P-Complete Approximation Problems

SARTAJ SAHNI AND TEOFILO GONZALEZ

University of Minnesota, Minneapolis, Minnesota

ABSTRACT For P-complete problems such as traveling salesperson, cycle covers, 0-1 integer programming, multicommodity network flows, quadratic assignment, etc, it is shown that the approximation problem is also P-complete In contrast with these results, a linear time approximation algorithm for the clustering problem is presented

KEY WORDS AND PHRASES P-complete, approximation algorithm, polynomial complexity, k-max cut, traveling salesperson, cycle covers, multicommodity network flows, quadratic assignment, integer programming, general partitions, k-min cluster, generalized assignment

CR CATEGORIES 5 23, 5 25, 5 32, 5 40

1. *Introduction*

Our notion of P-complete (polynomial-complete) corresponds to the one used in [19]. A problem, L, will be said to be P-complete iff the following holds: L can be solved in polynomial deterministic time iff the class of nondeterministic polynomial time languages is the same as deterministic polynomial time languages (i.e. P = NP). Knuth [14] suggests the terminology NP-complete (nondeterministic polynomial-complete). However, his notion of "completeness" is that of Karp [13]. Since the equivalence or nonequivalence of the two notions is not known, we will use the term NP-complete for problems that can be shown complete with Karp's definition and P-complete for those which require the definition of [19]. The reader unfamiliar with P-complete problems is referred to [13 and 19]. All problems that are NP-complete (i e. complete under Karp's definitions) are also P-complete [4 and 19]. The reverse is unknown. Since it appears that P $\neq$ NP, the P-complete problems probably have no polynomial solution Many of these problems, especially the optimization problems, are of practical significance Often, as in the case of the knapsack problem [20], approximate solutions (i.e. feasible solutions that are guaranteed to be "reasonably" close to the optimal) would be acceptable so long as they can be obtained "quickly" (e.g. by an $O(n^k)$ algorithm for small k). Johnson [11] and Sahni [20] have studied some P-complete problems with respect to obtaining "good" (i.e. polynomial) approximate algorithms. Formally we define an ϵ-approximate algorithm as follows.

Definition. An algorithm will be said to be an ϵ-approximate algorithm for a problem P_1 iff either (i) P_1 is a maximization problem and for every instance of P_1,

$$|(F^* - \hat{F})/F^*| \leq \epsilon, \quad 0 < \epsilon < 1,$$

An earlier version of some of the results in this paper was presented at the 15th IEEE Symposium on Switching and Automata Theory, 1974

This research was supported in part by the National Science Foundation under Grant DCR 74-10081

Authors' present addresses S Sahni, Department of Computer Science, 114 Lind Hall, University of Minnesota, Minneapolis, MN 55455, T Gonzalez, Department of Information and Computing Sciences, University of Oklahoma, Norman, OK 73069

or (ii) P_1 is a minimization problem and for every instance of P_1,

$$|(F^* - \hat{F})/F^*| \leq \epsilon, \quad \epsilon > 0,$$

where F^* is the optimal solution (assumed greater than 0) and $\hat{F}$ is the approximate solution obtained. □

Additional results on approximation algorithms for P-complete problems may be found in [2, 6, 8–10, 16, 18]. Some of these algorithms obtain ϵ-approximate solutions only for certain values of ϵ (i.e. $\epsilon \geq k$ for some k). For example, Graham's heuristic to sequence jobs on m identical processors is ϵ-approximate for $\epsilon \geq \frac{1}{3}(1 - 1/m)$. Other approximation algorithms [9, 10, 18, 20] obtain ϵ-approximations for any $\epsilon > 0$. An example is the $O(n/\epsilon^2 + n \log n)$ algorithm of Ibarra and Kim [10] for the 0/1 knapsack problem. General techniques for obtaining ϵ-approximate solutions for all ϵ are presented by Sahni [18] and Horowitz and Sahni [19]. The techniques are applicable to certain types of P-complete problems.

In this paper we shall look at some "natural" P-complete problems and show that the corresponding approximation problems are also P-complete One can easily construct "nonnatural" problems with this property. For example, consider the problem: Given a graph G, ϵ, $0 < \epsilon < 1$, and integer k, find that k-vertex subgraph G' which minimizes $f(G') = 1$ if G' is a clique and $f(G') = 1 + 2\epsilon$ otherwise.

For this problem it is easy to see that if G has a clique of size k then all ϵ-approximate solutions have $f(G) = 1$. If G has no such clique then $f(G) = 1 + 2\epsilon$. Hence from the ϵ-approximate solutions one can determine if G has a clique of size k, and so the approximation problem is P-complete for all ϵ. It is interesting that this should be true for naturally occurring problems. Garey and Johnson [6] show that obtaining ϵ-approximate solutions to the chromatic number problem is P-complete for all $\epsilon < 1$. The problems we shall consider are: traveling salesperson, cycle covers, 0/1 integer programming, multicommodity network flows, quadratic assignment, general partitions, k-min cluster, and generalized assignment. For these problems it will be shown that the ϵ-approximation problem is P-complete for all values of ϵ. (One should note that in the case of a maximization problem ϵ is restricted to $0 < \epsilon < 1$ as all feasible solutions are 1 − approximate.)

