COPPEAD/UFRJ

RELATÓRIO COPPEAD Nº 1

"A MATHEMATICAL REFORMULATION OF THE CLASSICAL ASSEMBLY LINE BALANCING PROBLEM AND ITS OPTIMUM-EFFICIENT SOLUTION"

Jorge A. Garcia Gomez*
João Lizardo R. H. Araujo*

April 1976

* Universidade Federal do Rio de Janeiro

This paper and this research series were supported by the FINEP, an agency of the Brazilian Government.

I. INTRODUCTION

Since 1954 many papers have been published reporting the development of new techniques for balancing line production systems; all of them oriented toward solving the classical formulation of the assembly line balancing problem, due to Salveson. Most of these papers report good results in applying these techniques to hypothetical and industrial line balancing problems. However, since the classical formulation does not distinguish among problem solutions - in terms of workload distribution - all efficient algorithms in fact, tend to overload the first stations in order to reduce search effort, looking for minimization the number of work stations at the line. The solution formed are thus typically unbalanced: first stations overloaded and final ones idle. The first portion of the paper describes the assembly line balancing problem. This is followed by a criticism to the classical formulation and then our proposed formulation. In the third portion we present some results on the A* class of algorithms relevant to our case. This is followed by a description of the proposed algorithm, the design of the experiments, some experimental results and the conclusions.

II. THE ASSEMBLY LINE BALANCING PROBLEM

Given a decision to use line production, management must first decide on the work that is to be done by an assembly line. This work is then broken down into a set of "work elements", each with a "work element time". Requirements are specified that some work elements must "precede" others. These requirements must be mutually consistent, as fully specified by the partial ordering assumption in the mathematical formulation to follow A "station" is a subset of the set of work elements. A "production line" is an assignment of the work elements to a sequence of stations with each element assigned to precisely one station, such that,if one work element must precede another, then either it is assigned to an earlier station in the sequence or both work elements are assigned to the same station. The length of the production line is the number of stations, and its "cycle time" is the maximum of the sums of work element times for work elements assigned to each station.

The following are two intimately related optimization problems:

I. Given a "required cycle time", to find a production line of minimum length, subject to its cycle time not exceeding the required cycle time.

II. Given a "required line length", to find a production line with minimum cycle time, subject to its line length not exceeding the required line length.

Mathematically, the preceding can be condensed as follows: Let be given a finite nonempty set of work elements Ω ,a partial ordering $\prec$ defined on Ω (precedence relationships), a real valued function $t(x) > 0$ for x in Ω (work element times). A station is a subset of Ω . A production line is a partition of Ω into stations, $\Pi(\Omega) = (S_1, S_2, \ldots, S_N)$, such that if $x \prec y$ and x is in S_i while y is in S_j, then $i \leq j$. The length of this production line is $L(S_1, S_2, \ldots, S_N) = N$ and its cycle time is:

$$\theta' = \max_i \{T(S_i)\}$$

where

$$T(S_i) = \Sigma \{t(x) \mid x \varepsilon S_i\}$$

is the work content of station S_i.

Problem I. Given $\theta \geqslant \text{Max}\ \{t(x) \mid x \varepsilon \Omega\}$, to find a production line, $\Pi(\Omega) = (S_1, S_2, \ldots, S_N)$ for which N is a minimum, subject to $\theta' \leqslant \theta$

ProblemII. Given an integer $N_0 > 0$, to find a production line $\Pi(\Omega) = (S_1, S_2, \ldots, S_N)$, for which θ' is a minimum subject to $N \leqslant N_0$

Most line balancing techniques are designed to obtain "good approximations" to solutions for problem I; that is, to find a production line of nearly minimum lenght. However, the application of these techniques to problem II is a more suitable way to compare them. This is acomplished by applying them directly to problem I, with various required cycle times, and seeking a "nearly" minimum cycle time subject to a required line length. Problem II solutions constitute the more critical line balancing results because the better techniques produce Problem II solutions with smaller cycle times than do the poorer techniques. If no attempt were made to obtain problem II solutions, many of the techniques would produce identical results for the large number of required cycle times that are not the nearly minimum cycle times.

Idle time is the amount of time that resources are not utilized for each unit of product to be reproduced by a production line. The expression for idle time is:

$$IT = N\Theta - \Sigma_{i=1}^{i=N} T(S_i);$$

The optimal solution of either problem I or problem II also minimizes idle time for specified cycle times and line lengths. In the above equation for idle time, any reduction in the line length N for specified cycle time Θ decreases idle time. Similarly, idle time is decreased by any reduction in the cycle time Θ for specified line length N.

Now, perhaps due to its lesser complexity Problem I has been rather extensively studied while Problem II has lain neglected. In order to fully appreciate the consequences of this "state of affairs" let us review the classical formulation of Problem I, due to Salveson:

Given a set of tasks Ω, a partial ordering $\prec$ defined on Ω (technological precedence relations), a mapping $t:\Omega \to R^+$ (execution times) a constant $\Theta>0$ (maximum allowable cycle time for the line), the classical assembly-line balancing problem is defined as the problem of finding a collection $\Pi(\Omega)=(S_1, S_2, \ldots, S_N)$ of subsets of Ω satisfying the five following conditions:

i) $\cup_{i=1}^{i=N} S_i = \Omega$

ii) $i, j \in \{1, 2, \ldots, N\}, i \neq j \to S_i \cap S_j = \phi$

((i), (ii) imply that $\Pi(\Omega)$ partitions Ω)

iii) $T(S_i) \overset{d}{=} \Sigma \{t(x_j) \mid x_j \in S_i\} \leq \Theta, i=1,2,\ldots,N$

iv) $x < y,\ x \in S_i,\ y \in S_j \rightarrow i \leq J$

v) Idle time, defined as

$$IT = \Sigma_{j=1}^{j=N} (\Theta - T(S_j)),$$

is minimized.

