Transient Scheduling of Single Armed Cluster Tools: Algorithms for Wafer Residency Constraints

Hong-Yue Jin[1], James R. Morrison[2]

Department of Industrial and Systems Engineering, KAIST
Daejeon, 305-701, Republic of Korea

[1]nevergiveup@kaist.ac.kr

[2]james.morrison@kaist.edu

Abstract— **The wafer handling robot actions in cluster tools used for semiconductor manufacturing should serve to maximize throughput while maintaining good wafer quality. Since excessive delay in a process chamber may cause deterioration in wafer quality, wafer delays should be maintained in an acceptable range, or preferably, should be minimized. We focus on addressing these concerns for all wafers in a lot, including those in both the transient and possibly cyclic regime. As the general problem is computationally complex, we first assume that the robot sequence is given and develop a multistage linear programming (LP) model to minimize the total makespan, subject to wafer residency constraints, and subsequently the average delay. Forging into less tractable territory, we next develop a branch and bound algorithm to find an optimal robot sequence with minimum wafer delay. This approach enables us to solve problems that were not previously solvable. Simulation studies demonstrate that when the number of process modules grows to more than five, the branch and bound algorithm may fail to find an optimal solution due to computational complexity. In this case, we suggest a transient sequence based on cyclic policies together with the LP model; it is within 2% of optimal.**

Keywords— **cluster tool, transient state, wafer delay constraint, linear programming, branch and bound**

I. Introduction

In 2011, the world semiconductor market revenue was US$307 billion according to Garner, Inc., and is expected to grow to US$344 billion by 2014 according to the International Data Corporation. To increase the productivity of semiconductor wafer fabrication, cluster tools are widely used. A cluster tool consists of processing modules (PM) and wafer handling robot(s) housed in a single chassis. We will consider cluster tools with single armed robots that perform three tasks: pick-up (or unload) a wafer from a processing module, move from one processing module to another, and place (or load) a wafer into a processing module. In addition to the processing modules and wafer handling robot, there are also input and output load locks that store wafers without processing them. There may be a wafer aligner, cooler and buffer.

We consider the case where the tool, starting from empty, processes a fixed number of wafers and then empties. There is a single wafer handling robot and wafers must receive service from a fixed number of process modules in order (serial processing). When wafers initially enter, there is a start-up period. During this period, wafer exit times may not exhibit periodic behaviour, i.e., wafers come out of the system at irregular times; we call this the initial transient period. After processing several wafers, the system may enter a periodic regime referred to as steady state. Once the final wafers in a lot enter the tool, another aperiodic regime begins. This final transient period ends when the last wafer exits the tool. The time difference between the entry of the first wafer and the exit of the last wafer is called the makespan. The makespan is the inverse of the throughput. It is our primary goal to maximize the throughput. Readers may refer to [1]-[4] for studies on steady state behaviour and [5-8] for studies on transient analysis.

Minimizing wafer delay is the secondary goal of this paper. Wafer delay occurs when a wafer has finished processing but the robot is not available to pick it up. It is thus the difference between the epochs of unloading a wafer from a processing module and the end of the process provided by that processing module. Especially for chemical processes, if a wafer stays too long in a processing module, then residual gas and heat may cause the wafer quality to deteriorate. To combat this phenomenon, constraints on the wafer delays – referred to as time windows – are common. The sequential optimization of throughput followed by wafer residency time seems to have first been proposed in [9-10] in the context of flow line models of manufacturing. The concept is helpful in our context as well.

Focusing on steady state analysis with wafer delay constraints, Kim, et al. [11] proposed a method to find the feasible range of process times in dual armed cluster tools with wafer delay constraints. Kim [12] extended this work to allow disruptive events in single-armed cluster tools, and developed stabilizing strategies for an efficient return to steady state. Wu, et al. [13] proposed an analytical method to check the schedulability of single armed cluster tools with wafer residency time constraints and developed an algorithm to find the optimal cyclic sequence when it is schedulable. This was extended to allow changing activity times ([14]).

