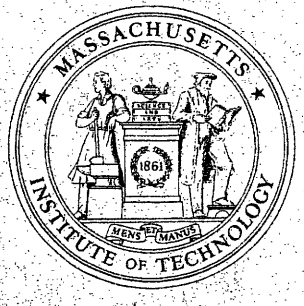
OPERATIONS RESEARCH CENTER

working paper



MASSACHUSETTS INSTITUTE OF TECHNOLOGY

The Theory of Cyclic Transfers

by

Paul M. Thompson and James B. Orlin

OR 200-89 August 1989

THE THEORY OF CYCLIC TRANSFERS

by

Paul M. Thompson*

and

James B. Orlin**

August, 1989

* Leavey School of Business and Administration
Santa Clara University, Santa Clara, CA 95053

** Sloan School of Management
MIT, Cambridge, MA 02139

ABSTRACT

This paper develops the theory of cyclic transfers, a neighborhood search strategy for a broad class of combinatorial problems. The problem class is characterized by a two-phase decision process: first, the optimal assignment of elements to clusters and, second, the optimal configuration of the elements within each cluster. Many important and difficult problems fall into this problem class, for example, facility location, multi-vehicle routing and scheduling, clustering, vertex covering, graph coloring, weighted matching and quadratic assignment problems.

We characterize a local search neighborhood in terms of cyclic transfers of elements among clusters, define data structures useful in searching the neighborhood, and show that the neighborhood search problem is itself NP-hard. We derive necessary and sufficient conditions for the existence of negative cost cyclic transfers in terms of negative cost cycles on an auxiliary graph. Finally, we examine algorithm implementation issues.

This paper develops and analyzes a generic type of local search algorithm for a broad class of combinatorial problems. In the first section, we define rooted partitioning problems, the combinatorial problem class to which our research applies, and discuss representative examples. In Section 2, we motivate local search as a solution methodology and review the local search literature for this class of combinatorial problems. In the remaining sections, we develop the theory of cyclic transfers, the neighborhood search strategy that is the focus of this paper, and discuss implementation of cyclic transfer algorithms. Section 3 introduces cyclic transfers, characterizes a class of local neighborhoods in terms of cyclic transfers, shows that a solution locally optimal in a cyclic transfer neighborhood is not necessarily globally optimal, defines data structures useful in searching cyclic transfer neighborhoods, and investigates the computational complexity of the neighborhood search problem. Section 4 derives necessary and sufficient conditions for the existence of negative cost cyclic transfers in terms of negative cost cycles on an auxiliary graph. Section 5 examines algorithmic implementation issues. Finally, Section 6 presents conclusions.

We believe that the methodology developed in this paper will prove useful for solving a variety of problems of practical importance. Indeed, related work [Thompson, 1988] [Thompson and Psaraftis, 1989], has shown the efficacy of our methods for several classes of rooted partitioning problems that arise in vehicle routing and scheduling.

1. The Combinatorial Setting

The generic class of combinatorial problems that we investigate in this paper we term **rooted partitioning problems**. Before defining this problem class, we introduce some notation. Let S be a set. An **m-partition** of S is a partition of S into m (possibly empty) subsets $I^1, I^2, \ldots, I^m$, i.e., the subsets $\{I^j : j=1 \text{ to } m\}$ are pairwise disjoint, and their union is the set S. Now, let S and T be disjoint sets, and let us refer to T as the root set, with $|T| = m$. A **rooted partition** of $S \cup T$ is an m-partition of $S \cup T$ into subsets $I^1, \ldots, I^m$ such that

for each $j = 1$ to m, $|I^j \cap T| = 1$.

We refer to I^j as the j-th cluster, and we refer to the unique element in $I^j \cap T$ as the root of the j-th cluster. We denote this element as $\text{ROOT}(I^j)$.

Rooted partitioning problems have the following mathematical structure. Let S and T be sets, and suppose that T is a root set with $|T| = m$. Let f be a function from $S \cup T$ into the reals, i.e.,

$$f : \{S^i + t_j\} \rightarrow \Re$$

for all subsets $S^i \subseteq S$ and all elements $t_j \in T$. Thus, f is a function both of the root and of the cluster's elements. The rooted partitioning problem is to

Minimize: $\sum (f(I^j): j = 1 \text{ to } m)$

Subject to: $I^1,\ldots,I^m$ is a rooted partition of $S \cup T$.

For cases where the cost function f is independent of the root but where the number of parts is fixed, the problem is really an unrooted problem. In this case, we define an analogous unrooted m-partitioning problem. If the cost function f is independent of the root and, in addition, the number of parts is not fixed in advance, we define an unrooted partitioning problem.

In our problem formulation, we allow f to implicitly model both the problem's objective function and the problem's constraints. We model the constraints by assigning arbitrarily large values to f for infeasible (rooted) partitions. In this way, f can model feasibility restrictions such as time window and capacity constraints. In addition, we permit functions f that are defined implicitly but not explicitly. For example, f(S) might be the minimum TSP cost of visiting all of the customers in S. Thus the evaluation of f(•) may in and of itself be an NP-hard problem.

Because both the feasibility constraints and the objective function may be modeled implicitly, the problem class is very broad. In fact, every combinatorial problem can be modeled in this way by looking for an optimal 1-partition and letting f(•) model both the objectives and the constraints. While this is true, however, it is not particularly helpful. Our theory offers no help in solving a problem in which the number of parts is one. It applies well only if the problem may be modeled naturally as a partitioning or rooted partitioning problem.

The practical importance of partitioning and rooted partitioning problems lies in the broad variety of operational contexts in which they occur. In the remainder of this section, we identify some important and representative examples of this problem class.

1. **Vehicle Routing and Scheduling** [Bodin et al., 1983], [Christofides, Mingozzi and Toth, 1979], [Golden and Assad, 1986], [Magnanti, 1981], [Solomon and Desrosiers, 1987]. We can formulate a vehicle routing and scheduling

problem as follows. Let S denote the set of customers and let T denote the set of vehicles. Let $f(I^j)$ reflect the cost of optimally scheduling the customers in I^j to vehicle $ROOT(I^j)$ while satisfying capacity, time, and other constraints. If the vehicle fleet is non-homogeneous (or based at more than one depot), then the problem is to find a rooted partition $I^1,\ldots,I^m$ of $S \cup T$ so as to minimize $\sum f(I^j: j = 1 \text{ to } m)$. If, however, the fleet is homogeneous and based at a single depot, then the cost of servicing a cluster of customers is independent of the vehicle serving the cluster. In this case, the mathematical formulation does not require one to specify a set of vehicles. In this case, the problem is a partitioning problem, i.e., a problem of finding a partition $I^1,I^2,\ldots$ of S that minimizes $\sum f(I^j: j = 1,2,\ldots)$.

A common route initialization procedure for vehicle routing problems is seed assignment [Jaw et al., 1982] [Fisher and Jaikumar, 1983] [Psaraftis et al., 1985]. Typically, a "seed" for a vehicle denotes the first customer that an algorithm assigns to the vehicle. Because the initial assignment profoundly affects the cost of subsequent assignments to a route, a good selection of seed assignments is critical. With this approach, once the seeds are assigned for a vehicle routing problem, the remaining scheduling problem is a rooted partitioning problem.

2. **Facility Location** [Francis and White, 1974] [Odoni, 1974] [Handler and Mirchandani, 1979] [Larson and Odoni, 1981]. Facility Location problems are naturally modeled as rooted partitioning problems. Let S denote the set of customers and let T denote the set of potential facility sites. The facility location problem is to find a rooted partition $I^1,\ldots,I^m$ of $S \cup T$ so as to minimize $\sum (f(I^j): j = 1 \text{ to } m)$, where $f(\cdot)$ reflects the cost of servicing the customers of cluster I^j if the facility $ROOT(I^j)$ is located optimally for the customers in I^j.

3. The Assignment Problem. The assignment problem [Ford and Fulkerson, 1962] [Papadimitriou and Steiglitz, 1982] may be stated as a rooted partitioning problem. Let S denote a set of persons and let T denote a set of tasks. The problem of assigning persons to tasks may be modeled as a problem of finding a rooted partition $I^1,\ldots,I^m$ of $S \cup T$ that minimizes $\sum (f(I^j): j = 1 \text{ to } m)$, where each cluster I^j has an element s^j of S and an element t^j of T, and $f(s^j,t^j)$ is the "cost" of assigning person s^j to task t^j.

4. **Exam Scheduling**. The version that we consider here is a special case of the quadratic assignment problem [Lawler, 1963] [Murty, 1976]

[Papadimitriou and Steiglitz, 1982]. The objective is to assign classes to m exam periods so as to minimize the total number of student conflicts that result from scheduling different classes in the same exam period. Let C_{ij} denote the number of students taking both class i and class j. Let $f(I^k)$ reflect the total number of student conflicts for period k of the exam schedule, i.e.,

$$f(I^k) = \sum_{i,j \in I^k} C_{ij}.$$

The problem is then to minimize $\sum_k f(I^k)$.

5. **Clustering problems.** This problem class includes data aggregation, clustering and other statistical problems [Charoen-Rajapark, 1987] [Hartigan, 1975] [Johnson and Wichern, 1982] [Lebart, Morineau and Warwick, 1984], as well as related problems in operations research [Garey and Johnson, 1979] [Mulvey and Beck, 1984]. Let S denote the set of elements to be clustered. The problem is to find an (unrooted) m-partition $I^1,\ldots,I^m$ of S that minimizes $\sum f(I^j: j = 1 \text{ to } m)$, where $f(\cdot)$ reflects the cost of cluster I^j. For example, $f(\cdot)$ could be a function of the information lost by aggregation for a data aggregation problem. These problems formulate as similar (unrooted) partitioning problems if the number of clusters is not specified in advance.

6. **Graph Coloring.** [Garey and Johnson, 1979] [Papadimitriou and Steiglitz, 1982]. We can state the graph coloring problem as an unrooted partitioning problem, as follows. Let $G = (S,A)$ denote a graph. Find a partition $I^1,I^2,\ldots.$ of S so as to minimize $\sum f(I^j: j = 1,2,\ldots.)$, where $f(\cdot)$ is given by

$$f(I^j) = \begin{cases} 0 & \text{if } I^j = \emptyset \\ 1 & \text{if } I^j \neq \emptyset \text{ and if arc } (r,s) \notin A \text{ for all } \{r,s\} \subseteq I^j \\ \infty & \text{otherwise} \end{cases}$$

As stated, the coloring problem is to minimize the number of colors needed to color the nodes of a graph, such that no arc in the graph has both endpoints with the same color. If we wish to determine such a coloring with exactly m colors, then we would formulate a similar m-partitioning problem.