2. *P-Complete Approximation Problems*

In this section we look at some P-complete problems and show that they have a polynomial time approximate algorithm iff P = NP. This would then imply that if P ≠ NP, any polynomial time approximation algorithm for these problems, heuristic or otherwise, must produce arbitrarily bad approximations on some inputs. The problems we look at are:

(i) Traveling salesperson: Given an undirected (directed) complete graph $G(N, A)$ and a weighting function $w: A \to Z$, find an optimal tour (i.e. an optimal Hamiltonian cycle). The optimality criteria we shall consider are:

(a) Minimize tour length (i.e. find the shortest Hamiltonian cycle).

(b) Minimize mean arrival time at vertices. The arrival time is measured relative to a given start vertex i_1 and the weights are interpreted as the time needed to go from vertex i to vertex j. If $i_1, i_2, \cdots, i_n, i_{n+1} = i_1$ is a tour (i.e. Hamiltonian cycle), then the arrival time Y_k at vertex i_k is

$$Y_k = \sum_{j=1}^{k-1} w(i_j, i_{j+1}), \quad 1 < k \leq n + 1.$$

The mean arrival time $\bar{Y}$ is then

$$\bar{Y} = (1/n) \sum_{k=2}^{n+1} Y_k = (1/n) \sum_{j=1}^{n} (n + 1 - j) w(i_j, i_{j+1}).$$

The objective is to find a tour $i_1, \cdots, i_n, i_1$ that minimizes $\bar{Y}$ (see [3, p. 56])

(c) Minimize variance of arrival times. Let $i_1, i_2, \cdots, i_n, i_{n+1} = i_1$ be a tour. Let Y_k and $\bar{Y}$ be as defined in (b) We wish to obtain a tour that minimizes the quantity $\sigma = (1/n) \sum_{k=2}^{n+1} (Y^k - \bar{Y})^2$.

(ii) Undirected edge disjoint cycle cover: Given an undirected graph $G(N, A)$, find the minimum number of edge disjoint cycles needed to cover all the vertices of N (i.e. minimum number of cycles such that each vertex of G is on at least one cycle).

(iii) Directed edge disjoint cycle cover: Same as (ii) except that G is now a directed graph.

(iv) Undirected vertex disjoint cycle cover: This problem is the same as (ii) except that now the cycles are constrained to be vertex disjoint.

(v) Directed vertex disjoint cycle cover: Same as (iv) except that G is now a directed graph.

(vi) 0-1 integer programming with one constraint.

(vii) Multicommodity network flows: We are given a transportation network [19] with source s_1 and sink s_2. The arcs of the network are labeled corresponding to the commodities that can be transported along them. Such a network is said to have a flow f iff f units of each commodity can be transported from source to sink. The problem here is to maximize f.

(viii) Quadratic assignment [5, p. 18]:

$$\text{minimize} \quad f(x) = \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \sum_{\substack{k,l=1 \\ k \neq l}}^{m} c_{i,j} d_{k,l} x_{i,k} x_{j,l}$$

$$\text{subject to (a)} \quad \sum_{k=1}^{m} x_{i,k} \leq 1, \quad 1 \leq i \leq n,$$

$$\text{(b)} \quad \sum_{i=1}^{n} x_{i,k} = 1, \quad 1 \leq k \leq m,$$

$$\text{(c)} \quad x_{i,k} = 0, 1 \quad \text{for all } i, k,$$

$$\text{where} \quad c_{i,j}, d_{k,l} \geq 0, 1 \leq i, j \leq n, 1 \leq k, l \leq m.$$

One situation in which this problem arises is that of optimally locating m plants at n possible sites. Thus $x_{ik} = 1$ iff plant k is to be located at site i. Condition (a) states that at most 1 plant can be located at any particular site. Condition (b) requires that every plant be assigned to exactly 1 site. If $c_{i,j}$ represents the cost of transporting 1 unit of goods from site i to site j, and $d_{k,l}$ is the amount of goods to be transported from plant k to plant l, then $f(x)$ represents the cost of transporting all the goods between plants.

(ix) k-general partition [15]: We are given a connected undirected graph $G(N, A)$, an edge weighting function $f: A \rightarrow Z$, a vertex weighting function $w: N \rightarrow Z$, a positive number W, and an integer $k \geq 2$. The problem is to obtain k disjoint sets $S_1, \cdots, S_k$ such that:

(a) $\bigcup_{i=1}^{k} S_i = N$,

(b) $S_i \cap S_j = \phi$ for $i \neq j$,

(c) $\sum_{j \in S_i} w(j) \leq W$ for $1 \leq i \leq k$, and

(d) $\sum_{i=1}^{k} \sum_{\substack{(u,v) \in A \\ u,v \in S_i}} f(u, v)$ is maximized.

Partitioning problems of this type are encountered in the assignment of logic blocks to circuit cards in computer hardware design and in the assignment of computer information to physical blocks of storage [15].