Focusing on transient analysis with wafer delay constraints, Kim, et al. [15] discussed how the latest starting policy minimizes the start-up period and earliest starting policy minimizes the close-down period while meeting the wafer delay constraints. However, without restricting attention to the backward sequence or assuming the tool is in steady state, there is no *systematic* approach to decide both *when* and *how* (in what sequence) to schedule the robot for the entire makespan with wafer delay constraints. We take steps to address this problem.

This paper is organized as follows. In Section 2, we propose a multistage linear programming model to answer the question of when to start each robot activity assuming a given sequence of activities. It employs an approach ([16]) enabling the modelling of a discrete-event system as an LP. In Section 3, we develop a branch and bound algorithm that uses an upper bound, lower bound, and feasibility check to address the larger question of what robot action sequence to use (we also obtain action times). In Section 4, we conduct numerical experiments to evaluate the performance of our branch and bound algorithm. Concluding remarks and future work are mentioned in Section 5.

II. Linear programming Model

A. Notation

Let m+1 be the total number of processing modules (PM), including the input and output load locks. The processing module index is in the set {0, 1, …, m}; 0 represents the input load lock and m represents the output load lock. Let n be the total number of wafers to be processed; the wafer index is in the set {1, …, n}. Wafers must receive service from each processing module in order. There is a single wafer handling robot. In this section, we will assume a given robot action sequence; it should be feasible (but our linear programming model will determine if it is not).

1) s =(s_1,s_2,…,s_k,…,s_{mn}) denotes a given robot action sequence, where $s_k \in$ {0,1,…,m-1} is the index of the processing module from which the k^{th} action starts. For example, s_5 = 2 means that the 5^{th} robot action in the sequence is to unload a wafer from the 2^{nd} processing module, move it to the 3^{rd} PM, and then load the wafer into the 3^{rd} processing module.

2) w =(w_1,w_2,…,w_k,…,w_{mn}) denotes the set of wafer indices associated with each of the robot actions, where $w_k \in$ {1,…,n} is the index of the wafer associated with the k^{th} robot action. For example, w_{13}=5 means that the 13^{th} robot action is handling the 5^{th} wafer.

For a given robot action sequence, the following pseudo-code determines the wafer index associated with each action.

```
Pseudo code 1:
Let wafer_index=1
(1) Initialize $w_i$ for i∈(1,2,…,mn) as 0.
(2) Update the value of each $w_i$ as follows.
For (j=1;j<=mn;j++){
        If $w_j$=0 then{//start of the first "if"
                $w_j$=wafer_index
                Set latest_index = j
                For (k=j+1;k<=mn;k++){
                        If $w_k$=0 and $s_k$=$s_{\text{latest\_index}}$+1 then
                                {//start of the second "if"
                                $w_k$=wafer_index
                                Update latest_index = k
                                }//end of the second "if"
                }
                Update wafer_index = wafer_index+1
                }//end of the first "if"
        }
```

For example, consider the case with two processing modules, an input load lock, an output load lock (m=3) and n=2 wafers. When the robot action sequence is s=(0, 1, 0, 2, 1, 2), we obtain w=(1, 1, 2, 1, 2, 2). That is, the 1st, 2nd and 4th robot actions move the first wafer, while the 3rd, 5th and 6th robot actions move the second wafer.

3) Table I provides the definition of other variables and constants.

TABLE I
TERMINOLOGY

Term	Type	Definition
u_{ij}	variable	Unloading epoch of wafer i from PM j
s_{ij}	variable	Start time of processing wafer i in PM j
f_{ij}	variable	Finish time of processing wafer i in PM j
r_i	variable	Robot available time for the i^{th} time
tw_j	constant	Maximum allowed wafer delay in PM j
$I_{\{s_{k+1} \neq s_k+1,\ k<mn\}}$	constant	If $s_{k+1} \neq s_k + 1$ and k<mn, then the value is 1; otherwise, it is 0.
p_j	constant	Process time of a wafer in PM j
δ	constant	Robot moving time
ε	constant	Robot pick / place time

B. Constraint

Chan, et al. [16] proposed a method to model the dynamics of discrete-event stochastic systems as optimization problems. They presented a procedure that maps a simulation event relationship graph into a mixed-integer program. Our linear programming model is a special case of this approach, where the following constraints define and enforce the desired system behaviour.

