Assignment 2

Linear and integer optimization with applications

Erling Hjermstad, 990118
<erlingh@student.chalmers.se>
Estimated time: 10 hours

Anton Lindén, 960201
<antoli@student.chalmers.se>
Estimated time: 10 hours

Matilda Widlöf, 980102
<widlof@student.chalmers.se>
Estimated time: 10 hours

May 13, 2022

Problem description

The purpose of this assignment is to minimize the total cost of maintaining a system consisting of several components which have varying lives and costs for new parts. This can be modeled in Julia which will give us a maintenance schedule over T time steps which tells us when to replace which component(s) in order to minimize the total cost. The maintenance schedule could look something like figure 1 where 3 different components are replaced at 4 maintenance occasions:

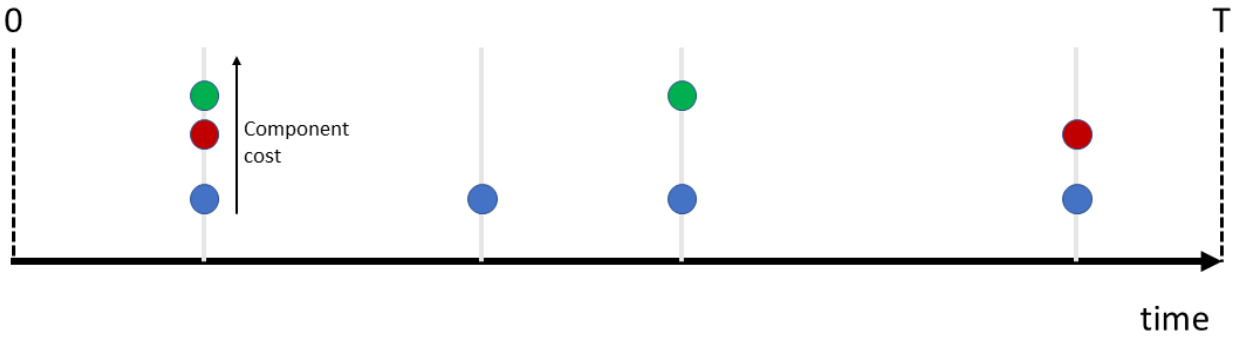

Figure 1: Maintenance schedule example for 3 components.

Our objective function is:

$$\text{minimize} \sum_{t=1}^{T} \sum_{i \in \mathcal{N}} c_{it} x_{it} + d_t z_t \tag{1a}$$

$$\text{subject to} \sum_{t=l+1}^{l+T_i} x_{it} \geq 1, \qquad l \in \{0, ..., T - T_i\}, \qquad i \in \mathcal{N} \tag{1b}$$

$$x_{it} \leq z_t, \qquad t \in \{1, ..., T\}, \qquad i \in \mathcal{N} \tag{1c}$$

$$x_{it}, z_t \in \{0, 1\}, \qquad t \in \{1, ..., T\}, \qquad i \in \mathcal{N} \tag{1d}$$

Sets

- $\mathcal{N}$ = components

Parameters

- T = total number of time steps
- T_i = life (time steps) of component i, $\quad i \in \mathcal{N}$
- c_{it} = cost of spare component i at time t, $\quad i \in \mathcal{N}, t \in \{1, ..., T\}$
- d_t = cost for performing a maintenance at time t, $\quad t \in \{1, ..., T\}$

Decision variables

- $x_{it} = 1$ if component i is replaced at time t else 0, $\quad i \in \mathcal{N}, t \in \{1, ..., T\}$
- $z_t = 1$ if maintenance is performed at time t else 0, $\quad t \in \{1, ..., T\}$

1

a.

Solving our objective function for three different cases of integrality requirements gives us the following results (Table 1):

Integrality requirements	Optimal objective value [euro]	Computational time [s]
x_{it}, z_t	762	6.64
z_t	762	5.48
None	724	0.08

Table 1: Solving a maintenance schedule. Computational times were averaged over 10 runs.

In the first case we have integrality requirements for both of our decision variables which means we have an integer linear programming problem (ILP). ILP problems have integral constraints on variables but the objective function is linear.