(x) k-min cluster (for $k \geq 3$). This is the document clustering problem with the minimization criteria. Given an undirected graph $G(N, A)$ and a weighting function $w : A \to Z$ (the nonnegative integers), find k disjoint sets $S_1, \cdots, S_k$ such that

$$\bigcup_{i=1}^{k} S_i = N; \quad S_i \cap S_j = \phi \quad \text{for } i \neq j$$

and

$$\sum_{i=1}^{k} \sum_{\substack{\{u,v\} \in A \\ u, v \in S_i}} w(u, v) \quad \text{is minimized.}$$

(xi) Generalized assignment [17]:

$$\text{minimize} \quad \sum_{i \in I} \sum_{j \in J} c_{i,j} x_{i,j}$$

$$\text{subject to} \quad \sum_{j \in J} r_{i,j} x_{i,j} \leq b_i \quad \text{for all } i \in I,$$

$$\sum_{i \in I} x_{i,j} = 1 \text{ for all } j \in J,$$

$$x_{i,j} = 0, 1.$$

In this formulation $I = (1, 2, \cdots, m)$ is a set of agent indices, $J = (1, 2, \cdots, n)$ is a set of task indices, $c_{i,j}$ is the cost when agent i is assigned task j, $r_{i,j}$ is the resource required by agent i to perform task j, and $b_i > 0$ is the amount of resource available to agent i. The decision variable is 1 if agent i is assigned to task j, and is 0 otherwise. This problem arises in the following situations: assigning software development tasks to programmers, assigning jobs to computer networks, scheduling variable length television commercials, etc.

In order to prove some of our results we shall use the following known P-complete problems:

(a) Hamiltonian cycle: Given an undirected (directed) graph $G(N, A)$, does it have a cycle containing each vertex exactly once [13]?

(b) Multicommodity flows: Given the transportation network of (vii) above, does it have a flow of $f = 1$ [19]?

(c) k-graph colorability (for $k \geq 3$): Given an undirected graph $G(N, A)$, do there exist disjoint subsets $S_1, \cdots, S_k$ such that $\bigcup_{i=1}^{k} S_i = N$ and if $\{i, j\} \in A$ then vertices i and j are in different sets [7]?

(d) k-partition (for $k \geq 2$): Given n integers $r_1, r_2, \cdots, r_n$, are there disjoint subsets $I_1, I_2, \cdots, I_k$ such that $\bigcup_{i=1}^{k} I_i = \{1, 2, \cdots, n\}$ and $\sum_{i \in I_1} r_i = \sum_{i \in I_l} r_i$, $2 \leq l \leq k$?

THEOREM 2.1. *The ϵ-approximation problem for (i)-(xi) above is P-complete.*

PROOF For each of the problems (i)-(xi), it is easy to see that if P = NP then the ϵ-approximation problem is polynomially solvable (as the exact solutions would then be obtainable in polynomial time). Consequently we concern ourselves only with showing that if there is a polynomial time approximation algorithm for any of the problems listed above then P = NP. Our approach is to separate feasible solutions to a given problem in such a way that from a knowledge of the approximate solution one can solve exactly a known P-complete problem

(i-a) Hamiltonian cycle α ϵ-approximate traveling salesperson (minimum length criteria): Let $G(N, A)$ be any graph. Construct the graph $G_1(V, E)$ such that $V = N$ and $E = \{(u, v) \mid u, v \in V\}$. Define the weighting function $w : E \to Z$ to be

$$w\{u, v\} = \begin{cases} 1 \text{ if } (u,v) \in A, \\ k \text{ otherwise.} \end{cases}$$

Let $n = |N|$. For $k > 1$, the traveling salesperson problem on G_1 has a solution of length n iff G has a Hamiltonian cycle. Otherwise, all solutions to G_1 have length greater than or equal to $k + n - 1$. If we choose $k \geq (1 + \epsilon)n$, then the only solutions approximating a solution with value n (if there was a Hamiltonian cycle in G_1) also have length n. Consequently, if the ϵ-approximate solution has length less than or equal to $(1 + \epsilon)n$, then it must be of length n. If it has length greater than $(1 + \epsilon)n$, then G has no Hamiltonian cycle.

(i-b) Hamiltonian cycle α ϵ-approximate traveling salesperson (minimum mean arrival time criteria)· We construct $G_1(V, E)$ as in (i-a) above Let the starting vertex i_1 equal 1. It is easy to see that G_1 has a tour with mean arrival time less than or equal to $(n + 1)/2$ iff G has a Hamiltonian cycle. If G has no Hamiltonian cycle then all tours in G_1 have mean arrival time greater than or equal to $k/n + (n - 1)/2$. Choosing $k > (1 + \epsilon)n(n + 1)/2$ sufficiently separates these two solutions. The only solutions approximating $n(n + 1)/2$ also have value $n(n + 1)/2$. Consequently if the ϵ-approximate solution has a value less than or equal to $(1 + \epsilon)(n + 1)/2$, then it must be of value $(n + 1)/2$ and G has a Hamiltonian cycle. If the value is greater than $(1 + \epsilon)(n + 1)/2$, G has no Hamiltonian cycle

(i-c) Hamiltonian cycle α ϵ-approximate traveling salesperson (minimum variance criteria). From the undirected graph $G(N, A)$ we obtain the undirected graph $G_1(N_1, A_1)$ with

$$N_1 = N \cup \{\alpha, \beta, \gamma, \delta\}, \quad A_1 = A \cup \{(r, \alpha), (\alpha, \beta), (\beta, \gamma), (\gamma, \delta)\} \quad \cup \{(\delta, z) \mid (r, z) \in A\},$$

where r is some arbitrary vertex in N. This construction is shown in Figure 1. From the construction it is evident that G_1 has a Hamiltonian cycle iff G has such a cycle. From the graph G_1 we obtain the traveling salesperson problem $G_2(N_2, A_2)$ with $N_2 = N_1$, $A_2 = \{(i,j) \mid i \neq j,\ i,j \in N_2\}$, and weighting function w . $A_2 \to Z$ defined by

$$w\{u, v\} = \begin{cases} 1 \text{ if } (u,v) \in A_1, \\ k \text{ if } (u,v) \notin A_1. \end{cases}$$

Lemma 2.1 obtains lower bounds on the variance (σ) of an optimal tour for G_2.