7. **N-Dimensional Weighted Matching.** [Garey and Johnson, 1979] [Papadimitriou and Steiglitz, 1982]. The N-dimensional weighted matching problem may be formulated as an unrooted m-partitioning problem, as follows. Let W,X,Y,...,Z be q sets, each of cardinality m. Let R be a subset of $(W \times X \times Y \times \cdots \times Z)$, and let $S = (W \cup X \cup Y \cup \cdots \cup Z)$. The problem is to find an m-partition of S that minimizes $\sum (f(I^j): j = 1 \text{ to } m)$, where $f(\cdot)$ is given by

$$f(I^j) = \begin{cases} C_j & \text{if } |I^j \cap W| = 1,\ |I^j \cap X| = 1, \ldots, |I^j \cap Z| = 1, \\ & \quad \text{and } I^j \in R \\ \infty & \text{otherwise,} \end{cases}$$

and where C_j is the cost of the matching I^j. If $\sum (f(I^j)\colon j = 1 \text{ to } m)$ is finite, then we have found an N-dimensional matching with weight $\sum(I^j)$.

8. **Vertex Cover** [Garey and Johnson, 1979] [Papadimitriou and Steiglitz, 1982]. Let $G = (V,S)$ be a graph, and let m be an integer. The vertex covering problem is to form an m-partition of S so as to minimize $\sum (f(I^j)\colon j = 1 \text{ to } m)$, where $f(\cdot)$ is given by

$$f(I^j) = \begin{cases} 0 & \text{if } I^j = \emptyset \\ 1 & \text{if all edges in } I^j \text{ share a common endpoint} \\ \infty & \text{otherwise} \end{cases}$$

If $\sum (f(I^j)\colon j = 1 \text{ to } m)$ is finite, then we have found a vertex cover.

2. RELATED WORK

In this section, we motivate local search as an appropriate solution strategy for partitioning and rooted partitioning problems. To this end, we survey the literature that is devoted to local search methods for these problems. We find that existing neighborhood search techniques for this problem class are highly problem-dependent and limited in their ability to find good solutions. In addition, exact solution methods are limited in the problem size that they can handle. These findings open the way for the development of a generic local search strategy for rooted partitioning problems.

The importance of studying approximate solution methods for rooted partitioning problems is underscored by their inherent computational complexity. Most interesting rooted partitioning problems belong to the difficult class of NP-hard problems. For example, of the problems discussed in Section 1.3, all but the assignment problem are NP-hard.

Computational work over the past two decades has demonstrated that optimal solutions to most NP-hard problems that arise in practice are unattainable in reasonable amounts of computation time, for even moderately sized problem instances. This has led researchers to focus on heuristic solution methods for these problems. Although heuristic methods do not guarantee optimal solutions, one may sometimes obtain confidence bounds on a solution's goodness through statistical techniques; or deterministic bounds through worst case, a posteriori, or other bounding methods. Moreover, a

reasonably good, feasible solution is often entirely adequate from a practical standpoint.

One effective and general heuristic solution strategy is local search (See, for example, [Papadimitriou and Steiglitz, 1982] and [Llewellyn, Tovey and Trick, 1987]). These algorithms attempt to improve an initial solution by iteratively making small changes in the current solution, while maintaining feasibility. Thus they transform the starting solution into a local optimum via a series of simple transformations.

The archetypical application of local search is the edge-exchange, or "λ-change," procedures of Croes [1958], Lin [1965] and Lin and Kernighan [1973] for the Traveling Salesman Problem (TSP). This methodology has been successful for extremely large problems. For example, Johnson, McGeoch and Rothberg [1987] report solution values within a few percent of optimality for TSPs with up to 100,000 nodes, using 3-changes and the variable-depth method of Lin and Kernighan [1973].

The success of edge exchange methods for the TSP has spurred researchers to adapt them to other vehicle routing and scheduling problems. For example, Kanellakis and Papadimitriou [1980] extend these methods to the asymmetric TSP. Savelsbergh [1984, 1985] generalizes the work of Lin [1965] to the time-constrained TSP. Independently, Baker and Schaffer [1986] extend the work of Croes [1958] and Lin [1965] for the TSP to the VRP with time windows. Psaraftis [1983] develops efficient k-interchange procedures for the precedence constrained TSP. Solomon, Baker and Schaffer [1986] propose accelerated branch exchange heuristics for the VRP with time window constraints.

It is an indication of the power of the edge exchange concept that it is successful on a wide variety of vehicle routing and scheduling problems. Its success rests, in part, on the relative simplicity of feasibility checks for unconstrained single-vehicle and multi-vehicle problems. Unfortunately, the concept is not easy to apply to complex, constrained fleet planning problems. For example, the task is much more difficult for problems with time windows [Baker and Schaffer, 1986] [Solomon, Baker and Schaffer, 1986]. However, a clever use of data structures may reduce the complexity of checking feasibility [Savelsbergh, 1985]. Similarly, the addition of precedence constraints complicates the extension of λ-change methods to the multi-vehicle case. For example, the k-interchange procedures of Psaraftis [1983] for a

single vehicle precedence constrained problem do not directly apply to multiple vehicle problems, except insofar as they can be used to improve the cost of individual tours. The complexity and size of λ-change neighborhoods precludes using these methods for medium or large sized multi-vehicle problems. In general, complicating factors such as multiple vehicles, capacity restrictions, time windows and precedence constraints, make feasibility checks more difficult, thereby increasing the effort required to search a λ-change neighborhood.

A related class of local search procedures for partitioning problems involves transferring or exchanging elements between clusters. For example, Kernighan and Lin [1970] propose a variable-depth "swap" algorithm for the Uniform Graph Partitioning problem. Their method involves sequences of swaps of elements between the two subpartitions of a problem instance. A related method is the algorithm of Armour and Buffa [1963] (see also [Francis and White, 1974]) for facility layout problems, which attempts to improve a given solution by iteratively interchanging department locations. Similar to this is the "swapper heuristic" of Bodin and Sexton [1983] for multi-vehicle dial-a-ride problems. Their algorithm transfers single demands between vehicle tours if a lower cost solution results. These methods represent a departure from edge exchange methods because they transfer cluster elements instead of graph edges.

The neighborhood search strategy that we develop for rooted and unrooted partitioning problems generalizes the methods of both Kernighan and Lin [1970] and Bodin and Sexton [1983]. Its application to vehicle routing and scheduling problems also generalizes the edge exchange method of Croes [1958] and of Lin [1965], as well as the variable depth method of Lin and Kernighan [1973]. We describe our strategy in the next few sections.

3. CYCLIC TRANSFER NEIGHBORHOOD SEARCH

This section introduces and develops what we term cyclic transfer algorithms, a general class of local search procedures for partitioning and rooted partitioning problems. In section 3.1, we introduce and define cyclic transfers. In Section 3.2, we define a class of local neighborhoods based on cyclic transfers. In section 3.3, we describe data structures that enable us to search the class of neighborhoods efficiently. Finally, in section 3.4, we

discuss both theoretical and implementation issues associated with cyclic transfers.

3.1. CYCLIC TRANSFERS

The central concept for cyclic transfers is the iterative attempt to improve the total cost of a given solution $[I^1,\ldots,I^m]$ to a partitioning or rooted partitioning problem by transferring small numbers of elements among clusters. We formalize this idea in the framework of neighborhood search by using cyclic transfers to define a local neighborhood.

Let $[I^1,\ldots,I^m]$ be a rooted partition of $S \cup T$. Let ρ be a cyclic permutation of a subset of $\{1,\ldots,m\}$ written in cyclic form. For example, $\rho = (2\ 5\ 3)$ maps 2 into 5, it maps 5 into 3 and it maps 3 into 2. We also write $\rho(2)=5$, $\rho(5)=3$, and $\rho(3)=2$. We refer to the simultaneous transfer of elements from I^j to $I^{\rho(j)}$ for each j, as a **cyclic transfer**. Figure 1 illustrates a cyclic transfer.

Several special cases of cyclic transfers are of interest. First, a **cyclic Q-transfer** is a cyclic transfer in which each subset transferred from I^j to I^k is a member of a set Q. For example, Q might be the collection of all pairs of elements of S. Or it might be modified dynamically at each iteration of a cyclic transfer algorithm. This latter possibility also permits Q to be user-defined according to the special features of a problem instance and the current solution.

We say that a set $q \in Q$ is **feasible** for a solution I to a rooted partitioning problem if all of the elements of q belong to a common part of I, say I^k. In this case $q \subseteq I^k$, which we write as $I(q) = I^k$. Thus, I(q) denotes the cluster to which each element of q is assigned for the current solution. Clearly, every subset involved in a cyclic Q-transfer is feasible for the current solution.

If a cyclic transfer involves the transfer of exactly k elements from from I^j to $I^{\rho(j)}$ for each j, then we refer to it as a **cyclic k-transfer**. For example, the cyclic transfer in Figure 1 is a cyclic 2-transfer. Kernighan and Lin [1970] used a cyclic 1-transfer algorithm for the uniform graph partitioning problem, where the cycle length is limited to two arcs.

A natural extension of cyclic transfers is to permit **"acyclic" transfers** in which there is a simultaneous transfer of elements along permutations

(rather than cyclic permutations) of subsets of {1,2,...,m}. However, the extension may be simulated as a cyclic transfer in which dummy elements are transferred. For example, if the subset transferred from I^j to $I^{\rho(j)}$ is a set of dummy elements, then no real elements are transferred from I^j to $I^{\rho(j)}$, and the chain of transfers becomes acylic. We will not focus further attention on acyclic transfers since they comprise a special case of cyclic transfers.

A fourth case of interest we call **k-transfers**. A k-transfer is the simple transfer of k elements from one cluster to another. This situation corresponds to an acyclic k-transfer involving two clusters (where only dummy elements are transferred from one of the clusters). Bodin and Sexton [1983] used this concept in their dial-a-ride problem heuristic.

