G. CIOBANU

A Multi-Project Programming Model With Resources Levelling

In any complex action, activities are grouped per projects. The starting terms and the concluding terms both for projects and activities are established so as to meet certain conditions in order to minimize the implementation time of a project and to cover uniformly the resources.

A multi-project is defined as an aggregate of projects conducted by a central body, whereas a project is defined as a number of activities having exactly one final goal. If a number of works with common resources is required to be programmed for a certain time interval, such items shall be defined as projects, and the multi-project shall be defined as the complete aggregate of the said projects, which also includes their sequence interconditioning. Each project can be plotted by a directed sub-graph, and the multi-project can be plotted by a graph containing the said sub-graphs whose interconnections appear to be the interconditioning of the projects. This paper considers the activities of the projects plotted by arcs. An activity cannot be programmed if not all the directly precedence activities have been programmed.

The paper is meant to describe an algorithm for scheduling the projects and the activities of each project so as to achieve the best levelling of the resources during a time interval imposed or equal to the length of the critical path in the graph of the multi-project. The projects may have common resources. Such a scheduling must meet the condition that the development shall be continuous and shortest, taking into account the interconditioning of the activities and the levelling of the resources against the aggregate of projects.

1. *Drawing up the graph.*

1.1. Consider a finite, directed, without circuits*) graph $\mathbf{G} = (X, U)$ plotted by the set of nodes (vertices) $X = \{x_1, x_2, \ldots x_n\}$ and the set of arcs $U = \{u\}$. The graph is assumed to have no isolated nodes. We write $X_s = \{x_{s1}, x_{s2}, \ldots, x_{sp}\}$ for the initial nodes of the graph, and $X_f = \{x_{f1}, x_{f1}, \ldots, x_{fq}\}$ for the final nodes. Each arc $u \in U$ is plotted by a couple of nodes $x_i \in X$, $x_j \in X$ which are joined by an arrow directed from x_i, named predecessor node, towards x_j, named successor node, and is written $u = (x_i, x_j)$.

We associate to each arc $u = (x_i, x_j)$ of graph $\mathbf{G}$ a positive number ($\geqslant 0$), named the length of the arc, writing it d_u or d_{ij}.

1.2. Consider Y to be a sub-set of X with $X_s \subset Y$ and $X_f \subset Y$. Consider the directed graph $\mathbf{G}^* = (Y, V)$ which has Y as the set of the nodes and arcs $V = \{v\}$ thus defined: there is arc $v = (x_i, x_j)$ in graph $\mathbf{G}^*$ with $x_i \in Y$ and $x_j \in Y$ if there exists in graph $\mathbf{G}$ a direct precedence or path from x_i to x_j with $x_i \in X$ and $x_j \in X$.

The length of arc $v = (x_i, x_j)$ is written D_v or D_{ij} and is calculated as follows:

a) in the logical order of graph G (directly covered), values

$$t_j^+ = 0 \text{ if } x_j \in X_s$$

$$t_j^+ = \max_{(x_i, x_j) \in U} \left[\begin{pmatrix} t_i^+ & \text{if } x_i \notin Y \\ 0 & \text{if } x_i \in Y \end{pmatrix} + \text{dij} \right] \text{ if } x_j \in X/X_s$$

are found;

b) $D_{ij} = t_i^+$ whatever $(x_i, x_j) \in V$ may be.

1.3. Consider sub-graphs $\mathbf{G}_k = (X_k, U_k)$ $k = 1, 2, \ldots, m$ of graph G to meet the following requirements:

a) $U_k \cap U_h = \emptyset$ if $k \neq h$.

b) Each sub-graph $\mathbf{G}_k$ has one initial node x_k pertaining to Y. Without restricting the generality of the problem, we consider one sole arc $(x_h, x_l) \in U_k$ to exist, having as its predecessor node the initial node x_k of sub-graph G_k.

c) Each sub-graph $\mathbf{G}_k$ may have one or more final nodes $\{x_{v_1}, x_{v_2}, \ldots, x^{vk}\} \subset X$ of which at least one $x_{vi} \in Y$ exists.

d) Several graphs may have common the initial node or the final nodes.

e) Sub-graphs $\mathbf{G}_k$ require to choose a set Y; the elements of Y are simultaneously initial nodes for some sub-graphs and final nodes for other sub-graphs. Nodes $x_\alpha \in Y$ are named interface nodes.

f) The final nodes $x_\mu \in Y$ are named interconditioning nodes of sub-graphs and their set is written X_μ.