LEMMA 2.1 *For $k > [(1 + \epsilon)(n)(n - 1)(n + 1)/3]^{1/2}$ and $\epsilon > 0$, the complete graph G_2 has a tour, with starting vertex β, with a variance $\sigma \leq (n - 1)(n + 1)/12$, iff G_1 has a Hamiltonian cycle. If G_1 has no Hamiltonian cycle, then the optimal tour for G_2 has $\sigma > (1 + \epsilon)(n - 1)(n + 1)/12 \cdot n = |N_2|$.*

PROOF If G_1 has a Hamiltonian cycle, then this cycle is a valid tour in G_2. All edges on this tour have weight 1 and mean arrival time $= \bar{Y} = (n + 1)/2$,

$$\sigma = (1/n) \textstyle\sum_1^n (i - \bar{Y})^2 = (1/n) \sum_1^n i^2 - \bar{Y}^2$$
$$= \frac{1}{6}(2n^2 + 3n + 1) - \frac{1}{4}(n + 1)^2 = (n - 1)(n + 1)/12.$$

If G_1 has no Hamiltonian cycle, then every tour in G_2 must include at least one edge with weight equal to k. Let the optimal tour be $\beta = i_1, i_2, \cdots, i_n, i_{n+1} = \beta$. We have three cases:

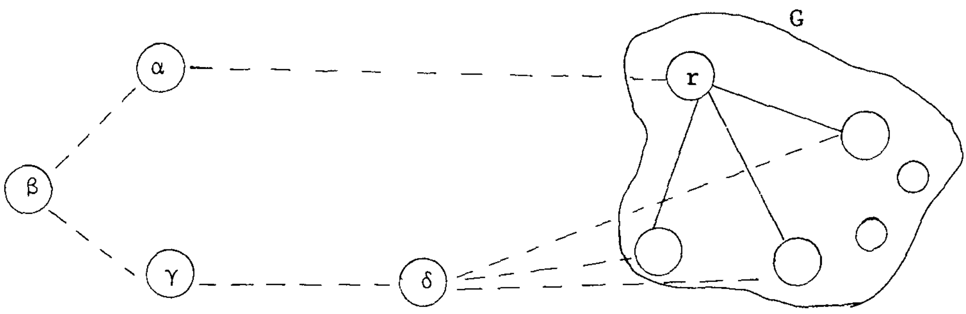

FIG 1 Construction of G_1 from G Broken lines indicate edges in G_1 which are not in G

Case 1. $w(\beta, i_2) = 1$, $w(i_j, i_{j+1}) = k$ for some j, $1 < j \leq n$: For this case we have $Y_2 = 1$ and $Y_{n+1} \geq k + n - 1$. If $\bar{Y} \geq k/2 + 1$, then $|Y_2 - \bar{Y}| \geq k/2$. If $\bar{Y} < k/2 + 1$, then $|Y_{n+1} - \bar{Y}| \geq k/2$. In either case we have

$$\sigma \geq (k/2)^2/n = k^2/(4n) > (1 + \epsilon)(n - 1)(n + 1)/12 \quad \text{for } k > [(1 + \epsilon)n(n - 1)(n + 1)/3]^{1/2}.$$

Case 2. $w(\beta, i_2) = k$ and all other edges have weight $= 1$: Since all other edges on the tour have weight 1, it follows that $i_n = \alpha$ or $i_n = \gamma$ as (α, β) and (γ, β) are the only two edges in A_2 incident to β and having weight 1 Without loss of generality we may assume $i_n = \alpha$. Since γ is a vertex on the tour, the tour enters γ via some edge (u, γ) and leaves via another edge (γ, v), $u \neq v$ and $v \neq \beta$. Also $u \neq \beta$ as $w(\beta, i_2) = k \neq 1$. From the construction of G_2 it is clear that the only edges in A_2 incident to γ with weight 1 are (β, γ) and (γ, δ). (γ, δ) is the only edge satisfying the requirements on u and v. Hence the second edge on the tour incident to γ must have weight $= k$. Hence there is no tour in G_2 with $w(\beta, i_2) = k$ and all other edges having a weight of 1.

Case 3. $w(\beta, i_2) = k$ and $w(i_j, i_{j+1}) = k$ for some j, $1 < j \leq n$: Now $Y_2 = k$ and $Y_{n+1} \geq 2k + n - 2$. If $\bar{Y} \geq 3k/2$, then $|Y_2 - \bar{Y}| \geq k/2$. If $\bar{Y} < 3k/2$, then $|Y_n - \bar{Y}| \geq k/2$. Hence $\sigma > (1 + \epsilon)(n - 1)(n + 1)/12$ (see case 1).

This takes care of all possibilities when G_1 has no Hamiltonian cycle. □

The reduction of (i-c) now follows from arguments similar to those of (i-a) and (i-b).

