MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Project MAC

Computation Structures Group Memo No. 48

Deadlock Avoidance in Multi-resource Systems

Prakash G. Hebalkar

April 1970

Abstract

This memo examines systems in which many kinds of resources are shared instead of only one kind. The algorithmic test for safeness of an allocation state is extended to this case. However it is found that an important property of the algorithm is lost in this process. It is shown, however, that this is inevitable regardless of what the algorithm used in this multi-resource case is. The property is that of linearity, a term defined in the text.

Memos 45 and 46 in this series described the demand graph model for representation of asynchronous systems of processes, which share resources from a pool, for a study of deadlock problems and presented some results for the case of a system with one type of resource. A question that arises immediately is whether the results extend to the case where more than one type of resource is shared from a pool. This situation can be represented by replacing the scalar demands associated with the arcs by vectors, the components of which represent demands for each of the n types of resource being shared. It will be noted that the results of memo 46 are structural and consequently extend directly to the case of multiple resource types as long as the demand graph consists of single chains, one per user.

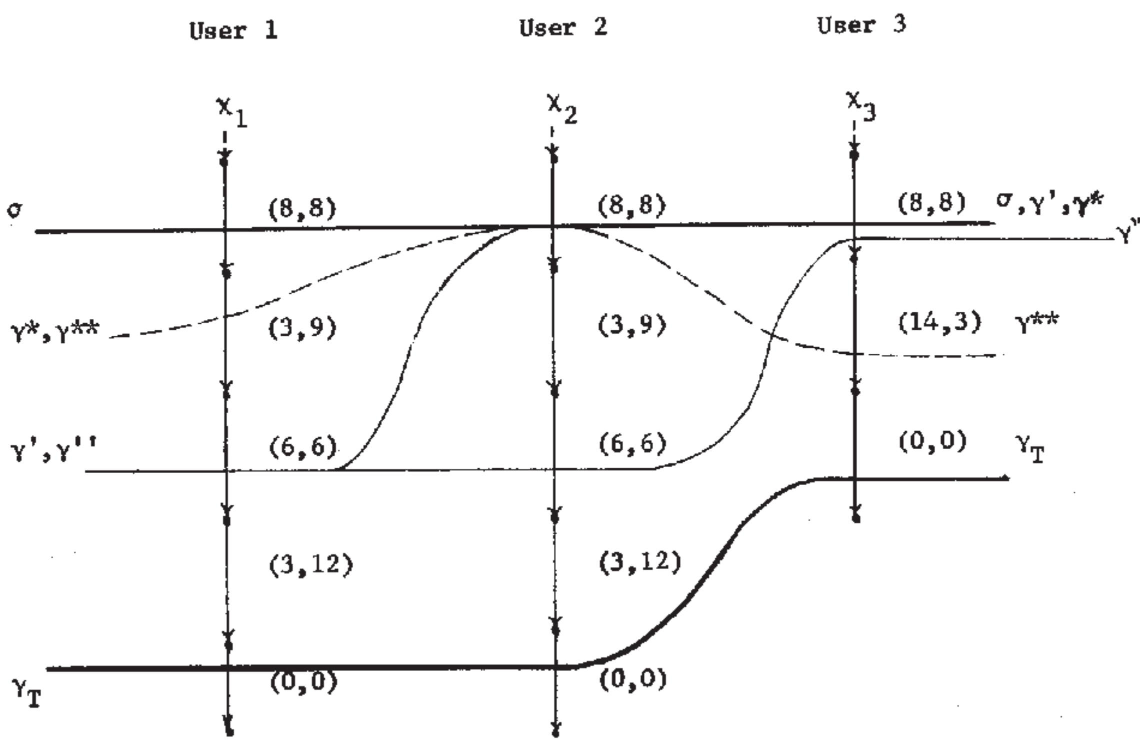
The nice properties of the safeness algorithm do not, however, hold in the case of multiple resource types (called the multi-resource case in what follows). Consider the example shown in figure 1a. Suppose one is investigating the safeness of the slice σ and one tries to apply the safeness algorithm given in memo 45 with vectors replacing all scalar demands therein. The modified algorithm reads:

Safeness Algorithm:

Let σ be the slice whose safeness is to be examined.

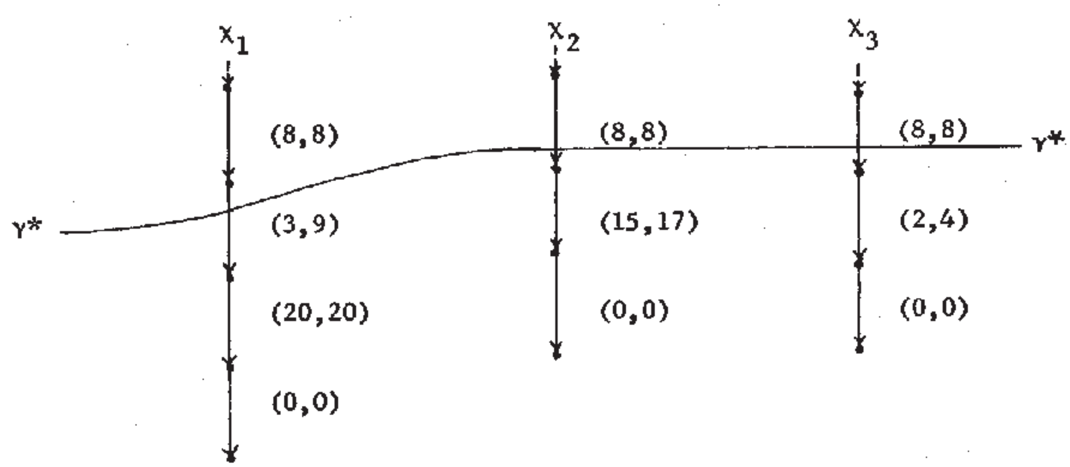
<u>Step 1</u>: Pick a chain χ_i of the d-graph in some way.

<u>Step 2</u>: Construct a sequence of moves (which consist of moving from an arc to the next one across a transition) down χ_i so that the slice resulting from each move is feasible and the last resulting slice, γ', has the property



Capacity = (25,25)

Figure 1a



Capacity = (25,25)

Figure 1b

$\underline{d}(\gamma' \cap \chi_i) \le \underline{d}(\sigma \cap \chi_i)$ where $\underline{d}$ (arc) means the vector of demand associated with the arc

and the sequence is maximal, i.e.

$\underline{d}(\text{i.s.}(\gamma) \cap \chi_i) \nleq \underline{d}(\gamma' \cap \chi_i)$ where $\le$ means every component of the left hand vector is less than the corresponding component of the right hand vector

and where i's is the immediate successor function

If such a sequence cannot be constructed, go to step 3; if it can be constructed, replace σ by γ' and repeat unless $\gamma' \equiv \gamma_T$, the bottom-most slice. If $\gamma' \equiv \gamma_T$, make a note of the success and quit.

Step 3: Pick another unused chain χ_j, $j \neq i$, and go to step 2. If none can be found, note the failure and quit.

With reference to figure 1a, the result of applying the algorithm to σ is to lead to γ' and then to γ'' whereat the algorithm fails. Can one conclude as in the earlier case, that σ is unsafe? Clearly not, for $\sigma \to \gamma^* \to \gamma^{**}$ shows a way of constructing a full feasible sequence from σ to σ_T. Thus the algorithm no longer has the prefix property for the slices γ', γ'', etc., which it produces, i.e., the property that they represent correct moves which need never be regretted. It is easily seen from the figure that modifying the test in step 2 of the algorithm to read:

$$\forall \zeta \; \{ \forall j, \ni 1 \le j \le n, \{ \underline{d} \; (\gamma' \cap \chi_i) \le \underline{d} \; (\zeta \cap \chi_i) \; \} \; \}$$ where

$\sigma \le \zeta \le \gamma'$ (i.e., ζ lies between slices σ and γ') and

$$\underline{d} \; (\text{i.s} \; (\gamma') \cap \chi_i) \nleq \underline{d} \; (\gamma' \cap \chi_i)$$

ensures that the prefix property is retained. In fact one can prove this, as shown in the lemma after the two definitions which follow. The safeness algorithm with this substitution will be called the Modified Safeness Algorithm.