We understand that:

1°) $(x_\alpha, x_\beta) \in U_\nu$ with $x_\beta \in X_\mu$ if $x_\alpha \in X_\nu$

2°) $x_\beta \in X_\nu$ and $x_\beta \in X_\lambda$ with $x_\beta \in X_\mu$ if $(x_\alpha, x_\beta) \in U_\nu$.

---

*) See Addendum.

2. *Defining the multi-project programming problem.*

2.1. A multi-project is a aggregate of interconnected projects $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$. Aggregate $\mathcal{P}$ is plotted by a graph G, and each project P_k is plotted by sub-graphs G_k. Thus we get the following correspondence: graph node ≡ project event; graph arc ≡ project activity; arc length ≡ activity length. Therefore, we shall observe the same notation (for instance, $u \in U$ is an arc as regards the graph and an activity as regards a project).

Consider $R_1, R_2, \ldots, R_g$ to be the resources used for achieving the multi-project, and $(r_1(u), r_2(u), \ldots, r_g(u)$ the intensities of the resources (quantity per time unit) required for performing activity $u \in \bigcup$; if $r_i(u) = 0$, then activity (u) doesn't make use of resource R_1.

We assume that projects P_k $(k = 1, 2, \ldots, m)$ may have common resources.

2.2. Through a multi-project programming with resources levelling we understand a scheduling of the activities of aggregate $\mathcal{P}$ of projects $P_1, P_2, \ldots, P_m$ under the following restrictions:

a) The activities of each project P_k shall be scheduled for a time interval (the length in time of project P_k):

$$T_k = L(P_h) = \max_h D_{hk}$$

where x_k is an initial node and $\{x_h\}$ are final nodes (interface nodes + interconditioning nodes) of sub-graph G_k.

b) The resources per aggregate $\mathcal{P}$ of projects P_k $(k = 1, 2, \ldots, m)$ to be levelled.

c) Time T for performing the aggregate of projects is imposed or equal to the length of the critical path in graph G.

If condition a) is not set because of the time reserves, on considering the whole aggregate $\mathcal{P}$ to be a sole project, by levelling resources there is a possibility to schedule the activities of each project P during a time interval $L > L(P_k)$ and, consequently, discordant breakdowns (time intervals) between the activities of the projects may occur, thus incurring disturbances into the organization and management work for the aggregate of projects and the good use of the resources.

3. *Calculating the utmost terms for starting projects and activities.*

3. 1. Case $x_i \in Y$. We make use of the critical path algorithm for graph G and we hold for activities $u = (x_i, x_g)$ with $x_i \in Y$ as the absolute minimal and maximal starting terms, which are written $\underline{K}_n$ and, respectively, $\overline{K}_n$.

If $K \in [\underline{K}_u, \overline{K}_u]$ taking into account restriction 1.3.b for a project P_v, for which $u \in P_v$ we have at the same time: K, as the starting term of project P_v, and K, as the starting term of activity (u).

3.2. Case $x_i \notin Y$. Starting from the final nodes X_f towards the initial nodes X_s in the reverse order of the topology of graph G (covered in the reverse way), values

$$t_i^* = t_i^+ \text{ if } x_i \in X_f$$

$$t_i^* = \min_{(x_i, x_j) \in U} \left[\begin{pmatrix} t_j^* \text{ if } x_j \notin Y \\ t_j^+ \text{ if } x_j \in Y \end{pmatrix} - dij \right] \text{ if } x_i \in X \backslash X_f$$

are calculated.

For each activity $u \in U_\nu$, $u = (x_i, x_j)$, $x_i \notin Y$ we define as a time reserve within project P_ν:

$$m_u = t_j^* - t_i^+ - dij,$$

and as (relative) starting terms:

minimal: $\underline{t}_u = t_i^+$,

maximal: $\bar{t}_u = \begin{cases} t_j^* - d_{ij} & \text{if } x_j \notin y \\ t_j^+ - d_{ij} & \text{if } x_j \notin y \end{cases}$

If K is the starting term for project P_ν, then the (absolute) terms for starting any activity $u = (x_i, x_j)$, $u \in P_\nu$, with $x_i \notin Y$ are:

minimal: $K + \underline{t}_u$, maximal: $K + \bar{t}_u$.