1) Initial condition constraints

(1) $f_{w_k,0}=0$, for $\forall w_k$

(2) $tw_0=\infty$

(3) $p_m=0$

(4) $r_0=0$

2) Primary constraints

For each k $\in$ {1,…,mn}:

(5) $u_{w_k,s_k} \geq r_{k-1}$

(6) $u_{w_k,s_k} \geq f_{w_k,s_k}$

(7) $u_{w_k,s_k} - f_{w_k,s_k} \leq tw_{s_k}$

(8) $r_k \geq u_{w_k,s_k} + 2\,\varepsilon + \delta + \delta \cdot I_{\{s_{k+1} \neq s_k+1, k<mn\}}$

(9) $s_{w_k,s_k+1} = r_k - \delta \cdot I_{\{s_{k+1} \neq s_k+1, k<mn\}}$

(10) $f_{w_k,s_k+1} = s_{w_k,s_k+1} + p_{s_k+1}$

Constraint (5) ensures that the robot unloads a wafer only when the robot is available. Constraint (6) requires the robot to wait until a wafer has finished processing before it unloads it. Constraint (7) enforces the wafer delay constraints. Constraint (8) calculates the robot available time for the next action. It is not equality because we may want to delay the loading to meet the wafer delay constraint. Constraint (9) allows a wafer to start processing once it is loaded into a processing module. Constraint (10) sets the finish time of processing to the start time plus processing time.

C. Objective Function

Our multistage LP model is constructed as follows.

(1) Minimize makespan:

$$\min s_{n,m}$$
s.t. constraints (1)~(10)

We call this linear programming model as τ_1, and denote its optimal value as V_{τ_1}. The resulting total wafer delay we denote as WD_{τ_1}. If feasible, this linear programming model provides the minimal makespan possible while meeting the time window constraints.

We subsequently seek to minimize wafer delays with the following linear programming model that we call τ_2.

(2) Minimize total wafer delays:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m-1}(u_{ij} - f_{ij})$$
s.t. constraints (1)~(10)

$$s_{n,m} = V_{\tau_1}$$

Let V_{τ_2} denote the optimal value of τ_2. This linear programming model minimizes the total wafer delay while subject to reaching the minimum makespan and meeting the wafer delay constraints. Since there are two objective functions and they are solved one by one, we call this a multistage linear programming model.

The multistage LP model gives higher priority to the makespan objective than to the total delay objective, with the assumption that the throughput maximization is more important as long as the wafer delay constraints are not violated. However, one may modify τ_1 and τ_2 to balance the trade-off between V_{τ_1} and V_{τ_2}. For example, importance factors, denoted as d_1 and d_2, may be given to V_{τ_1} and V_{τ_2}. Then minimizing $d_1 s_{n,m} + d_2[\sum_{i=1}^{n}\sum_{j=1}^{m-1}(u_{ij} - f_{ij})]$ would become the objective function, while the constraints (1)~(10) are kept the same.

Readers may refer to [9] and [10] for the use of sequential optimizations via linear programming in flow lines. In [9], Park, et al. proposed an optimization algorithm for flow lines to reduce wafer residency times and maximize throughput. In [10], Park, et al. developed a linear program to determine when to admit the preordered jobs into flexible flow lines, which resulted in significant reductions in wafer residency time, in-tool buffer occupation and hot lot queueing time.

D. LP Performance Evaluation

We consider some examples using ILOG CPLEX 12.4 with JAVA on a PC with Intel dual core CPU, 2.4GHz and 3GB RAM. Table II provides the results with the backward sequence. For odd PM indices such as the first and third PM, the process time is set as 100 s; for even PM indices such as the second and fourth PM, the process time is set as 150 s. Furthermore, tw_j = 10 s for all j, and δ = ε =1s. We define m' to be the number of processing modules excluding input and output load lock; thus, m'=m-1. Average data, such as optimal delay per wafer (V_{τ_2}/n), is rounded to two decimal points. Let C_T be the computation time. As it is shown in Table II, the LP model is reasonably tractable.