(Note that (v) is equivalent to requiring minimization of N, since $IT = N\Theta - T^*$, where $T^* = \Sigma \{t_x \mid x \in \Omega\}$).

Many algorithms have been proposed to solve this problem; some (as for instance Gutjahr-Nemhauser's) use Dynamic Programming and other associated techniques to find an optimal solution, while others use ad-hoc heuristics to find a solution of these, the sole method that coul be modified to guarantee optimality of the solution is Nevins'. We can roughly subsume the present situation thus: the solutions produced by heuristic methods are not guaranteed optimal (with the above mentioned exception), while the methods based on systematic search are too ineffi- cient to use in medium to large-scale problems.

III. CRITICISM OF THE CLASSICAL FORMULATION

Observe that according to the classical formulation any solution of Problem I is as good as any other; although in principle we could choose among them using Problem II as yardstick, in practice no algorithm in the literature provides a way of doing this other than by generating all solutions. The situation is worsened by the fact that efficient algorithms (such as Nevins') tend to eliminate all but a small subset of solutions from consideration. Therefore, the best we could do would be to compare two or three solutions and choose one of them without the slightest guarantee that a much better one (in the sense of Problem II) could be found.

Thus, in many solutions produced the cycle time of the assembly line will be near or equal to the maximum allowable cycle time, even though a solution may exist with significantly reduced cycle time.

This same fact could be observed from another viewpoint: if the cycle time is far from optimal this means that the workload is very unevenly distributed among stations - while some are overloaded, others remain idle a good part of the time.

This situation is clearly undesirable from all points of view: the bottleneck in the overloaded stations hampers efficiency and avoidable inequality among workloads is both unjust and disruptiv of work relations.

These are not purely academic considerations: since the classical formulation does not distinguish among solutions, <u>all</u> efficient algorithms in fact tend to overload the first stations in order to reduce search effort. The solutions found are thus typically unbalanced: first stations overloaded, and final ones idle.

Small wonder that in practice most assembly lines are balanced using manual or semi-manual procedures.

It seems to us, therefore, that a reformulation of the assembly line balancing problem is needed in order to eliminate this difficulty. According to the preceding discussion, the real problem to be solved is:

I') Given Θ (maximum allowable cycle time), find a solution to problem I that has minimum cycle time.

II') Given N (Maximum allowable number of stations), find a solution to problem II that has minimum number of stations.

More formally, let $\Pi(\Omega)$ be any line balancing; i.e., $\Pi(\Omega)$ satisfies

i) $\Pi(\Omega)=(S_1, S_2, S_3, \ldots, S_N)$; $S_i \subset \Omega$, $i=1,2,3,\ldots,N$;

$\cup_{i=1}^{i=N} S_i=\Omega$.

ii) For all, $j\varepsilon\{1,2,\ldots,N\}$, $i\neq j \rightarrow S_i \cap S_j = \phi$

iii) For all $x,y\varepsilon\Omega$, $x \lessdot y$, $x\varepsilon S_i$, $y\varepsilon S_j$, $\rightarrow i \leqslant j$

and let $N_\Pi=N$, $\Theta_\Pi=\max\ \{T(S_i)\ |\ i=1,2,\ldots, N\}$

be the number of stations and cycle time of the line respectively. If $A(\Omega)=\{\Pi(\Omega)\ |\ (i), (ii), (iii)\}$, the problem (I') is then:

IV) Given Θ, find $\Pi^*(\Omega)$ having $N_{\Pi^*} = N^*$, $\Theta_{\Pi^*} = \Theta^*$,

$$\text{where} \begin{cases} \Theta^*=\min\{\Theta_\Pi\ |\ \Pi(\Omega)\ \varepsilon\ A(\Omega),\ N_\Pi \leqslant N^*\} \\ \\ N^*=\min\{N_\Pi\ |\ \Pi(\Omega)\ \varepsilon\ A(\Omega),\ \Theta_\Pi \leqslant \Theta\} \end{cases}$$

and problem (II') is:

V) Given N, find $\tilde{\Pi}(\Omega)$ having $N_{\tilde{\Pi}} = \tilde{N}$, $\Theta_{\tilde{\Pi}} = \tilde{\Theta}$,

$$\text{where}\begin{cases}\tilde{N} = \min \{N_{\Pi} \mid \Pi(\Omega) \in A(\Omega),\ \Theta_{\Pi} \leqslant \tilde{\Theta}\} \\ \\ \tilde{\Theta} = \min \{\Theta_{\Pi} \mid \Pi(\Omega) \in A(\Omega),\ N_{\Pi} \leqslant N\}\end{cases}$$

