scheduling, parallel processing, manufacture, work stealing, greedy algorithms

2 Introduction

Norma AS has three processing lines, K1, K2 and S1 which produce a total of 151 components. The important factor in the processing lines is the temperature of the salt bath, which differs for each component. The processing lines have particular processing productivities for each component. K1 and K2 processing lines are very similar and have the same productivity and bath temperatures for the components which they can both produce, while S1 has about half the productivity of K1 and K2 for the components for which they can all produce, but also produces some items which neither K1 nor K2 can produce. While one can increase the throughput through the processsing lines by appropriate scheduling of tasks, total throughput may also be limited by postprocessing steps. In this first approach, we neglect the influence of the postprocessing step.

3 Model Problem Formulation

Given orders for a particular week, containing items i with quantities q_i, minimize the maximum time any production line is operational subject to:

- processing all the items
- there is a gap of 12 minutes between batches (number of items per batch are given)
- the temperature during the week on each processing line is a non-increasing function
- if the production item on a processing line changes, a 15 minute break is needed for changing the temperature of the salt bath fluid

- items that can be processed only on K1 or K2 and not on S1 are processed on K1 or K2
- items that can be processed only on line K1, K2 or S1 are processed on K1, K2 or S1 respectively

4 Solution to Model Problem

To solve this problem, an Octave/Matlab program was written to produce a sequence of processing steps. As input, the program takes two comma separated value files (CSV), one with order information and a second with item processing information. The program outputs the order in which items should be processed on each processing line, and the total time each processing line is active. Rather than aim for global processing time minimization, the program outputs a time for a feasible solution that satisfies all the constraints. This time is obtained by using a round robin scheduling algorithm to feed the processing lines. The program consists of the following steps:

- Read in item production information
- Read in weekly item orders
- Create three sorted lists of weekly item orders, one for each processing line sorted by the salt bath processing temperature of that line
- Repeatedly perform the following steps until all items are processed:
  - For the first machine, K1, take the highest temperature unprocessed order on its list, and process it entirely if it takes less than 4 hours, or process at most 4 hours of the order. Then update the lists of all machines to indicate that this item has been processed.
  - For the second machine, K2, take the highest temperature unprocessed order on its list, and process it entirely if it takes less than 4 hours, or process at most 4 hours of the order. Then update the lists of all machines to indicate that this item has been processed.
  - For the third machine, S1, take the highest temperature unprocessed order on its list, and process it entirely if it takes less than 4 hours, or process at most 4 hours of the order. Then update the lists of all machines to indicate that this item has been processed.

To check the quality of this solution, a lower bound on total processing time is also obtained which assumes minimal gap of 6 minutes between batches. This lower bound is found by taking the minimum processing time on lines K1, K2 or S1 for each ordered item and then summing these times. This lower bound time is used to give a parallel processing efficiency.

5 Results

Three test data sets were given. The following processing times were obtained:

Test set	Processing time, K1	Processing time, K2	Processing time, S1	Parallel Efficiency
1	110.0 hours	99.2 hours	108.9 hours	82.9%
2	91.9 hours	87.0 hours	91.9 hours	86.6%
3	120.3 hours	99.8 hours	110.5 hours	79.4%

6 Conclusions

The heuristic model, which is based on a variation of the current strategy seems to be reasonably successful. The workloads are reasonably balanced and the parallel efficiency is satisfactory. The heuristic model can be easily integrated into the current workflow and still allows for some flexibility in scheduling, for example if new items are added, or productivities are changed.

7 Further Work

At present, postprocessing steps have been neglected. These can also be a bottleneck, and in some cases, it maybe worth doing scheduling to maximize throughput on the postprocessing steps. An initial first step is to determine if these postprocessing steps are a bottleneck, and then to determine if scheduling can be easily changed to keep throughput high.

The present algorithm uses static scheduling, a dynamic scheduling algorithm would enable better efficiency and flexibility of the workflow. The current workflow also does not include production costs or sale prices of the items produced - adding these would enable better evaluation of the efficiency of the production process.

The present algorithm may also be evaluated by using it on publicly available datasets. Boysen et al. provide an archive of datasets[1].

The parallel efficiency metric is simple and does not fully take into account load balancing or production efficiencies of the different lines. More effort to find a simple to understand, simple to evaluate yet effective performance metric is needed. Metrics from the parallel computing field may be taken as possible starting points[4].

The present problem has a competitive programming flavor. It would be interesting to see how well such a problem is solved in such a context[2, 3, 5].