TABLE II
LP PERFORMANCE WITH THE BACKWARD SEQUENCE

m'	n	V_{τ_1}/n	WD_{τ_1}/n	V_{τ_2}/n	C_T
2	25	161.08s	9.60s	0s	<1s
2	1000	157.10s	8.97s	0s	3s
2	5000	157.02s	8.98s	0s	13s
3	25	165.20s	11.60s	0s	<1s
3	1000	157.21s	14.91s	0s	3s
3	5000	157.04s	14.49s	0s	18s
4	25	171.32s	7.20s	0s	<1s
4	1000	157.36s	9.99s	0s	5s
4	5000	157.07s	0s	0s	138s
5	25	175.44s	17.04s	14.28s	<1s
5	1000	157.46s	16.96s	16.93s	7s
5	5000	157.09s	16.99s	16.99s	189s
6	25	182.00s	20.40s	18.36s	<1s
6	1000	157.63s	20.98s	20.93s	11s
6	5000	157.13s	20.99s	20.99s	308s

From the data for the example given in Table II, we can see that WD_{τ_1}/n becomes positive when m' is increased to 5. This is reasonable because as the number of PMs increase, there tend to be more wafers in the system. The robot is busier and sometimes unavailable to handle the wafers immediately as they are available.

Note that there are two cases in which the LP model fails to provide a solution. The first case is when the given robot action sequence is itself infeasible. This case violates our assumption of starting with a viable sequence and can be fixed by providing a feasible one (which could be obtained using those inspired by steady state policies as in Section III.A below). The second case is when the time windows cannot be satisfied with the given feasible sequence. We next develop a branch and bound algorithm to identify an optimal sequence; the approach is less computationally tractable but guarantees an optimal solution if one exists.

III. Branch and Bound Algorithm

We next endeavour to develop a branch and bound algorithm to identify an optimal robot action sequence as well as to determine the actions times. We pursue two goals sequentially as before.

In order to increase the computational efficiency, we use three mechanisms: upper bound, lower bound, and feasibility check in our branch and bound algorithm. First, we find an upper bound on the optimal makespan value by choosing the best transient robot sequence inspired by the steady state sequences. Second, we apply a dynamic programming algorithm to find a lower bound on the makespan. If the lower bound exceeds the upper bound, we stop searching and move to another branch. Finally, we apply the linear programming model τ_2 to check the feasibility of the survived branches.

A. Upper Bound

We develop an algorithm to find an upper bound on the optimal makespan with wafer delay constraints. We do this by considering robot action sequences inspired by the one-unit cycle sequences.

For a given one-unit cycle sequence, we generate an entire sequence that includes all actions from the start of the first wafer to the end of the last one. For a given m, there are (m-1)! different one-unit cycle sequences, so the computation required is limited.

The entire sequence is generated as follows. Let s^{s-s} = (s_1^{s-s}, …,s_k^{s-s},…,s_m^{s-s}) denote the one-unit cycle sequence, where $s_k^{s-s} \in$ {0, 1, …, m-1}. Since it is a one-unit cycle sequence, each value in {0, 1, ..., m-1} occurs exactly once. Let s be the corresponding full sequence including all robot actions from start to finish of the work (which is initially empty when we start our algorithm to construct it).