These two problems are related in a very intimate way. To see this, let us denote $N^*(\Theta)$ and $\Theta^*(\Theta)$ the length and the cycle time of the solutions to problem (I') given Θ, and $\tilde{N}(N)$, $\tilde{\Theta}(N)$ their counter parts in problem (II') given N. We can enounce:

Lemma 1:

Let

$$S^*(\Theta,\Omega)=\{\Pi(\Omega) \mid \Pi(\Omega) \in A(\Omega),\ N_{\Pi}=N^*(\Theta), \Theta_{\Pi} = \Theta^*(\Theta)\}$$

be the set of solutions to (I'). Then

$$S^*(\Theta,\Omega) \subset A^*(\Theta,\Omega),$$

where

$$A^*(\Theta,\Omega)=\{\Pi(\Omega) \mid \Pi(\Omega) \in A(\Omega),\ \Theta_{\Pi} \leqslant \Theta,\ N_{\Pi} \leqslant N^*(\Theta)\}$$

Proof:

The result comes at once from the definition of $N^*(\Theta)$, keeping in mind that $\Theta^*(\Theta) \leqslant \Theta$.

We can analogously prove

Lemma 2:

Let

$$\tilde{S}(N,\Omega)= \{\Pi(\Omega) \mid \Pi(\Omega) \in A(\Omega),\ N_{\Pi}=\tilde{N}(N),\ \Theta_{\Pi}=\tilde{\Theta}(N)\}$$

be the set of solutions to (II'). Then

$$\tilde{S}(N,\Omega) \subset \tilde{A}(N,\Omega),$$

where

$$\tilde{A}(N,\Omega)=\{\Pi(\Omega) \mid \Pi(\Omega) \in A(\Omega),\ N_{\Pi} \leqslant N,\ \Theta_{\Pi} \leqslant \tilde{\Theta}(N)\}$$

and now we can enounce and prove the following proposition:

<u>Proposition 1</u>:

i) Given Θ, $S^*(\Theta,\Omega)=\tilde{S}(N^*(\Theta),\Omega)$

ii) Given N, $\tilde{S}(N,\Omega)=S^*(\tilde{\Theta}(N),\Omega)$

<u>Proof (i)</u>:

we have

$$\tilde{A}(N^*(\Theta),\Omega)=\{\Pi(\Omega) \mid \Pi(\Omega) \in A(\Omega),\ N_{\Pi} \leqslant N^*(\Theta),$$
$$\Theta_{\Pi} \leqslant \tilde{\Theta}(N^*(\Theta))\}$$

Now, substituing $N^*(\Theta)$ for N in the definition of $\tilde{\Theta}(N)$, comes $\tilde{\Theta}(N^*(\Theta))=\Theta^*(\Theta)$. Therefore

$$\tilde{A}(N^*(\Theta),\Omega)=\{\Pi(\Omega) \mid \Pi(\Omega) \in A(\Omega),\ \Theta_{\Pi} \leqslant \Theta^*(\Theta),\ N_{\Pi} \leqslant N^*(\Theta)\}$$
$$=A^*(\Theta^*(\Theta),\Omega) = S^*(\Theta,\Omega)$$

From lemma 2, we may write

$$\tilde{S}(N^*(\Theta),\Omega) \subset S^*(\Theta,\Omega).$$

But this gives

$$\tilde{N}(N^*(\Theta)) = N^*(\Theta)$$

and, using the definition of $\tilde{S}(N,\Omega)$, we have:

$$\tilde{S}(N^*(\Theta),\Omega) = S^*(\Theta,\Omega)$$

The second item is proved in exactly the same way, interchanging problems (I') and (II'), as well as lemma 1 for lemma 2. This completes the proof.

Q.E.D.

<u>Corollary 1:</u>

i) given Θ, there exist $N_0 = N^*(\Theta)$, $\Theta_0 = \Theta^*(\Theta)$ subject to $\tilde{N}(N_0) = N_0 = N^*(\Theta_0)$, $\tilde{\Theta}(N_0)=\Theta_0=\Theta^*(\Theta_0)$, and the pair (N_0,Θ_0) is not altered if we replace Θ by any $\Theta' \varepsilon [\Theta_0,\Theta]$.

ii) given N, there exist $N_0=\tilde{N}(N)$, $\Theta_0 = \tilde{\Theta}(N)$ subject to $\tilde{N}(N_0)=N_0=N^*(\Theta_0)$, $\tilde{\Theta}(N_0)=\Theta_0=\Theta^*(\Theta_0)$, and the pair (N_0,Θ_0) is not altered if we replace N by any $N' \varepsilon [N_0,N]$.

<u>Proof:</u>

immediate from proposition 1.

Corollary 2:

For every pair (N,Θ) such that $N>N^*(\Theta)>\tilde{N}(N)$ or, equivalently, $\Theta>\tilde{\Theta}(N)>\Theta^*(\Theta)$, we have $S^*(\Theta,\Omega)=\tilde{S}(N,\Omega)$.

Proof:

the equivalence comes at once from the definitions of problems (I') and (II'). We have thus

$$\tilde{N}(N)\leq N^*(\Theta)\leq N$$

$$\Theta^*(\Theta)\leq\tilde{\Theta}(N)\leq\Theta$$

and from corollary 1, comes:

$$\tilde{\Theta}(N)=\tilde{\Theta}(N^*(\Theta))=\Theta^*(\Theta)=\Theta^*(\tilde{\Theta}(N))$$

$$\tilde{N}(N)=\tilde{N}(N^*(\Theta))=N^*(\Theta)=N^*(\tilde{\Theta}(N))$$

substituting in the definitions of $S^*(\Theta,\Omega)$ and $\tilde{S}(N,\Omega)$ the conclusion comes at once.

