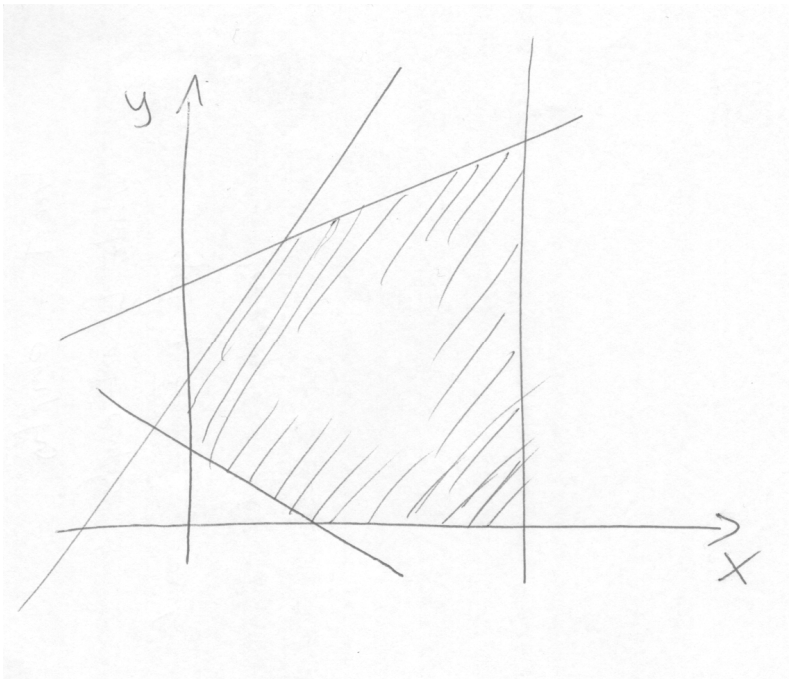
DECO

Discrete Event Control and Optimization

Exam SSY 220, Friday Jan 14, 08:30-12:30, M

Teacher: Martin Fabian, (772) 3716

Time when teacher present: 09:30, 11:30



Solutions and answers should be complete, written in English and be unambiguous and well motivated. In the case of ambiguously formulated exam tasks, the suggested solution with possible assumptions must be motivated. The examiner retains the right to accept or decline the rationality of assumptions and motivations.

In total the exam comprises 25 credits. For the grades 3, 4 and 5, is respectively required 10, 15 and 20 credits.

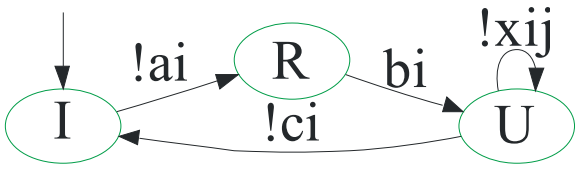
Solutions will be announced on the course web-page on the first week-day after the exam date. Exam results are announced through Chalmers' administrative routines. The corrected exams are open for review Friday Jan 21, 12:30 – 13:30 at the department.

Aids: None.

Task 1. Supervisor Synthesis

Three users share a common resource that they are not allowed to use simultaneously. Each user i ($i = \{1,2,3\}$) is very polite and asks for access and awaits permission before using the resource. When a user is done using the resource, it releases the resource.

Models of the users are given below. The *I* state represents the user being idle; the *R* state represents the user requesting the resource and the *U* state represents using the resource. The a_i (i = {1,2,3}), c_i and x_{ij} ($i, j = \{1,2,3\}, i \neq j$) events are uncontrollable, while the b_i events are controllable. Note that there are two x_{ij} events for each user; for instance, User1 has both x_{12} and x_{13}. Note also that we consider $x_{ij} = x_{ji}$, for example, x_{23} is the same event as x_{32}.



a) Give one or more specifications that each involve *only* the uncontrollable events x_{ij} ($i, j = \{1,2,3\}, i \neq j$, $x_{ij} = x_{ji}$) so that the simultaneous use of the resources is guaranteed to be avoided. (2p)

b) Using the specification(s) from a), synthesize a modular supervisor. (2p)

Specs involving only *the* x_{ij} *events and avoiding the UU states, can be formulated simply as specs with a single marked state, no transitions, and the* x_{ij} *event as the single event in the respective alphabet. This forbids the UU states, since these are the only ones where the* x_{ij} *events will be possible in the plant. Synthesis is then a simple matter of synchronizing* $User_i||User_j||Sp_{ij}$ *and then make this controllable. Note that non-blocking is not an issue, since all plant and spec states are marked.*

Task 2. Discrete Optimization

Both A* and Dijkstra's algorithm are "guided" branch-and-bound algorithms.

a) Describe how A* is guided and what properties that guarantee that the optimal solution is found. (3p)

b) Describe how Dijkstra's algorithm is guided. (2p)

Make sure to note the principle differences between the two algorithms.

A is guided by both the cost of getting to a node* n *(*g(n)*) and an estimate of getting from* n *to the goal (*h(n)*). To guarantee that the optimal path to the goal is found,*

- h(n) *must not over-estimate the true value* h*(n)*, that is* h(n) <= h*(n)*, and*
- h(n) *must me monotonic, that is, for two nodes* n_1 *and* n_2*, such that* n_2 *is reachable from* n_1 with weight $w(n_1,n_2)$, h(n_1) <= $w(n_1,n_2)$ + h(n_2).

For a system with a number of jobs routed through a number of resources, one such estimate comes from regarding each job as running by itself through its needed resources. Of course, the time to finish a single job must be shorter than finishing several simultaneous jobs competing for the resources.

Dijkstra's algorithm is principally the A algorithm with h(n)=0, that is, it does not use any estimate of the "future" as guide, it merely looks at the "past". Naturally, h(n)=0 is an estimate that never over-estimates the true value, and it guarantees monotonicity.*

For both algorithms, the node weights must be non-negative.

Task 3. Linear Programming

We have formulated LP-problems in a "standard form", looking like

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Az = b \\ & x \geq 0, x \in \mathbb{R} \end{aligned}$$

Regard the following LP-formulation

$$\begin{aligned} \max \quad & -x_1 - x_2 - x_3 \\ \text{s.t.} \quad & x_1 + x_2 \geq 4 \\ & -2x_2 - x_3 \leq 4 \\ & -x_1 - x_3 \leq -4 \\ & x_i \geq 0 \text{ and } x_i \in \mathbb{R} \end{aligned}$$

a) Formulate this as an LP-problem in the standard form given above. (2p)

b) Solve it. (3p)

c) Is the feasible set bounded or unbounded? (1p)

We know that max -f = min f, so we rewrite the objective as $-(-x_1 - x_2 - x_3) = x_1 + x_2 + x_3$.

We need also to include slack and surplus variables to get equalities. For convenience we also rewrite b *to have only non-negatives. So we get:*

$$\begin{aligned} \min \quad & x_1 + x_2 + x_3 \\ \text{s.t.} \quad & x_1 + x_2 - s_1 = 4 \\ & -2x_2 - x_3 + s_2 = 4 \\ & x_1 + x_3 - s_3 = 4 \\ & x_i, s_i \geq 0 \text{ and } x_i, s_i \in \mathbb{R} \end{aligned}$$

Writing this in matrix form is trivial.

Note that the constraint $-x_1 - x_3 \leq -4$ *is equivalent to* $x_1 + x_3 \geq 4$. *This constraint together with the constraint* $x_1 + x_2 \geq 4$ *means that the feasible set does not contain (0,0,0).*

The optimal value of the objective function is 4, which occurs when $x_1 = 4$, $x_2 = 0$, $x_3 = 0$.

Note that $-2x_2-x_3 = -(2x_2-x_3)$ *is always negative (when the* x_i *are non-negative) and so can be removed from the problem formulation. So we can actually solve the much simpler problem*

$$\begin{aligned} \min \quad & x_1 + x_2 + x_3 \\ \text{s.t.} \quad & x_1 + x_2 \geq 4 \\ & x_1 + x_3 \geq 4 \\ & x_i \geq 0 \text{ and } x_i \in \mathbb{R} \end{aligned}$$

which quite obviously has the above solution. Clearly, the feasible set is unbounded.

Task 4. Integer Programming

We have two integer programming problems

$$\begin{aligned} \min \quad & c_i^T z \\ s.t. \quad & A_i z = b_i \\ & z \geq 0, z \in \mathbb{Z} \end{aligned}$$

for $i = \{1,2\}$ with b_i and c_i integral. The A_i matrices look like

$$A_1 = \begin{bmatrix} 1 & 0 \\ -1 & -1 \end{bmatrix}, \text{ and } A_2 = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix}.$$

Can we say anything about the solution of the LP relaxation in relation to the IP solution for the two cases? Motivate your answer. (4p)

A1 is totally uni-modular and b_1 *is integral, so we know that the solution of the LP-relaxation will also be the solution to the IP-problem.*

A2 is not totally uni-modular, so we do not know what the LP-relaxation will tell us, other than that it is a lower bound on the optimum for the IP-problem.

Task 5. Modeling Logical Conditions

So called *zero-one variables* are integer variables constrained to take only the values 0 or 1. Such variables are typically used to model logical conditions in integer and/or linear programming problems. For each of the logical conditions below, write linear constraints involving 0-1 variables that model the condition.

a) If we produce products A or B, then we must also produce C *or* D (or both). (2p)

b) If we produce products A or B, then we must also produce both C *and* D. (2p)

c) Either the constraint $x < y$ or $y < x$, but not both, should hold. (Note that in this case you need to introduce also variables or constants other than the 0-1 variables.) (2p)

Let the variables δ_A, δ_B *etc be 0-1 variables denoting whether we produce any of the respective products A, B etc. Then, the first criterion (*a*) can be expressed as* $(\delta_A + \delta_B) - 2(\delta_C + \delta_D) \leq 0$. *Since these are 0-1 variables, a sum* $\delta_A + \delta_B$ *can only take the values 0, 1 or 2. Thus, the criterion says the following: When* $\delta_A + \delta_B = 0$ *no constraints are placed on* δ_C *or* δ_D*; when* $\delta_A + \delta_B = 1$ *or* 2*, then* δ_C *or* δ_D *or both must be* 1.

*The second criterion (*b*) can be expressed as*

$$\begin{aligned} (\delta_A + \delta_B) - 2\delta &\leq 0 \\ \delta - \delta_C &\leq 0 \\ \delta - \delta_D &\leq 0 \end{aligned}$$

This means that when producing A *or* B *(or both), then* $\delta = 1$*, and when* $\delta = 1$ *then both* $\delta_C = 1$ *and* $\delta_D = 1$*, thus we should be producing both* C *and* D.

The third criterion is expressed as

$$\begin{aligned} x &< y + M\delta \\ y &< x + M(1 - \delta) \end{aligned}$$

Here, M is a "sufficiently" large constant such that $x + M \approx M$ *(and similarly for y). When* $\delta = 1$ *then the criterion* $x < y + M\delta$ *places no constraint on x, but requires that* $y < x$*. On the other hand, when* $\delta = 0$*, no constraint is placed on y but* $x < y$ *is required.*