In this section we have introduced and defined the concept of cyclic transfers for rooted partitioning problems. Now, we define a local neighborhood based on this concept.

3.2. Cyclic Transfer Neighborhoods

Local search heuristics for combinatorial optimization problems are normally defined in terms of a local neighborhood [Papadimitriou and Steiglitz, 1982]. Formally, given an instance (E,c) of a combinatorial optimization problem, where E is the set of feasible solutions and c is the cost mapping, and given a feasible solution $r \in E$, a local search procedure chooses a neighborhood function

$$N : E \rightarrow 2^E$$

in which to search for a feasible solution $s \in N(r)$, for which $c(s) < c(r)$. If the search is successful, then the procedure may repeat the search at the new feasible solution s. If all feasible solutions in the neighborhood of r have cost at least as great as c(r), then a local optimum is attained.

We define the **cyclic transfer neighborhood** of a feasible solution r to a partitioning or rooted partitioning problem to be the set of feasible solutions reachable from r via a cyclic transfer. Rigorously, given an instance (E,c) of such a problem, the cyclic transfer neighborhood at a feasible solution $r \in E$ is

$$N(r) = \{s : s \in E \text{ and } s \text{ can be obtained from } r \text{ via a cyclic transfer}\}.$$

Analogous definitions hold for the cyclic-Q-transfer and cyclic-k-transfer neighborhoods of r.

Although we have defined cyclic transfers in a general setting, we do not propose to search the neighborhood for any but small values of k. Further, we will typically search the neighborhood approximately. The reason for limiting our search is that the neighborhood may be exponentially large. For example, if all clusters have a common number t of elements and if we consider only cyclic k-transfers involving p clusters, then the size of the neighborhood of r is

$$|N_{p,k}(r)| = (p-1)! \binom{m}{p} \binom{t}{k}^p .$$

The first term is the number of cyclic permutations of p clusters; the second term is the number of combinations of p clusters that can be taken from a set of m clusters; and the third term is the number of combinations of sets of k elements that can be taken, each set from one of p distinct clusters. This restricted neighborhood is exponentially large.

Let us now define the cost of a cyclic transfer. As above, we assume that cluster costs are independent. We say that the cost of a cyclic transfer is the change in objective function caused by the cyclic transfer. For example, suppose that $I = \{I^1, I^2, \ldots, I^p\}$ is the original set of p clusters involved in a cyclic transfer CT, and let $J = \{J^1, J^2, \ldots, J^p\}$ be the transformed set of clusters resulting from the cyclic transfer. Then the cost of CT is given by

$$\text{Cost}(CT) = \sum_{s=1}^{p} \left[f(I^s) - f(J^s) \right] .$$

Using these measures of neighborhood and cost, we may define local optimality for partitioning and rooted partitioning problems in terms of cyclic transfers. Let r be a feasible solution to a rooted partitioning problem. We say that r is **cyclic-transfer-optimal** if there are no negative cost cyclic transfers for r, that is, if

$$\sum [f(I^i) : I^i \in r] \leq \sum [f(I^j) : I^j \in s] \qquad \text{for all } s \in N(r).$$

Similarly, we say that r is **cyclic-Q-transfer-optimal** for some set Q (**cyclic-k-transfer-optimal** for some integer k) if there are no negative cost cyclic Q-transfers (cyclic k-transfers) for r.

Figure 2 demonstrates that the cyclic transfer neighborhood is not exact, i.e., a local optimum may not be a global optimum. Here there are 3 clusters comprised, as indicated, of the elements a_1, a_2, a_3, b_1, b_2, c_1, c_2. If $f(a_1,b_1,c_1) = 0$, and the cost of any other cluster is 1, then the solution of

Figure 2 is not optimal. But there are no negative cost cyclic transfers, hence the solution is cyclic transfer optimal. This implies that the cyclic transfer neighborhood is not exact, regardless of how many elements may be transferred from one cluster to another.

3.3. NEIGHBORHOOD SEARCH

In this section we develop a general heuristic methodology for searching a class of cyclic transfer neighborhoods. Our method uses an auxiliary graph to transform the search for negative cost cyclic transfers into a search for negative cost cycles on the graph. We formally relate properties of the auxiliary graph to properties of the cyclic transfer neighborhood. Finally, we show that the neighborhood search problem is itself NP-hard, and discuss why cyclic transfer neighborhood search is nonetheless a reasonable approach for solving rooted partitioning and related problems.

For notational ease, we present neighborhood search in terms of cyclic Q-transfers. The reader should note that our methods apply equally well to the general case where Q is not prespecified, as well as to special cases such as cyclic k-transfers.

Several alternate approaches exist for cyclic transfer neighborhood search for partitioning and rooted partitioning problems. First, consider an auxiliary graph H = (V,A), which has one vertex for each cluster of the partition, and one arc for each possible transfer of element subsets from one cluster to another. This approach has intuitive appeal because each arc represents the natural transfer of elements between clusters. Unfortunately, however, the costs of the arcs in H are not well defined. In particular, the cost of a transfer may be affected by other transfers that take place simultaneously. For example, the net change in travel distance due to moving a customer from one vehicle route to another may depend heavily on other changes made simultaneously to the two routes. These interaction effects are endemic to the structure of H, and preclude its effective use for rooted partitioning problems. For this reason, we elect not to pursue this approach further.

A more effective (and less intuitive) approach to searching cyclic transfer neighborhoods is the following. Let $I = (I^1, I^2, \ldots, I^m)$ represent a feasible solution to a rooted partitioning problem, i.e., a rooted partition

of S ∪ T. In order to identify a negative cost cyclic Q-transfer, we construct an **auxiliary graph** $G^Q = (V^Q, A^Q, C^Q)$, **where**

V^Q = {sets in Q feasible for I} has one vertex for each feasible set of Q

A^Q = {(i,j) : i,j∈V^Q, I(i) ≠ I(j), and {I(j)+i-j} is a feasible cluster}

C^Q = {C_{ij} : (i,j) ∈ A^Q} are arc costs (defined below)

The cost C_{ij} of each arc (i,j) in A^Q is equal to the increase (for a minimization problem) in the cost of the cluster I(j) due to simultaneously adding subset i to and removing subset j from I(j). This increase may be positive, in which case it is a "cost" in the normal sense of the word, or negative, in which case it is a "benefit". Rigorously,

$$C_{ij} = f(I(j)-j+i) - f(I(j)).$$

We emphasize that C_{ij} does not include any costs due to removing i from I(i) or due to adding j to any other cluster.

As a special case we denote the auxiliary graph for cyclic k-transfers by G^k. Here, V^k = {sets of k distinct elements of S all from the same cluster}, and A^k and C^k are defined as above.

In contrast with the graph H, the impact of any cyclic Q-transfer on cluster cost may be determined directly from the cost structure of G^Q. This is so because each arc cost accounts for interaction costs at its head vertex. We shall see that G^Q is very useful in finding solutions to partitioning and rooted partitioning problems.

Figure 2 shows the clusters of an example problem with seven elements and three clusters, A = $\{a_1,a_2,a_3\}$, B = $\{b_1,b_2\}$ and C = $\{c_1,c_2\}$. The corresponding cyclic 1-transfer and cyclic 2-transfer auxiliary graphs G^1 and G^2 appear in figures 3 and 4.

Whereas the size of the cyclic k-transfer neighborhood is exponentially large whenever the number of parts is not bounded, the size of the auxiliary graph is polynomially bounded for fixed k. In particular, the number of vertices in G^k is given by

$$|V^k| = \sum_{r=1}^{m} \binom{|I^r|}{k},$$

where $|I^r|$ is the number of elements in the r^{th} cluster. A lower bound on $|V^k|$ is given by the instance with exactly i elements per cluster. In this case, $\sum i_r = (m)(i)$, and the number of vertices in G^k is given by

$$|V^k| = m \binom{i}{k}.$$

Now we formally relate properties of the auxiliary graph to properties of the cyclic transfer neighborhood. As before, we assume that cluster costs are independent of one another. We define a **Cycle through Distinct Clusters** in G^Q to be a cycle whose vertices each correspond to element sets in different clusters. Using this definition, we may state necessary and sufficient conditions for the existence of a negative cost cyclic transfer at a feasible solution to a partitioning or rooted partitioning problem.

3.3.1. Theorem. Suppose that the cluster costs for a partitioning or rooted partitioning problem are independent, i.e., that cluster cost depends only on the elements of the cluster and not on the elements of other clusters. Then there is a negative cost cyclic Q-transfer for a given feasible solution if and only if there is a negative cost cycle through distinct clusters in the auxiliary graph G^Q.

Proof. Sufficiency: by definition of independent cluster costs, the cost of a cycle in G^Q is equal to the cost of the associated set of transfers of elements among clusters, if the cycle is through distinct clusters. But then the transfer is cyclic. The desired result follows.

Necessity: From the definition of the arc costs in G^Q, it is clear that the cost of a cyclic-Q-transfer at a feasible solution of a partitioning or rooted partitioning problem is equal to the cost of the corresponding cycle in G^Q, if cluster costs are independent. The result then follows immediately. ■

3.4. Computational Complexity Analysis

The following two lemmas and theorem show that each cyclic Q-transfer neighborhood search step is potentially NP-hard. Although we do not supply proofs for cyclic k-transfers, the reader should note that these results apply to this case equally well.

3.4.1. Lemma. **CYCLE THROUGH DISTINCT SUBPARTITIONS** is NP-complete.
INSTANCE: A directed graph $G = (V,A)$, and an m-partition $V^1, V^2, \ldots, V^m$ of V.
QUESTION: Does G contain a cycle that has at most one element of each of the subpartitions $V^1, V^2, \ldots, V^m$?

Proof. Transformation from HAMILTONIAN CIRCUIT. Given an instance of HAMILTONIAN CIRCUIT, we construct, in polynomial time, an instance of CYCLE THROUGH DISTINCT SUBPARTITIONS. Figure 5 illustrates the argument. Let $G_H = (V_H, A_H)$ be an instance of HAMILTONIAN CIRCUIT, with $n = |V_H|$ vertices. Construct a graph $G_S = (V_S, A_S)$ as follows. For each node $i \in V_H$, create n copies $i^1, i^2, \ldots, i^n$ in V_S. For each arc (i,j) in A_H, create n copies (i^k, j^{k+1}), where the nth copy is (i^n, j^1). Now, define a partition of G_S:

$$V^i = \{i^k \mid k=1,\ldots,n\} \quad \text{for } i \in V_H.$$

Thus, V^i is the set of copies of node i and $V^1, V^2, \ldots, V^n$ is an n-partition of V_S.