Q.E.D.

In other words, every solution of problem (I') given Θ is also a solution to problem (II') given an appropriate value of N.

This formulation is thus very attractive; nevertheless, these problems seem operationally harder to solve than either problem (I) or (II), as they are in fact combinations of both.

We propose to do this by adding to the classical formulatic of problem (I) a restriction relative to the distribution of workloads among the stations.

In more precise terms, let us note as $\tilde{S}$ the set of assignments satisfying the conditions (i) to (v) of problem (I), i.e., the set of optimal solutions to the classical problem.

Let us now assume given a convex increasing funcion $\ell: R^+ \to R^+$; we may now define, for each admissible assignment $\Pi(\Omega) = (S_1, S_2, \ldots, S_N)$, the <u>disequilibrium</u> in the sense of $\ell(.)$ as

$$D(\ell,\Pi)=\Sigma_{i=1}^{i=N} \ell(\Theta - T(S_i))$$

Note that $D(\ell,\Pi) > N(\Theta - T^*/N)$; if $\ell(.)$ is strictly convex, w shall have equality if, and only if, $T(S_i) = T^*/N$ for all i. An importan case is $\ell(.)=(.)^m$, $m > 1$; We shall then note $D(\ell,\Pi)$ as $D_m(\Pi)$.

We shall then say that a solution to the assembly-line balancing problem is <u>optimal in the sense of ℓ</u> (or ℓ-optimal) if $\Pi(\Omega$ satisfies Salveson's conditions (i) to (v), and also

$$\text{(vi) } D(\ell,\Pi) = \min\{D(\ell,\Pi') \mid \Pi'(\Omega) \in \tilde{S}\} = D^*(\ell)$$

in other words, if $\Pi(\Omega) \in \tilde{S}$ and $D(\ell,\Pi) = D^*(\ell)$.

It is easy to see that, if $\ell(.) = (.)$ (linear disequilibriu we obtain Salveson's formulation and that, for $\ell(.) = (.)^m$ with large m, we get problem (I'). Besides, every ℓ-optimal solution is also optimal for the classical problem.

Note also that, if $\ell(.)$ is strictly convex, an ℓ-optimal solution tends to have an approximately uniform distribution of workloads, even for fairly smooth $\ell(.)$; for instance, quadratic disequilibrium ($\ell(.)=(.)^2$) gives almost uniform solutions.

IV. SOME RESULTS ON THE A* CLASS ALGORITHM

The algorithm that we propose is based on the theory of heuristic search formalized by Hart et al., i.e., the theory of the A* algorithm. This is in fact intimately related to branch-and-Bound algorithms, and there has been some argument over whether they are not, in a way, equivalent. Without entering the merit of this discussion, we shall only note that Hart et al. have obtained stronger results (the "optimality" theorem) than classical branch-and-Bound theory. It so happens, as we shall see, that for the reformulated Classical Assembly Line Balancing Problem all of Hart's optimality conditions are fulfilled; before going on, however, we shall present the theoretical results most pertinent to our case.

IV.1 - Problem Graphs

Let us assume that we can formulate a problem in the following way: we know a finite set S of initial states, can recognize a set T of desired states (terminal states), and dispose of a (finite) set Γ of operations on states; that is, if x is a state of the problem and $\chi\varepsilon\Gamma$ is a valid operation on x, then $\chi x=(y,c)$ where y is a state of the problem and c is the cost of applying χ to x.

Under these conditions, the graph (weighted and directed) $G_s=(S,\Gamma)$, defined by

i) S is contained in the set of nodes of G_s, noted $V(G_s)$.

ii) $n_i \varepsilon V(G_s)$, $\chi\varepsilon\Gamma$, and $\chi n_i = (n_j, c_{ij}) \rightarrow n_j \varepsilon V(G_s)$ and c_{ij} is the cost of the arc $n_i n_j$,

is called graph generated by S and Γ, in the case (ii) above n_j is called <u>successor</u> of n_i. Solving a problem may then be viewed as partially exploring G_s.

IV.2 - Admissibility

We shall only consider graphs with arc-costs superior to some positive number δ (if G_s is finite, this condition may be relaxed; we shall require only that there be no circuit with negative cost), and we shall refer to them as δ-graphs. We shall further assume that costs are additive, i.e., if $\gamma=(n_0, n_1, \ldots, n_k)$, where n_{i+1} is a successor of n_i, is a path from n_0 to n_k it will have a cost $k_\gamma\ (n_0, n_k) = \Sigma_{i=1}^{k}\ c_{i-1,i}$. We note $k(n_0, n_k)$ for the minimum cost among all paths from n_0 to n_k. Some particu lar notations are also useful:

if n is a node of G_s we write

$$g(n) = \min_{s \varepsilon S} k(s,n) \quad ; h(n) = \min_{t \varepsilon T} k(n,t) \quad ; h(S) = \min_{s \varepsilon S} h(s)$$

the minimum cost of a path from S to T constrained to go through a node n is noted $f(n) = g(n) + h(n)$.

