Molde University College
Specialized University in Logistics

CSLP with sequence dependent setups and no setup carry-over

Problem Description

- In this variation of the capacitated lot sizing problem, the setup costs are dependent on the sequence in which the products are produced. It also includes setup times that are linearly dependent on the setup cost.
- Compared to all other studies in the field (most notably B. Almada-Lobo et al. (2007) and Gupta and Magnusson (2005)) the problem without the extremely complex setup carry-over has not yet been researched to the best of my knowledge.
- The problem without setup-carry-over between the periods is however of practical concern in many cases where perishable products are produced in a non-continuous environment. In these situations it is essential to clean the machine at the end of each period and thus the last produced product of the previous period is no longer of consequence at the start of the next period.

- I have created an advanced random instance generator that tries to mimic the types of problems found in real applications and also offers greater possibilities to explore the impact the data has on the difficulty of the problem.
- This case is inspired by my past working experience in a canned food factory in the Netherlands. I have however tried to generalize it as much as possible, in order not to over fit the solution method to the specific situation within that factory.

1. Formulation

Formulation with minimal number of variables and constraints.
Containing a total of $(n^2+3n+1)m$ variables, of which $(n^2+n)m$ are binary.
Excluding the objective and variable bounds, there are $(n^2+4n+3)m$ constraints.

1) minimize $\sum_{t=1}^{m} O_t \cdot o + \sum_{t=1}^{m-1}\sum_{i=1}^{n}\sum_{u=1}^{t} (Q_{iu} - d_{iu}) h_i + \sum_{t=1}^{m}\sum_{i=1}^{n}\sum_{j=0:j\neq i}^{n} S_{ijt} \cdot s_{ij}$

The objective function minimizes the inventory holding cost and setup cost

2) $\sum_{u=1}^{t} Q_{iu} \geq \sum_{u=1}^{t} d_{iu}, \quad \forall i \in 1..n, \forall t \in 1..m$

The difference between the total production and demand should be possitive in every period

3) $\sum_{i=1}^{n} Q_{it} + \sum_{i=1}^{n}\sum_{j=0:i\neq j}^{n} S_{ijt} \cdot s_{ij} \cdot r \leq a_t - O_t, \quad \forall t \in 1..m$

The total production time and setup time should be less than the capacity, unless overtime is used

4) $Q_{it} \leq \sum_{j=0:j\neq i}^{n} S_{ijt} \cdot \min\left\{\sum_{u=t}^{m} d_{iu}, a_t - s_{i0} \cdot r\right\}, \quad \forall i \in 1..n, \forall t \in 1..m$

If any production takes place, this should be part of the setup cycle for that period

5) $\sum_{j=0:j\neq i}^{n} S_{ijt} \leq 1, \quad \forall i \in 0..n, \forall t \in 1..m$

Each product can only be produced once in each period, so can only have one successor in the setup cycle

6) $\sum_{j=0:j\neq i}^{n} S_{ijt} = \sum_{i=0:i\neq j}^{n} S_{ijt}, \quad \forall i \in 0..n, \forall t \in 1..m$

As part of a cycle, there should be 1 successor and 1 predecessor for each node. For nodes that are not produced they can not be part of the cycle

7) $U_{jt} \geq U_{it} + 1 + (S_{ijt} - 1) \cdot n, \quad \forall i \in 1..n, \forall j \in 1..n : j \neq i, \forall t \in 1..m$

Classic Miller-Tucker-Zemlin subcycle elimination constraint. Assigns a unique sequence number to each setup in the cycle

8) $1 \leq U_{it} \leq n, \quad 0 \leq Q_{it}, \quad 0 \leq O_t, \quad S_{ijt} \in \{0,1\}$

Bounds on the variables

Remove the Miller-Tucker-Zemlin subcycle elimination constraints (7) and replace by the formulation proposed by Gavish and Graves.