Suppose that G_S contains a cycle, C_S, through distinct subpartitions. Observe that every cycle in G_S through distinct subpartitions contains exactly n vertices. Then C_S corresponds to a Hamiltonian Circuit in G_H. Suppose, now, that G_H contains a Hamiltonian Circuit, C_H. By construction, C_H corresponds to a cycle through distinct subpartitions in G_S.

Therefore HAMILTONIAN CIRCUIT transforms to CYCLE THROUGH DISTINCT SUBPARTITIONS. Since CYCLE THROUGH DISTINCT SUBPARTITIONS $\in$ NP, and HAMILTONIAN CIRCUIT is NP-Complete, CYCLE THROUGH DISTINCT SUBPARTITIONS is NP-Complete also. ■

3.4.2. Lemma. The problem of finding a negative cost cycle through distinct clusters in G^Q is NP-hard.

Proof. Recall that G^Q is not a complete graph because it contains no arcs that correspond to infeasible transformations. In fact, any graph G is the auxiliary graph G^Q for some Q. In particular, G^Q may have the structure of Figure 5. The problem then reduces from CYCLE THROUGH DISTINCT SUBPARTITIONS. ■

3.4.3. Theorem. The problem of finding a negative cost cyclic Q-transfer for a feasible solution to a partitioning or rooted partitioning problem is NP-hard.

Proof. By Lemma 3.3.1, finding a negative cost cyclic transfer is equivalent to finding a negative cost cycle through distinct clusters in G^Q. The result then follows from Lemma 3.4.2. ■

This result contrasts with the polynomial-time complexity of identifying

negative cost cycles in a graph. Such cycles may be determined in O(nm) time using the well-known label correcting algorithm (see, for example, [Ahuja, Magnanti and Orlin, 1989]) or it may be solved in $O(n^{1/2}m \log nC)$ time using the scaling algorithm of [Gabow and Tarjan, 1987] or [Ahuja and Orlin, 1988], where n is the number of arcs, m is the number of nodes, and C is the magnitude of the largest arc cost in the graph.

Despite this discouraging computational complexity result, we believe that it is still reasonable in practice to search for cycles through distinct clusters. First, large potential cost benefits may motivate one to expend great effort in order to marginally improve a given solution. Second, cyclic transfer techniques appear to be quite effective on practical problems, even though they are not guaranteeed to be optimal. Recent work [Thompson and Psaraftis, 1989] has successfully employed a hybrid approach of heuristic methods to search the cyclic transfer neighborhood for several important classes of vehicle routing and scheduling problems. Their results demonstrate the practical utility of approximate cyclic transfer neighborhood search.

4. SPECIAL CLUSTER COST STRUCTURES

This section derives conditions under which one may find negative cost cyclic transfers or prove there are none, in polynomial time. However, these conditions are quite restrictive, and do not apply to rooted partitioning problems other than the assignment problem and its close relatives.

4.1. Separability and Interaction

First we define two terms, "separability" and "interaction". Let i and j be vertices of G^Q corresponding to element sets from distinct clusters I and J, respectively.

Separability is an attribute of an auxiliary graph arc cost. We say that a cost C_{ij} is separable if (J+i) and (J-j) are feasible clusters, and if

$$f(J+i-j) - f(J) = [f(J+i) - f(J)] + [f(J-j) - f(J)].$$

Equivalently, C_{ij} is separable if

$$C_{ij} = A_i(J) + R_j(J),$$

where $A_i(J) = f(J+i) - f(J)$ = cost of adding element set i to J

and $R_j(J) = f(J-j) - f(J)$ = cost of removing set j from J.

This implies that the cost of adding i to J is independent of the presence of j on J, i.e.,

$$A_i(J-j) = A_i(J).$$

For example, consider the Euclidean vehicle routing problem illustrated in Figure 6, where f(T) is the cost of an optimal TSP tour on the node set T = {D,E,F,G}. Here, the auxiliary graph arc cost C_{AE} is separable because the optimal routings of both {T+A} and {T+A-E} insert A between G and D, regardless of the presence of E. C_{AD} is not separable, however, because the optimum routing of {T+A-D} inserts A between G and E, not between G and D.

Interaction is an attribute of a pair of auxiliary graph arc costs. It implies that the arc costs are dependent. Under our base assumption of independent cluster costs, arcs whose heads correspond to element sets in different clusters may not interact, since otherwise cluster costs would be dependent. Hence, we consider only arcs pointing into vertices of the same cluster as potentially interacting. Similarly, we define interaction only for arcs whose endpoints correspond to distinct sets of elements in the partitioning problem, since otherwise the transfers implied by the arcs would interfere with one another. Let E(v) denote the set of elements that corresponds to vertex v. Let j and q be vertices of G^Q belonging to cluster J, let i and p be vertices of G^Q not belonging to cluster J, and let

$$E(i) \cap E(j) = \emptyset$$

$$\text{and} \quad E(p) \cap E(q) = \emptyset.$$

We say that C_{ij} and C_{pq} interact if the sum $C_{ij}+C_{pq}$ does not represent the true cost of simultaneously adding i and p to J, and removing j and q from J. Formally, C_{ij} and C_{pq} interact if

$$f(J+i-j+p-q) - f(J) \neq [f(J+i-j) - f(J)] + [f(J+p-q) - f(J)]$$

or, equivalently, if

$$f(J+i-j+p-q) - f(J+i-j) \neq [f(J+p-q) - f(J)].$$

Consider the Euclidean Vehicle Routing Problem. Here, the cost of a cluster is the cost of an optimal TSP tour through the cluster's demand locations. In this problem, interaction occurs when the optimal order in which vehicle ROOT(J) visits the node set {J+i-j} changes if node p is added to and node q removed from J, where j and q correspond to distinct element sets in cluster J, and i and p correspond to element sets on other clusters. Interaction also occurs if the sequencing is the same, but if at least one node of the set i is inserted into j's route immediately next to a node of set

q. Non-interaction implies that the optimal insertion cost of i onto route J is independent of the presence of p and q, and that the optimal insertion cost of p onto route J is independent of i and j. In general, if rerouting is required for vehicle routes, then the costs of arcs into vertices of the same cluster will interact. If arc costs reflect simple insertions into existing routes, then costs will sometimes, but usually not, interact.

In Figure 6, which shows a route for an Euclidean Vehicle Routing Problem with two additional nodes, A and B, C_{AF} and C_{BD} interact, while C_{AD} and C_{GF} do not.

4.2. Logical Independence of Separability and Interaction

In this section, we show that, although arc cost separability and non-interaction are related concepts, they are logically independent.

First, we define two terms, **total separability** and **total non-interaction**. A rooted partitioning problem's auxiliary graph arc costs are **totally separable** if, for every auxiliary graph G^Q of each feasible solution to each problem instance, all arc costs $C_{ij} \in C^Q$ share the following property:

$$C_{ij} = A_i(ROOT(j)) + R_j(ROOT(j))$$

where $A_i(ROOT(j))$ is fixed for each choice of i and ROOT(j)

and $R_j(ROOT(j))$ is fixed for each choice of j.

This definition includes, as a special case, situations where all arc costs of all auxiliary graphs G^Q of all problem instances are separable. It also includes the more general case where the clusters {I(j)+i} or {I(j)-j} are infeasible. This is true, for instance, of the Hitchcock transportation problem, where there are demand constraints on the "clusters." In the general case, $A_i(ROOT(j))$ is the "effective cost" of adding i to cluster I(j), under the assumption that the addition of i to cluster I(j) is part of a feasible transformation of I(j). Similarly, $R_j(I(j))$ is the "effective cost of removing j from its cluster, under the assumption that the removal of j from cluster I(j) is part of a feasible transformation of I(j).

We say that a rooted partitioning problem's auxiliary graph arc costs are **totally nonseparable** if all arc costs of all auxiliary graphs for all problem instances are non-separable. Similarly, a rooted partitioning problem's auxiliary graph arc costs are **totally interacting** (respectively, **totally non-interacting**) if all arc costs of all auxiliary graphs for all problem

instances are interacting (respectively, non-interacting).

Consider the following examples of rooted partitioning problems.

EXAMPLE 1. $f(I) = |I|^2$.
In this case, all G^k arc costs are non-interacting, and the problem has total nonseparability.

To show non-interaction of all arc costs in G^k, observe that, for every feasible solution to every instance of the problem, every set of cyclic k-transfers leaves the number of elements in each cluster, and hence the cluster cost, unchanged. Therefore, the arc costs in G^k are non-interacting.

To show total nonseparability, consider the arc cost C_{ij}. Let the vertices i and j correspond to i and j elements in their respective clusters, and let J be the number of elements in I(j). We have

$$C_{ij} = (J + i - j)^2 - J^2 = i^2 + j^2 + 2J(i-j) - 2ij.$$

But, $$R_j(J) = (J-j)^2 - J^2 = j^2 - 2jJ,$$

and $$A_i(J) = (J+i)^2 - J^2 = i^2 - 2iJ,$$

so that $$A_i(J) + R_j(J) = i^2 + j^2 + 2J(i-j).$$

Then $$C_{ij} \neq A_i(J) + R_j(J),$$

implying total nonseparability.

EXAMPLE 2. The Hitchcock transportation problem, i.e., the problem of assigning n items to m subsets, subject to lower and upper bounds on the number of items assigned to subset I. In this case, we have total non-interaction and total separability because each assignment has fixed cost independent of any other assignments that are made, as long as the total set of assignments is feasible.

EXAMPLE 3. Multi-Vehicle Routing. Here we have total non-separability and total interaction.

EXAMPLE 4. The rooted partitioning problem in which there are elements of two "colors", say blue and red, and where the cost of cluster I is the square of the number of blue elements in the cluster, m_I, i.e.,

$$f(I) = (m_I)^2.$$

Consider the problem instance with two clusters, and consider the feasible

solution in which one cluster contains only blue elements (cluster B), and the other only red elements (cluster R). In this case, we have separability of all arc costs in G^k, and interacting arc costs as well.