References

[1] Nils Boysen and Malte Fliedner and Robert Klein and Armin Scholl, "Àssembly Line Balancing", `https://assembly-line-balancing.de/` 2019‘

[2] "Deadline24" `https://www.deadline24.pl/` 2019

[3] "Eesti Informaatika Olumpiaad" `http://eio.ee/` 2019

[4] Shigeo Orii, "Metrics for Evaluation of Parallel Efficiency Toward Highly Parallel Processing", Parallel Computing 36(1), 16–25, 2010 `http://dx.doi.org/10.1016/j.parco.2009.11.003`

[5] Steven Skiena, *The Algorithm Design Manual*, Springer, London, 2nd edition, 2008.

Appendix

```
clear all; clc;
% Read in a csv file to a matrix

filename ='MinimalProductionData.csv';
fid=fopen(filename,'r');
data=textscan(fid,'%f %f %f %f %f %f %f %f %f %f %f',...
                          'headerlines', 1,...
        'delimiter',',');
fclose(fid);
production_data=cell2mat(data);

filename ='MinimalOrderData.csv';

fid=fopen(filename,'r');
data=textscan(fid,'%f %f %f %f',...
                  'headerlines', 1,...
        'delimiter',',');
fclose(fid);
order_data=cell2mat(data);

% Time in minutes to increase temperature by 6 degree
heat_increase_time=6;
% Time in minutes to decrease temperature by 2 degree
heat_decrease_time=2;

week1_orders(:,1)=order_data(:,1);
week1_orders(:,2)=order_data(:,4);

%week2_orders(:,1)=order_data(:,1);
%week2_orders(:,3)=order_data(:,3);

%week3_orders(:,1)=order_data(:,1);
%week3_orders(:,4)=order_data(:,4);

% Add data to get production information
% Array headers:
% item, quantity, weight [kg], temp k1&k2 , temp s1,productivity k1 [Kg/hour],
% productivity s1 [KG/hour], amount in container [KG], maximum batch size [
    containers],
% production on k1, production on k2, production on s1
orderlength_k1=length(week1_orders);
orderlength_k2=length(week1_orders);
orderlength_s1=length(week1_orders);

maxorderlength=max(orderlength_k1,orderlength_k2);
maxorderlength=max(maxorderlength,orderlength_s1);

production_data_length=length(production_data);
% Augment order data matrix with processing properties of each item
for i=1:maxorderlength
  item=week1_orders(i,1);
  for j=1:production_data_length
    if production_data(j,1)==item
      for k=2:11
          week1_orders(i,k+1)=production_data(j,k);
      endfor
    endif
  endfor
endfor
%Determine best possible "serial" processing time
best_serial_time=0;
for i=1:maxorderlength
 if (((week1_orders(i,10)==1)||(week1_orders(i,11)==1)) && (week1_orders(i,12)==1))
   productivity=max(week1_orders(i,6),week1_orders(i,7));
 elseif (((week1_orders(i,10)==1)||(week1_orders(i,11)==1)) && (week1_orders(i,12)
     ==0))
   productivity=week1_orders(i,6);
 else
   productivity=week1_orders(i,7);
 endif
 weight_per_item=week1_orders(i,3);
 ordered_items=week1_orders(i,2);
 batch_size=week1_orders(i,9);
 weight=weight_per_item*ordered_items;
 container_capacity=week1_orders(i,8);
 containers=weight/container_capacity;
 batches=ceil(containers/batch_size);
 if(ordered_items>0)
   best_serial_time=best_serial_time+batches*7.0/60.0+weight/productivity;
 endif
endfor
% Sort by temp in k1 and k2 descending order
week1_orders_sort_k1=sortrows(week1_orders,-4);
% Add extra row to indicate if processed
week1_orders_sort_k1(:,13)=0;
week1_orders_sort_k2=week1_orders_sort_k1;
% Sort by temp in s1 descending order
```

```
week1_orders_sort_s1=sortrows(week1_orders,-5);
week1_orders_sort_s1(:,13)=0;
% simple processing time estimate, assumes temperature is always decreasing
processing_time_k1=0;
processing_time_k2=0;
processing_time_s1=0;
k1_count=1;
k2_count=1;
s1_count=1;
last_item_k1=-1;
last_item_k2=-1;
last_item_s1=-1;
k1_processed=0;
k2_processed=0;
s1_processed=0;
while( (k1_count<orderlength_k1) && ...
       (k2_count<orderlength_k2) && ...
       (s1_count<orderlength_s1))
   % Line K1
   already_processed=week1_orders_sort_k1(k1_count,13);
   can_process_on_k1=week1_orders_sort_k1(k1_count,10);
   item_k1=week1_orders_sort_k1(k1_count,1);
   ordered_items=week1_orders_sort_k1(k1_count,2);
   % Find next item to process
   while (((can_process_on_k1==0) || (already_processed>0) ...
            || (item_k1==0) || (ordered_items==0) ) ...
          && (k1_count<orderlength_k1) )
     k1_count=k1_count+1;
     already_processed=week1_orders_sort_k1(k1_count,13);