TABLE III
TERMINOLOGY FOR PSEUDO CODE 2

Term	Type	Definition
R_i	constant	Robot position after ith robot action in s
j	constant	Index of the full sequence s
max	constant	Maximum index of the feasible action
nw_i	constant	Number of wafers processed by PM i

```
Pseudo code 2:
1) Initialization
max = 0
R_0 = 0
j= 1
nw_i=0 for i∈(0,1,…,m-1)
2) Determine the value of each $s_k$ for each k∈(1,2,…,mn) as follows.
(2-1) Start-up and steady-state period
While $nw_0$<n, do the following.
For (i = 1; i <=m; i++) {
        If ($s_i^{s-s}$ <= $R_{j-1}$) or ($s_i^{s-s}$> $R_{j-1}$ and $s_i^{s-s}$ <= max), then
                {//start of first "if"
                Set $s_j$ = $s_i^{s-s}$
                Update $R_j$ = $s_i^{s-s}$ + 1
                Update $nw_{s_i^{s-s}}$ = $nw_{s_i^{s-s}}$ + 1
                If max < $R_j$, then {//start of second "if"
                        Set max = $R_j$
                        }//end of second "if"
                Update j = j+1
                }//end of first "if"
}
(2-2) Close-down period
While $nw_{m-1}$< n, do the following.
For (i = 1; i <=m; i++) {
        If $nw_{s_i^{s-s}}$ <n, then{//start of first "if"
                Set $s_j$ = $s_i^{s-s}$
                Update j= j+1
                Update $nw_{s_i^{s-s}}$ =$nw_{s_i^{s-s}}$+1
                }//end of first "if"
        }
```

Pseudo code 2 generates a full robot sequence from a steady state sequence. By checking the number of wafers processed by each PM, we ensure that every PM has processed exactly n wafers. The generated sequence is deadlock free because it satisfies the feasibility condition mentioned in [2]. In the start-up period, the term "max" plays the role of maximum index for a feasible robot action, and "s_i^{s-s} <= max" avoids robot unloading an unoccupied PM. Additionally, we avoid loading an occupied PM by walking through the s^{s-s} one by one and adding appropriate s_i . In the steady state, the algorithm iterates s^{s-s} so that no deadlock will occur. In the close-down period, "nw_{m-1}< n" ensures that the robot does not unload an unoccupied PM. Similarly in the start-up period, we avoid loading an occupied PM by adding each s_i according to the sequence of s_i^{s-s}. As such, the feasibility (deadlock free) condition is met for the sequences generated by Pseudo code 2.

We apply the linear programming model developed in Section II to all the sequences generated from the one-unit cycle sequences. Each is thus checked for feasibility with respect to the wafer delay constraints and the robot timing is adjusted to obtain the minimal makespan for that sequence. From all of these sequences generated from the one-unit cycle sequences, we select one that satisfies all of the LP constraints and achieves the minimal makespan. This makespan value will serve as an upper bound on the achievable performance for all possible robot action sequences and timings. Of course, if no such feasible policy is generated from the one-unit cycle sequences, the upper bound is considered as infinity.

Example 1: **Obtaining an upper bound.** *Consider the case m=4.*

First, enumerate all the possible one-unit cycle sequences. There are 3!=6 such sequences; they are {3,2,1,0}, {3,2,0,1}, {3,1,2,0}, {3,1,0,2}, {3,0,2,1,} and {3,0,1,2}.

Second, generate a robot action sequence for each one-unit cycle sequence via Pseudo code 2. There is one for a given one-unit cycle sequence. For example, consider the case m=3 and s_{s-s} = {2,1,0}. The resulting robot action sequence is s={0,1,0,2,1,0,2,1,2}.

Third, apply the multistage linear programming models from Section II to each s.

Fourth, select a best sequence from among these full robot actions sequences as follows. It must be feasible, obtain the minimum makespan from among the sequences, and achieve the minimal average wafer delay from among those with minimal makespan. □

The selected sequence serves as an upper bound for our branch and bound model. If we can prove that a branch in the branch and bound tree will provide a larger makespan than the upper bound, we will discard it. To this end, we next develop a lower bound.

B. Lower Bound

The minimum makespan of a sequence with wafer delay constraints cannot be less than the minimum makespan of the same sequence without wafer delay constraints, all other things being equal. Therefore, the minimum makespan without time windows can serve as a lower bound for our problem. Now the task is to find an efficient algorithm that helps obtain this lower bound. In [8], Wikborg et al. suggested an efficient dynamic programming (DP) algorithm to find an optimal robot sequence without time window constraints. The main idea is that when there are two identical states (same wafer occupancy in each chamber), we compare the ready times of each resource and discard the state that is inferior.