First, we show separability. For all sets of k blue elements b and all sets of k red elements r,

$$R_b(B) = (m_B-k)^2 - (m_B)^2 = k^2 - 2m_B$$
$$A_b(R) = k^2$$
$$R_r(R) = 0$$
$$A_r(B) = 0.$$

Further,

$$C_{br} = k^2$$
$$C_{rb} = (m_B-k)^2 - (m_B)^2 = k^2 - 2m_B,$$

implying

$$C_{rb} = A_r(B) + R_b(B) \qquad \text{for all r,b.}$$
$$C_{br} = A_b(R) + R_r(R) \qquad \text{for all r,b.}$$

Therefore all costs in the auxiliary graph are separable.

Now, we show interaction. Consider two sets of k blue elements, say b and d, and two sets of k red elements, say r and s. We have

$$f(B+r+s-b-d) - f(B) = (m_B-2k)^2 - (m_B)^2 = 4k^2 - 4km_B$$

but $$C_{rb} + C_{sd} = 2k^2 - 4km_B,$$

so that $$C_{rb} + C_{sd} \neq f(B+r+s-b-d) - f(B).$$

Therefore all arcs pointing into B interact with one another. Furthermore,

$$f(R+b+d-r-s) - f(R) = 4k^2$$

but $$C_{br} + C_{ds} = 2k^2,$$

so that $$C_{br} + C_{ds} \neq f(R+b+d-r-s) - f(R) \quad .$$

Therefore all arcs pointing into R interact with one another.

4.2.1. Theorem. Separability and interaction are logically independent.

Proof. Immediate, from Examples 1 through 4.

The previous theorem does <u>not</u> state that total separability and total interaction are logically independent. In fact, the following theorem and its corollary show that they are logically dependent.

4.2.2. Theorem. Separability of all auxiliary graph arc costs for all feasible solutions to an instance of a rooted partitioning problem implies

non-interaction of these costs.

Proof. Suppose that all arc costs for an instance of a rooted partitioning problem instance are separable, i.e., for each cluster J,

$$C_{ij} = A_i(J) + R_j(J) \qquad \text{for all } i \notin J,\ j \in J,$$

or, equivalently,

$$A_i(J) = A_i(J-j) \qquad \text{for all } i \notin J,\ j \in J.$$

First, observe that an inductive argument from $J=\emptyset$ proves that both $A_i(J)$ and $R_j(J)$ depend solely on the root of J, and not at all on the elements of J. Then, for all $j,q \in J$ and all $i,p \notin J$, we have

$$\begin{aligned} &f(J+p+i-q-j) - f(J) \\ &\quad = f(J) + A_p(J) + A_i(J) + R_q(J) + R_j(J) - f(J) \\ &\quad = f(J) + A_p(J) + R_q(J) - f(J) + A_i(J) + R_j(J) + f(J) - f(J) \\ &\quad = f(J+p-q) - f(J) + f(J+i-j) - f(J) \\ &\quad = C_{ij} + C_{pq}, \end{aligned}$$

so that C_{ij} and C_{pq} do not interact. This is the desired result. ■

This theorem easily generalizes to include all instances of a rooted partitioning problem. Formally, we have

4.2.3. Corollary. Total separability of auxiliary graph arc costs for a rooted partitioning problem implies that the costs are totally non-interacting.

Proof. Immediate. ■

These theorems show the logical independence of separability and non-interaction of auxiliary graph arc costs. However, in practice the two often occur together. For example, all of the examples of Section 1.3 except the assignment problem have arc costs that are simultaneously interacting and non-separable. Moreover, the assignment problem's auxiliary graph arc costs are totally separable and, hence, totally non-interacting. In fact, the assignment problem is the only problem with total separability of auxiliary graph arc costs. Formally,

4.2.4. Theorem. If a problem has auxiliary graph arc costs that are totally separable, then it is equivalent to an assignment problem.

Proof. Total separability implies that the cost of each arc in the

auxiliary graph is given by

$$C_{ij} = A_i(ROOT(j)) + R_j(ROOT(j)).$$

Then, for each cluster J,

$$f(J) = \sum_{j \in J} A_j(ROOT(J)),$$

and, further, for all elements i and all clusters J,

$$A_i(J) = R_i(J).$$

This we recognize as equivalent to an assignment problem. ■

It is interesting to note that the assignment problem is solvable in polynomial time, while most other rooted partitioning problems are NP-hard. This result suggests a relationship between computational complexity and auxiliary graph arc cost separability. Indeed, this theorem demonstrates that total separability of auxiliary graph arc costs implies polynomial-time solvability for a rooted partitioning problem.

4.3. The Effect of Separability and Interaction

In this section, we develop additional sufficient conditions for the existence of negative cost cyclic transfers and k-transfers. These theorems characterize the effect that separability and interaction have on the relationship between negative cost cycles in G^Q and negative cost cyclic transfers.

Two lemmas provide the base for sufficient conditions for cyclic transfers. First, we show that under conditions of separability, a cycle in G^Q that contains multiple element sets from a single cluster decomposes into a set of cycles through distinct clusters. Then we show that a negative cost cycle of separable costs in G^Q contains a negative cost cycle through distinct clusters. This provides the desired sufficient conditions.

4.3.1. **Lemma**. Suppose that there is a negative cost simple r-cycle C_r in G^Q, for $r \geq 4$. Suppose further that two or more vertices of C_r, say $\{N_1, N_2, \ldots, N_z\}$, correspond to elements from the same cluster. If the costs of the arcs of C_r into N_1, $N_2, \ldots,$ and N_z are separable, then there is a negative cost simple p-cycle in G^Q, containing exactly one of $\{N_1, N_2, \ldots, N_z\}$,

for some $p \leq r-2$, whose costs are separable.

Proof. Without loss of generality, denote C_r by the index set $[1,2,3,\ldots,v-1,v,v+1,\ldots,r,1]$, where 1 and v denote vertices on the same cluster. Let

$A_i(j)$ = the separable cost of adding element set i to the cluster of set j

R_j = the separable cost of removing element set j from its cluster.

Then the cost of C_r is given by

$$C[1,2,..,r,1] = C_{12} + C_{23} + \cdots + C_{v-2,v-1} + C_{v-1,v} + C_{v,v+1} + \cdots + C_{r-1,r} + C_{r1}.$$

By separability,

$$\begin{aligned} C_{v-1,v} + C_{r1} &= (A_{v-1}(v) + R_v) + (A_r(1) + R_1) \\ &= (A_{v-1}(v) + R_1) + (A_r(1) + R_v) \\ &= C_{v-1,1} + C_{rv}, \end{aligned}$$

so that

$$\begin{aligned} C[1,2,..,r,1] &= C_{12} + C_{23} + \cdots + C_{v-2,v-1} + C_{v-1,1} + C_{v,v+1} + \cdots + C_{r-1,r} + C_{rv} \\ &= C[1,2,\ldots,v-2,v-1,1] + C[v,v+1,\ldots,r-1,r,v]. \end{aligned}$$

Thus, C_r decomposes into two subcycles, $[1,2,\ldots,v-1,1]$ and $[v,v+1,\ldots,r,v]$. Since the original cycle has negative cost, then at least one of the subcycles has negative cost as well. Now, each subcycle contains at least two vertices. This means that each subcycle has at most r-2 nodes. Then there exists a negative cost p-cycle through one of the nodes on the repeated route, for $p \leq r-2$. By induction on r, we attain the desired result, since separability is maintained throughout. ■

4.3.2. Lemma. Suppose that there is a negative cost cycle of separable costs in G^Q. Then there is a negative cost cycle through distinct clusters in G^Q, whose costs are separable.

Proof. By flow decomposition, any cycle in a graph may be expressed as the sum of simple cycles. If the original cycle has negative cost, then at least one of the subcycles has negative cost as well. Then there is a negative cost simple cycle with separable costs in G^Q. Repeated application of Lemma 4.3.1 yields the desired result, since separability is maintained throughout. ■

4.3.3. Theorem. Sufficient conditions for the existence of a negative cost cyclic transfer at a feasible solution to a rooted partitioning problem are the following.

1. Cluster costs are independent and additive, and,
2. There is a negative cost cycle of separable costs in the auxiliary graph G^Q.

Proof. Immediate, from Theorem 3.3.1 and Lemma 4.3.2. ■

4.3.4. Theorem. Under conditions of separability, necessary and sufficient conditions for the existence of a negative cost cyclic transfer at a feasible solution to a rooted partitioning problem are the following.

1. Cluster costs are independent and additive, and,
2. There is a negative cost cycle in the auxiliary graph G^Q.

Proof. Immediate, from theorems 3.3.1 and 4.3.3. ■

This result contrasts with the necessary and sufficient conditions of Theorem 3.3.1. If all arcs in G^Q are separable, then the weaker conditions of Theorem 4.3.3 are necessary and sufficient as well. This demonstrates the fundamental importance of separability for rooted partitioning problems.

We now examine sufficient conditions for the existence of negative cost k-transfers. Recall that a k-transfer is the transfer of k elements from one cluster to another, and is equivalent to an acyclic transfer involving two clusters. The cost of the transfer of element set i from cluster I to cluster J is given by

$$C^{Tr}{}_i(J) = A_i(J) + R_i(I),$$

where $A_i(J)$ and $R_i(I)$ are as defined above.

4.3.5. Theorem. Suppose that there exists in G^Q a negative cost cycle of separable and non-interacting costs. Then there exists a negative cost k-transfer.

Proof. By Lemma 4.3.2, there is a negative cost cycle through distinct clusters in G^Q, say C^P, with separable costs. For notational convenience let node p+1 refer to node 1, and let C^P = [1,2,...,p,1]. By definition of separable and non-interacting costs and under conditions of independent and additive cluster costs, the cost of C^P, C[1,2,..,p,p+1], is given by

$$C[1,2,\ldots,p,p+1] = \sum_{i=1}^{p} [A_i(I(i+1)) + R_{i+1}(I(i+1))]$$

$$= \sum_{i=1}^{p} [A_i(I(i+1)) + R_i(I(i))]$$

$$= \sum_{i=1}^{p} C^{Tr}{}_i(I(i+1)) \quad < \quad 0.$$

The inequality follows by assumption: $C[1,\ldots,p,p+1] < 0$. Then at least one of $\{C^{Tr}{}_i(I(i+1)) : i = 1,2,\ldots,p\}$ is negative, since otherwise $C[1,2,..,p,p+1]$ must be nonnegative, a contradiction. Therefore there exists a negative cost k-transfer. ■

4.3.6. Theorem. Suppose that there exists in G^Q a negative cost cycle through distinct clusters, C_p, with separable costs. Then there exists a set of negative cost k-transfers whose total cost is at least as negative as C_p.

