De (2) et (3), il résulte l'existence du chemin $\gamma[s_0^*, n_{s_0^*}]$: contradiction avec (1), et le lemme est démontré.

Théorème 1. *Si le digraphe G est semi-fortement connexe, alors G possède un chemin pré-hamiltonien.*

Démonstration. Nous allons maintenant faire usage du lemme 2. Soit n_{i_1} un sommet de $\mathcal{N}$ dont tous les sommets de $\mathcal{N}$ sont des descendants. Soit n_{i_2} un sommet de $\mathcal{N} - \{n_{i_1}\}$ dont tous les sommets de $\mathcal{N} - \{n_{i_1}\}$ sont des descendants, etc. On obtient ainsi un classement n_{i_1}, $n_{i_2}, \ldots, n_{i_{p-1}}, n_{i_p}$ des sommets de $\mathcal{N}$, pour lequel $n_{i_{j+1}}$ est un descendant de n_{i_j} pour $j = 1, 2, \ldots, p-1$. Il existe donc un chemin allant de n_{i_j} à $n_{i_{j+1}}$, pour $j = 1, 2, \ldots, p-1$. La réunion de ces chemins constitue un chemin pré-hamiltonien de G.

Théorème 2. *Un digraphe semi-fortement connexe peut être rendu fortement connexe en lui ajoutant un seul arc.*

Démonstration. D'après le théorème 1, si le digraphe G est semi-fortement connexe, il possède un chemin pré-hamiltonien $\gamma_H = [n_{i_1}, \ldots, n_{i_2}, \ldots, n_{i_{p-1}}, \ldots, n_{i_p}]$. En lui ajoutant l'arc $<n_{i_p}, n_{i_1}>$, le chemin γ_H devient un circuit pré-hamiltonien, et, d'après le lemme 1, le digraphe G est rendu fortement-connexe.

BIBLIOGRAPHIE

1. B e r g e C. — *Théorie des graphes et ses applications*, Dunod, Paris, 1962.
2. M a r c u D. — *Un algoritm de determinare a componentelor tari conexe ale unui graf orientat finit*, Studii şi cercetări matematice, 28, 1 (1976).
3. M a r c u D. — *O aplicaţie a spaţiilor vectoriale la calculul numărului de componente conexe ale unui graf orientat finit*, Studii şi cercetări matematice, 28, 2 (1976).

Reçue 22 III. 1977

Université de Bucarest
Faculté de Mathématiques et Mécanique
R.S. România

Analele ştiinţifice ale Universităţii „Al. I. Cuza" Iaşi
Tomul XXIV, S. I a, f. 1, 1978

THE ASSIGNMENT PROBLEM WITH SOME COMPATIBLE JOBS

BY

PIERRE HANSEN, LEON KAUFMAN and MARC-PAUL ANTOINE

1. In the classical *assignment problem* [9], *M jobs* are to be assigned to *N machines*, each job must be assigned to exactly one machine and no two jobs may be assigned to the same machine. The objective is to minimize the sum of the *processing costs* for all jobs. In the generalization studied in this note, it is assumed that *some, but usually not all, pairs of jobs may be assigned to the same machine*; the other assumptions are the same as those of the assignment problem. It is shown that the problem so defined is *NP-complete*. An algorithm which combines the *hungarian method* with a *Lagrangian relaxation* approach is presented. Computational experience is reported on.

2. The problem just stated may be expressed mathematically as follows:

$$\text{Minimize } z = \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij}, \tag{1}$$

subject to the constraints

$$\sum_{j=1}^{N} x_{ij} = 1, \quad i = 1, 2, \ldots, M, \tag{2}$$

$$x_{mj} + x_{nj} \leqslant 2 - b_{mn}, \quad 1 \leqslant m < n \leqslant M;\ j = 1, 2, \ldots, N, \tag{3}$$

$$x_{ij} \in \{0, 1\},\ i = 1, 2, \ldots, M;\ j = 1, 2, \ldots, N. \tag{4}$$

where c_{ij} denotes the processing cost of job i on machine j, x_{ij} a boolean variable equal to 1 if job i is assigned to machine j and equal to 0 otherwise and b_{mn} a constant equal to 1 if the jobs m and n may not be assigned to the same machine and equal to 0 otherwise.