A <u>solution</u> to the problem is a path from S to T. It is said to be <u>preferred</u> if its cost is $f(S)$. A search algorithm is said to be <u>admissible</u> if, for every δ-graph having a finite solution, it can guarantee finding a preferred solution in a finite number of steps.

IV.3 - Heuristic Search. The A* Algorithm

We shall take "heuristic search" to mean the use of additional in formation about the problem to help guide the search. A good discussion of this may be found in Nilsson (1971). For our present purpose we need only assume that for every node n, already visited or successor to a visited node, we can calculate a "merit function" $\hat{f}(n)$, which may depend on external information as well as on the status of the search.

For every node we store

i) State description

ii) Flag F=0 (open) or 1 (closed)

iii) Value of $\hat{f}$

iv) Pointer P to its "best" predecessor

(the meaning of F and P will be made clear below)

A* Algorithm

```
1. For all s in S, calculate and store $\hat{f}(s)$ and do $F(s)=0$, $P(s)=nil$;
2. Choose the open node n having best merit. Resolve ties in any way you like,
   but always in favor of nodes in T, if there is no open node, exit with failure
3. If $n \in T$ do $F(n)=1$ and stop. The solution is obtained following the pointers
   backwards.
4. Otherwise do $F(n)=1$ and generate all successors to n; for each such m calculate
   $\hat{f}(m)$. If m is a new node or if the new value of $\hat{f}(m)$ is better than the old
   one, do $P(m)=n$, $F(m)=0$ and store $\hat{f}(m)$.
5. Go to 2.
```

In this algorithm, $\hat{f}(n)$ is usually an estimate of $f(n)=g(n)+h(n)$. An important special case is $\hat{f}(n)=\hat{g}(n) + \hat{h}(n)$, where $\hat{g}(n)$ is the least cost, found so far, of a path from S to n; $\hat{h}(n)$ is an estimate of $h(n)$. We must then store both $\hat{g}$ and $\hat{h}$, and do the test in step 4 on $\hat{g}$.

In this case we have the following theorem:

Theorem 3.1 (Hart et al.):

If $\hat{f}(n)=\hat{g}(n) + \hat{h}(n)$, and $\hat{h}(n) \leqslant h(n)$ for all nodes in G_s, then A* is admissible. Furthermore, only nodes having $\hat{f}(n) \leqslant f(s)$ are closed. This result is immediately extensible to finite graphs with non-negative arc-costs.

If, besides $\hat{h}(n) \leqslant h(n)$ for all n in G_s, the estimates also satisfy $\hat{h}(m) - \hat{h}(n) \leqslant k(m,n)$ for all m,n in G_s ("consistency"), the following results are true:

P.3.1. If $\hat{h}$ is consistent, A* only closes a node when an optimal path to it has been found (Hart)

P.3.2. If $\hat{h}$ is consistent, and the sequence of nodes closed by A* is $(N_1, \ldots, n_r)$

$$i<j \rightarrow \hat{f}(n_i) < \hat{f}(n_j). \quad \text{(Hart)}$$

IV.4 - Optimality

Let us introduce another concept: we shall say that an admissible algorithm A_1 is no more informed than another admissible algorithm A_2 if the information each of them disposes of allows lower bound estimates of h(n) for all n (respectively not $\hat{h}_1(n)$ and $\hat{h}_2(n)$) such that $\hat{h}_1(n) \leqslant \hat{h}_2(n) \leqslant h(n)$ for all n. We have then Hart et al's "Optimality Theorem":

Theorem 3.2.

Assume that A* uses a consistent $\hat{h}$, and let A be any admissible algorithm no more informed than A*. Then

(i) If the sequence of nodes closed by A* till stopping is $(n_1,\ldots,n_R)$ and Ec is the number of ties in step 2 with $\hat{f}(n)=f(s)$ (critical ties), calling N_A the number of nodes closed by A we have $N_A \geqslant R - Ec$

(ii) There is a tie - breaking rule for A* such that every node closed by A* will also be closed by A. (Hart)

V. RESOLUTION OF THE REFORMULATED C.A.L.B.P.

Initially, we define the problem graph:

STATES: A state $X=(X_1, X_2, \ldots, X_N)$ of the CALBP is a sequence of subsets from Ω, satisfying

(i) $X_i \cap X_j = \emptyset$ for $i \neq j$ and $X_i \neq \emptyset$ for all i

(ii) $T(X_i) \overset{d}{=} \sum_{x \in X_i} t(x) \leqslant \Theta$ for all i

(iii) for all $x,y \in \Omega$ and $x<y \rightarrow (x \in X_i, y \in X_j, j>i)$

$\text{or} (y \in \{\Omega - \cup_{i=1}^{i=N} X_i\})$

We convene that the empty list is an state

SUCCESSORS: A state $Y=(Y_1, Y_2, \ldots, Y_m)$ is called successor of the state $X=(X_1, X_2 \ldots, X_N)$ if, and only if:

(i) $M=N+1$

(ii) $Y_i = X_i$ for $i \leqslant N$

The transition cost C_{xy} is defined as

$$C_{xy} = \ell(\Theta - T(Y_{N+1})$$

Note that C_{xy} is always nonnegative.

