A Formal Description of the Transportation Problem

Thomas Dean and Lloyd Greenwald

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-92-14
March 1992

A Formal Description of the Transportation Problem

Thomas Dean* Lloyd Greenwald*
Department of Computer Science
Brown University, Box 1910, Providence, RI 02912

March 12, 1992

Abstract

In this paper we provide a compact, formal description for an important class of problems involving the deployment of diverse equipment (*e.g.,* planes, ships, trucks) to transport cargo (*e.g.,* personnel and material) throughout large geographical areas for private, government, and military purposes. The problem is particularly interesting for its combination of static and dynamic sources of complexity, and the fact that it allows and in some cases strongly encourages distributed solutions. The merit of this formalization is shown by its application to both understanding the complexities in existing restricted transportation problems and creating new natural restrictions on the general problem. Starting from the general formulation we are able to isolate the static and dynamic sources of complexity in transportation problems and relate them to existing research in computer science and operations research. In addition, we describe techniques for constructing algorithms for the transportation problem with provable performance guarantees.

---

*This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601, by the Air Force and the Advanced Research Projects Agency of the Department of Defense under Contract No. F30602-91-C-0041, and by the National Science foundation in conjunction with the Advanced Research Projects Agency of the Department of Defense under Contract No. IRI-8905436.

1 Introduction

In this paper, we present a formal description of the general problem addressed in the Transportation Planning and Scheduling Initiative sponsored by the Defense Advanced Research Projects Agency in conjunction with the Air Force's Rome Laboratory. The description provides a framework for constructing models of the static and dynamic aspects of the transportation problem. The purpose of the description is to facilitate technical discussion among participants in the initiative and within the larger community of researchers in planning, scheduling and control. For more background details on the initiative see [12, 21, 43, 17].

The specific problem faced by the initiative is concerned with directing the operators and equipment involved in transporting material and personnel for the United States military. This is the problem currently faced by the United States Transportation Command. The general formulation of the problem described in this paper addresses the wider range of issues facing those in government and industry responsible for the efficient and timely transportation of passengers and cargo. Both the general and the specific versions of the problem involve multiple agents responsible for the execution of transportation tasks, possibly multiple agents involved in directing the activities of agents executing such tasks, and the need to respond to changes given varying amounts of lead time.

Ultimately, we are interested in systems that can handle a wide range of challenges. In the case of a relatively static situation in which the transportation equipment is stable, the cargoes and their destinations are known somewhat in advance of their earliest shipping dates, and the execution of transportation tasks is relatively fault free, the system should be able to make highly efficient use of existing resources (*e.g.,* minimizing shipping time and fuel use). In uncertain situations, the system must be able to make decisions on the basis of partial or probabilistic information, and then monitor the situation as it evolves, prepared to countermand prior decisions should the situation warrant it. In the case of a rapidly changing situation, say, providing humanitarian aid in the aftermath of a major disaster or an ongoing conflict, the system should respond in a timely manner weighing the costs and benefits of spending additional time in reasoning and gathering information to support decision making.

In order to provide a compact, tractable description, we abstract from the many details of the specific problem faced by the U.S. Transportation Command. We do so, however, without eliminating essential sources of structure (which tend to make the problem easier) or complexity (which tend to make the problem harder). The description makes clear both the static structure of the problem involving locations and equipment for moving cargo between locations, and the dynamical structure of the problem involving the evolution of the system over time and the distributed nature of decision making and control.

The formal description of Section 2 provides the basis for subsequent explorations of the transportation problem, both within the remainder of this paper and beyond. For this reason it is made general enough to capture the entities and relationships of future transportation problems to arise, both static and dynamic. At the same time it is specific enough to provide useful structure for solving such problems. We demonstrate this in Section 3 by providing two special cases of the general transportation problem. Both of these cases can be expressed within the formalism developed in Section 2, yet the first case existed well before the formalism. The second case is constructed using the guidance of the formalism, yet describes a natural real-world transportation problem.

The remainder of the paper delves deeper into the complexity issues that arise from exploring the formalism. In Section 4, we isolate and explore the static sources of complexity in the transportation problem. We relate the static sources of complexity in the transportation problem to those in existing related research. We then discuss existing algorithmic techniques that can be used to address these static sources of complexity. We are especially concerned with techniques that can provide provable performance guarantees. In Section 5, we explore the dynamic sources of complexity. This is accomplished similarly to Section 4 by introducing related research and algorithmic techniques. In addition, we describe the space of solution concepts that can be explored in solving a particular transportation problem and discuss how the decision-making structure of the problem constrains the possible solutions.

The transportation problem has previously been explored by researchers both in computer science and operations research. There are many opportunities for sharing technology among these two disciplines. This paper takes a step in that direction by providing a standard formalization for researchers to agree on and by combining computer science analysis techniques with operations research approaches.

2 Formal Description of the Transportation Problem

In this section, we describe the class of problems being dealt with in the transportation planning initiative. We abstract from the particulars of the problem faced by the U.S. Transportation Command in an attempt to capture the fundamental sources of complexity. We make no attempt to model the process whereby problems in this class might be solved aside from recognizing that, in general, the decision-making process will involve more than one decision-making agent. This problem description makes no representational commitments; decision-making agents may choose to represent their knowledge about a problem instance in any way that they choose. The specifications in the following provide structure to constrain the set of possible physical models for the transportation domain. They by no means determine a model. Defining what features of the domain to model what to leave out is a complex and subtle problem that we do not address.

We begin by identifying two distinct aspects of the transportation problem. The first concerns the spatial layout of ports, capabilities of transportation assets, requirements for shipping equipment and personnel, and the dynamics governing the behavior of ports, transportation assets, and their cargoes. We call this the *basic dynamical system* and present it in Section 2.1. The second aspect concerns the agents responsible for controlling ports and transportation assets: their access to information, communication capabilities, and ability to initiate transportation activities, and the dynamics governing command, control, and communication. We call this the *decision-making structure* and present it in Section 2.2.

2.1 Basic Dynamical System

In keeping with terminology used by the U.S. Transportation Command, we use the term "asset" to refer to cargo handling equipment used to transport, store, or process cargo (*e.g.,* planes, ships, warehouses, forklifts) as well as support equipment used to refuel, maintain, or otherwise service cargo handling equipment. We use the term "package" to refer to any unit of cargo, including equipment, food, munitions, medical supplies, and personnel. Although we do not explicitly address the details of how items are shipped (*e.g.,* bulk items might be shipped on open wooden pallets or in sealed freight containers), packages can be assigned properties to facilitate modeling the dynamics of shipping.

At each point in time, each asset is in some location and each package is contained in some asset. Transportation assets can move between locations. Packages can be moved between assets by material handling assets. Locations, packages, and assets have time-varying properties that affect the performance of assets. For example, the readiness of an airplane is a time-varying property of a transportation asset. Material handling, maintenance, and refueling assets can change the properties of other assets.

Assets have programs that are changed from time to time by decision-making agents. A program can be anything from a set of protocols or standard operating procedures to an arbitrary control law. The program of an asset together with its properties and those of the locations that it visits and the packages that it carries determine the behavior of the asset over time. In Section 2.2, we discuss how an agent with the appropriate capabilities can change an asset's program. The only way in which an agent can exert any control over the dynamical system is by changing the programs of assets. The notions concerning the static structure of the transportation network are depicted graphically in Figure 1 and formalized in the following mathematical objects.

- a set of locations, L
- a set of location properties, QL
- a set of direct routes, R, between locations in L
- a set of direct-route properties, QR
- (L, R) corresponds to a graph with vertices, L, and edges, R
- a set of packages, P
- a set of package properties, QP
- a set of transportation, storage, and material handling assets, A
- a set of asset properties, QA
- a set of asset programs, U

Let $\star = L \cup R \cup P \cup A$ be the union of locations, routes, packages, and assets, and $Q\star = QL \cup QR \cup QP \cup QA$ be the union of location, route, package and asset properties.