(ii)–(v) The proofs for (ii)–(v) are similar. We outline the proof for (iv).

(iv) Undirected Hamiltonian cycle α ϵ-approximate disjoint cycle cover: Given an undirected graph $G(N, A)$, construct k copies $G_i(N_i, A_i)$ of this graph. Pick a vertex $v \in N$. Let $u^1, u^2, \cdots, u^d$ be the vertices adjacent to v in G (i.e. $(u^i, v) \in A$, $1 \leq i \leq d$). Define $H_j(V_j, E_j)$ to be the graph with

$$V_j = \bigcup_{i=1}^{k} N_i \quad \text{and} \quad E_j = \bigcup_{i=1}^{k} A_i \cup \{(u_i^j, v_{i+1}) \mid 1 \leq i < k\} \cup \{(u_k^j, v_1)\}.$$

Clearly, if G has a Hamiltonian cycle then, for some j, H_j has a cycle cover containing exactly one cycle (as for some j, (v, u^j) are adjacent in the Hamiltonian cycle and using the images of the edges of this cycle in the subgraphs G_i (except for the images of the edge (v, u^j), together with the edges $\{(u_i^j, v_{i+1}) \mid 1 \leq i < k\} \cup \{(u_k^j, v_1)\}$, one obtains a Hamiltonian cycle for E_j). If G has no Hamiltonian cycle, then the subgraphs G_i each require at least two disjoint cycles to cover their nodes. Consequently the disjoint cycle cover for j contains at least $k + 1$ cycles, $1 \leq j \leq k$. For any ϵ, one may choose a suitable k such that from the approximate solutions to H_j, $1 \leq j \leq k$, one can decide whether or not G has a Hamiltonian cycle (i.e. choose $k > (1 + \epsilon)$).

Note that the above proof also works for the case of edge disjoint cycle covers, since G has an edge disjoint cycle cover of size 1 iff it has a Hamiltonian cycle.

(vi) Just consider the reduction: 2-partition α ϵ-approximate 0-1 integer programming, i e.

$$\begin{aligned} \text{minimize} \quad & 1 + k(m - \textstyle\sum r_i\delta_i) \\ \text{subject to} \quad & \textstyle\sum r_i\delta_i \leq m, \\ & \delta_i = 0 \text{ or } 1, \\ & m = \textstyle\sum r_i/2. \end{aligned}$$

The minimum equals 1 iff there is a subset with sum equal to m; otherwise the minimum is greater than or equal to $1 + k$.

(vii) Multicommodity flows α ϵ-approximate multicommodity flows: In [19] it was shown that multicommodity flows with $f = 1$ was P-complete Given a multicommodity network N as in [19] we construct k copies of it and put them in parallel. Another network with a flow $f = 1$ is also coupled to the network as in Figure 2.

Clearly the multicommodity network of Figure 2 has a flow of $k + 1$ iff N has a flow of 1. If N does not have a flow of 1, then the maximum flow in the network of Figure 2 is 1. Hence the approximation problem for multicommodity flows is P-complete.

(viii) Hamiltonian cycle α ϵ-approximate quadratic assignment: Let $G(N, A)$ be an undirected graph with $m = |N|$. The following quadratic assignment problem (QAP) is constructed from G:

$$n = m,$$
$$c_{i,j} = \begin{Bmatrix} 1 & \text{if } j = i + 1 \text{ and } i < m \quad \text{or if} \quad i = m \text{ and } j = 1 \\ 0 & \text{otherwise} \end{Bmatrix}, 1 \le i, j \le n,$$
$$d_{k,l} = \begin{Bmatrix} 1 & \text{if } (k, l) \in A, \\ \omega & \text{otherwise} \end{Bmatrix}, \quad 1 \le k, l \le m.$$

The total cost, $f(\gamma)$, of an assignment, γ, of plants to locations is $\sum_{i=1}^{n} c_{i,j} d_{\gamma(i)\gamma(j)}$, $j = (i \bmod m) + 1$, when $\gamma(i)$ is the index of the plant assigned to location i. If G has a Hamiltonian cycle $i_1, i_2, \cdots, i_n, i_1$, then the assignment $\gamma(j) = i_j$ has a cost $f(\gamma) = m$. In case G has no Hamiltonian cycle, then at least one of the values $d_{\gamma(i),\gamma((i \bmod m)+1)}$ must be ω and so the cost becomes greater than or equal to $m + \omega - 1$. Choosing $\omega > (1 + \epsilon)m$ results in optimal solutions with a value of m if G has a Hamiltonian cycle and value greater than $(1 + \epsilon)m$ if G has no Hamiltonian cycle. Thus from an ϵ-approximate solution it can be determined whether or not G has a Hamiltonian cycle.