Definition: The set of extensions E_D of a demand graph D with respect to a slice γ of the demand graph is the set of all demand graphs having (i) the same number of chains and (ii) the same number of arcs, with the same vectors of demand associated with them, up to the slice γ but arbitrary many arcs (≥ 1 per chain), having arbitrary demand vectors (subject, however, to the same capacity vectors) associated with them below γ. Any single demand graph in this set is called an extension of D with respect to γ. Figure 1(b) shows an extension of the demand graph of figure 1(a) with respect to the slice γ^*.

Definition: A slice γ of a demand graph D which can be reached by a sequence of feasible slices from σ is said to have the prefix property with respect to D and σ iff

$$\forall D' \in E_D \quad \{ \sigma \text{ is safe with respect to an extension of } D' \text{ of } D \text{ with respect to } \gamma \Rightarrow \exists \text{ a sequence of feasible slices from } \gamma \text{ to } \gamma_T^{D'} \text{ (the bottom-most slice of } D') \}.$$

In words the prefix property assures correctness of the partial sequence from σ to γ, in constructing a full sequence from σ, without having to look at the part of the demand graph below γ.

Lemma 1: A slice γ of D, which has at least two arcs on it with non-zero demand vectors associated, has the prefix property with respect to D and γ iff

$$\{ \forall i,j,\zeta \quad \underline{d}\,(\gamma \cap \chi_i) \leq \underline{d}\,(\zeta \cap \chi_i) \} \text{ where } 1 \leq i \leq m,\ 1 \leq j \leq n \text{ and } \sigma \lneq \zeta \leq \gamma, \text{ i.e., } \zeta \text{ lies between } \sigma \text{ and } \gamma$$

i.e., iff $\forall i$ every component of $d(\gamma \cap \chi_i) \leq$ the corresponding component of $d(\zeta \cap \chi_i)$.

<u>Proof</u>: <u>Direct</u>: Suppose σ is safe with respect to an extension D' of D with respect to γ. Then $\exists$ a full sequence of feasible slices from σ to $\gamma_T^{D'}$, call it Σ. Let $\sigma_o \in \Sigma$ be the first slice in Σ to use an arc just past γ (i.e., it is the first element of $\Sigma \ni \sigma_o \nleq \gamma$). Then it is clear that $\sigma \cap \chi_i \leq \sigma_o \cap \chi_i \leq \gamma \cap \chi_i$

$$i \neq j \quad 1 \leq i \leq m$$

$$\text{and } \gamma \cap \chi_j \leq \sigma_o \cap \chi_j$$

where $\sigma_o \cap \chi_j$ is the arc just past γ that lies on σ_0.

Thus if γ satisfies the property above then clearly

$$\underline{d}\ (\gamma \cap \chi_i) \leq \underline{d}\ (\sigma_o \cap \chi_i) \qquad (1)$$

and since σ_o is feasible

$$\sum_{k=1}^{m} \underline{d}\ (\sigma_o \cap \chi_k) \leq \underline{C} \qquad (2)$$

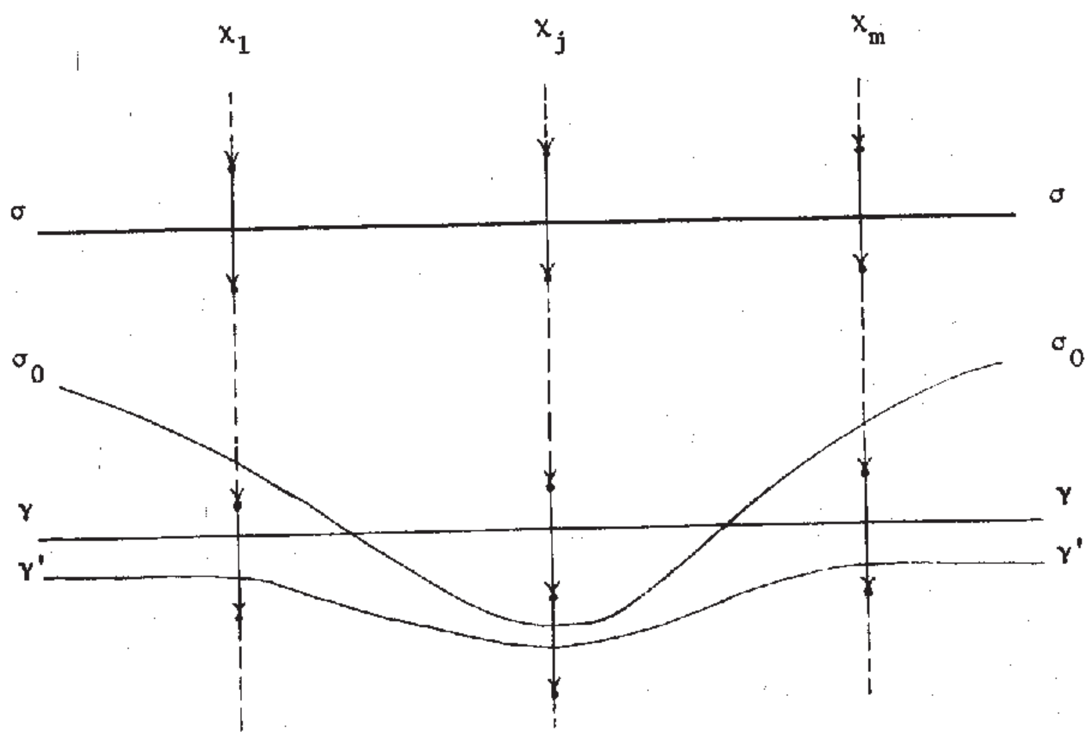
where $\underline{C}$ is the capacity vector

From (1) and (2)

$[\gamma - \gamma \cap \chi_j] \cdot \sigma_o \cap \chi_j$ is feasible, where the expression on the left represents the slice γ' in figure 2(a). Thus there is a feasible extension of $\sigma \rightarrow \gamma$ to γ'. Now, since

$$\sum_{k=1}^{m} \underline{d}\ (\gamma' \cap \chi_k) \leq \sum_{k=1}^{m} \underline{d}\ (\sigma_o \cap \chi_k) \text{ from (1)}$$

it follows that all moves down any chain that were possible (i.e. resulted in feasible slices) from σ_o are possible from γ. Thus the tail Σ' of Σ, the part following σ_o, is applicable to the sub-sequence $\sigma \rightarrow \gamma$ in the sense that all the moves represented are possible. Applying Σ' (less the moves that have already been made when one starts at γ) to the sub-sequence $\sigma \rightarrow \gamma$ one has a full sequence from σ that has $\sigma \rightarrow \gamma$ as a prefix.



χ_1
χ_j
χ_m
σ
σ
σ_0
σ_0
γ
γ
γ'
γ'

Figure 2a

Converse: Suppose γ does not satisfy the stated condition. Then

$\exists\ \chi_i \cdot \alpha_i,\ j \ni$

$$[\underline{d}\ (\gamma \cap \chi_i]_j > [\underline{d}\ (\alpha_i)]_j \quad \text{where } \sigma \cap \chi_i \leq \alpha_i \leq \gamma \cap \chi_i$$