In the second case we only have integrality requirements for the maintenance decision variables z_t while the component replacement decision variables x_{it} are linear. This means we have a mixed integer linear programming problem (MILP) where some of the variables have integrality requirements and others don't. Here we reach the same optimal objective value as in the previous case because even if we allow partial replacement of components, for example $x_{i\hat{t}=1} = 0.6$ at time step $\hat{t}$, we still have to wait for our integer maintenance decisions. From the first case we know that the optimal z_t values give us full replacements of components anyways. The computational time is also slightly faster because linear variables are usually easier to deal with since their feasible regions are convex.

In the third case there are no integrality requirements on our variables which gives us a linear problem (LP). The objective function is now lower because we are allowing for partial replacement of components and "partial" maintenance performances. This gives us a better solution but it is not realistic to model a maintenance schedule in this way. Either you perform a maintenance or you don't. Same goes for the components which most likely can't be replaced in partial pieces. The computational time is also a lot faster since everything is linear in this case which gives us less constraints to worry about.

b.

The maintenance model is scaled down in this part of the problem and we also add an extra constraint eventually:

$$z_1 + z_4 + x_{12} + x_{22} + x_{13} + x_{23} \geq 2 \tag{2}$$

Given parameters and results (Table 2) can be found below.

Parameters

- c_{it} = [1, 1, 2, 1; 1 100 100 1]
- d_t = [10, 10, 1, 10]
- T_i = [3, 4]

Integrality requirements	Optimal objective value [euro]	Computational time [s]
x_{it}, z_t	14.0	0.0101
None	13.5	0.0084
None + extra constraint	14.0	0.0001

Table 2: Scaled down maintenance schedule. Computational times were averaged over 100 runs.

Looking at the parameters for this model we can see that component 1 should be replaced at time step 3 because the cost for performing maintenance is 10 times cheaper and the lifespan for this component is exactly 3 time steps.

The cost for a new component is also still relatively cheap. Component 2 can be replaced at either time step 1 or 4 where the cost for a new component is 100 times cheaper than usual. This gives us an optimal objective value of

$$(2+1)+(1+10)=14\ euro$$

which is correct according to Table 2 (first row).

Next we removed all integrality requirements on our variables which yielded a slightly lower optimal objective value because partial replacement of components and maintenance performances were now allowed, just like in part **a**.

Lastly we added an extra constraint which is apparently not affecting the optimal objective value for the ILP problem. The LP problem is getting affected on the other hand. It looks like the extra constraint is essentially forcing the correct optimal objective value for the ILP problem but for a much cheaper computational cost (approx. 100 times faster) which is good considering the ILP problem is a more realistic way to model a maintenance schedule.

c.

As long as the inequality

$$z_1+z_4+x_{12}+x_{22}+x_{13}+x_{23}<2 \tag{3}$$

never holds for any integer values of the decision variables we know (2) is a valid inequality. From the problem description we also know the model has additional constraints

$$\begin{cases} \sum_{t=l+1}^{l+T_i} x_{it} & \geq 1 \\ x_{it} & \leq z_t \end{cases}$$

Using (3) together with the above constraints we can check a few cases:

$$\begin{cases} z_1+z_4+x_{22}+x_{23} & \geq 1 & (\textit{component 2 replaced within 4 time steps}) \\ z_1+x_{12}+x_{13} & \geq 1 & (\textit{component 1 replaced within 3 time steps}) \\ \rightarrow 2z_1+z_4+x_{12}+x_{22}+x_{13}+x_{23} & \geq 2 & \\ z_1=0 \rightarrow z_4+x_{12}+x_{22}+x_{13}+x_{23} & \geq 2 & (3)\ \textit{does not hold} \\ z_1=1 \rightarrow z_4+x_{12}+x_{13} & \geq 1 & (3)\ \textit{does not hold} \end{cases}$$

So (2) is a valid inequality.

2

a.

Figure 2 illustrates the computing time as a function of T which are the total number of time steps in the planning period. We have varied the total time from $T = 50$ up to $T = 200$, and iteratively computed the computing time as can be seen in the figure. Each iteration had a time limit of 3600 seconds. The maintenance decisions z_t had integrality requirements, while the component decisions x_t had relaxed integrality requirements, which makes this a mixed integer linear programming problem.

We also limited the MIPGap to be 5e-5 which means that the MIP solver (using the branch and bound algorithm) terminates when the gap between the lower and upper bounds of the optimal value are less than 5%. This was done in order to save time when solving the model while not losing too much accuracy. A log-scale was used for the axis of the computing times in order to illustrate the result. Using this log-scale, the graph illustrates a somewhat linear trend. This indicates that the underlying data increases exponentially.

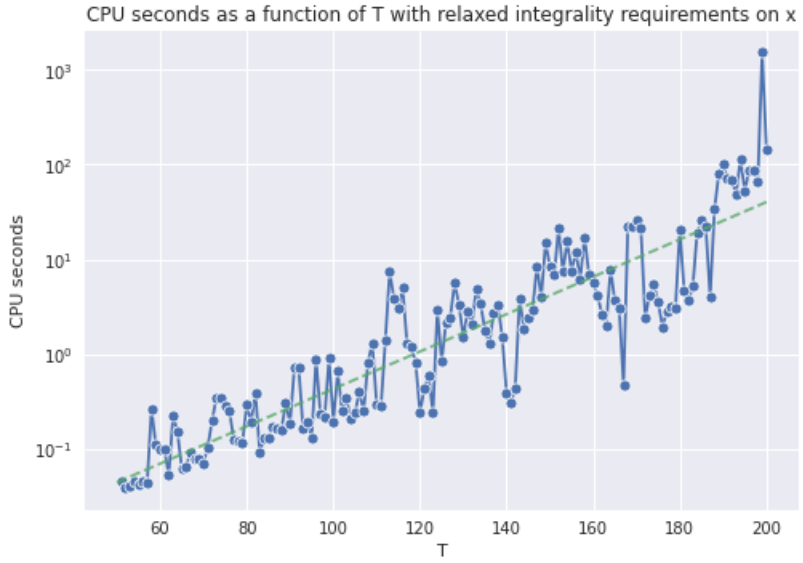

Figure 2: CPU seconds as a function of T with integrality requirements on z_t.

b.

Figure 3 illustrates the computing time as a function of T when varying the total time from $T = 50$ up to $T = 700$. This was done with relaxed integrality requirements on both the component replacement decisions x_t and the maintenance decisions z_t, which makes this a linear programming problem. Figure 3 shows a somewhat exponential increase of the CPU seconds when increasing T.

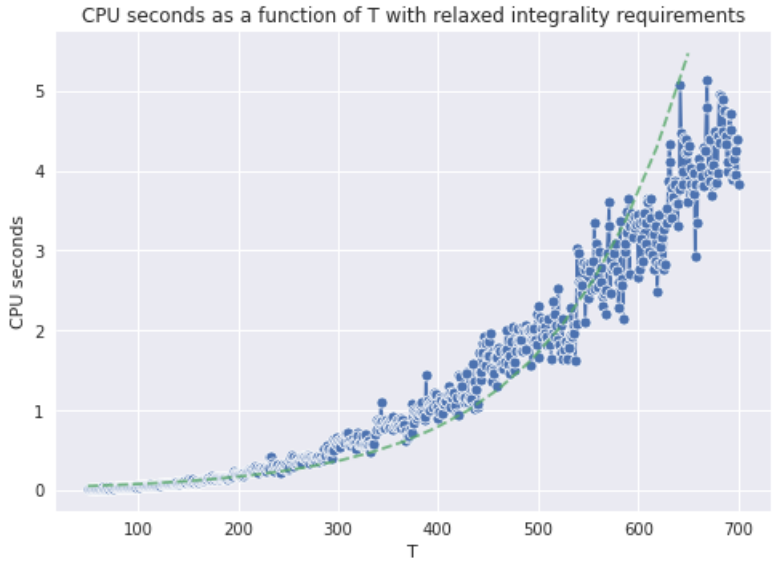

Figure 3: CPU seconds as a function of T with relaxed integrality requirements.

c.

As can be seen when comparing figure 2 and figure 3 there is a huge difference in the computing times. This difference is due to the fact that in figure 2 we solved a MILP problem and in figure 3 we solved a LP problem. In a LP problem less constraints need to be fullfilled than in a MILP problem, which is why the MILP are more difficult to solve and therefore takes much longer time. In a LP problem the feasible region are convex and you can solve it by looking only at the vertices of the feasible region. However, to solve a MILP the branch and bound algorithm is used, which creates binary search trees in order to solve the problem. These search trees can grow exponentially and are the reason for the huge difference in solving time between the different problems.

d.

By investigating the branch and bound algorithm we found that just a fraction of the solving time is used in the presolving step. The presolving step involves problem reductions in order to reduce the size of the problem that is done before starting the branch and bound procedure.

When the branch and bound process starts, an optimal feasible solution is found very early in the process. The branch and bound algorithm divide the feasible region into subproblems, and this solution is an optimal solution to such a subproblem.

The absolute majority of the solving time is used for verifying the optimality of the found solution. After finding an optimal solution, the branch and bound algorithm creates new subproblems in a search tree with new potential feasible and optimal solutions. Since these trees can grow exponentially these may take a lot of time searching through. However, according to [2] it is sometimes possible to stop the search in a part of the search tree if the subproblem for instance does not have a feasible solution. The search can also be stopped, for a minimization problem like ours, if the optimal solution of the subproblem is larger than or equal to the solution we have already found.

3

In *Preventive maintenance scheduling of multi-component systems with interval costs*, Gustavsson et al. (2014), the PMSPIC model was introduced which generalizes the opportunistic replacement problem (ORP) in equation set 1.

$$\text{minimize} \sum_{t\in\mathcal{T}} d_t z_t + \sum_{i\in\mathcal{N}} \sum_{(s,t)\in\mathcal{I}} c^i_{st} x^i_{st}, \tag{4a}$$

$$\text{subject to} \sum_{s=0}^{t-1} x^i_{st} \leq z_t, \qquad i \in \mathcal{N}, \qquad t \in \mathcal{T}, \tag{4b}$$

$$\sum_{s=0}^{t-1} x^i_{st} = \sum_{r=t+1}^{T+1} x^i_{tr}, \qquad i \in \mathcal{N}, \qquad t \in \mathcal{T}, \tag{4c}$$

$$\sum_{t=1}^{T+1} x^i_{0t} = 1, \qquad i \in \mathcal{N}, \tag{4d}$$

$$x^i_{st} \in \{0,1\}, \qquad i \in \mathcal{N}, \qquad (s,t) \in \mathcal{I}, \tag{4e}$$

$$z_t \in \{0,1\}, \qquad t \in \mathcal{T}. \tag{4f}$$

Where

$$\mathcal{I} = \{(s,t) | 0 \leq s < t \leq T+1; s,t \in \mathbb{Z}\},$$
$$\mathcal{T} = \{1, ..., T\},$$
$$\mathcal{N} = \{1, ..., n\} \text{ for n components.}$$

This model assigns each component i an interval cost c^i_{st} for running from time s to t without maintenance. For this to degenerate into the ORP, where a component's lifetime is set explicitly, the interval costs-matrix must be designed carefully. When set according to the rule $c^i_{st} := c_{it}$ when $t - s \leq T_i$, and $c^i_{st} := T(\max_{u\in\mathcal{T}}(d_u) + n \cdot \max_{j\in\mathcal{N},u\in\mathcal{T}}(c_{ju})) + 1$ when $t - s > T_i$ it can be shown that the two models are equivalent [1]. Here T_i is the lifespan of component i and c_{it} is the cost of replacing component i at time t.

a.

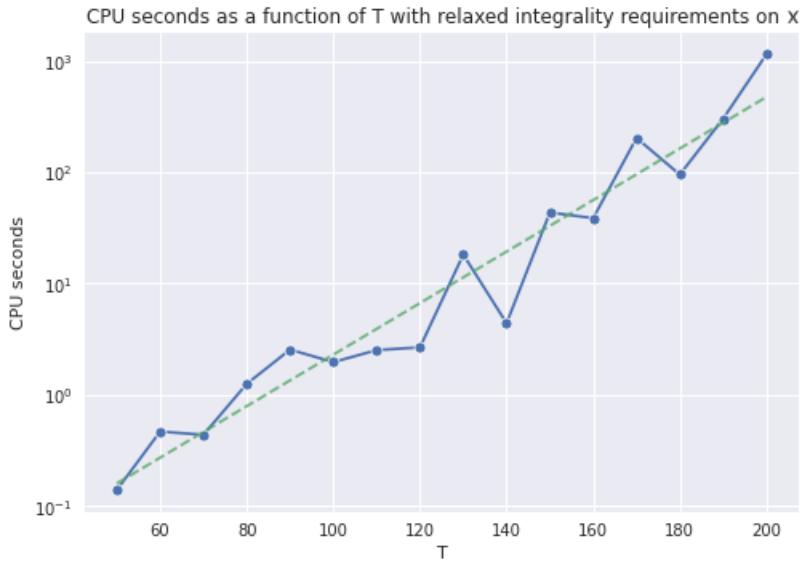

Figure 4: Computational time plotted against length of maintenance scheduling with relaxed integral requirements on x^i_{st}.

Repeating the tests done for the ORP in section 2a and b yields the results shown in figure 4 and 5. This time fewer variations of T were inspected as the new model was slower computationally compared to the previous implementation. The same bounds and hyperparameters were used. As in the ORP, the LP-problem is computationally

less demanding compared to the MILP-problem. This can be seen by the time it takes to solve the problem with $T = 200$ for both cases. The LP is faster by approximately 2 orders of magnitude.

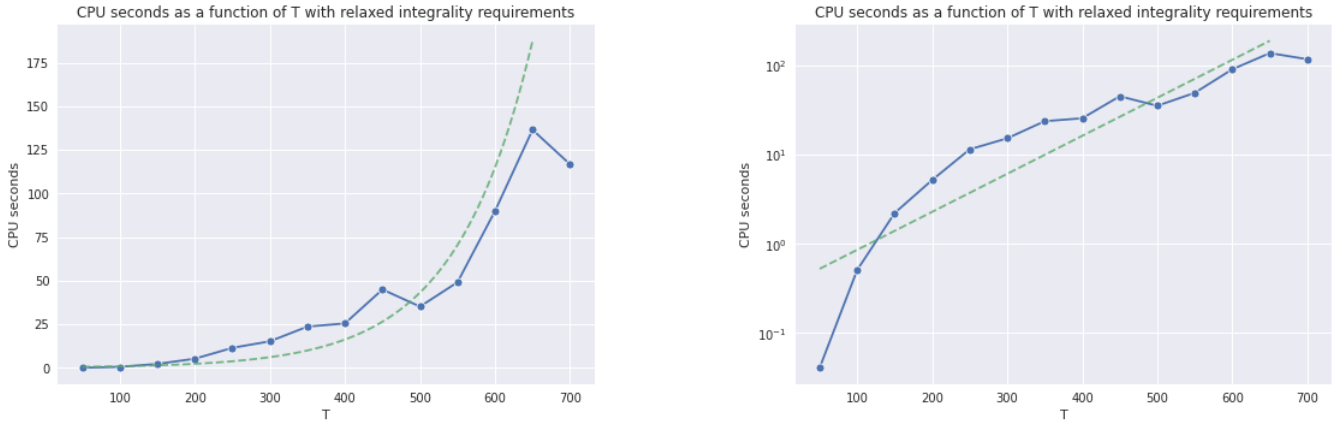

Figure 5: Computational time plotted against length of maintenance scheduling with relaxed integral requirements on z_t and x^i_{st}. Plotted with linear and logarithmic y-scale, respectively.

b.

Interestingly, when running the model on the given datasets it finds a different solution than the original model. However, the objective value of the given solutions are identical. After inspection of the proposed maintenance schedules it is apparent that both are equally good solutions. The proposed solutions are illustrated in figure 6. Manipulating the data to introduce a single optimal solution leads both models to find the same optimal solution.

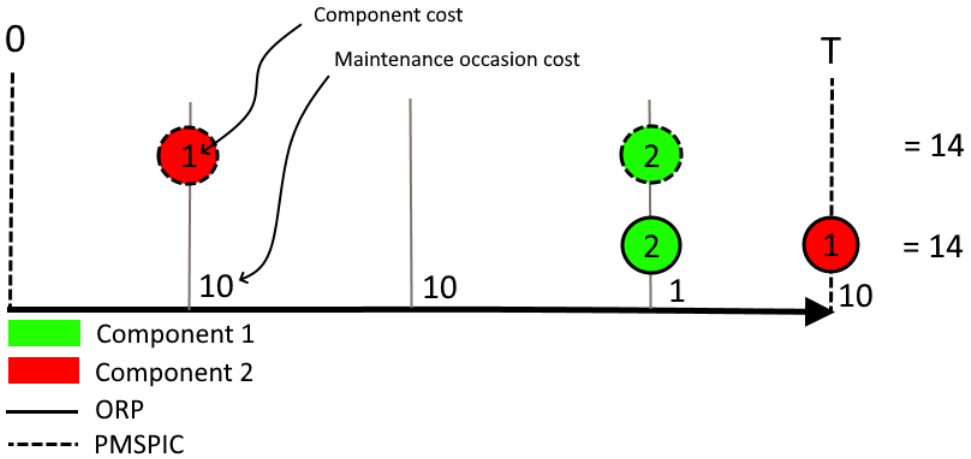

Figure 6: Comparison of the different solutions found by the PMSPIC and ORP for the small dataset with $T = 4$.

The implementation of the generalized model runs somewhat slower than the more specific model for our problem, as can be seen in figure 7. This can be due to the increase in dimensionality of the variables. The size of the maintenance schedule x is $\Theta(nT^2)$ in the PMSPIC formulation, while x is only $\Theta(nT)$ in the ORP. The same goes for the cost-matrix c. A comparison for the large dataset with $T = 50$ reveals that the ORP has 550 variables, while the PMSPIC has 26 060. A factor of approximately $T \in \Theta(T)$ more.

c.

When z is binary, x is limited to [0, 1] due to constraint 4b and positivity. Furthermore, for a fixed maintenance occurrence schedule $z \in \{0, 1\}^T$ the cost minimization can be solved independently for each component. These

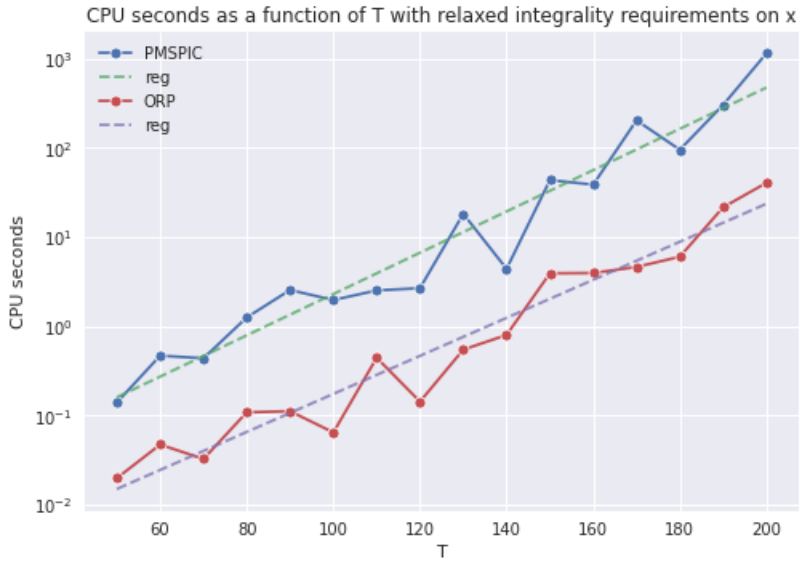

Figure 7: Comparison of computational times for the PMSPIC and ORP with increasing schedule length T.

minimization problems can be posed as network flow problems. Figure 8 illustrates this. As a network flow problem constraint 4d can be interpreted as the flow out of the start-node has to be 1. Constraint 4c says that the flow into a node must equal the flow out of the node, for all internal nodes. This is standard in network flows. As we wish to minimize the cost the best solution will be to choose the path minimizing the sum of costs on its way.

The z-values can be viewed as values enabling or disabling the nodes in our graph, i.e. if they can be included in our path. If the cheapest path is unique, then the flow in that path will be 1 due to constraint 4d and 4c. Therefore our x-values will take binary values; 1 if included in the path, 0 if not. If there are two or more equally cost-efficient paths, then there are an infinite amount of linear combinations of the paths minimizing the cost sum due to the convexity of our problem. All of these solutions will be on the form $x^* = \sum_i \alpha_i x_i,\ \sum_i \alpha_i = 1,\ \alpha_i \geq 0$, where x_i are extreme and optimal points in our solution space. In this case x^* does not necessarily need to take integer values in an optimal solution, as shown in figure 9. However, the extreme points are all binary in nature meaning there always are one or more binary optimal solutions. The fact that all extreme points are binary follows from the nature of the network flow. Only integral flows are the ones which cannot be expressed as a linear combination on the form of x^*, where x_i are other flows. This means that only integer paths can be extreme points in our problem by the definition of extreme points.

If there are multiple available cheapest paths, then there will exist one or more configurations in our occurrence schedule space $\{0,1\}^T$, with fewer enabled nodes which thus yields a lower objective function due to less maintenance occurrence cost $\sum_{t\in\mathcal{T}} d_t z_t$. In other words, the paths visiting the fewest nodes also exist in a z^* configuration where only the nodes they visit are enabled. Note that it might not be possible to reduce the number of enabled nodes due this action being dependent on other components. Also note that there still can exist multiple optimal solutions globally, since there can be multiple equally cheap, equally short paths with equally cheap maintenance occurrences. For these solutions x will again take binary values as there is only a single available path in the relevant z^*-schedule.

As discussed above, there always exists at least one integral optimal point. In fact, when using solver algorithms like the simplex method, only integral solutions will be found, as the algorithm searches extreme points along constraints decreasing the cost function. Since all extreme points are binary, such an optimal solution will be found. Therefore a binary condition on z is sufficient to enforce binary optimal solutions also in x. In other words the binary condition on x can be relaxed, which speeds up solving of the problem as shown in table 1.

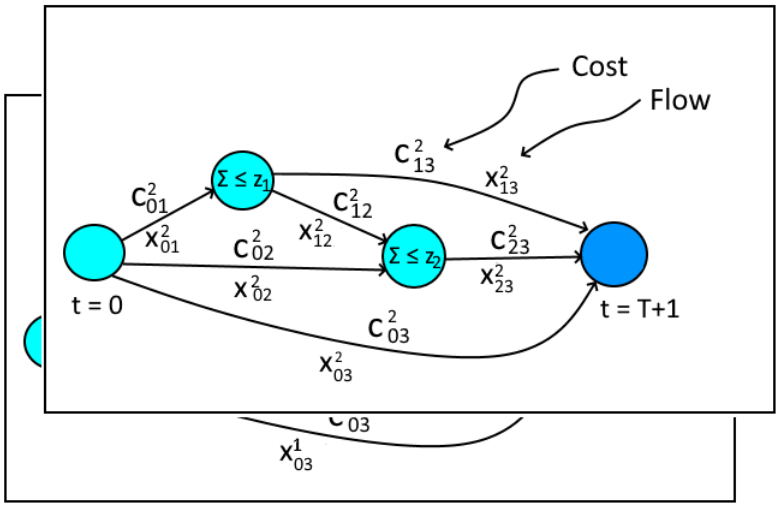

Figure 8: The PMSPIC with T = 2 and 2 components as two network flow problems.

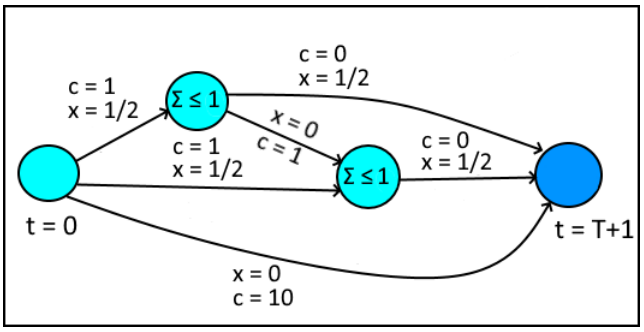

Figure 9: The PMSPIC with T = 2 and 2 components as a network flow problem where a non-integral optimal solution exist. Indexes are omitted for brevity. Note that there exist 2 binary optimal solutions as well, one for each of the two included paths in this solution.

4

a.

One way of implementing a requirement for the remaining life r is to increase the time steps $(T+r)$ and implement a new constraint which disallows maintenance to be performed past time step T. This ensures that our model tries to find a solution where all the components have to survive until time step $T+r$. The added constraint for maintenance performance in our Julia model looks like this:

$$z_t t \leq T \qquad t \in \{1, ..., T+r\} \tag{5}$$

We can verify our implementation by setting r to a value which is larger than the shortest component survival time. In the given data component number 10 has the shortest survival time at 11 time steps. If we set $r = 11$ our Julia model throws an expected error that no solution exists because component 10 would have broken down by time step $T + 11$.

b.

Figure 10 and 11 below show our results for all valid values of r. The optimal objective value is increasing as the requirement for remaining life increases. It is also increasing in a step-wise manner because sometimes we can achieve a higher remaining life requirement without the need for extra maintenance occasions.

Figure 10: Optimal objective value as a function of remaining life.

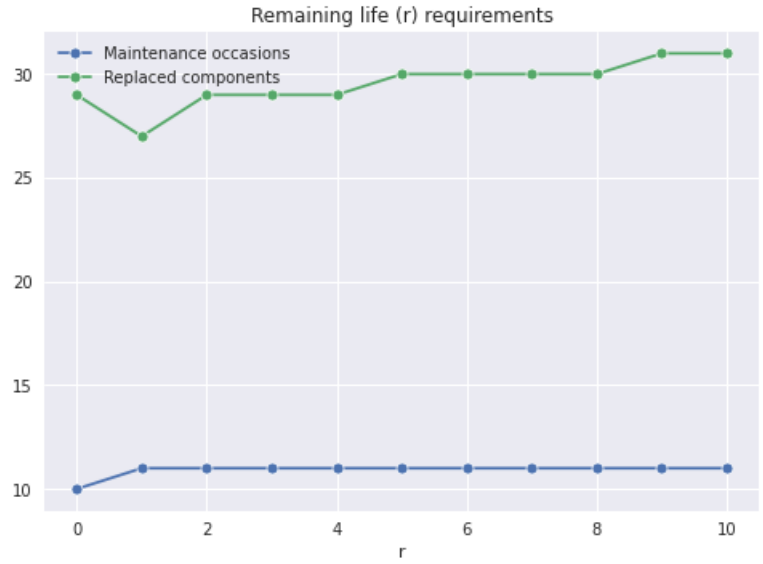

Figure 11: Maintenance occasions and replaced components as a function of remaining life.

c.

Relevant values of r for this specific problem are in the range:

$$0 \leq r < \min_{i \in \mathcal{N}} T_i = 11$$

References

[1] E. Gustavsson et al. "Preventive maintenance scheduling of multi-component systems with interval costs". In: *Computers & Industrial Engineering* 76 (2014), pp. 390–400. DOI: http://dx.doi.org/10.1016/j.cie.2014.02.009.

[2] J. Lundgren, M. Rönnqvist, and P. Värbrand. *Optimization*. Studentlitteratur AB, Lund, 2012.