Proof. As in the previous theorem, let node p+1 refer to node 1, and let $C^P = [1,2,\ldots,p,1]$. Then the cost of C^P, $C[1,2,..,p,p+1]$, is given by

$$C[1,2,\ldots,p,p+1] = \sum_{i=1}^{p} C^{Tr}{}_i(I(i+1)) \quad < \quad 0.$$

But

$$\sum_{i=1}^{p} C^{Tr}{}_i(I(i+1)) \quad \geq \quad \sum_{i=1}^{p} \left[C^{Tr}{}_i(I(i+1)) : CTr_i(I(i+1)) < 0 \right]$$

Therefore the set of k-transfers $\{CTr_i(i+1) : CTr_i(i+1)<0\}$ has cost at least as negative as C_p. ■

The results of this section point out the importance of separability and interaction when dealing with rooted partitioning problems. In related computational work [Thompson, 1988] [Thompson and Psaraftis, 1989], we develop algorithms that exploit the presence of separability in specific rooted partitioning problems.

5. FINDING NEGATIVE COST CYCLIC TRANSFERS

In previous sections, we proved that the search for negative cost cyclic transfers transforms into a search for negative cost cycles through distinct clusters in G^Q. Furthermore, we showed that this problem is NP-hard, except under restricted conditions such as total separability. In this section, we

investigate the suitability of optimization and heuristic methods for solving the general problem in practice.

5.1. Suitability of Optimization Approaches

Several well known optimization techniques solve, in polynomial time, the problem of finding minimum cost sets of cycles in a graph. For example, the general problem of finding a minimum cost collection of vertex-disjoint simple cycle cover transforms into an assignment problem, as follows. Construct a bipartite graph G' with two vertices i_a and i_b for each vertex i in the original graph G. For each arc (i,j) in G, construct arc (i_a, j_b) in G' and set its cost equal to the cost of arc (i,j) in G. Then, for each i, construct arc (i_a, i_b) in G' and set its cost equal to zero. This problem is solvable in $O(n^{1/2}m \log nC)$ time [Gabow and Tarjan, 1987] [Ahuja and Orlin, 1988], where n and m are the number of edges and vertices, respectively, in the graph, and where C is the magnitude of the largest edge weight.

Alternatively, shortest path algorithms are able to identify negative cost cycles in a graph in polynomial time. Of these, label correcting methods and dynamic programming methods require O(nm) time (see, for example, [Ford and Fulkerson, 1962] or [Ahuja, Magnanti and Orlin, 1989]). Indeed, label correcting is a generalization of dynamic programming. Typically better (but not strongly polynomial) $O(n^{1/2}m \log nC)$ algorithms are due to [Gabow and Tarjan, 1987] [Ahuja and Orlin, 1988], where n and m are the number of edges and vertices, respectively, in the graph, and where C is the magnitude of the largest edge weight. If we wish to find single negative cost cycles, then these methods are appropriate. With nonseparable costs, however, it is difficult to adapt these techniques to finding negative cost cycles through distinct clusters. Nevertheless, these methods can be used to establish that there is no negative cost circuit, or to provide bounds on the maximum improvement possible with a cyclic transfer.

Finally, there are other enumeration methods, such as branch and bound and complete enumeration. These are powerful and have proven effective for a wide variety of difficult combinatorial problems, but are computationally expensive. For this reason, these methods are not suitable for iterative algorithms such as cyclic transfer neighborhood search.

5.2. Heuristic Approaches

We now examine some heuristic approaches for finding negative cost cycles through distinct clusters in G^Q.

Truncated enumeration is a classical heuristic tool. Traditionally, one enumerates feasible solutions until reaching a computation time limit, at which point one stops. Applying this method to the problem of finding a negative cost cycle through distinct clusters in G^Q, one would enumerate cycles through distinct clusters until finding one with negative cost. In this case the cycle found would probably not be the most negative one.

Another possibility for heuristic search is to limit the length of the cycles under consideration, e.g., to two or three. By construction, all 2-cycles and 3-cycles are through distinct clusters, since there are no arcs in G^Q between vertices that correspond to the same cluster. This fact simplifies the enumeration process, since we may then avoid explicit checking for cluster repetition on a cycle.

A third avenue is variable depth search. This general methodology has worked very well on the Traveling Salesman Problem [Lin and Kernighan, 1973] and related problems. Applied to our approach, we could vary either the set Q itself, the or the cycle length.

A variable-Q heuristic would start with an initial set Q, and a cycle through distinct clusters in G^Q. Then it would attempt to improve the cost of the cycle by changing the number of elements transferred among the clusters represented on the cycle, i.e., by varying Q. This could be done on an arc-by-arc basis or simultaneously for the entire cycle. The major difficulty with this approach is that computing the relevant arc costs at each iteration of the algorithm is computationally burdensome. Indeed, experimental work [Thompson, 1988] has shown that this approach is inferior to variable cycle-length methods.

The second type of variable-depth cyclic transfer search is "variable-p," where the cycle length p varies. This approach iteratively attempts to improve the cost of a cycle by increasing its length, i.e., by replacing individual arcs on the cycle with pairs of arcs. In order to maintain a cycle through distinct clusters, one may add only vertices of clusters not yet represented on the cycle. This check marginally increases total computation time. Moreover, holding Q fixed considerably reduces the overall computation

time vis-a-vis the variable-Q approach.

These considerations lead us to propose the following approach for finding negative cost cycles through distinct clusters in G^Q. First, identify the most negative cost 2-cycle or 3-cycle in G^Q using a dynamic programming recursion. Then use a variable-p approach to attempt to simultaneously reduce the cycle's cost and increase its length, while maintaining a cycle through distinct clusters. Recent work [Thompson, 1988] [Thompson and Psaraftis, 1989], provides computational results using this method for several vehicle routing and scheduling problems. Overall, it proves to be an efficient method for finding negative cost cyclic transfers, and appears to be an effective means of solving practical rooted partitioning problems.

6. CONCLUSION

In this paper, we have developed the theory of cyclic transfers, a novel and effective class of local search algorithms for a broad group of combinatorial problems, namely partitioning and rooted partitioning problems.

First, we defined the class of partitioning and rooted partitioning problems. We illustrated the breadth and importance of this problem class by presenting a variety of practically and theoretically motivated rooted partitioning problems. Then we demonstrated the limitations of optimizaton and heuristic methods for solving these problems.

Next, we defined cyclic transfers, used them to characterize a search neighborhood for rooted partitioning problems, and developed a data structure, the auxiliary graph G^Q, to aid in searching the cyclic transfer neighborhood. We showed that searching the neighborhood is an NP-hard problem. Then we derived necessary and sufficient conditions for the existence of negative cost cyclic transfers in terms of negative cost cycles in the auxiliary graph. Finally, we investigated polynomial-time algorithms for searching restricted portions of cyclic transfer neighborhoods.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge Professors Harilaos N. Psaraftis and Amedeo R. Odoni of MIT for their valuable comments during the course of this research.

This research has been supported by a United Parcel Service Foundation fellowship, the Center for Transportation Studies at M.I.T., the Office of Naval Research contract number N00016-83-K-0220, an AFOSR grant 88-0088, and an NSF Presidential Young Investigator fellowship.

REFERENCES

Ahuja, R.K., T.L. Magnanti and J.B. Orlin (1989), "Network Flows," Working Paper No. 2059-88, Sloan School of Management, M.I.T., Cambridge, Mass., to appear in Handbooks in Operations Research and Management Science, Volume 1, Optimization.

Ahuja, R.K. and J.B. Orlin (1988), "New Scaling Algorithms for the Assignment and Minimum Cycle Mean Problems," Technical Report, Sloan School of Management, M.I.T., Cambridge, Mass.

Armour, G.C. and E.S. Buffa (1963), "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities," Mgmt. Sci. 9, 294-309.

Baker, E.K., and J.R. Schaffer (1986), "Solution Improvement Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," American Journal of Mathematical and Management Science, 6:3 & 4, 261-300.

Bodin, L.D., B.L. Golden, A.A. Assad and M. Ball (1983), "Routing and Scheduling of Vehicles and Crews: The State of the Art," Comp. Opns. Res. 10(2), 63-211.

Bodin, L.D. and T.R. Sexton (1983), "The Multi-Vehicle Subscriber Dial-A-Ride Problem," Working Paper 82-005, College of Business and Management, University of Maryland at College Park.

Chandy, K. and T. Lo (1973), "The Capacitated Minimum Spanning Tree," Networks 3(2), 173-182.

Charoen-Rajapark, C. (1987), Clustering to Minimize (Maximize) the Within-Cluster (Between-Cluster) Sum of Squares: Complexity, Optimization Algorithms, and Heuristics, Ph.D. thesis, MIT, Cambridge, Mass.

Christofides, N., A. Mingozzi and P. Toth (1979), "The Vehicle Routing Problem," Chapter 11 in Combinatorial Optimization, ed. by N. Christofides, A. Mingozzi, P. Toth and C. Sandi; John Wiley & Sons, Bath.

Croes, G.A., (1958), "A Method for Solving Traveling-Salesman Problems," Opns. Res. 6, 791-812.

Florian, M. and M. Klein (1971), "Deterministic Production Planning with Concave Costs and Capacity Constraints," Mgmt. Sci. 18, 12-20.

Ford, L.R.,Jr., and D.R. Fulkerson (1962), Flows in Networks, Princeton University Press, Princeton, N.J.

Gabow, H.N. and R.E. Tarjan (1987), "Faster Scaling Algorithms for Graph Matching," Technical Report, Department of Computer Science, Princeton University, Princeton, N.J.

Garey, M.R. and D.S. Johnson (1979), Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., San Francisco.