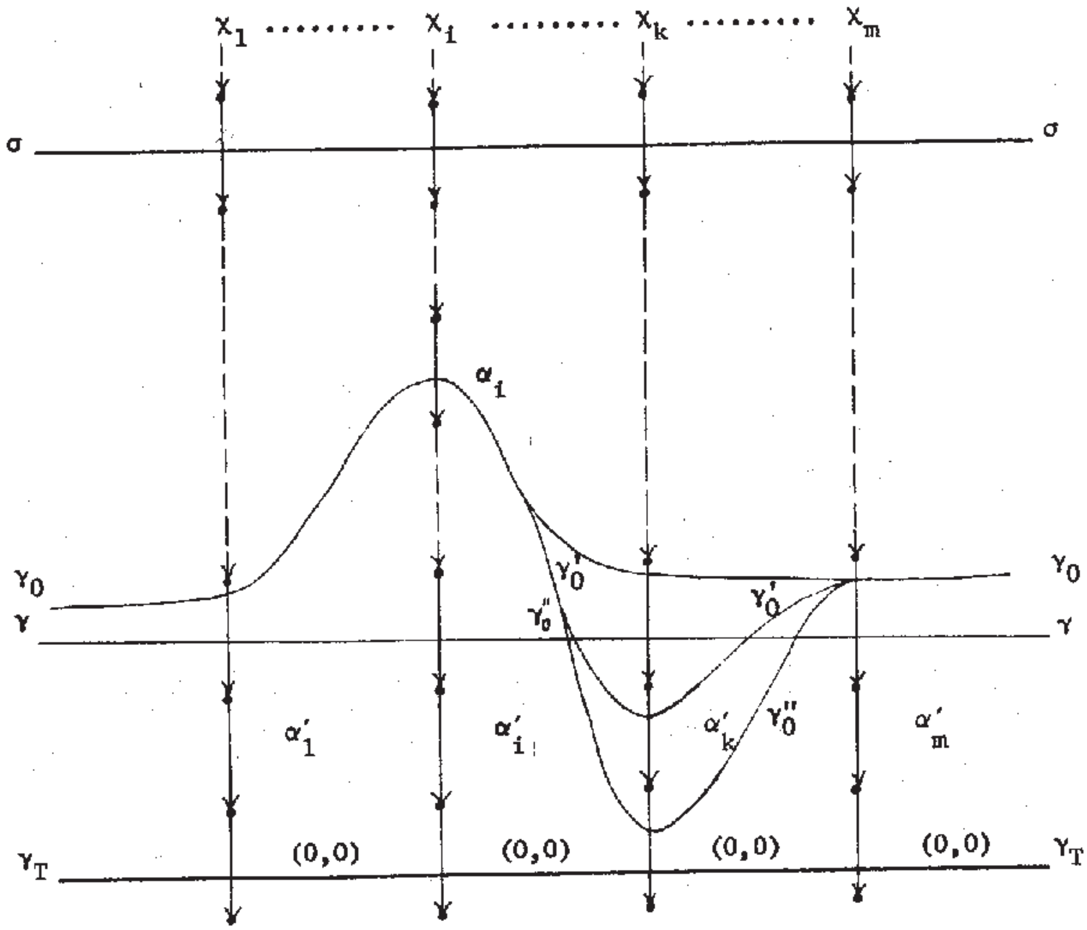
Let γ_0 be the slice $[\gamma - \gamma \cap \chi_i] \cdot \alpha_i$, i.e. the slice obtained by replacing $\gamma \cap \chi_i$ by α_i.

It will be shown that there exists an extension D' of D wrt γ for which σ is safe but there does not exist a full sequence from σ to $\gamma_T^{D'}$ having $\sigma \rightarrow \gamma$ as a prefix. D' is shown in figure 2b in its general form and in figure 2c as a specific example. It is easy to see that σ is safe since there is a full sequence from σ to γ_T by way of γ_0, γ_0' and and γ_0''. However there is no way of constructing a feasible sequence from γ to γ_T.

Thus there exists an extension of D wrt γ for which there is no full sequence from σ to γ_T by way of γ even though σ is safe. Thus γ does not have the prefix property. Q.E.D.

Note: There is a degenerate case, viz that of a slice having only one non-terminal slice, when the slice always has the prefix property.

The prefix property assures one that partial sequences of feasible slices can be extended to full ones. There remains the problem, however, of constructing that extension. The logical action is to continue to apply the Modified Safeness Algorithm so as to construct the extension. However, what happens when the algorithm fails to yield an extension? Can one conclude that σ is unsafe? Unfortunately this is not the case as figure 3 illustrates. There the algorithm provides no extension of the sequence $\sigma \rightarrow \gamma$. However $\gamma \rightarrow \gamma' \rightarrow \gamma'' \rightarrow \gamma'''$ indicates one extension. Thus the modified



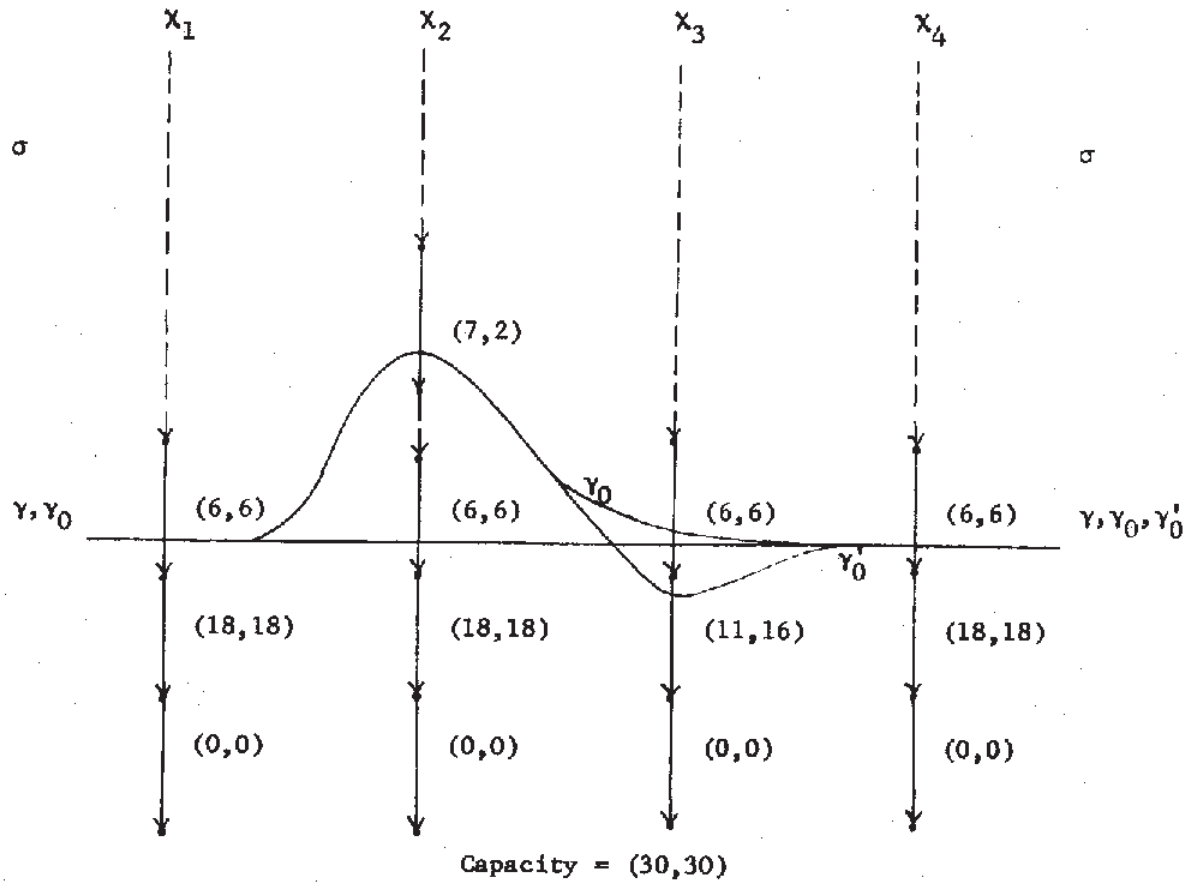
For $\ell \neq k$ $\forall r \left\{ [\, \underline{d}(\alpha'_\ell) \,]_r = [\, \underline{C} \,]_r - \sum_{\substack{s=1 \\ s\neq \ell,k}}^{m} [\underline{d}(\gamma \cap X_s)]_r \right\}$ $1 \leq r \leq n$ and $\underline{C}$ is the capacity vector

i.e. the slice $[\, \gamma - \gamma \cap X_\ell \,] \cdot \alpha'_\ell$ has been made infeasible

For $\ell = k$ $\forall r \left\{ [\underline{d}(\alpha'_k)]_r = [\, \underline{C} \,]_r - \sum_{\substack{s=1 \\ s\neq i,k}}^{m} [\, \underline{d}(\gamma \cap X_s) \,]_r - [\, d(\alpha_i) \,]_r \right\} 1 \leq r \leq n$

i.e. the slice $[\gamma - \gamma \cap X_k] \cdot \alpha'_k$ is made infeasible but γ'_0 is feasible

Figure 2b



X_1
X_2
X_3
X_4
σ
σ
(7,2)
γ_0
γ, γ_0
(6,6)
(6,6)
(6,6)
(6,6)
$\gamma, \gamma_0, \gamma_0'$
γ_0'
(18,18)
(18,18)
(11,16)
(18,18)
(0,0)
(0,0)
(0,0)
(0,0)
Capacity = (30,30)

