Deadlock-free Rescheduling in Flexible Manufacturing Systems

Hoda A. ElMaraghy (1) andTarek Y. ElMekkawy
Intelligent Manufacturing Systems (IMS) Centre,
University of Windsor, Ontario, Canada

Abstract

The dynamic nature of manufacturing makes rescheduling essential in today's complex production environment, particularly in flexible and re-configurable systems. Research on optimizing schedules, that includes deadlock avoidance, is rather limited. Furthermore, the deadlock problem is mostly ignored in research on rescheduling. A rescheduling algorithm, that uses time Petri-Nets and the minimal siphons concept, was developed to deal with sources of disturbance such as machine breakdowns in real-time. It guarantees a deadlock-free new schedule. The existence of alternative routes, availability of material handling facilities, and the limitation of buffer capacities were taken into consideration. The developed algorithm modifies only the affected portion of the original schedule, rather than rescheduling all jobs, in order to limit changes to the original schedule and reduce the impact on the response time.

Keywords: Flexible manufacturing systems, Rescheduling and Time Petri Nets

1 INTRODUCTION

The continuous customer demands for larger variety of products, faster production rates, and higher delivery commitments are a challenge for most modern industries. The performance of a production system greatly depends on the availability of efficient rescheduling capability in order to deal with Changes in real-time. The re-configuration of manufacturing systems requires not only hardware changes but also software adaptation including replanning and rescheduling.

A variety of conditions, demands and constraints necessitate revisions to existing schedules. Some of the common reasons for rescheduling are [Li et al., 1993]: machine breakdown; rush order arrival, shortage of materials, quality problems, over- or under-estimation of process time, order cancellation, due date changes, and being behind or ahead of the current schedule. Manual rescheduling is not practical for complex and automated manufacturing systems. It can be done using computer tools such as electronic Gantt chart, simulation, expert systems, algorithmic methods as well as hybrid approaches. Early surveys of real-time scheduling include Harmonosky and Robohn (1991), Basnet and Mize (1994), Shukla and Chen (1996), Harmonosky (1995), Szelke and Kerr (1994) and Gonzalez (1995).

Lin (2000) presented a dynamic controller with a manufacturing system capability database. Wiendahl and Breithaupt (2000, 2001) applied control theory principles to production control and scheduling. Jain and ElMaraghy (1997) developed local rescheduling GA-based algorithms that generate a new schedule without re-evaluating all tasks in the original schedule. Li et al. (1993) proposed a rescheduling algorithm based on the construction of a scheduling binary tree and a net change concept adopted from MRP systems. Yamamoto and Nof (1985) used a regeneration method to reschedule the entire set of operations, including those unaffected by the disturbance. Tipi and Bennett (1999) presented a simulation approach for dynamic and stochastic scheduling using ARENA software.

The developed rescheduling algorithm extends the work of Li et al. (1993). They did not consider any change in the existing operations sequence of each machine, alternate routings or the deadlock problem. Infinite capacity of resources such as material handling and buffers is often assumed albeit unrealistic in most cases. However, limiting such resources is a major cause of deadlocks in manufacturing systems that complicate the rescheduling task. Therefore, the developed rescheduling algorithm, presented in this paper, takes into consideration the limitations of both material handling devices and buffers and aims at achieving the highest possible system performance in the least response time while guaranteeing a new deadlock-free schedule.

2 DEADLOCK-FREE RESCHEDULING ALGORITHM

A Gantt chart is used to define the original schedule. The Petri Net model of the system is used as an input to the rescheduler. The following assumptions are used: 1) initial deadlock-free schedule is available; 2) no pre-emption or job splitting is allowed on the alternative resource; 3) machine breakdowns are the only source of disturbance; 4) setup time of any operation is only resource dependent; and 5) resources process only one operation at a time.