An algorithm for a combinatorial problem is said to work in *polynomial time* if the number of operations it requires is bounded by a polynomial in the numbers of variables and of constraints of the problem. S. C o o k [4] and R. K a r p [7] [8] have shown that a large class of difficult combinatorial problems are equivalent in that no polynomial time algorithm is known for any of them and the existence of a polynomial time algorithm for one of them would imply the existence of such algorithms for all of them. Problems in this class are called *NP-complete*, and include *linear 0-1 programming*, the *chromatic number* of a graph problem and many others.

Theorem. *The problem (1) — (4) is NP-complete.*

Proof. a) As the problem (1) — (4) is a particular linear $0-1$ program, a polynomial algorithm for linear $0-1$ programming would allow to solve it in polynomial time.

b) Let $G = (V, E)$ denote a graph[1] (cf. C. Berge [2]) with *vertex* set V and *edge* set E. Associate to G a problem (1)—(4) with $B = (b_{mn})$ equal to the *adjacency matrix* of G, i.e. $b_{mn}=1$ if v_m and v_n are adjacent in G and $b_{mn} = 0$ otherwise, and with $c_{ij} = c_j$ for all $i = 1, 2, ..., M$ and each $j = 1, 2, ..., N$ and $c_{j+1} > Mc_j$ for $j = 1, 2, ..., N-1$. Clearly, in any optimal solution X^* of the problem so defined the $x^*_{ij} = 1$ for each j correspond to stable sets of G and the largest j such that, for some i, $x^*_{ij} = 1$ is equal to the chromatic number of G. Thus any polynomial algorithm for problem (1)—(4) would allow to solve the chromatic number problem, which is NP-complete, in polynomial time.

The proof of this theorem suggests an alternate interpretation of problem (1)—(4). It may be considered as the problem of *assigning colors to the vertices of a graph G at minimum cost*, the cost of each color and vertex pair being given, in such a way that no pair of adjacent vertices are assigned the same color.

Applications of problem (1)—(4) are easy to find; M.P. Antoine [1] describes such an application to the scheduling of university examinations. The „jobs“ considered are the examinations to be passed, the „machines“ correspond to the available periods and the coefficients c_{ij} represent the weighted sum of the disutilities, subjectively expressed by the professors and/or students, for the assignment of examination i to period j. If at least one student takes both the examinations m and n they are considered as incompatible, i.e. cannot take place in the same period.

3. As problem (1)—(4) is NP-complete, a branch-and-bound algorithm seems adequate to solve it. Such an algorithm will be efficient only if it uses tight bounds. To obtain such bounds we construct a *Lagrangian relaxation* of (1)—(4). Let $\bar{G} = (V, \bar{E})$ denote the *complementary graph* of the graph $G = (V, E)$ the adjacency matrix of which is $B = (b_{mn})$. Let us color the vertices of $\bar{G}$ with an exact or a heuristic coloring algorithm (e.g. by ranking the vertices in the order proposed by R. Brown [3] and applying a *simple sequencial coloration* procedure [11]). Let V_k, $k = 1, 2, ..., P$ denote the sets of vertices of color k. Then partition the set of constraints (3) into those which correspond to pairs of vertices in the same V_k (*primary constraints*) and those which correspond to pairs of vertices in different V_k (*liaison constraints*). Introducing the liaison constraints into the objective function yields the problem:

[1] Recall that a set $A \subseteq V$ of vertices of G is *stable* iff $\forall v_j, v_k \in A$ $\{v_j, v_k\} \notin E$, i.e. no two vertices of A are *adjacent* and that the *chromatic number* of G is the smallest number of stable sets in a partition of V into stable sets.