8) $U_{ijt} \leq S_{ijt} \cdot n, \quad \forall i = 1..n, \forall j = 0..n : j \neq i, \forall t \in 1..m$

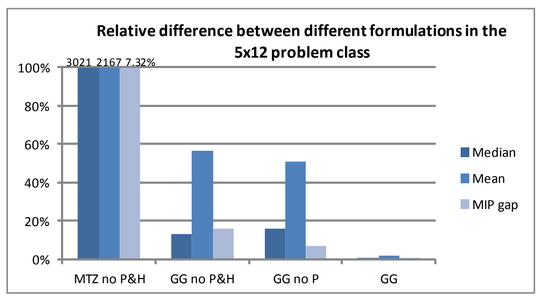
Limit the maximum sequence to *n* if the setup link is part of the production cycle, 0 else

9) $\sum_{j=0:j\neq i}^{n} U_{ijt} - \sum_{j=1:j\neq i}^{n} U_{jit} \geq \sum_{j=0}^{n} S_{ijt}, \quad \forall i = 1..n, \forall t = 1..m$

The difference in sequence for the link in and out should be 1 if this product is produced

The inventory in each period is represented as the difference between total production and demand up to that period. This can be represented by a new inventory variable H in the objective and change the positive inventory constraint (2) to a standard inventory balance constraint.

The summation of setup links out of a product to indicate if a product is produced in a period, as in constraints (4) and (9) can be replaced by one continuous production variable *P* with an upper bound of 1.
This variable is also placed with an equality sign as the RHS of constraints (4) and (5) to give the regular in and out degree constraints.



Include specific subcycle elimination constraints that remove subcycles that are know to be cheap. This improves the relaxation and cuts part of the search tree.
A simple Branch & Cut algorithm is used to solve the TSP problem provided by the setup cost matrix. The subcycles that are added to solve this problem iteratively are saved and added to the CLSP problem.
The following explicitly subcycle elimination constraints are then added to the formulation for these found cycles:

10) $\sum_{i\in V_k}\sum_{j\notin V_k} S_{ijt} \geq W_{kt}, \quad \forall k \in 1..c, \forall t \in 1..m$

Force at least one link to be used out of every subcycle group.

11) $W_{kt} \geq P_{it}, \quad \forall k \in 1..c, \forall i \in 1..v_k, \forall t \in 1..m$

Find if any product in this subcycle is produced. Only for these groups a flow out should be forced

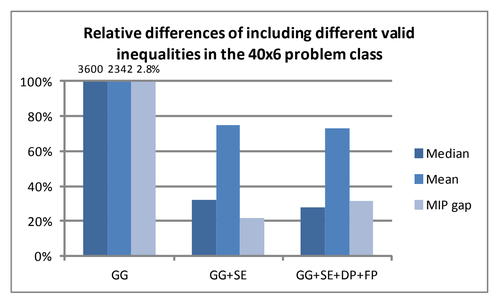
This includes new continuous variables *W* that indicates if any product in the subgroup is produced in that period.

The first produce constraint forces at least one full setup to be made before the first non-zero demand period. Improving the relaxation, that can be poor because of the 'big M' constraint (4).

12) $\sum_{t=1}^{\max_{y\in 1..m}\left\{u:\sum_{u=1}^{y-1} d_{iu}=0\right\}} P_{it} \geq 1, \quad \forall i \in 1..n$

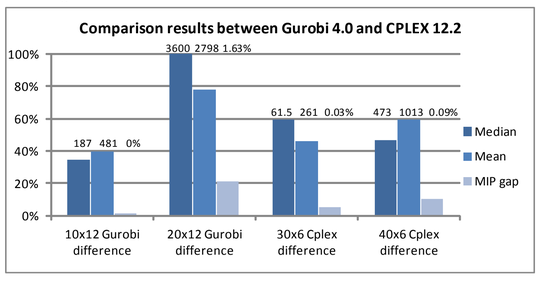
The dummy produce constraint forces the clean setup state to be 'produced' if any other product is produced. Cleaning the machine is expensive and was previously only forced in because of the subcycle elimination constraints.

13) $P_{0t} \geq P_{it}, \quad \forall i \in 1..n, \forall t \in 1..m$



2. Solvers and solver settings

- In frequent comparative tests on mixed integer programming problems by H. Mittelmann Gurobi has consequently beaten CPLEX.
- Comparative studies on the different problem classes of this problem gave large differences between the performance of the Gurobi 4.0 and CPLEX 12.2 solvers, but not always in the advantage of Gurobi.



- For problems with short time horizon and many products CPLEX clearly outperforms the Gurobi solver. But for longer time horizon problems, the CPLEX wasn't able to be competitive against Gurobi.