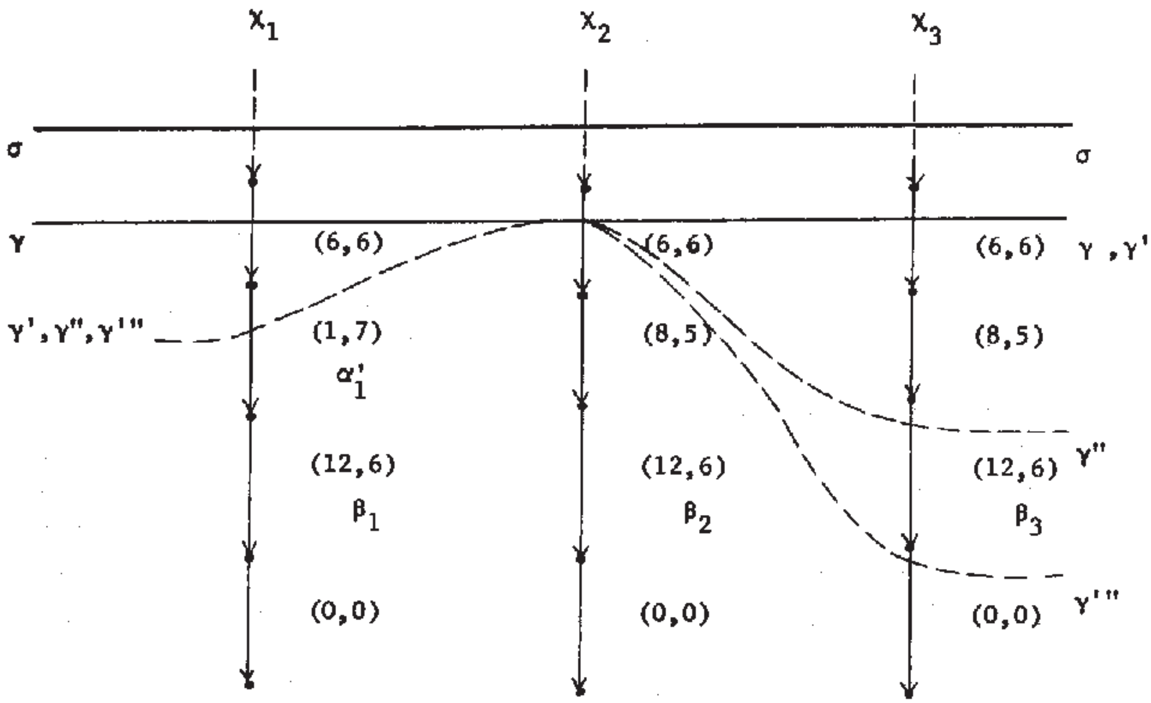
Figure 2c

safeness algorithm has to be augmented so that it uses a different strategy when it fails in the form in which it has been stated. Suppose one adds a step, call it step 2', which is to be used in place of 2 in this case:

Step 2': Construct a sequence of moves down χ_i so that the slice resulting from each move is feasible and the last resulting slice γ' has the property

$$\exists\, j \quad \ni \quad [\underline{d}\ (\gamma \cap \chi_i)]_j < [\underline{d}\ (\sigma \cap \chi_i)]_j$$

In other words, when attempts to find an arc on a chain which can be reached by a feasible sequence of slices moving down that chain from γ and which is the best overall, fail one wants to try <u>crutches wrt that slice γ</u>, which are better at least with respect to some one component of demand (like α_1' in figure 3), with the hope that one can overcome the <u>barriers</u> discovered (one on each chain) in the initial attempts (β_1, β_2, β_3 in figure 3). In general there are several arcs such as α_i' between $\gamma \cap \chi_i$ and β_i and it is clear that some may be of no use in that moving down to them does not result in an adequate release of critical resources to overcome any barrier. In fact figure 1a shows that one may have to move to more than one of the arcs α_i' in order to be able to cross the barrier (the sequence is $\gamma \rightarrow \gamma' \rightarrow \gamma'' \rightarrow \gamma''' \rightarrow \gamma''''$). In general up to (m-1) of these crutches may be needed to overcome the barrier. Figure 4b shows another complication in that 1 crutch suffices to overcome the barrier but more crutches are needed to reach the bottom-most slice.



Capacity = (20,20)

Figure 3

The two figures above (4a and 4b) illustrate two different problems. Figure 4b emphasizes the fact that (intermediate) slices using a crutch do not have the prefix property on account of the result of Lemma 1. It seems logical then to seek crutches that facilitate crossing of a barrier and then immediately try to reach a slice which has the prefix property and which can now be reached. In order to keep the algorithm local (a term defined a little later), however, knowledge of the specific arc which defines a barrier on a chain is assumed to be unavailable. The complete algorithm is specified next.

<u>Augmented Safeness Algorithm:</u>

Step 0: Let σ be the slice whose safeness is to be examined. If $\sigma \equiv \gamma_T$ note the success and quit; if not go to step 1.

Step 1: Pick a chain χ_i of the demand graph (say χ_1) and go to step 2.

Step 2: Construct a sequence of moves from σ down χ_i so that the slice resulting from each move is feasible and a slice γ' is reached which has the two properties

(i) $\forall\ \delta\ \ \{\underline{d}\ (\gamma' \cap \chi_i) \leq \underline{d}\ (\delta \cap \chi_i)\}\ \ \sigma \lneq \delta \lneq \gamma'$

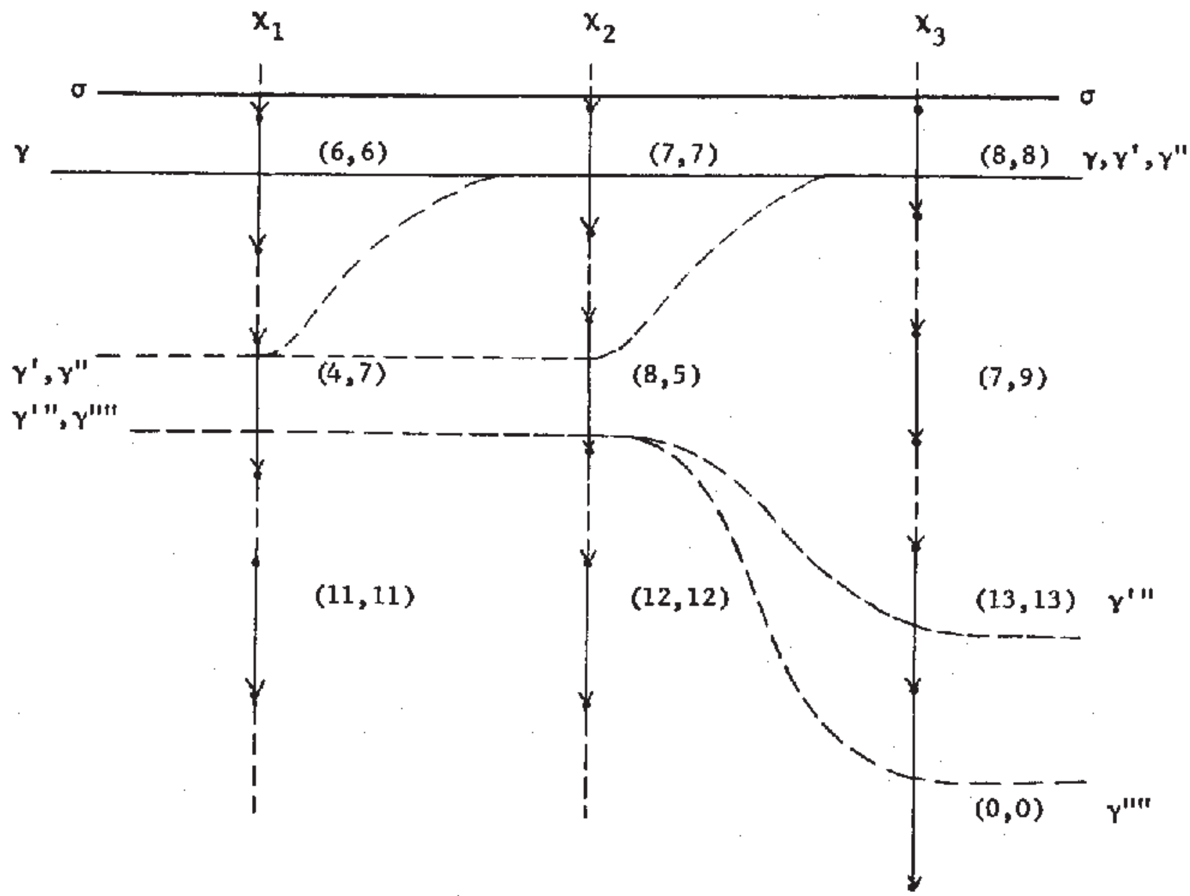
and

(ii) $\underline{d}\ (\text{i.s.}(\gamma') \cap \chi_i) \nleq \underline{d}\ (\gamma' \cap \chi_i)$ where

"i.s." is the immediate successor function.