$$(5)\quad \text{Minimize } z' = \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij} x_{ij} - \sum_{k=1}^{P} \sum_{p \neq k=1}^{P} \sum_{v_m \in V_k} \sum_{v_n \in V_p} \sum_{j=1}^{N} b_{mn} \lambda_{mnj} (1 - x_{mj} - x_{nj})$$

subject to the constraints (2) and (4) as well as the constraints (3) which are primary and where the λ_{mnj} are non-negative. For given values of the λ_{mnj}, problem (5), (2)—(4) decomposes into P rectangular assignment problems corresponding to the sets V_k, $k = 1, 2, ..., P$ and with the constraints (2) for $v_i \in V_k$ and the primary constraints (3) for $v_m \in V_k$, $v_n \in V_k$. These subproblems may be solved easily by a variant of the hungarian method [10]. It follows from a result of A. Geoffrion [5] that the value z'^* of the optimal solution of (5), (2)—(4) is a lower bound on the optimal values z^* of (1)—(4); this bound can be made sharp by choosing adequate values for the λ_{mnj}. As the best values are unknown a priori, a method of successive approximations is used (cf. M. Held, Ph. Wolfe and H. Crowder [6]). The precise rules of the algorithm are the following:

a) *Initialisation.* Construct the Lagrangian relaxation (5), (2)—(4) of problem (1)—(4). Set the $\lambda_{mnj} = 0$ for all m, n and j with $b_{mn} = 1$. Set $z_{opt} = L$, an arbitrarily large value. Free all variables x_{ij}.

b) *Feasibility and optimality test.* Solve the P rectangular assignment problems corresponding to (5), (2)—(4), taking the values of the fixed variables, if any, into account. Let X^*_k denote the optimal solution of problem k and z^*_k its value, $k = 1, 2, ..., P$. If one of the assignment problems is infeasible or if $z'^* = \sum_{k=1}^{P} z^*_k - \sum_{m,n,j} \lambda_{mnj} \geq z_{opt}$ go to f).

c) *Modification of the λ_{mnj}.* If there is no liaison constraint $x_{mj} + x_{nj} \leq 1$ which is violated, go to d). If the λ_{mnj} have been modified more than 5 times since the last branching or backtracking go to e). Otherwise, if $\lambda_{mnj} \neq 0$, set $\lambda_{mnj} := 2\lambda_{mnj}$ if $x^*_{mj} = x^*_{nj} = 1$ and $\lambda_{mnj} := 1/2\, \lambda_{mnj}$ if $x^*_{mj} = x^*_{nj} = 0$. If $\lambda_{mnj} = 0$ and $x^*_{mj} = x^*_{nj} = 1$ with $v_m \in V_k$, $v_n \in V_p$ compute by post-optimality analysis the increments ∂_k and ∂_p given to z^*_k and to z^*_p by imposing $x_{mj} = 0$ and $x_{nj} = 0$; set $\lambda_{mnj} = 1/2\,(\partial_k + \partial_p)$. Return to b).

d) *Resolution test.* Set $z_{opt} = \sum_{k=1}^{P} z^*_k - \sum_{m,n,j} \lambda_{mnj}$; $X_{opt} = X^*$ union of the solutions X^*_k for $k = 1, 2, ..., P$. Go to f).

e) *Branching.* Consider the violated liaison constraint $x_{mj} + x_{nj} \leq 1$ with $v_m \in V_k$, $v_n \in V_p$ for which $\max(\partial_k, \partial_p)$ is largest.

Set $x_{mj} = 0$ if $\partial_k < \partial_p$ and $x_{nj} = 0$ otherwise. Return to b).

f) *Backtracking.* Consider the last variable x_{mj} (or x_{nj}) fixed at 0 during branching step; if no such variable remains, stop. Otherwise free all variables fixed at 1 after that variable was fixed, fix $x_{mj} = 1$ and $x_{nj} = 0$ (or $x_{nj} = 1$ and $x_{mj} = 0$). Return to b).

4. Let us consider a problem with 5 jobs and 3 machines, the data of which are the following:

$$C = \begin{vmatrix} 7 & 1 & 3 \\ 8 & 2 & 7 \\ 2 & 1 & 5 \\ 8 & 4 & 5 \\ 3 & 2 & 1 \end{vmatrix} \quad B = \begin{vmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{vmatrix}$$

a) Two assignment problems, corresponding to the 3 first and 2 last lines of C are obtained : the liaison constraints are $x_{1j} + x_{5j} \leq 1$ and $x_{2j} +$ $+ x_{4j} \leqslant 1$, $j = 1, 2, 3$. b) $x^*_{13} = x^*_{22} = x^*_{31} = 1$ and $z^*_1 = 7$: $x^*_{42} = x^*_{53} = 1$ and $z^*_2 = 5$. c) The constraint $x_{22} + x_{12} \leq 1$ is violated. $\bar{c}_1 = 3$, $\bar{c}_2 = 2$; $\lambda_{242} = 2.5$, hence 2.5 is added to c_{22} and to c_{21}. c) $x^*_{13} = x^*_{22} = x^*_{31} = 1$, $z^*_1 = 9.5$; $x^*_{43} = x^*_{52} = 1$. $z^*_2 = 7$. d) The optimal solution is $x^*_{13} = x^*_{22} =$ $= x^*_{31} = x^*_{43} = x^*_{52} = 1$, the other $x^*_{ij} = 0$. $z_{opt} = 9.5 + 7 - 2.5 = 14$.

5. The algorithm of section 3 has been programmed in FORTRAN IV and tested on a CDC 6 500 computer. A series of test problems have been solved with 50 jobs and 50 constraints, integer c_{ij}'s randomly chosen with equal probability in the range [1, 50] and a density for B (i.e. a ratio of 1's to number of off-diagonal positions in B) between 0.0 and 0.6. Computation times range from 1.2 to 66.6 seconds and rise when the density of B is larger than 0.4. Several series of similar problems, with a density of 0.7 and increasing numbers of jobs and machines have also been solved with variants of the algorithm, i.e. different initial values for the λ_{mnj} and a few more or a few less iterations in computing their last values. The overall computing time appears to be rather insensitive to these changes, and is around 6 seconds for problems with 30 jobs and 30 machines.

REFERENCES

1. A n t o i n e, M-P. — *Modeles mathématiques et algorithmes pour les problèmes d'horaires d'examens*, Mémoire de fin d'études, Faculté des Sciences Appliquées, Université Libre de Bruxelles (1975).
2. B e r g e, C. — *Graphes et hypergraphes*, Paris : Dunod (1970).
3. B r o w n, R. — *Chromatic Scheduling and the Chromatic Number Problem*. Management Science, 18 (1972) 456—463.
4. C o o k, S.A. — *The Complexity of Theorem Proving Procedures*. Proc. 3-rd. Annual ACM Symposium on Theory of Computing (1971) 151—158.
5. G e o f f r i o n, A. — *Lagrangian Relaxation for Integer Programming*, Mathematical Programming Study, 2 (1974) 82—114.
6. H e l d, M., W o l f e P. and C r o w d e r H. P. —*Validation of Subgradient Optimization*, Mathematical Programming, 6 (1974) 62—88.
7. K a r p, R. — *Reducibility among Combinatorial Problems*, in R.E. Miller and J.W. Thatcher (eds) *Complexity of Computer Computations*, New York : Plenum Press (1972) 85—103.
8. K a r p, R. — *On the Computational Complexity of Combinatorial Problems*, Networks, 5 (1975) 45—68.
9. K u h n, H. — *The Hungarian Method for the Assigument Problem*, Naval Research Logistics Quarterly, 2 (1955) 83—97.
10. K u h n, H. — *Variants of the Hungarian Method for Assigument Problems*, Naval Research Logistics Quarterly, 3 (1956) 253—258.
11. M a t u l a, D. W., M a r b l e, G., and I s a a c s o n, J. D. — *Graph Coloring Algorithms*, in R.C. Read (ed) *Graph Theory and Computing*, New York : Academic Press (1972) 109—122.