(ix) k-partition α ϵ-approximate k-general partition: We prove this for $k = 2$. The proof is similar for other values of k. From the 2-partition problem the following 2-general partition problem is constructed (see also Figure 3):

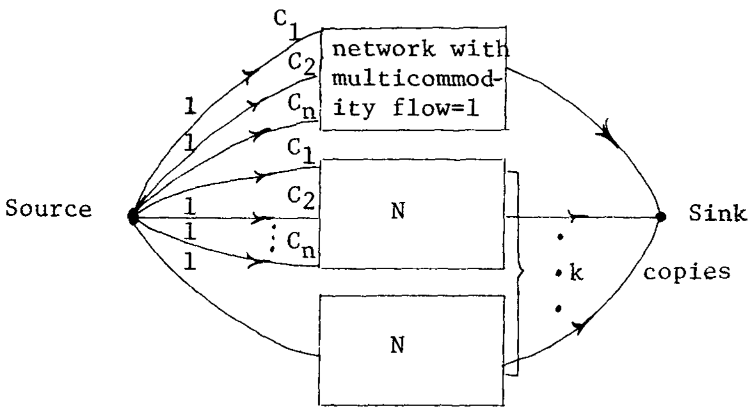

FIG 2 Reduction for ϵ-approximate multicommodity flows

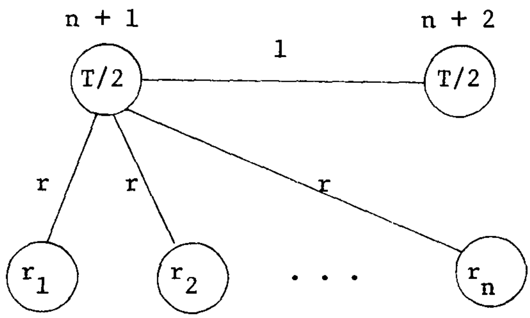

FIG 3 Reduction for ϵ-approximate k-general partition Numbers in vertices represent vertex weights, numbers on edges represent edge weights

$$N = \{1, 2, \cdots, n+2\},$$
$$A = \{(i, j) \mid 1 \le i \le n, j = n+1\} \cup \{(n+1, n+2)\},$$
$$f(u, v) = \begin{cases} r & \text{if } (u, v) \in A \text{ and } 1 \le u \le n, \\ 1 & (u, v) = (n+1, n+2), \end{cases}$$
$$w(j) = \begin{cases} r_j & 1 \le j \le n, \\ T/2 & j \ge n+1, \end{cases} \text{ where } T = \sum_1^n r_i,$$
$$W = T.$$

Clearly there is a solution of value greater than or equal to $nr/2$ iff the multiset $\{r_1, \cdots, r_n\}$ has a partition. If there is no partition of this multiset, then the solution value is 1. A suitable choice for r yields the desired result.

(x) l-chromatic number α ϵ-approximate l-min cluster, for all $l \ge 3$: Let $G(N, A)$ be an undirected graph. The following l-min cluster problem $G_1(N_1, A_1)$ is constructed:

$$N_1 = N,$$
$$A_1 = \{(u,v) \mid u \ne v \text{ and } u,v \in N_1\},$$
$$w(u,v) = \begin{cases} 1 & \text{if } (u,v) \notin A, \\ k & \text{otherwise.} \end{cases}$$

If G is l-colorable, then the l-min cluster problem has a solution with value less than n^2. If G is not l-colorable, then the minimum solution value is greater than or equal to k. Choosing $k > (1+\epsilon)n^2$ yields the desired result.

(xi) 2-partition α ϵ-approximate generalized assignment: From the partition problem $S = \{a_1, a_2, \cdots, a_n\}$, construct the following generalized assignment problem:

$$\begin{array}{ll} c_{1,i} = c_{2,i} = 1,\ c_{3,i} = k, & \text{for } 1 \le i \le n, \\ r_{1,i} = r_{2,i} = r_{3,i} = a_i & \text{for } 1 \le i \le n, \\ b_1 = b_2 = T/2,\ b_3 = T, & \text{where } T = \sum a_i \end{array}$$

Clearly there is a solution of value n iff the multiset $\{a_1, a_2, \cdots, a_n\}$ has a partition. If there is no partition of this multiset, then the solution value is greater than k. Choosing $k > (1+\epsilon)n$ yields the desired result.

This completes the proof of the theorem. □

3. *Conclusions*

In this paper we have shown that many combinatorial problems are, in a sense, much more difficult than mere P-completeness would imply. If P ≠ NP, not only can you not find optimal solutions in polynomial time, you cannot even guarantee coming close. Thus some P-complete problems may be harder than others.

In addition to the problems studied here, many others may have this property. Any P-complete problem for which no good heuristic is known is a candidate. In particular, no good heuristics are known for such important problems as maximum cliques, job shop and flow shop sequencing, etc. Results concerning such problems would be of much interest.

As with P-completeness, however, one must be wary of concluding that similar problems behave in similar ways. For instance, although the general one-constraint 0-1 integer programming problem apparently does not have polynomial time ϵ-approximate algorithms for any $\epsilon > 0$, the knapsack problem, which is a special case, can be ϵ-approximated in linear time for *all* $\epsilon > 0$ [10]. For another example, consider obtaining ϵ-approximate solutions for the k-min cluster problem. We have shown this to be P-complete for all $\epsilon > 0$. If instead of trying to minimize the number of edges within the clusters we try to maximize the number of edges between clusters, we get what is apparently an equivalent problem—the optimal solutions are the same. However, as can be shown (see Appendix) for this "k-max cut" problem, there exists a simple linear time ϵ-approximate algorithm for all $\epsilon \geq 1/k$. Finally, consider the traveling salesperson problem with minimum tour length criteria. Our results show that the approximation problem is P-complete. However, Rosenkrantz, Stearns, and Lewis [16] have shown that when the edge weights satisfy the triangular inequality, ϵ-approximate solutions may be obtained in polynomial time for certain values of ϵ.