Upon a machine breakdown occurrence, the alternative machines that can perform the interrupted operation are determined. Routing the interrupted operation to one of these alternatives (if any) is investigated. If it is decided to route the interrupted operation to one of the alternatives, the original process plan of the succeeding operations (including material handling facilities, buffers and machines allocation) of the job of the interrupted operation will become obsolete. Therefore, a new process plan for these succeeding operations is defined and the schedule is updated. The schedule is fed to the developed rescheduling algorithm as a Gantt chart. The algorithm comprises three parts. The *Control Routine* receives the input data upon machine breakdown, and modifies the original schedule. The *Updating Routine* finds the set of operations affected by the disturbance, and updates the Gantt chart. The *Verification Routine* checks whether the new schedule is deadlock-free. The following notations are used in the developed algorithm:

- $O_{i,j}$: operation j of job i.
- $O_{k,w}$: next operation on resource r after operation $O_{i,j}$.
- $pt_{i,j}$: processing time of $O_{i,j}$.

- st_r: setup time of *DAO* on machine r: $r \in$ *Alt_route_set*.
- $s_{i,j}$: planned starting time of $O_{i,j}$.
- $e_{i,j}$: planned ending time of $O_{i,j}$.
- *dt*: time of disturbance.
- *edt*: the expected down time of the broken machine.
- DAO: directly affected operation. The operation that is interrupted due to the machine breakdown.
- *Alt_route_set*: the set of machines which can serve *DAO*.
- ewt_r: expected waiting time of *DAO* to start processing on machine *r*: $r \in$ *Alt_route_set*.
- ept_r: expected processing time of *DAO* on machine *r*: $r \in$ *Alt_route_set*.
- ast_r: additional setup time of *DAO* on machine *r*: $r \in$ *Alt_route_set*.
- est_r: expected starting time of *DAO* on *r* ($est_r = dt + ewt_r$).
- ect_r: The expected completion time of *DAO* by machine *r* ($ect_r = est_r + ept_r + st_r + ast_r$).
- *Affected_operations*: Set of affected operations due to the disturbance.
- $OT_{i,j}$: Type of operation $O_{i,j}$: $O_{i,j} \in$ *Affected_operations*. $OT_{i,j}$ has the mapping: $0 \rightarrow$ *DAO*, $1 \rightarrow$ succeeding operation of the same job of *DAO*, and $2 \rightarrow$ any other operation.
- $RT_{i,j}$: ending time of previous operation on the same resource of operation $O_{i,j}$: $O_{i,j} \in$ *Affected_operations*.
- $JT_{i,j}$: ending time of previous operation of the same job of operation $O_{i,j}$: $O_{i,j} \in$ *Affected_operations*.
- *RD*: binary variable represents the routing decision.
- *FT*: binary variable represents the result of the deadlock existence test with the mapping: $0 \rightarrow$ the schedule has a deadlock, and $1 \rightarrow$ the schedule is deadlock-free;
- *M_Affected_operations*: Set of affected operations, due to the disturbance, that are already modified.

Control Routine:

```
1- Find the operation ($O_{i,j}$) that is interrupted due to the
     machine breakdown. Let DAO=$O_{i,j}$. Add DAO to
     Affected_operations and M_Affected_operations.
2- Evaluate the Alt_route_set of DAO.
3- If (Alt_route_set = $\phi$) go to step (9).
4- Find the machine r: $r \in$ Alt_route_set, with the
     minimum ect. Alt_route_set=Alt_route_set-r.
5- If ($ect_r \geq e_{DAO}+edt$) go to step (3)
6- Let RD =1 and use the Updating routine.
7- Use the Verification routine. If (the new schedule is
     NOT deadlock-free) then restore the data and go to
     step (3).
8- If there are other operations scheduled on the
     broken machine during its downtime, then let DAO
     = the operation with the earliest starting time, and
     go to step (2), otherwise stop.
9- Let RD = 0 and use the Updating routine.
```