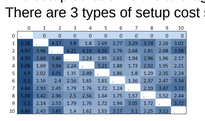
- Changing the solver setting for Gurobi and CPLEX from their default settings also slightly improved the performance.
- Very aggressive generation of cuts in both CPLEX and Gurobi works best for the most difficult problem classes, while a slightly more conservative cuts strategy works better for the problems that are know to be easier.
- CPLEX by default does a breadth-first search, changing this to 'best-estimate search' gives significantly better results for the easier problems, but convergence is much lower for hard problems.
- Changing the Gurobi search strategy to 'favor proving optimality' works rather surprisingly well for hard problems where optimality can not be proved for many instances.

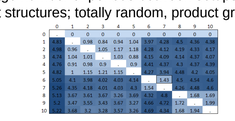
Commercial solver (Black box)

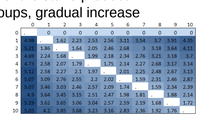
3. Instance data

- So far the effects of the data on the complexity of the problem, is only represented by the influence of the size of the instance.
- To analyze the effects of the data further, a regression model is created where the total run-time of the instance is estimated by using characteristics of the solution.

- In order to normalize the residuals, the dependent variable is the logarithm of the total elapsed time to solve the model.
- The model uses the following independent variables:
  - The cleaning fraction, which is the percentage of setup cost related to cleaning the machine at the end of the day
  - The utilization is the average percentage of available time used for setups or production
  - The setup duration is the average number of periods between setups of the same product
  - There are 3 types of setup cost structures; totally random, product groups, gradual increase







- The following table shows the results of this regression for 90 instances in the 10x12 instance class. The resulting model has an R square of 0.481 (adjusted 0.449)

	Coefficients	*St. Error*	*P*
Intercept	-1.217	0.705	0.088
Cleaning fraction	5.319	0.750	0.000
Utilization	1.204	0.593	0.046
Setup duration	-0.177	0.073	0.018
Group data	-0.295	0.138	0.035
Valley data	-0.200	0.143	0.164

- The sensitivity of the solution to the data is also important. Many parameter values are not exactly know in real situations.
- The sensitivity can be measured by its effect on the total objective value, but also on the effect on the matrix of binary produce variables.
- Currently I am investigating the effects of small changes in the inventory cost, setup cost, and setup time, but this work is not yet finished at this moment.

Compare strategies

- Comparing different 'treatments' (formulations or settings) applied to the same instance requires a 'paired' test statistic.
- The paired t-test can not be used because the differences between treatments are not normal distributed. Also it is hard to handle differences in MIP gap between treatments in the same score.
- The non-parametric Wilcoxon Signed Rank test is used instead. Instances with a difference in MIP gap get a fixed maximal time penalty plus the difference in MIP gap. This guarantees that these instances always get the highest ranks.
- Example:

Time	MIP gap	Time	MIP gap	Difference	Signed rank
3600	1.4%	3600	1.18%	3602.2	2
3600	0.83%	2112	0%	3608.3	3
555	0%	1006	0%	-451	-1

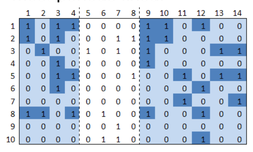
- The probability of a certain sum of negative signed ranks occurring by incident can be computed, and so the result can be found statistically significant or not.

Results

- The gradual steps of reformulating this very hard research problem have resulted in an extreme improvement of the performance in the commercial Gurobi solver
- Adding simple valid inequalities allows problems of up to 40 products, a practical size problem for the canned food industry considered, to be solved to optimality within one hour
- Using the CPLEX solver for short time horizon problems further improves the performance of these problems with up to 50% while Gurobi outperforms on the longer time horizon problems
- This research shows that detailed analysis of the formulation can result in practical size problems being solved to optimality with out of the box commercial solvers
- Detailed analysis of the data gives better insight in what makes a problem instance hard, and this insight can be used to use certain solver settings for certain hard or easy problems.

Future work

- The next part of my research will focus on relaxation techniques for this problem and finding Pareto efficient methods.
- An improvement heuristic will be developed later based on variable neighborhood search by solving the problem with the binary solution for most periods fixed.



- The final goal of this research is to solve the rolling horizon stochastic version of this problem where demand gets increasingly more unknow in later time periods.

Erik van Eikenhorst