In order to observe the same notation for the case $x_i \in Y$ we shall write $= \underline{t}_u = \underline{K}_u$ and $\bar{t}_u = \bar{K}_u$.

4. *Scheduling projects and activities*

4.1. By project and activities scheduling we understand the statement of an order in time for performing the activities classified per projects which are in a precedence relationship. Such order in time is yielded by the starting terms t_u of activities $u \in U$. Further, we shall describe an algorithm for ascertaining the terms under the restrictions shown so far.

The activities of graph G are assumed to be disposed in a sequential list in a logical order *), where each activity will be marked by a current number depending upon the predecessor relationships in the graph. We shall write:

K_i, for the starting time of project P_i;

Q_h, for the minimal starting time in scheduling the activities having x_h as predecessor node;

t_u, for the starting term of activity $u \in U$.

The algorithm consists in the following: activities (u) are covered in agreement with the order stated in the sequential list of graph G; for each activity is ascertained an interval $[I_1, I_2]$ within which it can start, contingent upon vector Q_h and the activity pertaining to a certain project P_i; conside ring thestarting terms to be uniformly distributed along the ascertained interval, in agreement with this distribution the starting term $K \in [I_1, I_2]$ of the activity is generated; in case the activity considered has a preɛecessor node and an interface node ($\in Y$), the generated starting term K shall also be the starting term of project P_i to which it pertains ($K_i = K$). It is obvious that the reasons for which restrictions have been imposed in drawing up the graph are given in the algorithm.

At the same time with the ascertaining of the starting term of each activity, profiles S_1, $S_1, \ldots$, S_g of the resources $\{R_1, R_2, \ldots R_g\}$ are also calculated.

By the profile S_i of a resource R_i we understand the required number of resource units $S_i(j)$ in each time unit j from the beginning till the com-

*) In the increasing order of the rank of the activities.

pletion of the multi-project, in agreement with activities scheduling, and it is calculated with the aid of formula:

$$S_i(j) = \sum_{u \in A} s_i(u) \qquad \begin{matrix} i = 1, 2, \ldots, g \\ j = 1, 2, \ldots, T \end{matrix}$$

where

$$A = \{u \mid t_u \leqslant j < t_u + d_u, u \in U\}$$

4.2. The algorithm for ascertaining a scheduling is achieved through the following steps:

a) We make $p = 1$ and $Q_h = 0$ $(h = 1, 2, \ldots, n)$.

b) We consider activity $u = (x_\alpha, x_\beta)$ with the current number p; let P_v be the project containing activity u.

c) If $x_\beta \notin Y$, we go to step e).

d) We generate a random number $t_u \in [Q_\alpha, \bar{t}_u]$, make $k_v = t_u$ and go to step g).

e) If activity $u = (x_\alpha, x_\beta)$ has the time reserve equal to nil $(m_{\alpha\beta} = 0)$. or has no resources to be optimized, we make $t_u = Q_\alpha$ and go to step g).

f) We generate a random number $t_u \in [Q_\alpha, K_v + \bar{t}_u]$

g) Activity u is programmed for term t_u. We calculate the minimal starting terms

$$Q_\beta = max\,(Q_\beta, t_u + d_{\alpha\beta})$$

and the profiles of the resources if $d_{\alpha\beta} \neq 0$

$$S_l(j) = S_l(j) + s_l(u)$$

for $j = t_u, t_u + 1, \ldots, t_u + d_{\alpha\beta} - 1$ and those $l = 1, 2, \ldots, g$ for which $s_e(u) \neq 0$

h) We make $p = p + 1$ and go to step b).

5. *Levelling the resources.*

5.1. By scheduling the projects and the activities with the resources levelling we understand to ascertain a scheduling in agreement with the algorithm described above, so as to use the resources as uniformly as possible. By defining the objective, the economic content becomes obvious: a better operative programming. The completion term T of the aggregate of projects is set, by technological organizational reasons, to be longer or equal to the length of the critical path in graph G.