Updating Routine

```
1.     While (Affected_operations $\neq \phi$)
1.1.   Select an operation ($O_{i,j}$) from Affected_operations
          that has the earliest starting time
1.2.   If (RD=1 AND $OT_{i,j}$=0){Let r be the resource of $O_{i,j}$;
          $s_{i,j}=est_r$; $e_{i,} = ect_r$}
1.3.   If(RD = 0 AND $OT_{i,j}$ = 0){$e_{i,j} = e_{i,j} + edt$}
1.4.   If (RD =1 AND $OT_{i,j}$ =1){Evaluate the Alt_route_set
          of $O_{i,j}$; Find the resource r:$r \in$Alt_route_set, with
          the minimum ect; Let r be the resource of $O_{i,j}$; $s_{i,j}$
          = $est_r$; $e_{i,j} = ect_r$;}
1.5.   If ($OT_{i,j}$ = 2){If($RT_{i,j} > JT_{i,j}$){$s_{i,j}=RT_{i,j}$;}; Else {$s_{i,j}=JT_{i,j}$;
          $e_{i,j}=s_{i,j}+pt_{i,j}+st_{i,j}$}}
1.6.   If ($O_{i,j}$ is NOT the last operation of its job AND
          $s_{i,j}>s_{i,j+1}$ )
1.6.1  If($O_{i,j+1} \notin$ Affected_operations){Add $O_{i,j+1}$ to
          Affected_operations and
          M_Affected_operations}
1.6.2  If($O_{i,j+1} \notin$ Affected_operations){Add $O_{i,j+1}$ to
          M_Affected_operations}
1.6.3  $OT_{i,j+1}$ =2; $JT_{i,j+1}=e_{i,j}$
1.7.   If($O_{i,j}$ is NOT the last operation of the resource r
          AND there is an operation $O_{k,w}$ scheduled on r
          with $e_{i,j} > s_{k,w} > s_{i,j}$)
1.7.1     If($O_{k,w} \notin$Affected_operations){Add $O_{k,w}$ to
          Affected_operations}
1.7.2     If($O_{i,j+1} \notin$Affected_operations){Add $O_{i,j+1}$ to
          M_Affected_operations}
1.7.3     $OT_{k,w}$ = 2; $RT_{k,w} = e_{i,j}$
1.8.      If($OT_{i,j} \neq 0$ AND $O_{i,j}$ is NOT the first operation of
          its job AND $s_{i,j}>e_{i,j-1}$)
1.8.1.    $e_{i,j-1}= s_{i,j}$
1.8.2.    If($O_{i,j-1}$ is NOT the last operation of the resource
          r AND there is an operation $O_{l,m}$ scheduled on r
          with $e_{i,j-1}>s_{l,m}>s_{i,j-1}$){If($O_{l,m} \notin$Affected_operations){Add $O_{l,m}$ to
          Affected_operations};
          If($O_{i,j+1} \notin$Affected_operations){Add $O_{i,j+1}$ to
          M_Affected_operations};$OT_{l,m}$=2;$RT_{l,m}=e_{i,j-1}$}
1.9.      Affected_operations=Affected_operations-$O_{i,j}$
1.10.     End while
```

Verification Routine:

```
1. Let FT=1 ;
2. While (M_Affected_operations $\neq \phi$)
3. Select an operation $O_{i,j}$ from M_Affected_operations
   and remove it from M_Affected_operations; let T=$s_{i,j}$;
4. Let the marking of each resource place r be $N_r$ and
   every operation place is zero ;
5. Find the set of operations that are currently served
   or starting at T. Let the marking of their places equal
   1 and of their resources equal $N_r$-1 ;
6. Find the set of operations that are ending at T. Let
   the marking of their places equal 0 and of their
   resources r equal $N_r$+1;
7. If (there is an empty siphon at T), let FT=0 and stop;
8. If ($T \neq e_{i,j}$), let T= $e_{i,j}$ and go to step (5);
9. End while;
```

The *Control Routine* is the core of the rescheduling algorithm. The interrupted operation (*DAO*), due to machine breakdown, is found and the set of alternative machines (*Alt_route_set*) that can perform it is evaluated. If the *Alt_route_set* is empty, the interrupted operation will be kept on the broken machine and the schedule will be modified based on the expected machine downtime (*edt*). If the *Alt_route_set* is not empty, the expected completion time (ect_r) of *DAO* on every member of *Alt_route_set* is estimated using the following formula: