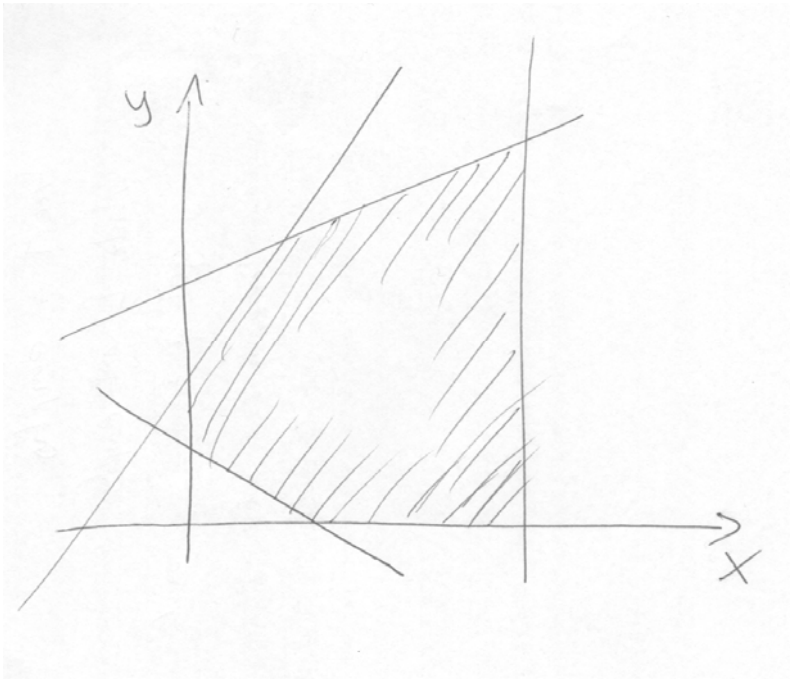
DECO

Discrete Event Control and Optimization

Exam SSY 220, Friday Oct 22, 14:00-18:00, M

Teacher: Martin Fabian, (772) 3716

Time when teacher present: 15:00, 16:30



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.

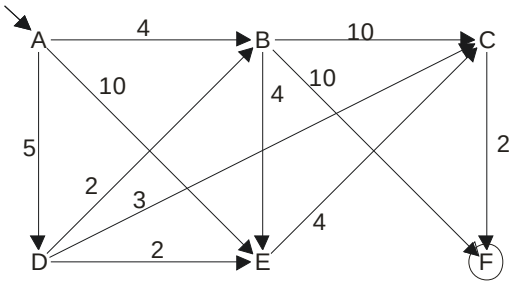
In total the exam comprises 25 credits. For the grades 3, 4 and 5, is respectively required 10, 15 and 20 credits.

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 Nov 05, 12:30 – 13:30 at the department.

Aids: None.

Task 1. Discrete Optimization

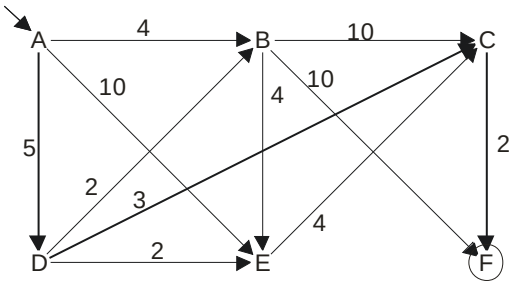
Regard the following weighted directed graph. The initial node is A, and the final node is F. The numbers represent the cost of taking the respective transition from one node to the other.



a) Using Dijkstra's algorithm, find the least cost path through the graph. Show on each iteration which node is taken out from and put in to the queue, and also what the queue looks like. (3p)

b) Using the A* algorithm, find the least cost path through the graph. Show on each iteration which node is taken out from and put in to the queue, and also what the queue looks like. Note that you need to define a useful heuristic estimate yourself; do not forget to write it down in your solution. (3p)

The optimal path is marked with thick arrows below. This should be the same for both tasks.