Appendix

Here we study the clustering problem of [12]. A set of n documents is represented by a weighted undirected complete graph G. The vertices are labeled 1 through n with vertex i corresponding to document i, and the weight of the edge (i, j), $w(i, j)$ is a measure of the dissimilarity of the documents i, j. The objective is to partition the set of n documents into k disjoint clusters (groups) such that the total dissimilarity among clusters (i.e. $\sum w(i,j)$ for i,j in different clusters) is maximized. Sometimes we may be interested in obtaining n/k clusters for some constant integer k. We first define these two problems formally.

(a) k-max cut ($k \geq 2$)[1]: Given an undirected graph $G = (N, A)$, find k disjoint sets, S_i, $1 \leq i \leq k$, such that $k \bigcup_{i=1}^{k} S_i = N$ and $\sum_{\substack{\{u,v\}\in A \\ u\in S_i \\ v\in S_j \\ i<j}} w\{u,v\}$ is maximized.

(b) $[n/k]$-max cut: Same as (a) with k replaced by $[n/k]$.

Using the proof techniques of Karp [13], one may easily show that these two problems are P-complete.

We now present an approximation algorithm for the k-max cut and $[n/k]$-max cut problems. Consider the algorithm MAXCUT below. (Intuitively, this algorithm begins by placing one vertex of G into each of the l sets S_i, $1 \leq i \leq l$; the remaining $n - l$ vertices are examined one at a time. Examination of a vertex, j, involves determining the set S_i, $1 \leq i \leq l$, for which $\sum_{m\in S_i} w\{m, j\}$ is minimal. Vertex j is then inserted/assigned to this set.) A similar algorithm for this problem appears in [12].

Algorithm MAXCUT(l, G)

comment. l · number of disjoint sets, S_i, into which the vertices, $N = (1, 2, \ldots, n)$, of the graph $G(N, A)$ are to be partitioned, SOL the value of the vertex partitioning obtained, $w\{i,j\}$ weight of the edge $\{i, j\}$; $SET(i)$ the set to which vertex i has been assigned ($SET(i) = 0$ for all vertices not yet assigned to a set), $WT(i)$ used to compute $\sum_{m\in S_i} w\{m, j\}$, $1 \leq i \leq l$

This algorithm assumes that the graph $G(N, A)$ is presented as n lists $v_1, v_2, \cdots, v_n$ Each list v_i contains all the edges, $\{i, j\} \in A$, that are adjacent to vertex i No assumption is made on the order in which these edges appear in the list **end comment.**

Step 1 [Initialize]

If $l \geq n$ **then do**

$SOL \leftarrow \sum_{\{i,j\}\in A} w\{i, j\}$

$SET(i) \leftarrow i,\ 1 \leq i \leq n$

stop

end

else do

$WT(i) \leftarrow 0,\ 1 \leq i \leq l$

$SET(i) \leftarrow i,\ 1 \leq i \leq l$

$SET(i) \leftarrow 0,\ l + 1 \leq i \leq n$

$SOL \leftarrow \sum_{\substack{\{i,j\}\in A \\ 1\leq i<j\leq l}} w\{i, j\}$

$j \leftarrow l + 1$

end

[1] The k-max cut problem is also a generalization of the "grouping of ordering data" problem studied in [1] [1] restricts the set S_i to be sequential, i e if $i, j \in S_l$ and $i < j$, then $i + 1, i + 2, \ldots, j - 1 \in S_l$ [1] presents an $O(kn^2)$ dynamic programming algorithm for this

Step 2 [process edge list of vertex j]
  for each edge $\{j, m\}$ on the edge list of vertex j **do**
    if $SET(m) \neq 0$ **then** $WT(SET(m)) \leftarrow WT(SET(m)) + w\{j, m\}$;
    end
  $d_j \leftarrow$ degree of vertex j = # of edges adjacent to vertex j

Step 3 [find the set for which $\sum_{m \in S_i} w\{j, m\}$ is minimal]

  look at $WT(a)$, $1 \leq a \leq \min\{d_j + 1, l\}$ and determine i such that $WT(i)$ is minimal in this range (Note that if $d_j + 1 \leq l$ then at least one of $WT(a)$, $1 \leq a \leq d_j + 1$, must be 0 and minimal For $d_j + 1 \geq l$ all $WT(a)$ are looked at and the minimal found.)

Step 4 [assign vertex j to set S_i]
  $SET(j) \leftarrow i$

Step 5 [update SOL and reset WT]
  for each edge $\{j, m\} \in A$ for which $SET(m) \neq 0$ **do**
  if $SET(m) \neq i$ **then** $SOL \leftarrow SOL + w\{j, m\}$
  $WT(SET(m)) \leftarrow 0$
  end

Step 6 [next vertex] $j \leftarrow j + 1$
  if $j \leq n$ **then go to** step 2
    else terminate algorithm
end MAXCUT

LEMMA A1. *The time complexity of algorithm MAXCUT is $O(l + n + e)$ on a random access machine (n is the number of vertices, e the number of edges, and l the number of groups into which the vertices are to be partitioned).*

PROOF.