In [8], the DP algorithm obtains an optimal solution for a problem with 5 process steps and 1000 wafers within 1 second. As the speed of this algorithm is substantially faster than others in existence, we employ it to obtain our lower bound.

Since the DP algorithm does not consider wafer delay constraints, it might discard the states that we want to keep. As such, we do not discard the states in the branch and bound tree at this stage. For each state, we obtain the minimum makespan from the DP algorithm without wafer delay constraints. If this lower bound (LB) on makespan exceeds the upper bound (UB), then we discard the state.

Here we briefly introduce the branching strategy that is used in [8]. Let n_i denote the number of wafers in the i^{th} PM. Then, the state $(n_0, n_1,\ldots,n_m)$ shows the wafer occupancy of each PM at some point of time. For instance, (4,0,1,0) means that there are 4 wafers in the input load lock, 1 wafer in the second PM, and no wafers in the other PMs. The next possible robot action is either 0 or 2; the robot can either unload a wafer from the input load lock or a wafer from the second PM. As such, from a given state, we can determine the next possible robot action(s) and the subsequent next state. Each subsequent state is the descendent of the original state. The robot action to reach that descendent is embedded in the arrow from parent to descendent. The result is a reachability graph.

<u>Example 2:</u> Lower bounds. *Consider the case m=3, n=4, p_1=100s, p_2=150s, δ=ε=1s. Using the preceding, an upper bound sequence can be obtained as s={0,1,0,2,1,0,2,1,0,2, 1,2}. Its minimal makespan and total wafer delay are 730s and 0s.*

Figure 1 shows the reachability tree or branch and bound tree. Here, the states are a vector of four integer values. Each value is the number of wafers in the corresponding process module. The DP algorithm provides a lower bound on the makespan of states (3,0,0,1), (2,0,0,2) and (1,0,0,3) as 833s. This exceeds the upper bound of 730s, so these states are discarded. As a result, only one branch remains; it is the upper bound sequence, s={0,1,0,2,1,0,2,1,0,2,1,2}. In Figure 1, states with colours are the states that have been discarded based on the UB and LB comparison. □

For the surviving states, we extract the robot sequence from the reachability graph and apply the multistage linear programming model in Section II to check its feasibility for the wafer delay constraints. (We are guaranteed at least one feasible branch if there was a feasible sequence obtained from the extended one-unit cycle sequences.) The branch and bound algorithm with upper bound, lower bound and feasibility check follows.

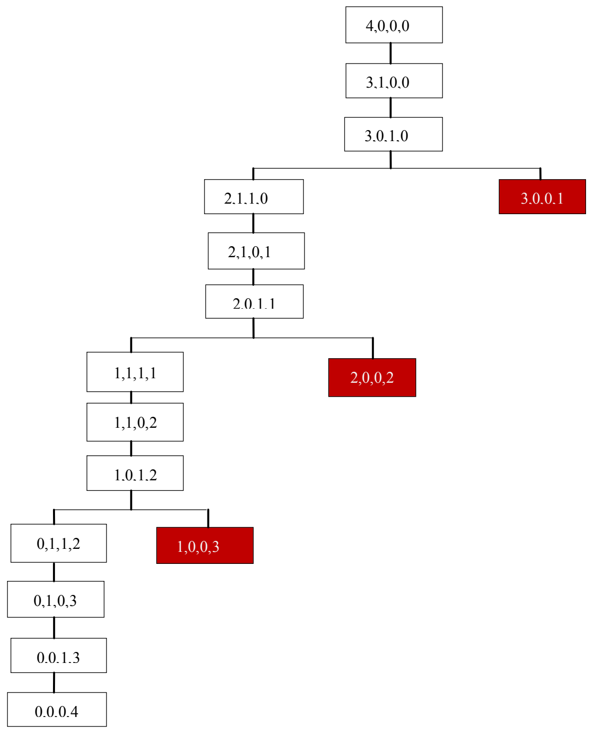


Fig. 1 Reachability graph using the UB and LB elimination strategy

First, the DP algorithm of [8] finds the minimum makespan for each state without considering the wafer delay constraints. Denote this makespan as V_{lb}.