INITIAL STATE:

The initial state s is the empty sequence (a sequence without elements)

TERMINAL STATE: A state is called terminal if it is a partition of Ω; And it is called preferred if the number of components of that partition is minimum.

The problem is then reduced to finding a minimum cost path between s and the set of preferred states. Note that G_s has the following properties [2]:

P.4.1 - G_s is an anti-symmetric tree. Thus, there exists just one path from s to any state n (whence $\hat{g}(n)=g(n)$), and at most one path from state n to any other state m.

P.4.2 - If we associate N^*_x (minimum length of a solution passing through X) to each state $X=(X_1, X_2,\ldots,X_N)$, and if $Y=(Y_1, Y_2,\ldots,Y_N)$ is such that

$$\bigcup_{i=1}^{i=N} Y_i = \bigcup_{i=1}^{i=N} X_i \quad \text{and} \quad g(X)\leqslant g(Y),$$

then $N^*_x=N^*_y$ and $f(X) \leqslant f(Y)$. In this case, we say that Y is dominated by X.

In this way, we define for all state $X=(X_1, X_2, \ldots, X_N)$

$$\hat{h}(x) \overset{d}{=} (\hat{N}(x) - N).\ell(\Theta - \frac{T^* - T(x)}{\hat{N}(x) - N})$$

Where
$$T(x) = \sum_{i=i}^{N} T(x_i)$$

and
$$\hat{N}(x) = N + \left\lceil \frac{T^* - T(x)}{\Theta} \right\rceil^+$$

Observe that the estimatives $\hat{N}(x)$ and $\hat{h}(x)$ have the following properties [2]

i) $\hat{N}(x) < N^*(x)$ for all x

ii) $Y \varepsilon \Gamma(x) \rightarrow \hat{N}(y) > \hat{N}(x)$

iii) $\hat{h}(x) < h(x)$ for all x

iv) $Y \varepsilon \Gamma(x) \rightarrow \hat{h}(y) + C_{xy} > \hat{h}(x)$ (consistency)

VI. THE A^*_{CALBP} ALGORITHM

The Algorithm for the CALBP is therefore as follows:

For each state $X = (X_1, \ldots, X_N)$, we store:

(i) description of X

(ii) $g(X)$, $N(X)$, $T(X)$

(iii) $\hat{N}(X)$, $\hat{h}(X)$, $\hat{f}(X)$

(iv) pointer $P(X)$ to its predecessor

(v) Flag $F(X) = 0$ (open) or 1 (closed)

1. Initialization: let s be the empty list; store s and do :

$$F(s) \leftarrow 0,\ g(s) \leftarrow 0,\ N(s) \leftarrow 0,\ T(s) \leftarrow 0,\ P(s) \leftarrow \emptyset;$$

Compute $\hat{N}(s) \leftarrow \left\lceil \frac{T^*}{\Theta} \right\rceil^+$, $\hat{h}(s) \leftarrow \hat{f}(s) \leftarrow \hat{N}(s) \cdot \ell(\Theta - \frac{T^*}{\hat{N}(s)})$;

let $\bar{N} \leftarrow \hat{N}(s)$.

2. Selection: Among all open nodes Y having $\hat{N}(Y) = \bar{N}$, choose an X with minimum $\hat{f}(X)$, resolving ties in favor of nodes with $\hat{N}(X) = N(X)$ and arbitrarily otherwise; if there are no open nodes exit with failure.

3. $F(X) \leftarrow 1$. If $\hat{N}(X) = N(X)$ exit with success (The solution is obtained following the pointers). Else generate ΓX.

4. For every $Y \in \Gamma X$ evaluate $T(Y_{N+1})$, $C_{XY} \leftarrow \ell(\Theta - T(Y_{N+1}))$; do $N(Y) \leftarrow N(X)+1$, $g(Y) \leftarrow g(X)+C_{xy}$, $T(Y) \leftarrow T(X)+T(Y_{N+1})$; let $\Delta N_Y \leftarrow \left\lceil \frac{T^*-T(Y)}{\Theta} \right\rceil^+$ and do $\hat{N}(Y) \leftarrow N(Y)+\Delta N_Y$, $\hat{h}(Y)\Delta N_Y \cdot \ell\ (\Theta - \frac{T^*-T(Y)}{\Delta N_Y})$, $\hat{f}(Y) \leftarrow g(Y)+\hat{h}(Y)$, $P(Y) \leftarrow X$;

   Now test for dominances: if there exists an open node Z that do minates Y, do not store Y; else store Y, do $F(Y) \leftarrow 0$ and eliminate any nodes dominated by Y; take next element in ΓX and repeat (4) until exhausted.

5. Update $\bar{N} \leftarrow \min \{\hat{N}(X) \mid F(X) = 0\}$;
   Return to 2

Observe that the restriction to nodes having $\hat{N}(X) = \bar{N}$, in step (2), guarantees we will stop at a desired state, since $\hat{N}(X) \leqslant N^*(X)$. Besides, the solution will minimize disequilibrium (else let $\bar{N}_f$ and $\hat{f}_f$ be the length and disequilibrium of the solution reached; if this is not optimal there is an open node X having $N^*(X) = \bar{N}_f$ and $f(X) < \hat{f}_f$ which was not selected; this is impossible since $\bar{N}(X) \leqslant N^*(X)$ and $\hat{h}(X) \leqslant h(X)$ for all X).