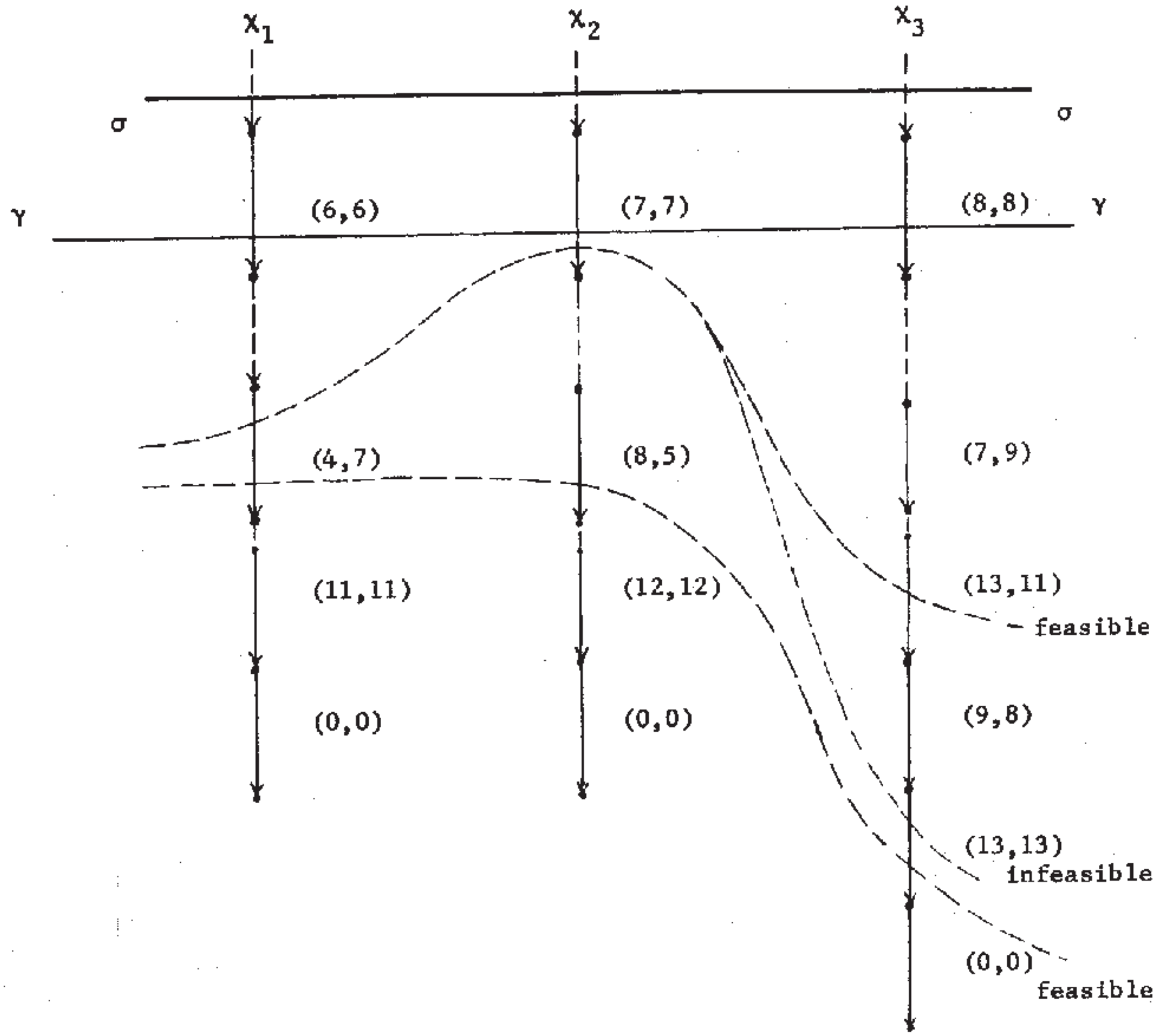
If such a sequence can be constructed replace σ by γ' and go to step 0.

If such a sequence cannot be found go to step 3.



Capacity = (25,25)

Figure 4a



Capacity = (25,25)

Figure 4b

Step 3: Pick an unused chain χ_j, $j \neq i$, (say $j = i+1$) and go to step 2 if such a j can be found. If none can be found call the Complementary Algorithm with ω and γ_0 set equal to γ. If it returns successfully set $\sigma = \gamma^*$ and go to step 0 else note the failure and quit.

<u>Complementary Algorithm</u>:

This algorithm takes three parameters: a demand graph D, a slice ω and a slice γ. The first parameter will be consider to have been passed to the algorithm at the first call or implicitly and, hence, is not mentioned in calls to the algorithm. The algorithm returns with a slice γ^* having the prefix property wrt ω and a sequence of moves from ω to γ^* when it is successful. In case of failure no value is returned.

Step 1: Pick a chain χ_i (say χ_1) and go to step 2.

Step 2: Construct a sequence of moves from γ down χ_i so that each resulting slice is feasible and a slice γ^* is reached which satisfies the condition:

$$\forall j \ni [\underline{d}\,(\gamma^* \cap \chi_i)]_j < [\underline{d}\,(\omega \cap \chi_i)]_j$$

i.e. $\gamma^* \cap \chi_i$ is a crutch of wrt ω.

If such a sequence can be constructed go to step 3, if not go to step 4.

Step 3: If γ^* has the prefix property wrt ω take the successful return and return with the sequence from ω to γ^*. If it does not have the prefix property call the Complementary

Algorithm with γ_0 set equal to γ^*, and ω unchanged. Go to step 4 in case of unsuccessful return otherwise go to step 3.

Step 4: Pick another unused chain χ_j (say χ_{i+1}). If such a chain can be found go to step 2. If not, take the unsuccessful return.

It will be noticed that the Complementary Algorithm is recursive and uses a tree growing approach to the construction of a sequence of feasible moves to a slice with the prefix property. The number of possible branches at a node is the number of chains having crutches that are accessible from the current slice by a sequence of moves down the corresponding chain. When one of these crutches is chosen the resulting slice becomes a successor node to the prior one and the process repeats until a slice with the prefix property is found. (See figure 5b).

It is shown in lemma 2 below that the Augmented Safeness Algorithm is always successful whenever σ is safe (conversely, if it is successful, σ is safe.)

Lemma 2: The Augmented Safeness Algorithm applied to a slice σ and demand graph D is successful $\Leftrightarrow$ σ is safe in D.

Proof: By the definition of safeness the forward implication is true.

Now for the reverse implication. Any failure of the algorithm implies that a slice γ, which has the prefix property wrt σ was reached whereat the

Complementary Algorithm was applied and failed. Failure of the latter means that at every leaf of the tree grown was a slice δ from which no crutches could be reached, i.e. every chain i had an arc which is a feasibility barrier β_i' in terms of moves down that chain from δ and there is no crutch between β_i' and $\delta \cap \chi_i$. Now let $\beta_1, \beta_2, \cdots \beta_m$ be the farthest (i.e. lowest down) of these barriers (there is one for each leaf of the tree) on the chains $\chi_1, \chi_2, \cdots \chi_m$ respectively.

Suppose now that σ were safe. Then $\exists$ a sequence ∇ of slices from σ to γ_T, the terminal slice. Thus there is a slice σ_0 which is the first slice in ∇ to use one of the β_i.

Consider $\sigma_0 \cap \chi_j$ $(j \neq i)$.