     can_process_on_k1=week1_orders_sort_k1(k1_count,10);
     item_k1=week1_orders_sort_k1(k1_count,1);
     ordered_items=week1_orders_sort_k1(k1_count,2);
   endwhile
   if((k1_count<orderlength_k1))
     batch_size=week1_orders_sort_k1(k1_count,9);
     container_capacity=week1_orders_sort_k1(k1_count,8);
     weight_per_item=week1_orders_sort_k1(k1_count,3);
     productivity=week1_orders_sort_k1(k1_count,6);
     weight=ordered_items*weight_per_item;
     containers=weight/container_capacity;
     batches=ceil(containers/batch_size);
     temperature_k1=week1_orders_sort_k1(k1_count,4);
     ptime=weight/productivity;
     partialprocess=0;
     % if the processing time is greater than 4 hours, process only upto 4 hours
     if (ptime>4)
       weight=productivity*4;
       items=floor(weight/weight_per_item);
       weight=items*weight_per_item;
       containers=weight/container_capacity;
       batches=ceil(containers/batch_size);
       partialprocess=1;
     endif
     if partialprocess==1
       ptime=weight/productivity;
     endif
     printf('Line K1 processing item %d k1_count is %d items are %d, processing time
          %f, temperature is %f \n',...
     item_k1,k1_count,ordered_items,ptime,temperature_k1);
     if(k1_processed==0)
       temperature_change_k1=0;
     else
       temperature_change_k1=temperature_k1-old_temperature_k1;
     endif
     % store temperature value
     old_temperature_k1=temperature_k1;
     % get time to change the temperature
     temperature_change_time_k1=heat_decrease_time*temperature_change_k1/60.0;
     % update processing time
     if last_item_k1==item_k1
       processing_time_k1=processing_time_k1+(weight/productivity) + ...
                          temperature_change_time_k1 + batches*12.0/60.0;
     else
       processing_time_k1=processing_time_k1+(weight/productivity) + ...
                          temperature_change_time_k1 + 15.0/60.0 + batches
                               *12.0/60.0;
     endif
     if (partialprocess==0)
     % remove item from list if processed all of it
       week1_orders_sort_k1(k1_count,13)=1;
       % mark item as processed on line k2 list
       for j=1:orderlength_k2
         if(week1_orders_sort_k1(k1_count,1)==week1_orders_sort_k2(j,1))
           week1_orders_sort_k2(j,13)=1;
         endif
       endfor
       % mark item as processed on line s1 list
       for j=1:orderlength_s1
         if(week1_orders_sort_k1(k1_count,1)==week1_orders_sort_s1(j,1))
           week1_orders_sort_s1(j,13)=1;
         endif
       endfor
       k1_count=k1_count+1;
    else
      % reduce quantities if processed only some of the order
       week1_orders_sort_k1(k1_count,2)=week1_orders_sort_k1(k1_count,2)-items;
       % mark item as processed on line k2 list
       for j=1:orderlength_k2
         if(week1_orders_sort_k1(k1_count,1)==week1_orders_sort_k2(j,1))
```

```
183             week1_orders_sort_k2(j,2)=week1_orders_sort_k2(j,2)-items;
184           endif
185         endfor
186         % mark item as processed on line s1 list
187         for j=1:orderlength_s1
188           if(week1_orders_sort_k1(k1_count,1)==week1_orders_sort_s1(j,1))
189             week1_orders_sort_s1(j,2)=week1_orders_sort_s1(j,2)-items;
190           endif
191         endfor
192       endif
193       k1_processed=k1_processed+1;
194       last_item_k1=item_k1;
195     endif
196
197     % Line K2
198     already_processed=week1_orders_sort_k2(k2_count,13);
199     can_process_on_k2=week1_orders_sort_k2(k2_count,11);
200     item_k2=week1_orders_sort_k2(k2_count,1);
201     ordered_items=week1_orders_sort_k2(k2_count,2);
202     % Find next item to process
203     while ((((can_process_on_k2==0) || (already_processed>0) ...
204             || (item_k2==0) || (ordered_items==0) ) ...
205            && (k2_count<orderlength_k2) )
206       k2_count=k2_count+1;
207       already_processed=week1_orders_sort_k2(k2_count,13);
208       can_process_on_k2=week1_orders_sort_k2(k2_count,11);
209       item_k2=week1_orders_sort_k2(k2_count,1);
210       ordered_items=week1_orders_sort_k2(k2_count,2);
211     endwhile
212     if((k2_count<orderlength_k2))