On the other hand, notice that both $\Delta N_y (< N^*(Y) - N(Y))$ and $\hat{h}(Y)$ satisfy the consistency assumption (from properties (ii) and (iv)). This allows the conclusion that this algorithm is Hart-optimal for the reformulated CALBP among all <u>admissible</u> algorithms using no more information than:

(i) cycle time Θ

(ii) $T^* = \Sigma\ \{t_x \mid x \varepsilon \Omega\}$

(iii) For every state $X = (X_1, \ldots, X_N)$,

i.i.i.a. $N(x)=N$ (number of stations in x)

i.i.i.b. $T(X_i)=\Sigma\{t_x | x \ \varepsilon\ X_i\}$, $i=1,2,\ldots,N$

VII. THE EXPERIMENTS

The efficiency of a search algorithm oriented towards solving the CALBP depends strongly on the structure of the precedence graph, as well as on the problem cycle time. In order to account for these factors and test more rigorously the proposals contained in this work two different kinds of experiments, i.e., precedence graphs, were selected, to wit

a) networks extracted from the literature and

b) networks generated through pseudo-random methods.

The networks extracted from the literature represent well-known problems, carrying the name of the researchers who designed them for the purpose of testing the performance of the algorithms proposed by them in certain pathological cases. Among the networks taken from the literature stand out Jackon's 11 element network, Gutjahr and Nemhauser's 10 element network and Tonge's 21 element network.

The networks generated through pseudo-random mechanisms were included in order to obtain experimental results also for non-pathological cases.

The principal parameters of each network were generated as follows: the number of elements from a uniform distribution, the task times from another uniform law, and the successors to each task from repeated Bernoulli urns.

For large scale problem it is often necessary to prune the search tree in order to lessen the storage needed to solve a problem. With the purpose of testing the sensitivity of the algorithm with respect to pruning we limited the number of successors per node to three, these being selected, after their generation, in increasing order of deviation - that we define as the absolute value of the difference between the workload assigned to the last station and the average workload T*/N - in order to select nodes leading to more uniform distribution of idle time. This pruning criterium revealed an out-standing property: in no case the pruning affected the solutions arrived at; this was true also for the Nevins algorithm, which we selected for comparison with the A^*_{CALBP}.

From the many heuristic algorithms proposed for the CALBP (before this date) the only one that employs, in a way, the concepts of ordered search is Nevins. In his paper Nevins reports a series of experiments for comparing his algorithm with the main existing methods at the time (1972), showing his method to be more efficient than any of the others; for this reason we selected it alone for the purpose of comparison. It is worth observing that we used for Nevins the same pruning criterium as in the A^*_{CALBP} to maintain comparability.

VIII. PERFORMANCE INDEXES

To evaluate the quality of solutions obtained we designed several performance indexes. The first is the quadratic disequilibrium D_2 (Π) as defined in (§2), generated by the solution $\bar{X}$ found; here

$$\bar{D} = D_2 (\Pi) = \Sigma_{j=1}^{j=\bar{N}} (\Theta - T(\bar{X}_j))^2, \qquad \bar{X} = (\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_{\bar{N}})$$

Since $\bar{D} > \bar{N} (\Theta - T^*/\bar{N})^2 = D^*$ and as $\ell(.) = (.)^2$ is strictly convex, equality occurs if and only if $T(\bar{X}_j) = T^*/\bar{N}$ for all j. We may thus consider $\bar{D} - D^*$ (or $\bar{D}/D^* - 1$) as an index of the additional disequilibrium introduced by $\bar{X}$ over the lower bound D^*.

A second index is the difference between the number of stations of the solution $\bar{X}$, found, denoted by $\bar{N}$, and the theoretical lower bound $N_0^* = \lceil T^*/\Theta \rceil^+$. Through this index we may evaluate the performance of algorithms in problems (precedence graphs) for which the lower bound is not attainable.

The number of generated nodes G refers to all nodes analysed by the algorithm as possible candidats for expansion, and not to those effectively stored; since we effect a dominance test as described in (P.42), the nodes dominated by some other node are deleted. The number of nodes effectively stored by the end of the search is denoted by G'. The difference G - G' is thus an index os selectivity for the dominance criterion.

The number of closed nodes T gives an approximate measure of the effort expended to find the solution. We did not consider computer time as a performance criterium, since it reflects longely the programmer's ability and machine characteristics, thus making comparison between two algorithms difficult.

IX. RESULTS

					A^*_{CALBP}				NEVINS			
PROBLEM	Θ	N^*_0	$\bar{N}$	D^*	$\bar{D}$	G	G'	T	$\bar{D}$	G	G'	T
JACKSON (11 elements)	7	7	8	12.5	20	149	52	47	22	88	37	25
	9	6	6	10.7	14	123	47	26	26	195	46	39
	10	5	5	3.2	6	49	29	9	10	43	17	8
	11	5	5	16.2	19	57	36	9	51	22	22	6
	14	4	4	25.0	26	54	38	6	42	24	23	4
	15	4	4	49.3	50	61	42	6	126	31	32	4
	19	3	3	40.3	41	60	47	4	101	38	31	3
	21	3	3	96.3	97	64	48	4	289	37	37	3
	22	3	3	133.3	134	70	50	4	328	39	39	3
	23	2	2	0.0	0	45	45	2	0	42	4	2
GUTJAHR & NEMHAUSER	6	7	8	12.5	20	72	34	32	20	47	29	23
	7	6	7	17.3	23	57	32	21	29	38	27	15
	8	5	5	0.8	4	21	20	6	4	27	11	8
	9	5	5	9.8	13	33	27	7	37	20	20	5
	10	4	5	28.8	32	45	31	9	56	48	25	11

X. ANALYSIS OF RESULTS

The following observations were verified both in experiments with punning and without it; therefore we shall not discriminate between them.