With the aid of algorithm 4.2 we can find a sub-set $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots \mathcal{O}_N\}$ of the set of possible schedulings, N being a chosen number. For each scheduling $\mathcal{O}_k$ through step f) of the algorithm the resources profiles written $\mathcal{S}_k = \{S_1, S_2, \ldots, S_g\}$ are ascertained. We must choose that scheduling $\mathcal{O}_k$ for which profiles $\mathcal{S}_k$ shall meet best the requirements of uniformity in using the resources; consequently, we must define criteria enabling such a choice. By setting a criterion we seek to define "the best" scheduling $\mathcal{O}_k$ out of set $\mathcal{O}_k$ for which the non-uniformity of profiles $\mathcal{S}_k$ is minimal.

5.2. Consider $E_1^i, E_2^i, \ldots, E_g^i$ to be values indicating the non-uniformity of resources $R_1, R_2, \ldots, R_g$ for profiles $\mathcal{S}_i$, in agreement with scheduling $\mathcal{O}_i$.

A levelling criterion is a rule for choosing $\mathcal{S}_i$ to be "the most uniform", contingent upon the non-uniformity indicators E^i.

Consider two schedulings $\mathcal{O}_i$ and $\mathcal{O}_j$. In agreement with profiles $\mathcal{S}_i$ and $\mathcal{S}_j$ we have indicators E_k^i, E_k^j $(k = 1, 2, .., g)$. Consider:

$$\xi_k = \begin{cases} 1 \text{ if } E_k^i < E_k^j \\ 0 \text{ if } E_k^i = E_k^j \\ -1 \text{ if } E_k^i > E_k^j \end{cases} \quad k = 1, 2, .., g$$

The levelling criterion is defined thus: profiles $\mathcal{S}_i$ are more uniform than profiles $\mathcal{S}_j$ if

$$\sum_{k=1}^{g} \xi_k > 0 \qquad (i)$$

and the best scheduling is $\mathcal{O}_i$.

If $\mu_k \in [0,1]$ is the degree of importance of resource R_k, the levelling criterion (i) becomes

$$\sum_{k=1}^{g} \mu_k \xi_k > 0 \qquad (ii)$$

In case we have equality $(=0)$ in relationships (i) and (ii), ruling criteria must be introduced, defining them in the same way as the levelling criterion with relationships (i) and (ii) but the non-uniformity indicators E_k^i should have been calculated in a different way. We can assume a sequence of ruling criteria before getting an inequality which is to decide which scheduling we can shall choose.

5.3. As non-uniformity indicators for profile S_k of a resource R_k we assume, for instance:

a) $$E_k = \sum_{j=1}^{T} [S_k(j)]^2$$

i.e. the sum of the squared requests regarding resource R_k for each time unit;

b) $$E_k = \sum_{j=1}^{T-1} \left| S_k(j+1) - S_k(j) \right|$$

i.e. the sum of absolute variations;

c) $$E_k = \max_{j \in [1,T]} S_k(j)$$

i.e. the maximum vertex of the profile,

d) $$E_k = \sum_{j=1}^{T} [f_u(j) - S_k(j)]^2$$

.e. the sum of the squared distances from the profile to the available f_k of resource R_k.

e) $$E_k = \sum_{j=1}^{T} h_{kj} \text{ where } h_{kj} = \begin{cases} S_k(j) - f_k(j); & \text{if } S_k(j) \geqslant f(j); \\ 0 & \text{if } S_k(j) < f_k(j) \end{cases}$$

i.e. the sum of the excess available S_k of resource R_k.

These non-uniformity indicators of a resource profile can be ranked according to the goal aimed at and to their importance. The first indicators shall be used in order to define the levelling criterion and the remaining ones shall be used for successive ruling criteria.

5.4. As a rule, a better solution will be obtained if a small number of resources is levelled and a big set of possible schedulings is explored.

The random generation of the starting terms for activities and projects enables a quick (scheduling) solution possible for the problem, and offers the possibility of drawing up a most diversified set of schedulings $\mathcal{O}$ with the aid of algorithm 4.2. for N times (N as big as possible); by checking the levelling (ruling) criterion, the best scheduling is chosen each time.

ADDENDUM

Finding the circuits in a graph

Consider a finite, directed graph $\mathbf{G}(X, U)$. A circuit is a finite path $\mu = [x_1^\mu, x_2^\mu, \ldots x_k^\mu]$ in which the initial node x^μ coincides with the terminal node x_k^μ, covering it in the sense of the arcs. In case there are circuits in the graph, the topological sorting of the arcs is not performed and the graph is amended.

With the aid of set U of the arcs of graph $\mathbf{G}$ we make up a table where on each arc of the graph the current number i, the predecessor node $I(i)$ and the successor node $J(i)$ will appear. We consider the tables of arcs classified in a sequentially increasing order after the antecedent node.

The algorithm for finding the circuits in a graph $\mathbf{G}$ is made up of the following steps:

Initializations: $k = 0$, $S(l) = 0$ $(l = 1, 2, \ldots, \lambda)$

1) We consider the first arc with the current number i for which $S(i) = 0$. If there is no arc with $S(i) = 0$, the algorithm is finished.

2) We look for the first arc j with $I(j) = J(i)$.

a) If such an arc j does not exist, go to 6);

b) If $S(j) = -1$, go to 5).

3) We make $k = k + 1$, $P(k) = I(i)$ and $i = j$

4) If $J(i) \in \{P(1), P(2), \ldots, P(k)\}$ we go to 7), otherwise go to 2).

5) We look for another arc j for which $I(j) = J(i)$.

a) If such an arc j exists, go to 2 b);

b) If such an arc j doesn't exist, then we make $k = k - 1$ ξ and

— if $k = 0$, go to 1);

— if $k \neq 0$, we look for the first arc j with $I(j) = P(k)$ and go to 2a).

6) We make $S(i) = -1$ and go to 5).

7) We print the circuit

$$P(l), P(l+1), \ldots, P(k), \text{ where } P(l) = J(i)$$

and go to 6).

REMARK. When we have to do with a circuit, because in step 6 the arc with the current number i is eliminated (that is, we make $S(i) = -1$, another circuit containing this eliminated arc shall no more be rendered evident.

REFERENCES

1. A. A. B. Pritsker, L. J. Watters, and P. M. Wolfe: *Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach.* Management Science, 16, 1, 1969.
2. R. Langer: *Ein mehrprojekt plannungsmodell bei beschrankten Ressourcen und variable Vorgangsdauer.* Rechentechnik Datenverarbeitung, 8, 1970.
3. L. S. Lasdon and R. C. Terjung: *An Efficient Algorithm for Multi-Item Scheduling.* Operations Research 19, 4, 1971.
4. Gh. Ciobanu: *Project Scheduling with Resource Allocation.* Economic Computation and Economic Cybernetics Studies and Research, No. 2, 1971.

MODELE DE PROGRAMMATION MULTIPROJET AVEC NIVELLEMENT DES RESSOURCES

Dans cet article, on présente un modèle de programmation multiprojet qui repose sur les méthodes d'analyse du chemin critique. Dans ce modèle le déroulement de chaque projet est continu et aussi court que possible, et pour l'ensemble des projets, les ressources sont nivellées. Le planning de durée des projets et des activités se fait à base de simulation, en générant de façon aléatoire les dates de commencement. On expose les bases mathémathiques de la programmation multiprojet, les algorithmes d'ordonnancement des projets et des activités, ainsi que les algorithmes de nivellement des ressources.

On présente, dans l'annexe, un algorithme pour la determination rapide des circuits dans un graphique-réseau.

МОДЕЛЬ МНОГОПРОЕКТНОГО ПРОГРАММИРОВАНИЯ С ВЫРАВНИВАНИЕМ РЕСУРСОВ

В настоящей статье приводится модель многопроектного программирования, имеющая в своей основе методы анализа критического пути, по которому развертывание каждого проекта должно быть непрерывным и наиболее коротким, а по совокупности проектов ресурсы должны быть выравнены. Планирование во времени проектов и видов деятельности производится на основе симулирования путем случайного установления сроков начала. Указываются математические основы многопроектного программирования, а также алгоритмы составления документов на выполнение проектов и видов деятельности и выравнивания ресурсов.

В приложении приводится алгоритм быстрого определения цепей в сетевом графике.