Golden, B.L. and A.A. Assad (1986), "Perspectives on Vehicle Routing: Exciting New Developments," Working Paper, College of Business and Management, University of Maryland, College Park.

Hadley,G. and T.M. Whitin (1963), Analysis of Inventory Systems, Prentice Hall, Englewood Cliffs, N.J.

Handler, G.Y. and P.B. Mirchandani (1979), Location on Networks: Theory and Algorithms, The MIT Press, Cambridge, Mass.

Hartigan, J.A. (1975), Clustering Algorithms, John Wiley & Sons, New York.

Jaw, J.J., A.R. Odoni, H.N. Psaraftis and N.H.M. Wilson (1982), "A Heuristic Algorithm for the Multi-Vehicle Many-to-Many Advance Request Dial-A-Ride Problem," Working Paper MIT-UMTA-82-3, M.I.T., Cambridge, Mass.

Johnson, D.S., L.A. McGeoch, and E.E. Rothberg (1987), "Near-Optimal Solutions to Very Large Traveling Salesman Problems," presented at the TIMS-ORSA joint national meeting, New Orleans, May.

Johnson, R.A. and D.W. Wichern (1982), Applied Multivariate Statistical Analysis, Prentice-Hall, Englewood Cliffs, N.J.

Kanellakis, P.C. and C.H. Papadimitriou (1980), "Local Search for the Asymmetric Traveling Salesman Problem," Opns. Res. 28(5), 1086- 1099.

Kernighan, B.W. and S. Lin (1970), "An Efficient Heuristic Procedure for Partitioning Graphs," BSTJ 49(2), February, 291-307.

Kirkpatrick, S., C.D. Gelatt, Jr. and M.P. Vecchi (1983), "Optimization by Simulated Annealing," Science 220, 671-680.

Kolen, A.W.J., A.H.G. Rinnooy Kan and H.W.J.M. Trienekens (1987), "Vehicle Routing With Time Windows," Opns Res. 35(2), 266-273.

Larson, R.C. and A.R. Odoni (1981), Urban Operations Research, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Lawler, E.L. (1963), "The Quadratic Assignment Problem," Mgmt. Sci. 9, 586-599.

Lebart, L., A. Morineau and W. Warwick (1984), Multivariate Descriptive Statistical Analysis, John Wiley & Sons.

Lin, S. (1965), Computer Solutions to the Traveling Salesman Problem," Bell System Tech. J. 44, 2245-2269.

Lin, S. and B.W. Kernighan (1973), "An Effective Heuristic Algorithm for the Traveling Salesman Problem," Opns. Res. 21, 498- 516.

Llewellyn, D.C., C. Tovey and M. Trick (1988), "Local Optimization on Graphs," DAM, to appear.

Magnanti, T.L. (1981), "Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects," Networks 11, 179-213.

Mulvey, J.M. and M.P. Beck (1984), "Solving Capacitated Clustering Problems," Eur. J. Opnl. Res. 18:339-348.

Murty, K.G. (1976), Linear and Combinatorial Programming, John Wiley & Sons, New York.

Odoni, A.R. (1974), Location of Facilities on a Network: A Survey of Results, Technical Report TR-03-74, Operations Research Center, MIT, Cambridge, Mass.

Papadimitriou, C.H. and K. Steiglitz (1982), Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Psaraftis, H.N. (1983), "K-Interchange Procedures for Local Search in a Precedence-constrained Routing Problem," Eur. J. Opnl. Res. 13, 391-402.

Psaraftis, H.N., J.B. Orlin, D. Bienstock and P.M. Thompson (1985), "Analysis and Solution Algorithms of Sealift Routing and Scheduling Problems," Working Paper #1700-85, Sloan School of Management, M.I.T., Cambridge, Mass.

Savelsbergh, M.W.P. (1984), "Local Search in Routing Problems with Time Windows," Report Os-R8409, Department of Operations Research and System Theory, Centre for Mathematics and Computer Science, Amsterdam.

Savelsbergh, M.W.P. (1985), "Local Search in Routing Problems with Time Windows," Ann. Opns Res. 4, 285-305.

Solomon, M.M., E.K. Baker, J.R. Schaffer (1986), "Vehicle Routing and Scheduling Problems with Time Window Constraints: Efficient Implementations of Solution Improvement Procedures," Working Paper #87-03, College of Business Administration, Northeastern University, Boston, Mass.

Solomon, M.M. and J. Desrosiers (1987), "Time Window Constrained Routing and Scheduling Problems: A Survey," Working Paper, February.

Thompson, P.M. (1988), Local Search Algorithms for Vehicle Routing and Other Combinatorial Problems, PhD Thesis, M.I.T., Cambridge, Mass.

Thompson, P.M. and H.N. Psaraftis (1989), "Local Search Algorithms for Multi-Vehicle Routing and Scheduling Problems," in preparation.

Wagner, H.M. and T. Whitin (1958), "Dynamic Version of the Economic Lot Size Model," Mgmt. Sci. 5, 89-96.

Francis, R.L. and J.A. White (1974), Facility Layout and Location: An Analytical Approach, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Figure 1. The Effect of a Cyclic 2-Transfer