213       batch_size=week1_orders_sort_k2(k2_count,9);
214       container_capacity=week1_orders_sort_k2(k2_count,8);
215       weight_per_item=week1_orders_sort_k2(k2_count,3);
216       productivity=week1_orders_sort_k2(k2_count,6);
217       weight=ordered_items*weight_per_item;
218       containers=weight/container_capacity;
219       batches=ceil(containers/batch_size);
220       temperature_k2=week1_orders_sort_k2(k2_count,4);
221       ptime=weight/productivity;
222       partialprocess=0;
223       % if the processing time is greater than 4 hours, process only upto 4 hours
224       if (ptime>4)
225         weight=productivity*4;
226         items=floor(weight/weight_per_item);
227         weight=items*weight_per_item;
228         containers=weight/container_capacity;
229         batches=ceil(containers/batch_size);
230         partialprocess=1;
231       endif
232       if partialprocess==1
233         ptime=weight/productivity;
234       endif
235       printf('Line K2 processing item %d k2_count is %d items are %d, processing time
              %f, temperature is %f \n',...
236       item_k2,k2_count,ordered_items,ptime,temperature_k2);
237       if(k2_processed==0)
238         temperature_change_k2=0;
239       else
240         temperature_change_k2=temperature_k2-old_temperature_k2;
241       endif
242       % store temperature value
243       old_temperature_k2=temperature_k2;
244       temperature_change_time_k2=2.0*temperature_change_k2/60.0;
245       % Update processing time
246       if last_item_k2==item_k2
247         processing_time_k2=processing_time_k2+(weight/productivity) + ...
248                            temperature_change_time_k2 + batches*12.0/60.0;
249       else
250         processing_time_k2=processing_time_k2+(weight/productivity) + ...
251                            temperature_change_time_k2 + 15.0/60.0 + batches
                                   *12.0/60.0;
252       endif
253       if (partialprocess==0)
254       % remove item from list if processed all of it
255         week1_orders_sort_k2(k2_count,13)=1;
256         % mark item as processed on line k2 list
257         for j=1:orderlength_k1
258           if(week1_orders_sort_k2(k2_count,1)==week1_orders_sort_k1(j,1))
259             week1_orders_sort_k1(j,13)=1;
260           endif
261         endfor
262         % mark item as processed on line s1 list
263         for j=1:orderlength_s1
264           if(week1_orders_sort_k2(k2_count,1)==week1_orders_sort_s1(j,1))
265             week1_orders_sort_s1(j,13)=1;
266           endif
267         endfor
268         k2_count=k2_count+1;
269       else
270       % update item quantity
271         week1_orders_sort_k2(k2_count,2)=week1_orders_sort_k2(k2_count,2)-items;
272         % mark item as processed on line k2 list
273         for j=1:orderlength_k1
274           if(week1_orders_sort_k2(k2_count,1)==week1_orders_sort_k1(j,1))
275             week1_orders_sort_k1(j,2)=week1_orders_sort_k1(j,2)-items;
276           endif
277         endfor
278         % mark item as processed on line s1 list
279         for j=1:orderlength_s1
```

```
280             if(week1_orders_sort_k2(k2_count,1)==week1_orders_sort_s1(j,1))
281               week1_orders_sort_s1(j,2)=week1_orders_sort_s1(j,2)-items;
282             endif
283           endfor
284         endif
285         k2_processed=k2_processed+1;
286         last_item_k2=item_k2;
287       endif
288
289       % Line S1
290       already_processed=week1_orders_sort_s1(s1_count,13);
291       can_process_on_s1=week1_orders_sort_s1(s1_count,12);
292       item_s1=week1_orders_sort_s1(s1_count,1);
293       ordered_items=week1_orders_sort_s1(s1_count,2);
294       % Find next item to process
295       while (((can_process_on_s1==0) || (already_processed>0) ...
296               || (item_s1==0) || (ordered_items==0) ) ...
297              && (s1_count<orderlength_s1) )
298         s1_count=s1_count+1;
299         already_processed=week1_orders_sort_s1(s1_count,13);
300         can_process_on_s1=week1_orders_sort_s1(s1_count,12);
301         item_s1=week1_orders_sort_s1(s1_count,1);
302         ordered_items=week1_orders_sort_s1(s1_count,2);
303       endwhile
304       if((s1_count<orderlength_s1))
305         batch_size=week1_orders_sort_s1(s1_count,9);
306         container_capacity=week1_orders_sort_s1(s1_count,8);
307         weight_per_item=week1_orders_sort_s1(s1_count,3);
308         productivity=week1_orders_sort_s1(s1_count,7);
309         weight=ordered_items*weight_per_item;
310         containers=weight/container_capacity;
311         batches=ceil(containers/batch_size);
312         temperature_s1=week1_orders_sort_s1(s1_count,5);
313         ptime=weight/productivity;
314         partialprocess=0;
315         % if the processing time is greater than 4 hours, process only upto 4 hours
316         if (ptime>4)
317           weight=productivity*4;
318           items=floor(weight/weight_per_item);
319           weight=items*weight_per_item;
320           containers=weight/container_capacity;
321           batches=ceil(containers/batch_size);
322           partialprocess=1;
323         endif
324         if partialprocess==1
325           ptime=weight/productivity;
326         endif
327         printf('Line S1 processing item %d s1_count is %d items are %d, processing time
                %f, temperature is %f \n',...
328         item_s1,s1_count,ordered_items,ptime,temperature_s1);
329         if(s1_processed==0)
330           temperature_change_s1=0;
331         else
332           temperature_change_s1=temperature_s1-old_temperature_s1;
333         endif
334         % store temperature value
335         old_temperature_s1=temperature_s1;
336         % calculate time to change temperature
337         temperature_change_time_s1=heat_decrease_time*temperature_change_s1/60.0;
338         % Update processing itme
339         if last_item_s1==item_s1
340           processing_time_s1=processing_time_s1+(weight/productivity) + ...
341                              temperature_change_time_s1 + batches*12.0/60.0;
342         else
343           processing_time_s1=processing_time_s1+(weight/productivity) + ...
344                              temperature_change_time_s1 + 15.0/60.0 + batches
                                     *12.0/60.0;
345         endif
346         if (partialprocess==0)
347           % remove item from list if processed all of it
348           week1_orders_sort_s1(s1_count,13)=1;
349           % mark item as processed on line k1 list
350           for j=1:orderlength_k1
351             if(week1_orders_sort_s1(s1_count,1)==week1_orders_sort_k1(j,1))
352               week1_orders_sort_k1(j,13)=1;
353             endif
354           endfor
355           % mark item as processed on line k2 list
356           for j=1:orderlength_s1
357             if(week1_orders_sort_s1(s1_count,1)==week1_orders_sort_k2(j,1))
358               week1_orders_sort_k2(j,13)=1;
359             endif
360           endfor
361           s1_count=s1_count+1;
362         else
363           % update item quantity
364           week1_orders_sort_s1(s1_count,2)=week1_orders_sort_s1(s1_count,2)-items;
365           % update item quantity on line k1 list
366           for j=1:orderlength_k1
367             if(week1_orders_sort_s1(s1_count,1)==week1_orders_sort_k1(j,1))
368               week1_orders_sort_k1(j,2)=week1_orders_sort_k1(j,2)-items;
369             endif
370           endfor
371           % update item quantity on line k2 list
372           for j=1:orderlength_s1
373             if(week1_orders_sort_s1(s1_count,1)==week1_orders_sort_k2(j,1))
374               week1_orders_sort_k2(j,2)=week1_orders_sort_k2(j,2)-items;
375             endif
376           endfor
```

```
377         endif
378         s1_processed=s1_processed+1;
379         last_item_s1=item_s1;
380       endif
381
382     endwhile
383     parallel_efficiency=100.0*best_serial_time/(processing_time_k1+...
384                                                 processing_time_k2+...
385                                                 processing_time_s1);
386     printf('On K1, the processing time is at most: %f hours.\n',processing_time_k1)
387     printf('On K2, the processing time is at most: %f hours.\n',processing_time_k2)
388     printf('On S1, the processing time is at most: %f hours.\n',processing_time_s1)
389     printf('Parallel efficiency %f\n',parallel_efficiency)
```