$\sigma \cap \chi_j \leq \sigma_0 \cap \chi_j \leq \beta_j$ by the choice of σ_0

If $\sigma_0 \cap \chi_j \leq \gamma \cap \chi_j$

then $\underline{d}(\sigma_0 \cap \chi_j) \geq \underline{d}(\gamma \cap \chi_j)$ because γ has the prefix property wrt σ

$\therefore$ $[\sigma_0 - \sigma_0 \cap \chi_j] \cdot [\gamma \cap \chi_j]$ is feasible since σ_0 is ------------------ $\underline{1}$

If $\gamma \cap \chi_j < \sigma_0 \cap \chi_j$

and if $\nexists\, k \ni [\underline{d}(\sigma_0 \cap \chi_j)]_k < [\underline{d}(\gamma \cap \chi_j)]_k$

i.e. if $\sigma_0 \cap \chi_j$ is not a crutch wrt γ

then $\underline{d}(\gamma \cap \chi_j) \leq \underline{d}(\sigma_0 \cap \chi_j)$ and therefore

$[\sigma_0 - \sigma_0 \cap \chi_j] \cdot [\gamma \cap \chi_j]$ is again feasible ------------------------ $\underline{2}$

From 1 and 2 one sees that one can move the slice σ_0 from $\sigma_0 \cap \chi_j$ to $\gamma \cap \chi_j$ whenever

$\{\sigma_0 \cap \chi_j \geq \gamma \cap \chi_j$ and $\sigma_0 \cap \chi_j$ is a crutch wrt $\gamma\}$

is not true. Call the resulting slice γ^*

Then γ^* is feasible and lies below γ. Moreover

(i) γ^* uses only crutches in addition to arcs in γ

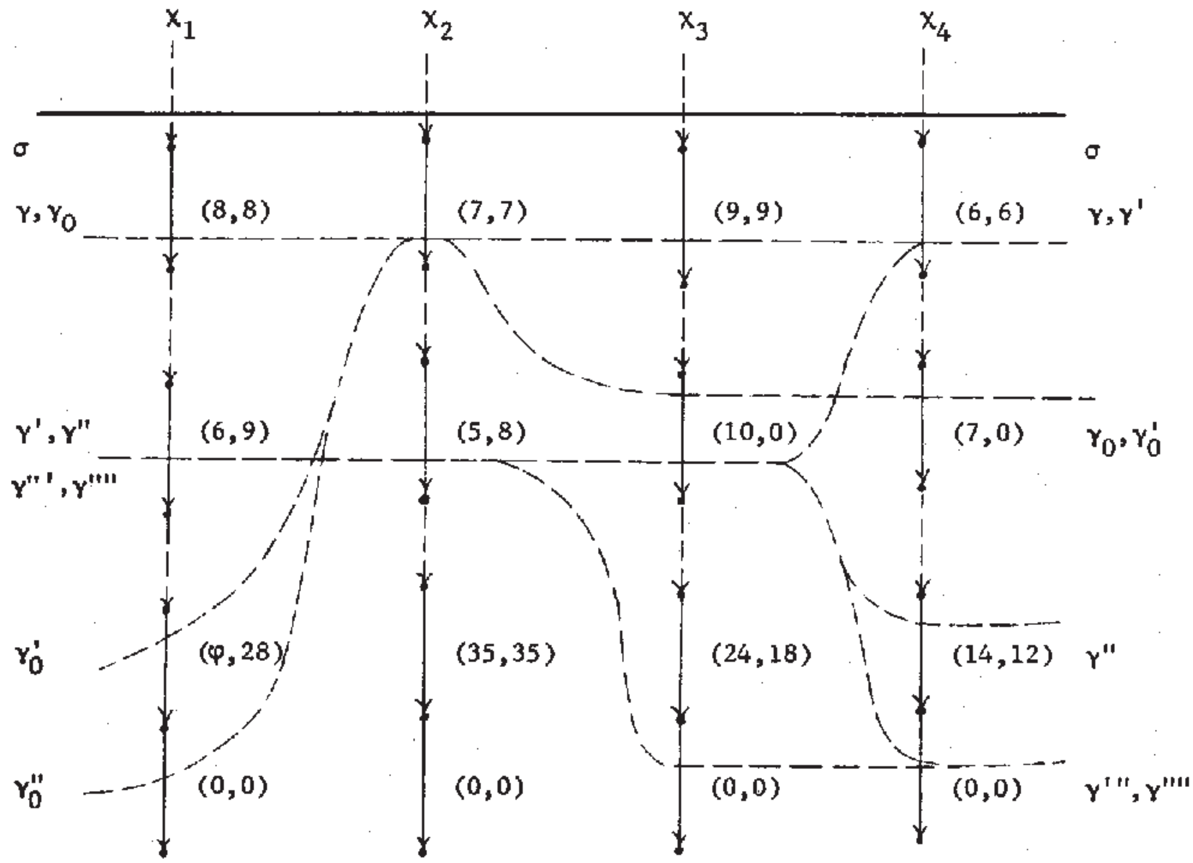
(ii) γ^* can be reached by a feasible sequence from σ since σ_0 can be and since $\forall k \;\; \underline{d}(\gamma^* \cap \chi_k) \leq \underline{d}(\sigma_0 \cap \chi_k) \quad 1 \leq k \leq m$

and finally γ^* can be reached by a feasible sequence from γ since γ has the prefix property wrt D and σ.

Thus γ^* must have been reached by the Complementary Algorithm applied at γ. However by definition β_i was the lowest barrier on χ_i wrt all leaves of the tree the algorithm examines. Thus there is no such γ^* -- a contradiction.

Hence Σ cannot exist, i.e. σ is unsafe. Q.E.D.

Reconsider the Complementary Algorithm. How far down does it need to look before it finds a slice which has the prefix property or realizes that one cannot be reached using that particular path in the tree? Unfortunately figure 4c shows that inability to reach such a slice may not be perceived until all but two chains have been gone down (it is clear that if one can reach a slice having only one non-terminal slice then it has the prefix property by virtue of the fact that all demands have to be less than the capacity of the system)! Thus the amount of backtracking that may be required for a single trial is considerable.



$\varphi \leq 11$ Capacity = (35,35)

Here one can get from γ to γ'''' before realising that the move $\gamma \rightarrow \gamma'$ was a mistake. One discovers upon further trials that σ is safe since $\gamma_0 \rightarrow \gamma_0 \rightarrow \gamma_0' \rightarrow \gamma_0''$ is one way of extending the sequence $\sigma \rightarrow \gamma$.

<u>Figure 4c</u>

Moreover the number of possible trials is also quite large as figure 4a points out. That figure shows that up to m-1 mediocre arcs may need to be used to overcome a barrier.

The discussion so far points out by example that when the Modified Safeness Algorithm reaches a slice γ beyond which it cannot proceed, then one has no choice but to experiment with crutches with the aim of reaching a slice γ^* with the prefix property, when the algorithm can be brought back into force. The amount of backtracking involved in reaching γ^* from γ appears to be large. A formal result regarding this backtracking is stated below, but some definitions are necessary for its statement and proof, and these follow. The term "algorithm" in the definitions and theorem refers to an algorithm for the construction of a full sequence of feasible slices for the purpose of checking the safeness of an arbitrary slice σ of a demand graph D.

Definition: A local algorithm is one which has a knowledge of the part of the demand graph above the current slice as the only (knowledge) input in making a decision regarding what move (in the literary sense) to make next.

For instance a local algorithm does not know or cannot "see" the entire remaining portion of the demand graph and thus make only the correct move (in the defined technical sense). Similarly a local algorithm does not have recall abilities in respect of past moves so that it cannot just sweep down the chains one at a time and thereby gain (and store) knowledge of the whole or part of the remaining portion of the demand graph. Were one to assume such an ability it is clear that an arbitrarily large memory would be required to store the information, and since any realistic memory has finite capacity such an assumption is clearly unrealistic. It is therefore convenient to assume zero recall capability (regarding futile moves). It should be clear now that both the Modified and Augmented Safeness Algorithm are local when the strategies outlined in parentheses in their definition are used.

Definition: A local algorithm is said to be a limited- backtracking algorithm if one can partition the sequence of slices it produces into sub-sequences whose initial and terminal slices have the prefix property wrt the demand graph and the slice σ whose safeness is being investigated.

In other words the construction of a full sequence proceeds as in figure 5a rather than as in figure 5b, where $\sigma \to \gamma_1$, $\gamma_1 \to \gamma_2$ represent the sub-sequences mentioned in the definition, while γ_1', γ_2' $\cdots$ are just plain intermediate slices.

