A novel approach to include limited equipment connectivity in State-Task Network models

Mikael Nyberg
CENTER OF EXCELLENCE IN OPTIMIZATION AND SYSTEMS ENGINEERING AT ÅBO AKADEMI UNIVERSITY

ÅBO DECEMBER 8 2011
Table of Content

- Introduction
  - STN models
  - Limited equipment connectivity
  - Task Splitting
- State Splitting
  - Novel approach
  - Mathematical formulation
  - Benefits
  - Improvements
- Conclusions
State Task Network models

- **STN models**
  - Are used for modeling batch process in both discrete and continuous time
- **An STN graph consists of**
  - Task nodes described as rectangles
  - State nodes described as circles
  - Directed arches
- **States represent commodities**
- **Tasks transform one or more states into a new state**
  - Tasks are preformed on units
- **Arches describe batches of commodities moving through the graph**

**Example: A simple STN graph**

![Simple STN Graph](image)
Limited equipment connectivity

- STN graphs don’t explicitly include units in the graph
  -Units are included as an index in the mathematical model

- Limited equipment connectivity
  -When at least one unit in a production stage is not connected to all units in the next stage
  -Can only occur in multi-stage systems
  -Common in many industries

Example: Limited equipment connectivity

```
Unit M1 → Unit N1
Unit M2 → Unit N2
```

Mikael Nyberg: Limited equipment connectivity in State-Task Networks
Center of Excellence in Optimization and Systems Engineering at Åbo Akademi University
Task Splitting

- Kondili et al. proposed task splitting for including limited equipment connectivity in STN models
- Task Splitting does the following:
  - Duplicates tasks for units in the later stage to include limited connectivity
  - Task Splitting does not require any modifications to the mathematical model, only additional tasks are required
- The method has two drawbacks:
  - Increases the number of binary variables
  - Prohibits merging of batches

Example: Task Splitting

```
Task 1
Unit M1
Unit M2

Task 2
Unit N1
Unit N2

State M1
State M2

Task 1_1
Available units: M1

Task 1_2
Available units: M2

Task 2_1
Available units: N1

Task 2_2
Available units: N1, N2
```
State splitting

- Includes limited connectivity by splitting states and divides batches into sub batches instead of splitting tasks
- Requires:
  - A new set of continuous variables
  - Reformulation of the mass balance constraint
  - Introduction of a new set of balance constraints

Example: State Splitting
State Splitting - three ways to do it

For every unit configuration with limited connectivity State Splitting can be done in (at least) three ways:

- Forward State Splitting (SS-F)
- Backward State Splitting (SS-B)
- Inter State Splitting (SS-I)

The advantage of SS-F and SS-B are that one of them is always able to produce an STN model with a minimal number of additional variables.

The advantage of SS-I is that all limited equipment connectivity configurations are dealt with the same way:
- Reduces the modeling effort and simplifies the resulting mathematical model

All three State Splitting methods can be mixed if need be.

Example: State Splitting methods

- SS-B: 3 new var./time step
- SS-F: 3 new var./time step
- SS-I: 3 new var./time step
Mathematical formulation

- State Splitting for a discrete time STN model

New continuous variables

\[
S_{s,j',t} \quad \forall \ t, (s,j') \in J_s^s \quad \text{amount of state } s \text{ available for unit } j'
\]

\[
B_{i,j,j',t} \quad \forall \ t, j', j \in J_j, \quad \text{size of batch } i \text{ produced by unit } j \text{ going to unit } j'
\]

Reformulated mass balance constraint

\[
S_{s,j',t} = S_{s,j',t-1} + \sum_{i \in T_s} \sum_{j \in J_j} B_{i,j,j',t-p_{i,s}} - \sum_{i \in T_s} \sum_{j' \in K_i} B_{i,j',t} \quad \forall \ t, (s,j') \in J_s^s
\]

New batch balance constraint

\[
B_{i,j,t} = \sum_{j' \in J_j} B_{i,j,j',t} \quad \forall \ t, j, i \in I_j^s
\]
Benefits of State Splitting

- State Splitting overcomes both drawbacks of Task Splitting as:
  - Only continuous variables are added to the model
    - The addition of only continuous variables reduces the increase in computation time compared to Task Splitting
    - The number of additional variables is roughly the same as for Task Splitting
  - Merging (and splitting) of batches is possible
    - The possibility to merge batches increases the flexibility of the mathematical model
    - In some cases this improves the solution quality
- The only drawback of State Splitting is a slight increase in model complexity

Comparison: Number of variables

<table>
<thead>
<tr>
<th>Unit configuration</th>
<th>Method</th>
<th># of binary *</th>
<th># of cont. *</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit M1 → Unit N1</td>
<td>Task Splitting</td>
<td>5</td>
<td>19</td>
</tr>
<tr>
<td>Unit M2 → Unit N2</td>
<td>State Splitting</td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

*Number of variables / time step
Computational results for a discrete time STN model using State Splitting

- A simple 2-commodity production planning problem
  - Objective function: maximize profit
  - 4 different unit configurations to test the different State Splitting methods compared to Task Splitting
  - Solved with 11 MILP solvers using GAMS 23.7

Unit configuration 1 - solution times (Cplex)
More results

Unit configuration 2 - solution times (Cplex)

Unit configuration 3 - solution times (Cplex)
Improving solution quality

- State Splitting is able to produce better results than Task Splitting when:
  - The optimal solution includes at least one occurrence of batch merging
  - This is not possible in Task Splitting due to the following:
    - Tasks are duplicated
    - The allocation constraint only allows task to be executed on each unit
  - In State Splitting this can be done because no tasks have been duplicated

**Example: Task duplication**
Improving solution quality

Unit configuration 1 - solution quality

Unit configuration 2 - solution quality

Unit configuration 3 - solution quality
Conclusions

- Solution times
  - Task Splitting vs. State Splitting
    - In all but one unit configuration at least one State Splitting method produced faster results than Task Splitting
    - In 50% of the cases all State Splitting methods were faster than Task Splitting
    - The results are consistent regardless of solver
  - SS-B vs. SS-F vs. SS-I
    - SS-I introduces more additional variables in all test cases compared to both SS-B and SS-F
      - This does not seem to impact solution efficiency
      - Good news as SS-I is easier to implement than SS-B or SS-F
Conclusions

- Solution quality
  - Task Splitting vs. State Splitting
    - In 81% of all test problems State Splitting produced better results than Task Splitting
    - The average improvement was 2.01%
  - SS-B vs. SS-F vs. SS-I
    - Solution quality is independent of State Splitting method

- So far the performance of State Splitting is very promising both in regards of solution speed and quality
  - More tests are needed to before any general conclusion can be made concerning improvements solving efficiency
  - The improvements in solution quality are certain
The future

- Continue testing the State Splitting approach on:
  - Large scale discrete time STN problems
  - Small and large scale continuous time STN problems
- Investigate if State Splitting can be used in some of the following areas:
  - Zero Wait production planning problems
  - Detailed storage modeling
  - Blocking
Thank you!

Questions?