Received 15. VI. 1977

Faculté Universitaire Catholique de Mons and University of Brussels, Belgium

Analele ştiinţifice ale Universităţii „Al. I. Cuza" Iaşi
Tomul XXIV, S. I a, f. 1, 1978

SUR L'INTERSECTION DES LANGAGES ALGEBRIQUES

PAR

VLADIMIR V. TOPENTCHAROV, ROUMEN L. LOZANOV

Nous nous proposons dans cette note de démontrer qu' étant donné les langages L_1 et L_2 des formes sentencielles de deux grammaires algébriques G_1 et G_2 non ambiguës, il existe deux grammaires algébriques non ambigues G'_1 et G'_2 et deux morphismes F'_1 et F'_2 tels que, si $L_1 \cap L_2$ est aussi un langage de formes sentencielles algébrique et non ambigu alors $L(G'_1) = L(F'_1G'_1) = L_1$ et $L(G'_2) = L(F'_2G'_2) = L_2$ et $F'_1G'_1 \cup F'_2G'_2$ soit non ambigue.

La terminologie et les notions pour les catégories sont celles de [2] et pour les langages formels — de [3].

1. Soit C la catégorie dont les objets C_0 sont les grammaires algébriques et les flèches — les morphismes préservant strictement la longueur des mots (morphismes „lettre en lettre"); on démontre ([3]) que C est à produits, produits fibrés et sommes. Une grammaire algébrique (ou context-free) est donc un quadruplet $G = <O, X, P, S>$, où O est l'alphabet total, X-l'alphabet terminal, P-un sous-ensemble fini de $(O - X) \times O^*$, O^* étant le monoïde libre des mots sur O et $S \subset O - X$ est l'ensemble des symboles initiaux.

Le résultat suivant, qui est le point de départ du théorème que nous nous proposons de démontrer, nous a été communiqué sans démonstration par M. N i v a t (Univ. Paris VII, Paris).

Proposition. *Soit G_1 et G_2 deux grammaires algébriques non ambigues engendrant les langages $L_1 = L(G_1)$ et $L_2=L(G_2)$. S'il y a deux morphismes F_1 et F_2 préservant les alphabets terminaux et tels que $F_1G_1 \cup F_2G_2$ soit une grammaire non ambigue, alors $L_1 \cap L_2$ est un langage algébrique.*

Démonstration. Soit G_1 et G_2 deux objets quelconques de C. La construction du produit $\bar{G} = G_1 \times G_2$ est connue [3] : $\bar{O} = O_1 \times O_2$. $\bar{X} = X_1 \times X_2$, $\bar{S} = S_1 \times S_2$, $P = \{(w, v) : ((p_1(w), p_1(v)), (p_2(w), p_2(v))) \in P_1 \times P_2) \subset (\bar{O} - \bar{X}) \times \bar{O}^*$, où p_1 et p_2 sont les projections canoniques de $\bar{O}^*$. Considérons deux morphismes $f_i : G_i \to G_3$, $i = 1, 2$ et soit $((p_1, p_2), (f_1, f_2))$ le produit fibré correspondant G_0, où

$$O_0 = (a_1, a_2)\ \bar{O};\ f_1(a_1) = f_2(a_2);$$

$$X_0 = \bar{X} \cap O_0;$$

$$S_0 = \bar{S} \cap O_0;$$

$$P_0 = \bar{P} \cap (O^*_0 \times O^*_0);$$

$$\begin{tikzcd} & G_0 \arrow[dl, "p_1"'] \arrow[dr, "p_2"] & \\ G_1 \arrow[dr, "f_1"'] & & G_2 \arrow[dl, "f_2"] \\ & G_3 & \end{tikzcd}$$