Step	Time per execution	Total time
1	$O(n + e + l)$	$O(n + e + l)$
2	$O(d_j)$	$O(e)$
3	$O(d_j + 1)$	$O(e + n)$
4	$O(1)$	$O(n)$
5	$O(d_j)$	$O(e)$
6	$O(1)$	$O(n)$

Hence the total time is equal to $O(n + e + l)$.

LEMMA A2. *Algorithm MAXCUT is a $1/k$-approximate algorithm for the k-max cut problem.*

PROOF. If $n \leq k$, then MAXCUT generates the optimal solution value. So assume $n > k$. Define the internal weight of the set S_i to be $\sum_{u<v, u,v \in S_i} w\{u, v\}$. Then the total internal weight (TIW) equals $\sum_{i=1}^{k}$ internal weight (S_i). The external weight (EW) equals $\sum_{u<v, u \in S_i, v \in S_j, i \neq j} w\{u, v\}$. In step 4, when vertex j is assigned to set i, either $WT(i) = 0$ (corresponding to $d_j < l$) or $WT(i) \leq \sum_{1 \leq m \leq k} WT(m)/k$, i.e. if the total internal weight increases by $WT(i)$, then the external weight increases by at least $(k - 1)WT(i)$. Consequently, at termination $TIW < EW/(k - 1)$ (note that $SOL = EW$). But the optimal value of the solution is less than or equal to $TIW + EW$. Let F^* be the optimal. $EW = SOL$ is the approximation obtained by MAXCUT. The worst case occurs when TIW approaches $EW/(k - 1)$. Hence $|(F^* - SOL)/F^*| < 1/k$. □

From Lemma A2 it follows that Algorithm MAXCUT is a k/n-approximate algorithm for the n/k-max cut problem. While approximately optimal clusters may be found in linear time using the maximization criteria, one of the results of this paper is that finding approximately optimal clusters under the minimization criteria is P-complete.

ACKNOWLEDGMENTS. Organizational changes suggested by the referees have improved the readability of this paper. In particular, the concluding section (Section 3) was contributed by one of the referees. We would like to thank the referees for their interest and helpful comments

REFERENCES

1 BODIN, L D. A graph theoretic approach to the grouping of ordering data. *Networks 2* (1972), 307-310
2 BRUNO, J , COFFMAN, E G , AND SETHI, R Scheduling independent tasks to reduce mean finishing-time *Comm ACM 17*, 7 (July 1974), 382-387.
3 CONWAY, R W., MAXWELL, N.L , AND MILLER, L W *Theory of Scheduling* Addison-Wesley, Reading, Mass , 1967
4 COOK, S A The complexity of theorem-proving procedures Conf. Record of Third ACM Symp on Theory of Computing, 1970, pp 151–158.
5 GARFINKEL, R S., AND NEMHAUSER, G L *Integer Programming* Wiley, New York, 1972
6 GAREY, M R , AND JOHNSON, D S The complexity of near-optimal graph coloring. *J ACM 23*, 1 (Jan 1976), 43–69
7 GAREY, M R , JOHNSON, D S , AND STOCKMEYER, L J Some simplified NP-complete problems *Theoretical Comput Sci* (to appear)
8 GRAHAM, R L Bounds on multiprocessing timing anomalies *SIAM J Appl Math 17*, 2 (March 1969), 416–429
9 HOROWITZ, E , AND SAHNI, S Exact and approximate algorithms for scheduling nonidentical processors *J ACM 23*, 2 (April 1976), 317-327
10 IBARRA, O H , AND KIM, C E Fast approximation algorithms for the knapsack and sum of subset problems. *J ACM 22*, 4 (Oct. 1975), 463-468.
11 JOHNSON, D Approximation algorithms for combinatorial problems. *J Comput. Syst Scis 9*, 3 (Dec 1974), 256-278
12 JOHNSON, D.B , AND LAFUENTE, J M A controlled single pass classification algorithm with application to multilevel clustering Scientific Rep #ISR-18, *Information Science and Retrieval*, Cornell U , Ithaca, N Y , Oct 1970, pp XII-1-XII-37
13 KARP, R M Reducibility among combinatorial problems In *Complexity of Computer Computations*, R E Miller and J W Thatcher, Eds , Plenum Press, N Y , 1972, pp 85-104
14 KNUTH, D E. A terminological proposal ACM SIGACT News 6, 1 (Jan 1974), 12-18
15 LUKES, J A Combinatorial solution to the partitioning of general graphs *IBM J Res and Develop 19*, 2 (March 1975), 170-180
16 ROSENKRANTZ, D J , STEARNS, R E , AND LEWIS, P M Approximate algorithms for the travelling salesperson problem 15th Annual IEEE Symp on Switching and Automata Theory, 1974, pp 33-42
17 ROSS, G T , AND SOLAND, R M A branch and bound algorithm for the generalized assignment problem *Math Programming 8* (1975), 91-103
18 SAHNI, S. Algorithms for scheduling independent tasks *J ACM 23*, 1 (Jan 1976), 116-127
19 SAHNI, S Computationally related problems. *SIAM J Comput 3*, 4 (Dec 1974), pp 262-279
20 SAHNI, S Approximate algorithms for the 0/1 knapsack problem *J ACM 22*, 1 (Jan 1975), 115-124

RECEIVED JULY 1975, REVISED JANUARY 1976