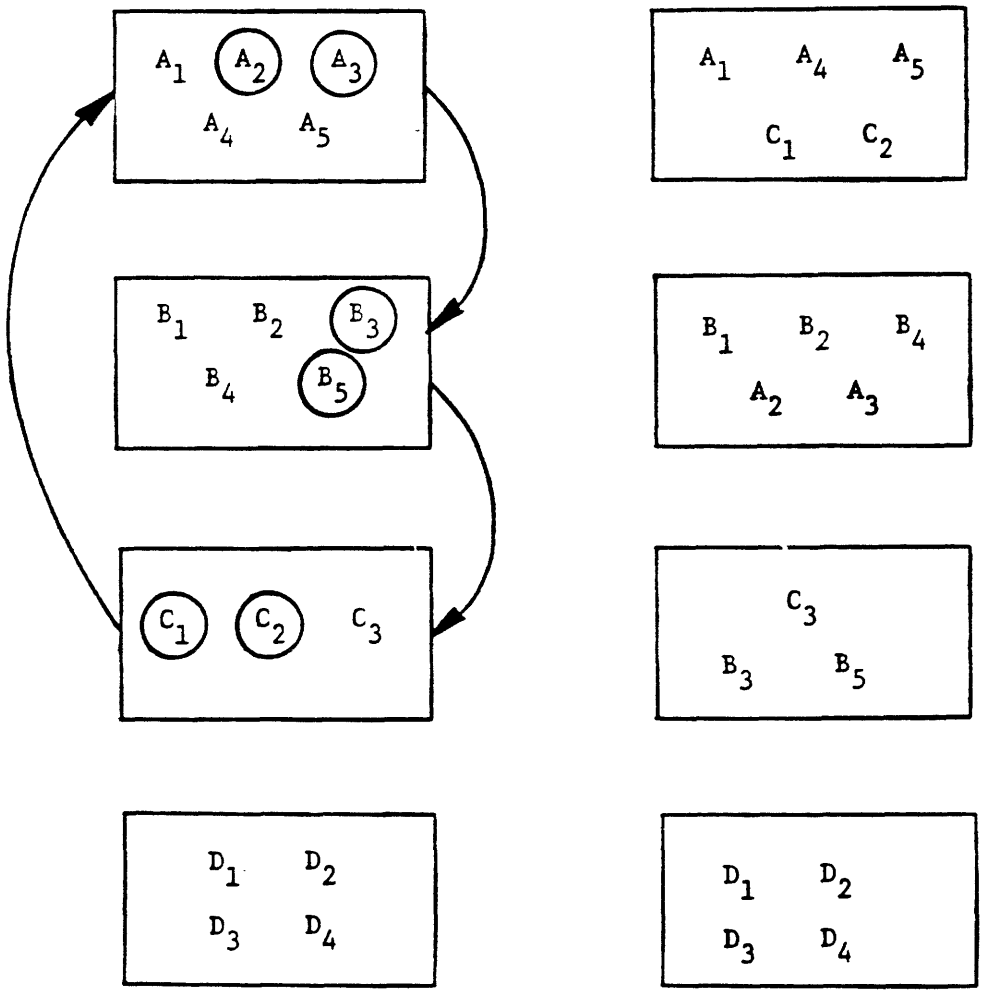


Figure 2. The Cyclic Transfer Neighborhood is Not Exact

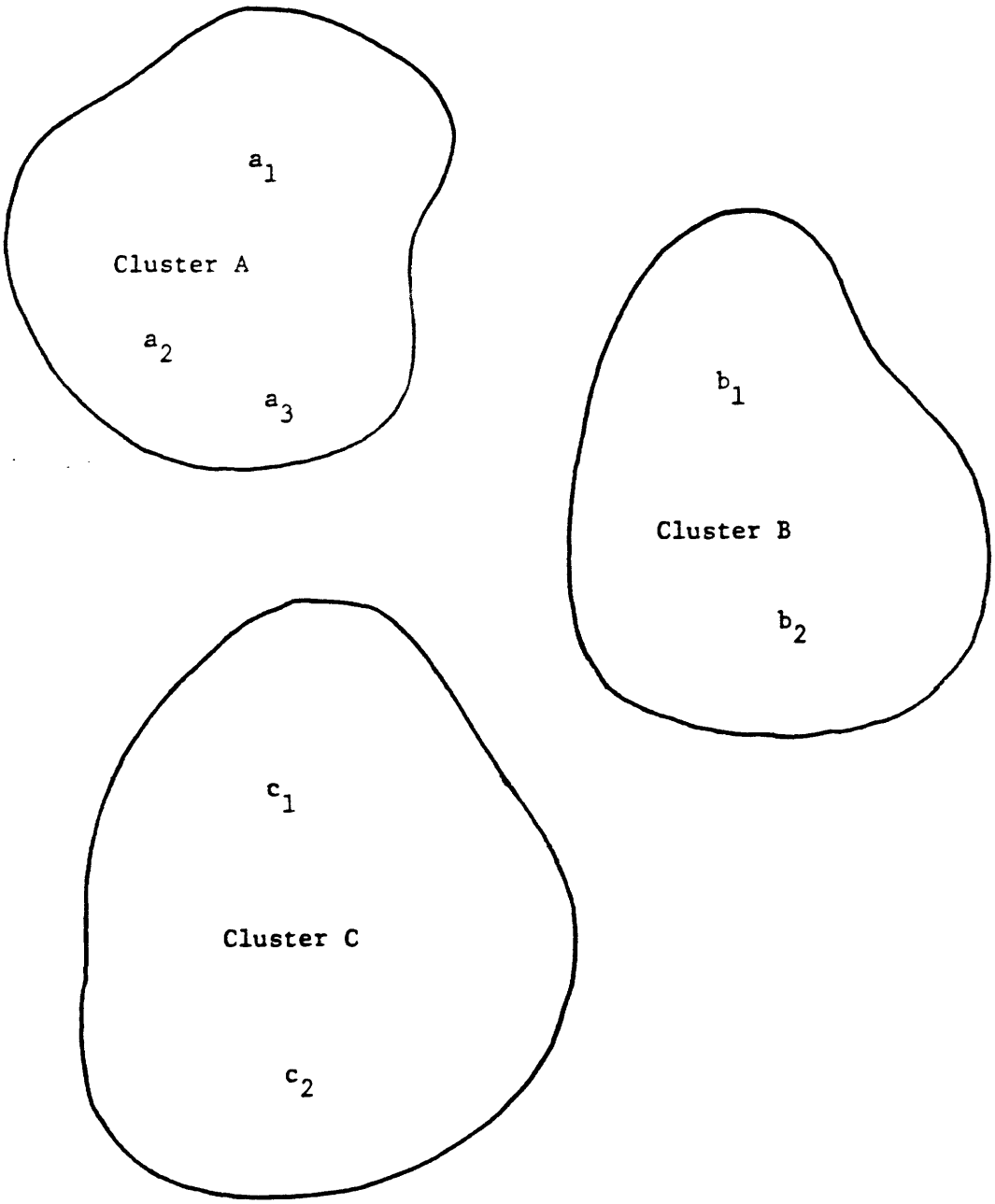


Figure 3. An Auxiliary Graph G^1

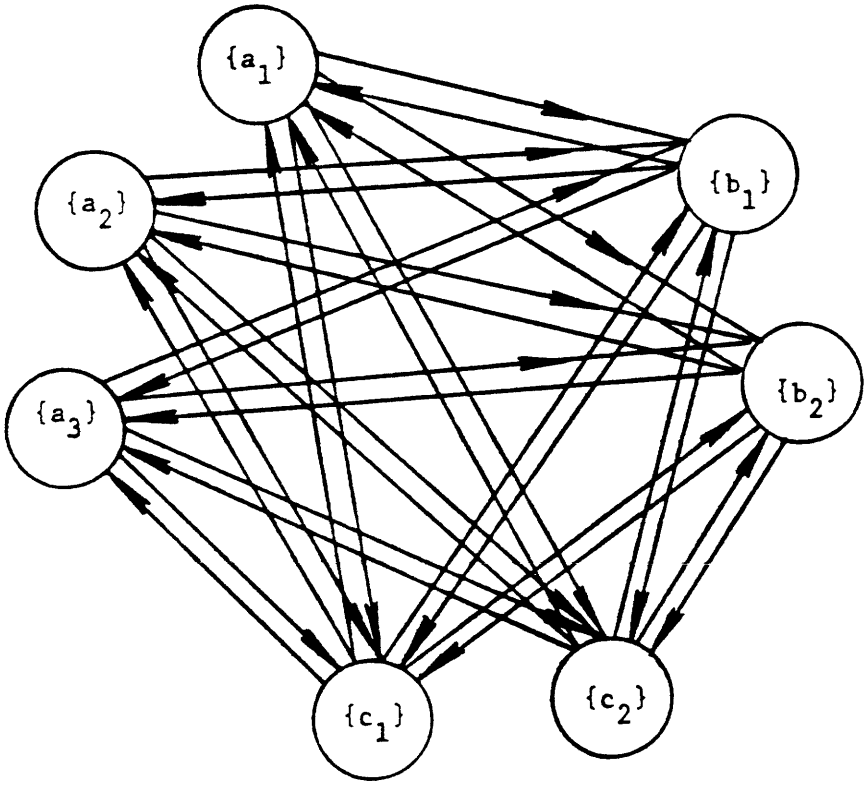


Figure 4. An Auxiliary Graph G^2

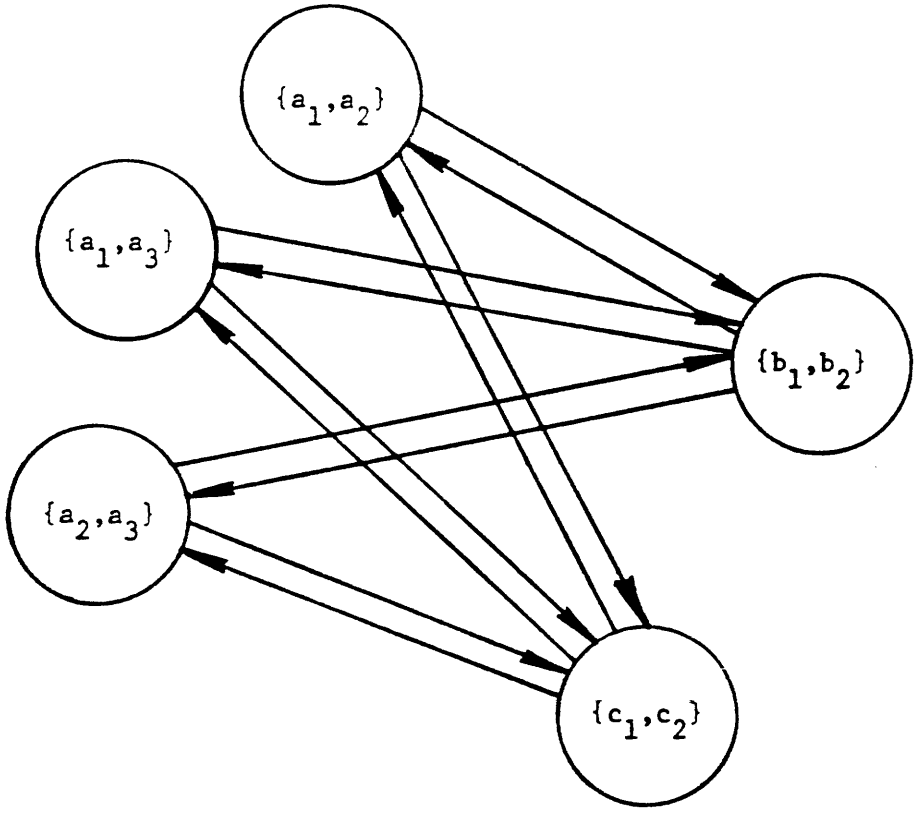


Figure 5. HAMILTONIAN CIRCUIT Transforms to CYCLE THROUGH DISTINCT SUBPARTITIONS

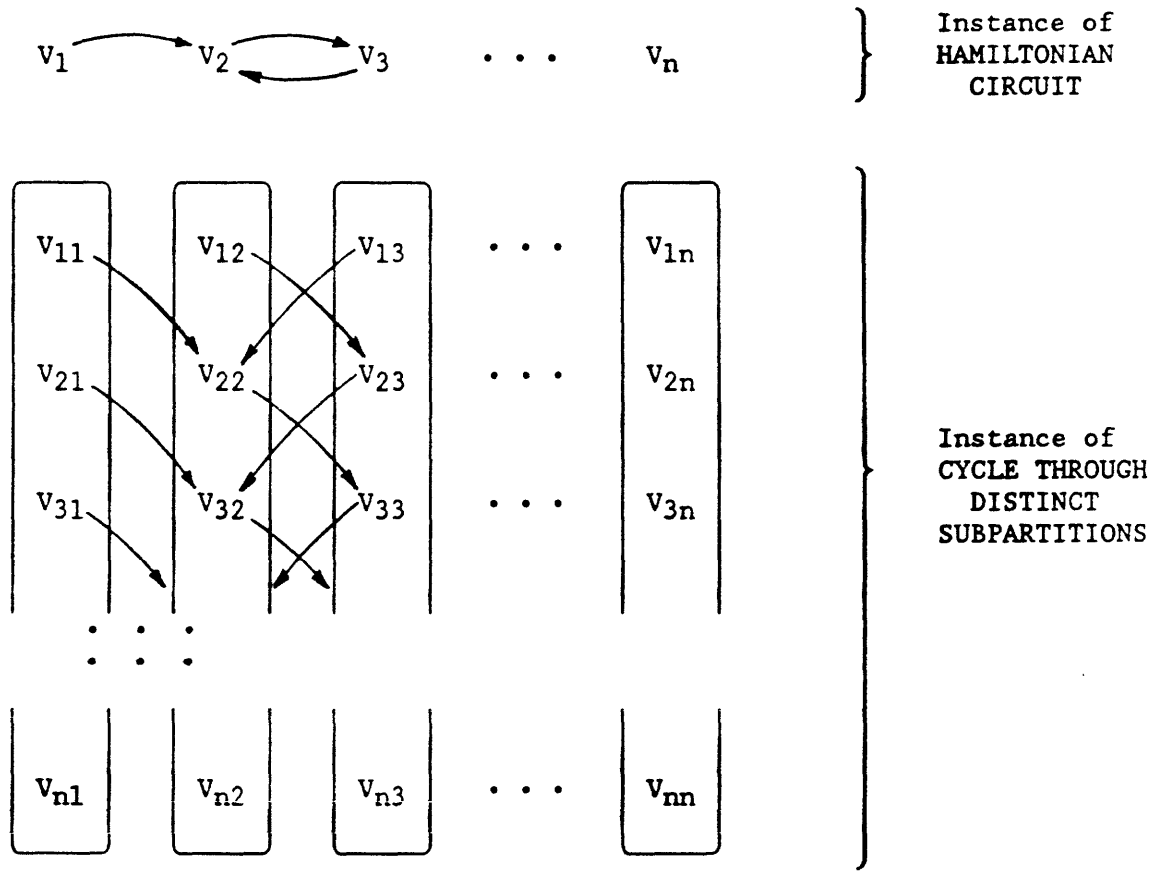


Figure 6. A Euclidean VRP Subtour With Two Additional Nodes