Definition: A limited backtracking algorithm is said to be linear if the maximum number of sub-sequences examined, before the correct sub-sequence to adjoin to the partial sequence Σ already constructed is found (or it is discovered that none exists), is of the form

$$A.f(n_1, n_2, n_3, \cdots n_m)$$

where A is a constant and $f(n_1, n_2, \cdots n_m)$ is linear in the n_i i.e. of the form $\sum_{i=1}^{m} a_i n_i$ and the a_i are integer constants, and where the n_i are some appropriate (relevant) number of arcs on each chain below γ, the slice terminating the partial sequence Σ.

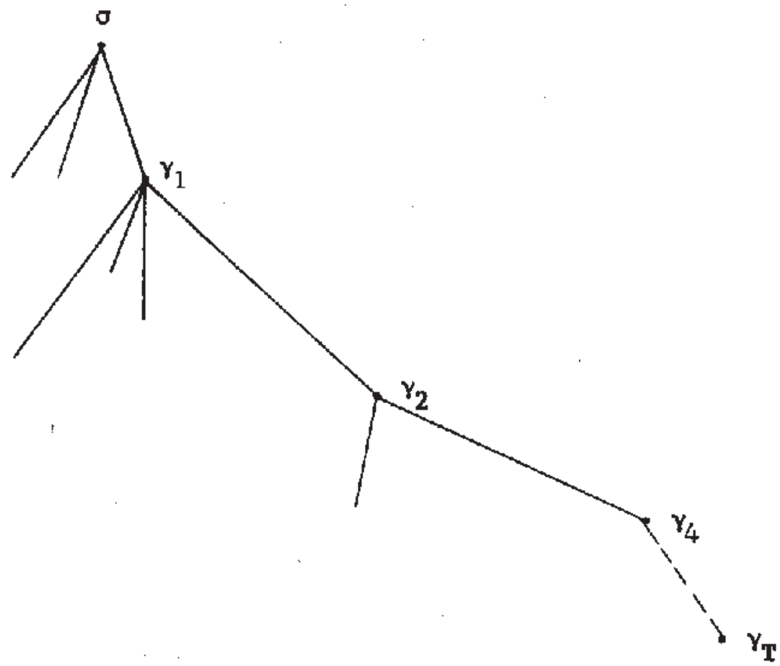


Figure 5a

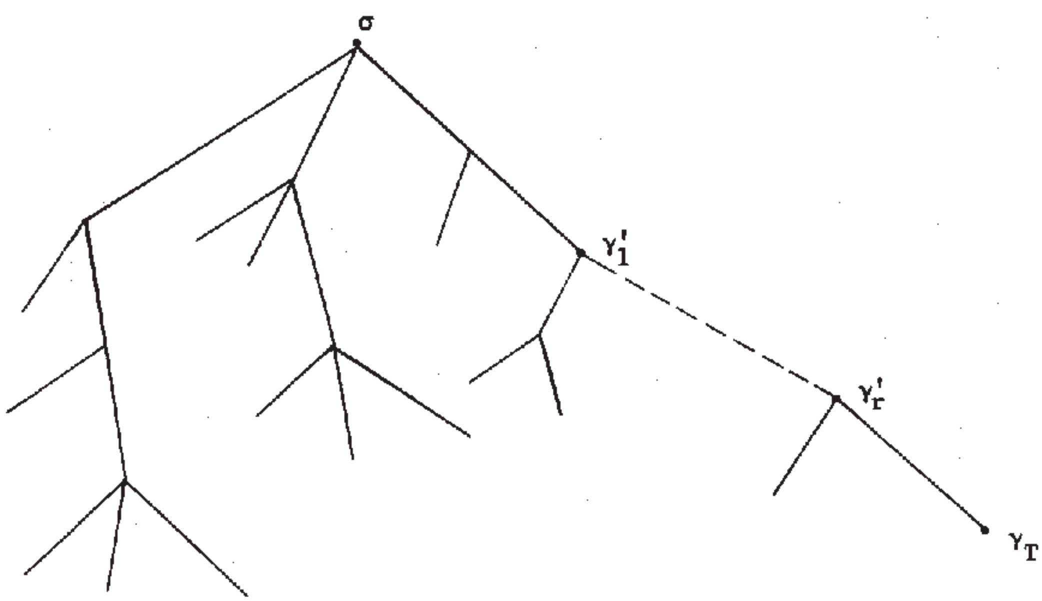


Figure 5b

If f increases faster than the sum of the n_i, the algorithm is said to be <u>of higher order</u>.

For example in the single resource type case the number of sub-sequences examined is m, i.e. A=m and $f(n_1, n_2, \ldots n_m) = 1$. An example of an algorithm of higher order is found the case of multiple resource types. (This statement is clarified in the theorem below.) The n_i in the case of the Complementary Algorithm are the number of crutches below γ (and above the next slice having the prefix property say).

<u>Theorem</u>: In the multi-resource case there does not exist a linear limited-backtracking algorithm (for testing the safeness of an allocation state or a slice of the demand graph)

<u>Proof</u>: The proof involves demonstration by means of a counter-example that the number of wasted trials is of higher order. Refer to figure 6 which shows the counterexample. The example will be clarified in the discussion. The slice γ represents the end of a partial sequence constructed somehow (perhaps by the Modified Safeness Algorithm) from the slice σ whose safeness is being tested. Slice γ has the prefix property. As a special case σ may be γ. By virtue of construction there are no arcs between $\gamma \cap \chi_i$ and β_i, for any i, that have demand vectors which are less than or equal to the demand vector associated with $\gamma \cap \chi_i$. Note that $(\underline{d}(\beta_i) \not\leq \underline{d}(\gamma \cap \chi_i)$ too . Thus the nearest slice with the prefix property and lying below γ lies below $\beta_1 \beta_2 \cdots \beta_m$ the barrier slice. Thus any limited backtracking algorithm, which by definition has to

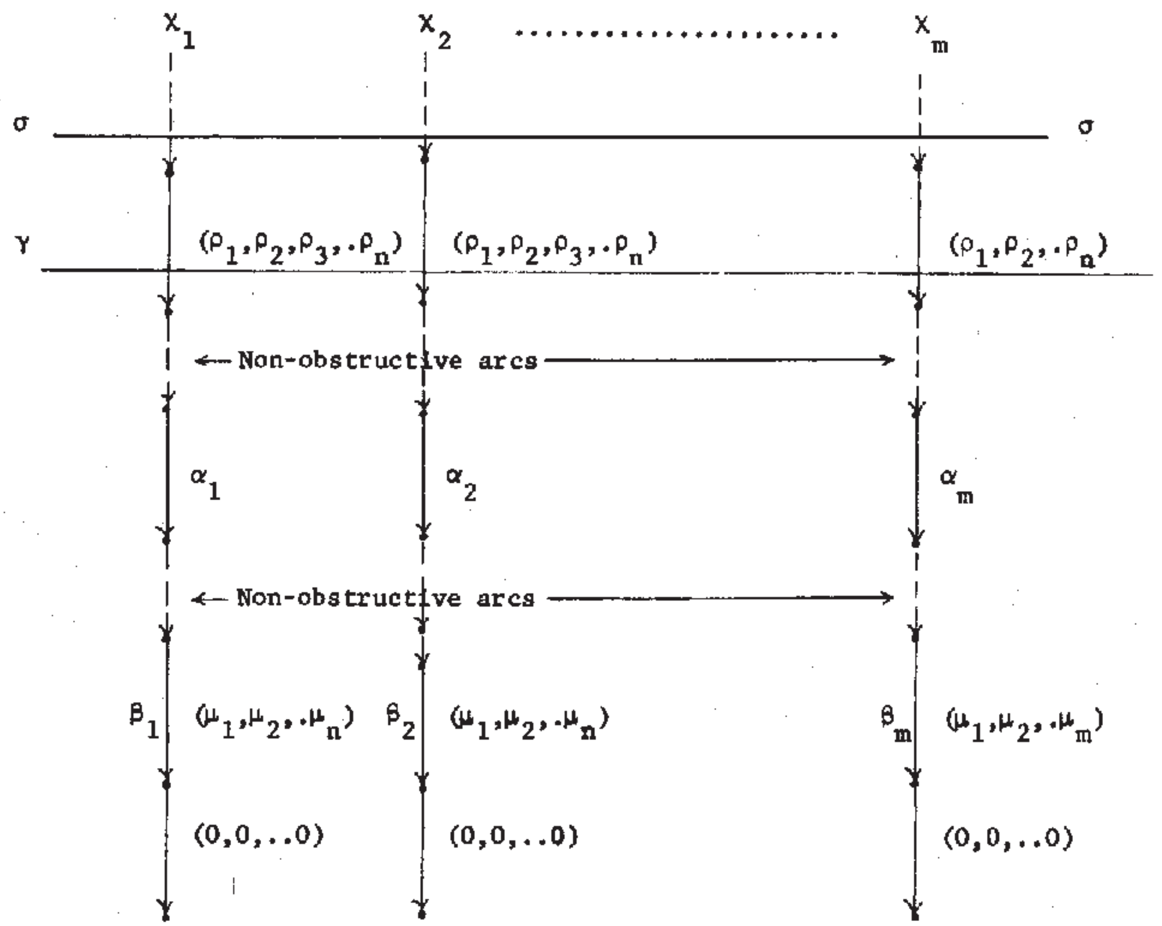
look for a prefix slice which is accessible from γ, has to construct a sub-sequence from γ to γ' where γ' is the first slice in this sub-sequence satisfying $\gamma' \not\geq \beta_1 \ \beta_2 \ \beta_3 \cdots \ \beta_m$ i.e. $\exists_j \ni \gamma' \cap \chi_j \not> \beta_j \quad 1 \leq j \leq n$