X.1 - With Respect To The Mathematical Reformulation

i) $\bar{D}$ (A^*_{CALBP}) < $\bar{D}$ (Nevins) in <u>all</u> experiments (often remarkably smaller); thus A^*_{CALBP} <u>consistently</u> generates assembly lines that are more efficiently balanced, which comes from using a more selective objective function. This difference tends to grow for large cycle times. The explanation lies in that in this case the number of solutions having minimal idle time increases greatly and Nevins' criterium tends to prefer solutions having idle time concentrated in the last work stations.

ii) $\bar{N}$ (A^*_{CALBP}) $\equiv$ $\bar{N}$ (Nevins) in <u>all</u> experiments; both methods obtained solutions having minimum number of stations (that is to say, minimum idle time); Thus A^*_{CALBP} finds solutions that are both <u>optimal</u> in the traditional sense and <u>most efficiently</u> balanced.

X.2 - With Respect To The Search

i) $G(A^*_{CALBP}) > G(\text{Nevins})$. The total number of nodes generated by A^*_{CALBP} was slightly larger than that by Nevins. This is due mainly to the different dominance criterion used; thus Nevins' algorithm eliminates all nodes that lead to an idle time to that of some node already stored (compare the G' for both algorithms). This greater selectivity is attained at the cost of the quality of the solution, as we saw. Even in the most unfavorable cases, nevertheless, $G(A^*_{CALBP}) < 2G$ (Nevins); the selectivity of the heuristics used by the A^*_{CALBP} longely compensates its weaker dominance criterion.

ii) In what concerns the number of nodes closed we may repeat the observations above. Pruning, although it did not affect the quality of solutions for all experiments, did reduce both storage needs and number of closed nodes for the larger problems.

XI. CONCLUSIONS

XI.1 - With Respect To The Problem Reformulation

The objective function proposed was shown to be an optimizing modification for the solution of the CALBP in the sence of discriminating, among all solutions with minimum idle time, those having minimum dispersion of workload among stations (in other words, the best balanced solutions)

ii) Since the reformulation demands minimum idle time (or, equivalently, minimun number of stations), and solutions having least disequilibrium tend to have least real cycle time Θ', A^*_{CALBP} is the only algorithm in the litterature that attempts to solve simultaneously problems (I) and (II).

XI.2 -.With Respect To The A^*_{CALBP} Algorithm

i) A^*_{CALBP} is optimal (in Hart's sense) for solving the reformulated classical assembly line balancing problem among all admissible algorithms using no more information than the following.

i.1) Cycle time Θ

i.2) $T^* = \Sigma\{t_x \mid x \in \Omega\}$

i.3) for each state X $(X_1, X_2, \ldots, X_N)$

i.3.1. $N(X) = N$ (number of stations in X)

i.3.2. $T(X_i) = \Sigma\{t_x \mid x \in X_i\}$, $i=1,2,\ldots,N$

ii) A^*_{CALBP} represents a class of algorithms oriented towards solving the CALBP, among which is a modification of Nevins' algorithm - corresponding to linear disequilibrium (idle time) as the objective function. For each objective function we have a subclass of algorithms whose members are distinguished one from the other according to the way in which they resolve ties.

iii) The pruning, effected according to the criteria exposed in this paper, <u>did not affect at all</u> the optimality of solutions obtained through the A^*_{CALBP}. This criterium thus seems reliable enough to use in large scale problems.

REFERENCES

1. ARAÚJO,J.,L., "Evaluating and Acelerating Heuristic Search: A Proposal", Technical Report Nº 1.74, Department of Systems Engineering, Universidade Federal do Rio de Janeiro, January 1974.

2. GARCÍA, J. A., "Uma Nova Formulação do Problema Clássico de Balanceamento de Linhas de Montagem e Sua Solução Ótima e Eficiente", D.Sc. Dissertation, Programa de Engenharia de Produção, Universidade Federal do Rio de Janeiro, June 1975.

3. GUTJAHR,A.L., AND, NEMHAVSER,G.L., "An Algorithm for the Line Balancing Problem" , Management Science, Vol.11, (November 1964).

4. HART,P., NILSSON,N.J., RAPHAEL,B., "A Formal Basis for The Heuristic Determination of Minimal Costs Paths", IEE TSC, Vol.1, SSC-4, Nº 2(1968), p. 100-107.

5. JACKSON,J.R., "Computing Procedure for Line Balancing Problems",Management Science, Vol.2, Nº3 (April 1956).

6. NEVINS,A.J., "Assembly Line Balancing Using Best Bud Search", Management Science, Vol.18, Nº9 (May 1972).

7. SALVESON,M.E., "The Assembly Line Balancing Problem", Journal of Industrial Engineering, Vol.6, nº3 (May-June 1955)