The workings of Dijkstra's algorithm is shown to the left, below, and A is to the right.*

Dijkstra's Algorithm		*A**	
A[0,-]	B[4,A] D[5,A] E[10,A]	A[0,10,-]	D[5,5,A] B[4,12,A] E[10,4,A]
B[4,A]	D[5,A] C[14,B] E[8,B] F[14,B]	D[5,5,A]	C[8,2,D] B[4,12,A] E[7,4,D]
D[5,A]	E[7,D] F[14,B] C[8,D]	C[8,2,D]	F[10,0,C] B[4,12,A] E[7,4,D]
E[7,D]	C[8,D] F[14,B]	F[10,0,C]	E[7,4,D] B[4,12,A]
C[8,D]	F[10,C]		
F[10,C]			

The first element within the brackets is the current cost of the node, the last element is the current parent, and the middle element is the estimate. Here I use as estimate the weighted sum of the shortest, in terms of transitions, path to the goal. This is maybe not a reasonable estimate, since calculating it requires to know everything*, which is what we want to avoid, but using the shortest path to the goal as estimate will give A* the same behavior as Dijkstra's algorithm. The used estimate fulfills the requirements we place on estimates, monotonic and not over-estimating, to give us the optimal solution, and it also shows that the tighter the estimate, the better A* performs.*

Task 2. Linear and Integer Programming

Regard the following four optimization problems:

$$z_1^* = \max\ c^T x \quad \text{s.t.}\ Ax \le b,\ 0 \le x \le 1 \qquad z_2^* = \max\ c^T x \quad \text{s.t.}\ Ax \le b,\ 0 \le x$$

$$z_3^* = \max\ c^T x \quad \text{s.t.}\ Ax \le b,\ 0 \le x, \text{integer} \qquad z_4^* = \max\ c^T x \quad \text{s.t.}\ Ax \le b,\ x \in \{0,1\}$$

The z_i^* (for $i \in \{1,2,3,4\}$) are the different optimal objective function values.

a) Determine the relative relationship (in magnitude) between the z_i^* (for $i \in \{1,2,3,4\}$). Note that it may not be possible to relate all of them to each other. (2p)

b) Assume that *A* is totally uni-modular and that *b* is integer, then do the same as in a). (2p)

The least constrained problem is the 2nd one, so that z_2^* has potentially the largest value. The most constrained problem is the 4th one, so this has potentially the smallest value. As for the 1st and 3rd problems, we cannot say much about their relative magnitudes, except that they fall in-between the solutions to the 2nd and 4th problems. Thus we have

$$z_4^* \le \left\{ \begin{matrix} z_1^* \\ z_3^* \end{matrix} \right\} \le z_2^*$$

When A *is totally uni-modular and* b *is integer, the solutions to the 1st and 2nd problems are integer and so must be the same as the solutions to the 4th and 3rd problems, respectively. Thus we have that $z_4^* = z_1^* \le z_2^* = z_3^*$.*

Task 3. Visibility Graph

Two trains, T_1 and T_2, share mutual tracks in two sections.

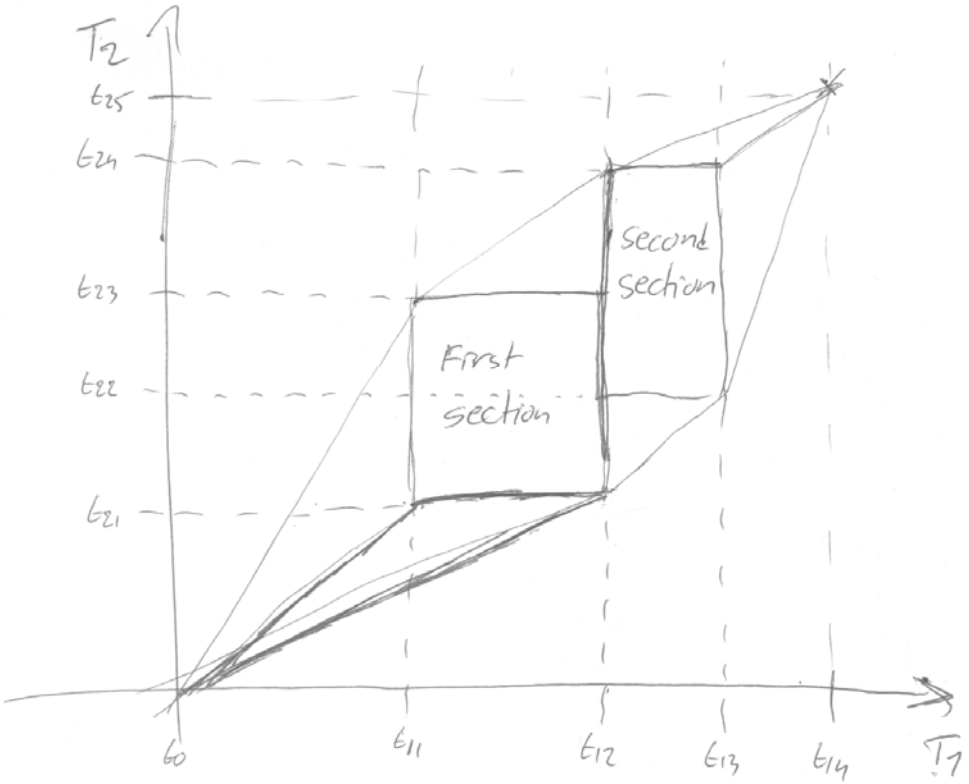
Starting at time t_0 train 1 enters the first shared section at time t_{11} then exits the first shared section to immediately enter the second shared section at time t_{12}. T_1 then exits the second shared section at time t_{13} and reaches its destination at time t_{14}.

Train 2, also starting at time t_0, enters the first shared section at time t_{21} and then enters the second shared section at t_{22} without yet exiting the first shared section (it's a long train). The first shared section is exited at time t_{23}, while the second shared section is exited at time t_{24}. T_2 reaches its destination at time t_{25}.

The times for the respective trains are consecutively larger as their indices increase, but we do not know much about the relative timings between the trains; we cannot say whether t_{11} is larger or smaller than t_{21}, for instance.

a) Draw the visibility graph for the system of the two trains. (2p)

b) Examine the first, lower part of the train system, including the times t_0, t_{11}, t_{12}, and t_{21}. View the times as coordinates in the visibility graph. Quantify when the path $<t_0,t_0>$, $<t_{11},t_{21}>$, $<t_{12},t_{21}>$ is better or worse (in terms of time) than the path $<t_0,t_0>$,$<t_{12},t_{21}>$. (3p)



We are to compare the two lower paths, up to the lower right hand corner of the first section obstacle

The upper path to there has the cost $\max(t_{11}, t_{21}) + (t_{12} - t_{11})$
The lower path to there has the cost $\max(t_{12}, t_{21})$

There are three possibilities

① $t_{21} \leq t_{11} < t_{12}$

Upper path cost is then $t_{11} + t_{12} - t_{11} = t_{12}$
Lower path cost is then t_{12}
} equally good

② $t_{11} < t_{21} \leq t_{12}$

Upper path cost is then $t_{21} + t_{12} - t_{11} = t_{12} + (t_{21} - t_{11})$
Lower path cost is then t_{22}
Thus, lower path is better than upper path when $t_{21} - t_{11} > 0$

③ $t_{11} < t_{12} < t_{21}$

Upper path cost is then $t_{21} + t_{12} - t_{11}$
Lower path cost is then t_{21}
Thus, lower path is better than upper path when $t_{12} - t_{11} > 0$

and these things hold for these two cases

Task 4. Modular Supervisory Control Theory

We have two plants P_1 and P_2, and two supervisors S_1 and S_2. Let us also regard their respective synchronous compositions, $P := P_1 \| P_2$ and $S := S_1 \| S_2$. For simplicity we can assume that all automata have the same alphabet Σ. The uncontrollable events are given by $\Sigma_u \subseteq \Sigma$.

a) Show that when S is controllable with respect to P_1 and Σ_u, then S is also controllable with respect to P and Σ_u. (2p)

b) Show that when S_1 and S_2 are both controllable with respect to P and Σ_u, then also S is controllable with respect to P and Σ_u. (2p)

Controllability is formally expressed as $L(S)\Sigma_u \cap L(P) \subseteq L(S)$ *when all alphabets are equal. Also, when all alphabets are equal, it holds that* $L(S_1) \cap L(S_2) = L(S_1 \| S_2)$.

For a) *we have:*

$$L(S)\Sigma_u \cap L(P_1) \subseteq L(S) \Rightarrow$$
$$L(S)\Sigma_u \cap L(P_1) \cap L(P_2) \subseteq L(S) \Leftrightarrow$$
$$L(S)\Sigma_u \cap L(P_1 \| P_2) \subseteq L(S)$$

For b) *we have:*

$$\left.\begin{array}{l} L(S_1)\Sigma_u \cap L(P) \subseteq L(S_1) \\ L(S_2)\Sigma_u \cap L(P) \subseteq L(S_2) \end{array}\right\} \Rightarrow$$
$$\left[L(S_1)\Sigma_u\right] \cap \left[L(S_2)\Sigma_u\right] \cap L(P) \subseteq L(S_1) \cap L(S_2) \Leftrightarrow$$
$$\left[L(S_1) \cap L(S_2)\right]\Sigma_u \cap L(P) \subseteq L(S_1 \| S_2) \Leftrightarrow$$
$$L(S_1 \| S_2)\Sigma_u \cap L(P) \subseteq L(S_1 \| S_2)$$

Note that the equivalences hold (as given) only for equal alphabets; for non-equal alphabets the notation gets a little more intricate, but the results still hold. Thus, there is really no loss of generality in assuming equal alphabets. That concatenation with an alphabet distributes over intersection (that is, $\left[L(S_1)\Sigma_u\right] \cap \left[L(S_2)\Sigma_u\right] = \left[L(S_1) \cap L(S_2)\right]\Sigma_u$*) would really need to be proven, but we can take this for a fact. Note though that this does not hold if* Σ_u *were an arbitrary language instead of an alphabet (in fact, it holds for languages of equal length strings, which is one way to view an alphabet; a language of 1-length strings).*

Task 5. Optimization Problem Modeling

We have 5 crates C_i that are to be loaded with 3 products P_j that cannot be mixed, so each crate can only contain one type of product. For each product, there is a quantity demand d_j which ideally should be met but need not be, and if it is not then there is a penalty p_j associated with the deviation from the demand. There is also a maximum allowed deviation from the demand, n_j. Each crate has a capacity c_i which cannot be exceeded.

The indices above range as $i := 1 \ldots 5$ (number of crates) and $j := 1 \ldots 3$ (number of products). The constants d_j, p_j, n_j, and c_i are all real numbers, equal to or larger than 0.

a) Formulate a problem of minimizing the total penalty cost while loading the crates, each with only one type of product (no mixing). (5p)

b) How would you characterize the problem in terms of different types of optimization problems? (1p)

Decision variables

y_{ij}, 0-1 variables, 1 if C_i contains P_j

x_{ij}, real variable, the amount of P_j in C_i

s_j, real variable, the deviation from demand d_j

Objective function to minimize

$$z = \sum_j s_j p_j$$

Constraints:

Only one type of product in each crate

$$y_{i1} + y_{i2} + y_{i3} = 1$$

Maximum penalty

$$s_j \leq n_j$$

Maximum crate capacity

$$x_{ij} \leq c_i y_{ij}$$

Meet the demands

$$\left(\sum_i x_{ij}\right) + s_j = d_j$$

General variable constraints

$$0 \leq x_{ij}$$

$$0 \leq s_j$$

$$y_{ij} \in \{0, 1\}$$

This is a MILP-problem