Second, the LP of Section II is used to obtain an upper bound on the makespan. Denote it as V_{ub}.

Third, if V_{lb}=V_{ub} and the total wafer delay of the upper bound is zero, then we have an optimal solution. Otherwise, we generate descendant states. For each generated state, find its lower bound and discard the state if it exceeds the upper bound. Also, check the feasibility of each branch and discard one that violates the wafer delay constraints.

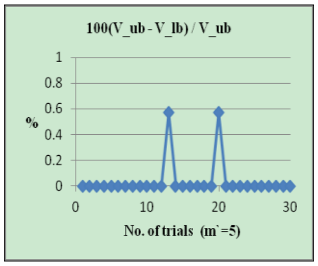
IV. Performance Evaluation

To assess the computational efficiency of the proposed approach, we studied several cases. We created 30 randomly generated cases each for m'=3, 4, 5 and 6.

For each case, the process times, time windows and robot move times were uniformly distributed and independent of all others. The stage processing times, time windows, δ and ε were uniformly distributed in the range [50, 300], [0, 10], [1, 2] and [1, 2] seconds, respectively. For m'≤4, we obtained an optimal solution for all the 60 cases. For m'≥5, there were 6 cases out of the 60 trials in which the algorithm did not come to a solution due to the heavy computational load. The average computation times to obtain an optimal solution were 1, 1.8, 5.9 and 24 seconds for m'=3, 4, 5 and 6, respectively, excluding the timed out cases.

For the cases where the branch and bound algorithm failed to converge, we use the upper bound as a good candidate.

Figures 2 and 3 show the difference between the upper bound and lower bound for m'=5 and m'=6. The maximum gap between the lower bound and upper bound in all the trial cases is within 2%. Since the optimal solution must have value between the upper bound and lower bound, the difference between the upper bound and optimal makespan is at most 2%.



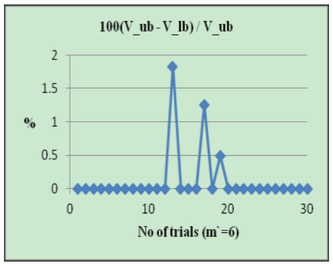


Fig. 2 UB vs. LB for m'=5

Fig. 3 UB vs. LB for m'=6

V. Conclusion

We proposed a multistage linear programming (LP) model that consists of two LPs. The first LP minimizes the total makespan of any given robot sequence while meeting wafer delay constraints, if that is possible. If the first LP is feasible, we apply the second LP to minimize the total wafer delays. Both LP models adjust the timings of each robot action to meet the wafer delay constraints.

Next we developed a branch and bound algorithm to find an optimal robot sequence. A DP approach was employed to obtain a lower bound on the makespan. An approach based on extending the one-cycle sequences into the transient regime was used to identify an upper bound on the makespan. The multistage LP was used to check the time window feasibility of each branch. The lower bound, upper bound and feasibility check are employed to eliminate fruitless paths.

This approach is relatively simple but gives optimal or good solutions in a relatively short time without assuming the backward sequence.

Since our focus is on single armed circular cluster tools, future work may extend the approach to dual armed cluster tools, linear cluster tools or multi-cluster tools. It would also be useful to allow wafer reentrance or lots with different recipes.

Acknowledgment

This work was supported in part by Korea Research Foundation (KRF) Grants N01110132-00 and N01120191.00.

References

[1] Terry L. Perkinson, Peter K. McLarty, Ronald S. Gyurcsik, and Ralph K. Cavin III. "Single-wafer cluster tool performance: an analysis of throughput." *IEEE Transactions on Semiconductor Manufacturing,* Vol.7, No.3, August 1994.