The demand graph has been constructed so that there is exactly one set of k crutches which must be used to overcome the barrier. By non-obstructive arcs are meant arcs whose demand vectors are such as to permit access from γ to any slice using ℓ of the crutches shown (for any $\ell \leq m-1$) and the remaining arcs from γ. These non-obstructive arcs may be crutches themselves, in fact they probably contain some of the n_i crutches that are presumed to exist on each chain between $\gamma \cap \chi_i$ and β_i. The skeptic may assume that the non-obstructive arcs are absent in which case $n_i = 1$ ($\forall$ i $\ni 1 \leq i \leq m$).

To further simplify understanding of the example, figure 7 shows a special case of figure 6.

The construction of figure 5 is quite general in that k can be an arbitrary integer between 1 and m-1. Now suppose a limited back-tracking algorithm is given. Since it is local it must examine the combinations of the crutches in some order and for each combination of r crutches it tries out some moves. However since there is only one combination that works, all other trials are wasted. The number of trials wasted can be made non-linear by choosing a value of k appropriate to the algorithm.

For example consider an algorithm that uses the crutches 1 at a time, 3 at a time, etc. up to m-1 or m-2 (whichever is odd) at a time and then 2, 4, 6 $\cdots$ at a time.



Capacity = $(C_1,C_2,\ldots.C_n)$

$\mu_\ell = 0 \quad$ for $\ell \neq j,h$

$\mu_j = \{ C_j-(m-1)\rho_j\} + k \qquad \mu_j > 0$

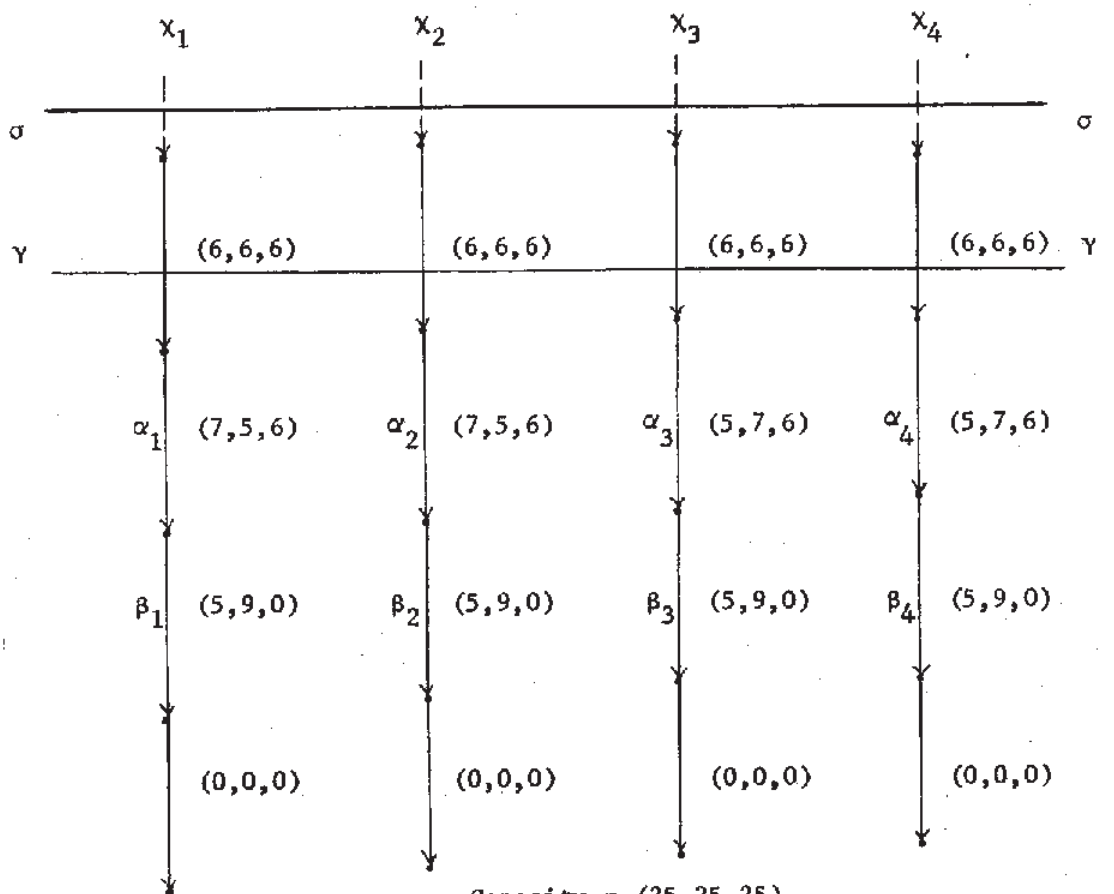
$\mu_h = \{ C_h-(m-1)\rho_h\} - k \qquad \mu_h \geq 0$

Of the arcs α_i (i) exactly k have the demand vector $(\rho_1,\rho_2,.\rho_h+1,.\rho_j-1,.\rho_n)$

(ii) the rest have the demand vector $(\rho_1,\rho_2,.\rho_h-1,.\rho_j+1,.\rho_n)$

The critical resource is the j^{th}. Specification of the h^{th} component is required to ensure that each α_i is a crutch but $\underline{d}(\alpha_i)\neq\underline{d}(\gamma\cap X_i)$

<u>Figure 6</u>



Capacity = (25,25,25)

Here m=3, n=3, k=2, h=1, j=2 in terms of the notation of figure 6.

Figure 7

Pick $k = 2$. Then the number of wasted trials

$$= \sum_{\substack{1 \\ (r \text{ odd})}}^{m'} \text{no. of combinations of } r \text{ crutches at a time} \qquad m' = m-1 \text{ or } m-2$$

$$= \text{the sum of the coefficients of } x^1, x^3, x^5, x^7 \ldots x^{n-1}$$

$$\text{in } (1 + n_1.x)\ (1 + n_2.x)\ (\quad)\ (\quad) \cdots (1 + n_m.x)$$

In case $n_i = 1$ for all i $(1 \le i \le m)$ the right hand side becomes 2^{m-1} (versus m).

Thus it has been shown that the number of futile trials is non-linear for the counter-example. Q.E.D.

<u>Comment</u>: The proof above is really quite conservative for figures 4b and 4c showed that merely being able to cross the barrier is not a guarantee of being able to reach a slice with the prefix property without further backtracking.

The theorem above indicates that the Modified Safeness Algorithm is in a sense optimal. As long as it succeeds the number of sequences examined in vain is at most m-1 and so the algorithm is linear. When it fails it is necessary to use crutches in a trial and error fashion to get past the barrier and then quickly reach a slice with the prefix property (say by use of the Complementary Algorithm), when the algorithm can be used again. Finally one should note the following:

<u>Comment</u>: It is clear that if no combination of crutches (from 1 to m-1 of them) permits crossing of a barrier then γ (and hence σ) is unsafe.

Reference:

[1] Hebalkar, P.G., "Coordinated Sharing of Resources in Asynchronous Systems", Project MAC Computation Structures Group Memo No. 45, January 1970.