[2] Milind W. Dawande, H. Neil Geismar, Suresh P. Sethi, and Chelliah Sriskandarajah. "Throughput optimization in robotic cells", *Springer`s International Series in Operations Research & Management Science*, Volume 101, 2007

[3] Tae-Eog Lee, Hwan-Yong Lee, and Yong-Ho Shin. "Workload balancing and scheduling of a single-armed cluster tool," *Proceedings of the 5th Asian-Pacifica Industrial Engineering and Management Systems Conference*, Gold Coast, Australia, Dec. 2004.

[4] Dae-Kyu Kim, Yu-Ju Jung, Chihyun Jung, and Tae-Eog Lee. "Cyclic scheduling of cluster tools with non-identical chamber access times between parallel chambers," *IEEE Transactions on Semiconductor Manufacturing*, vol. 25, no. 3, pp. 420-431, 2012.

[5] Young-hun Ahn, and James R. Morrison. "Analysis of circular cluster tools: Transient behavior and semiconductor equipment models", *Proceedings of the 6th Annual IEEE Conference on Automation Science and Engineering*, Toronto, Canada, August 2010, pp. 39-44.

[6] Hyun-Jung Kim, and Tae-Eog Lee. "Scheduling cluster tools with ready time constraints for consecutive small lots," *Proceedings of the 2011 IEEE Conference on Automation Science and Engineering*, Trieste, Italy, Aug. 2011, pp. 96-101.

[7] Jun-Ho Lee, and Tae-Eog Lee. "Scheduling transient periods of single-armed cluster tools," *Proceedings of the 2012 IEEE Conference on Robotics and Automation*, Saint Paul, MN, USA, May. 2012, pp. 5062-5067.

[8] Uno Wikborg, and Tae-Eog Lee. "Non-cyclic scheduling for timed discrete event systems with application to single-armed cluster tools using Pareto-optimal optimization", accepted for publication in *IEEE Transactions on Automation Science and Engineering*

[9] Kyungsu Park, and James R. Morrison. "Control of wafer release in multi cluster tools", Proceedings of the 2010 8th IEEE International Conference on Control and Automation (ICCA 2010), Xiamen, China, June 2010, pp. 1481-1487.

[10] Kyungsu Park and James R. Morrison, "Scheduling of job release in flexible flow lines: An LP approach and applications to semiconductor wafer fabrication", in revision for the IEEE Transactions on Automation Science and Engineering (IEEE).

[11] Ja-Hee Kim, Tae-Eog Lee, Hwan-Yong Lee, and Doo Byeong Park.. "Scheduling analysis of time-constrained dual-armed cluster tools," *IEEE Transactions on Semiconductor Manufacturing,* Vol. 16, No. 3, p. 521 - 534, 2003.

[12] Ja-Hee Kim, "Stable Schedule for a single-armed cluster tool with time constraints," 4th *IEEE Conference on Automation Science and Engineering,* Key Bridge Marriott, Washington DC, USA, Aug. 2008.

[13] Naiqi Wu, Chengbin Chu, Feng Chu, and Meng Chu Zhou. "A petri net method for schedulability and scheduling problems in single-arm cluster tools with wafer residency time constraints", *IEEE Transactions on semiconductor manufacturing*, vol.21, No.2, May 2008.

[14] Yan Qiao, NaiQi Wu, and MengChu Zhou. "Petri net modeling and wafer sojourn time analysis of single-arm cluster tools with residency time constraints and activity time variation", *IEEE Transactions on Semiconductor Manufacturing*, Vol.25, No.3, Aug. 2012.

[15] Tae-Kyu Kim, Chihyun Jung, and Tae-Eog Lee. "Scheduling start-up and close-down periods of dual-armed cluster tools with wafer delay regulation," *International Journal of Production Research*, vol. 50, no. 10, pp. 2785-2795, 2012.

[16] Wai Kin (Victor) Chan, and Lee Schruben. "Optimization models of discrete-event system dynamics", Operations Research, Vol. 56, No. 5, September-October 2008, pp. 1218-1237