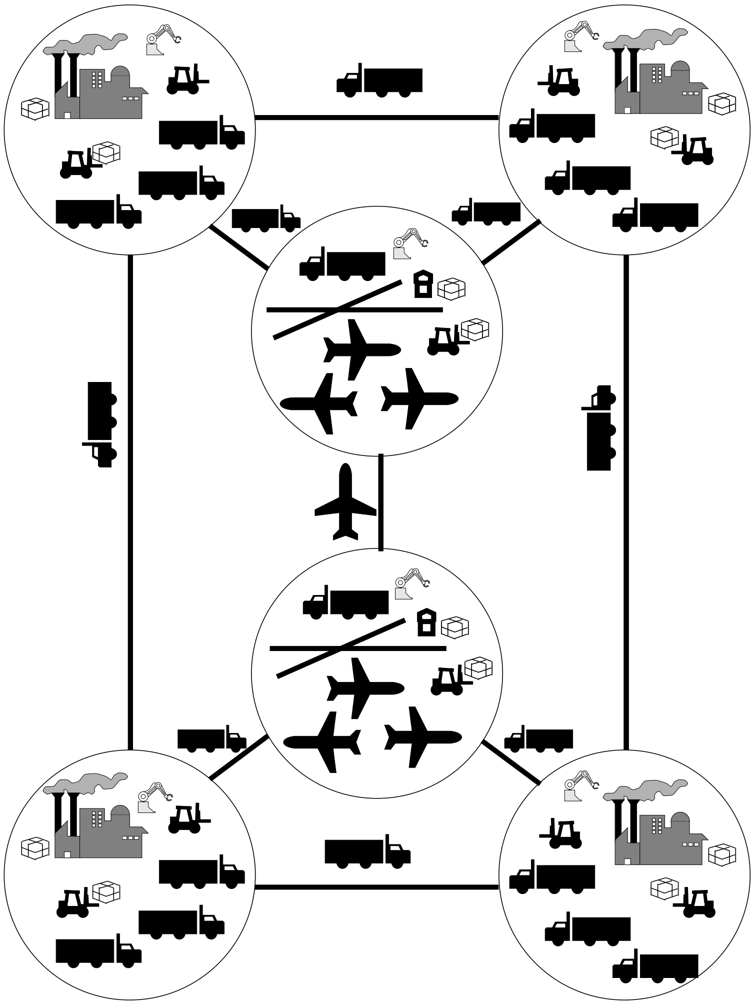


Figure 1: Transportation network

We assume that all properties are boolean; this assumption is made for no other reason than to simplify the presentation.

It is often the case that transportation assets are treated as cargo and loaded onto other assets for transport to locations where the transported assets are not available or are in short supply. We do not require that assets and packages be disjoint. For assets that are occasionally cargo, we may wish to provide a property to indicate that certain capabilities are disabled while in transit as cargo.

It should also be noted that packages are in some sense programmable. By attaching routing, handling, source and destination information, a package can be transported without any asset being explicitly programmed to transport that package. In the above formulation, routing information would be represented as a property of packages.

We assume that the sets of assets and packages are fixed. We model packages and assets being lost, found, created, or destroyed by assuming a large pool of virtual assets and packages assigned properties that determine their various states of accessibility.

Now that we have the static structure of the transportation network, we need to describe how things change over time. To account for change over time, we borrow from the systems theory literature by defining an appropriate state space and dynamical system [22]. The state space is determined by a set of state variables corresponding to the location of each asset, the asset containing each package, and the status of each property for each location, route, asset, and package. The set of state variables is determined by the following functions.

- a location function, $l : A \rightarrow L$,
- a container function, $c : P \rightarrow A$, and
- a property status function, $q : Q\star \times \star \rightarrow \{\text{true}, \text{false}\}$,

To deal with assets traveling between locations, we could include in L a pseudo location corresponding to being enroute and add a route property to QA, or we could extend the range of l to include routes as well as locations.

A state vector consists of $|A|$ location variables, one for each asset, $l(a_1), \ldots, l(a_{|A|})$, $|P|$ container variables, one for each package, $c(p_1), \ldots, c(p_{|P|})$, $|QA| * |A|$ asset property variables, one for each combination of asset and asset property, $q(g_1, a_1)$, $\ldots$, $q(g_1, a_{|A|})$,

…, $q(g_{|QA|}, a_1)$, …, $q(g_{|QA|}, a_{|A|})$, where $g_i \in QA$, plus additional variables for package, location, and route properties. We denote by S the state space formed by taking the cross product of the set of values for each of the state variables (*e.g.,* for each $a \in A$, L is the set of values for $l(a)$). For a given state, $s \in S$, we can refer to, say, the location of a particular asset, $a \in A$, using $l(a)[s] \in L$. To complete the specification of the basic dynamical system, we introduce time, control, and change as follows.

- a set of time points, T
- a control function, $u : A \times T \rightarrow U$
- a state transition function, $f : S \times U \times T \times T \rightarrow S$
- a set of state space histories, $H = \{h | h : T \rightarrow S\}$.

The structure of time can be discrete or continuous. We can refer to the value of a state variable given a time and history as $l(a)[h(t)]$.

We distinguish between state and control in keeping with standard practice in the control literature. For a description of the motivation and use of this formalism for modeling dynamical systems see [14, 18, 22].

Given a state, an initial time, and a final time, the transition function determines a final state for a fixed u. There is a lot folded into the state transition function; among other things, it determines the behavior of overloaded transportation assets, and the behavior of material handling assets that attempt to handle packages for which they were not designed. In the case of a stochastic system, the state transition function starts with an initial distribution on states and determines a final distribution.

The transition function is a *simulator* for the transportation domain; it can be specified as a difference equation, a system of differential equations, or a finite state machine; it may account for the weight, volume, fragility, and shelf life of cargo, the speed, fuel consumption, and capacity of transportation assets, and the effects of weather, equipment failure, and hostile forces. The amount of detail that the transition function will model is determined by the individual researcher; we do not prescribe any particular state transition function in this formalization. For example the transition function may specify the impact on a plane when atmospheric moisture levels rise, or this detail may be ignored if not warranted.

Exactly what goes in and what stays out of the transition function depends on the goals of the modeler. There currently exists a set of simulation tools that approximate a state transition function for the transportation problem.

Next, we have to establish performance criteria for comparing solutions to various instances of the transportation problem. Traditionally, a particular instance of the transportation problem has associated with it a set of requirements that must be satisfied. Requirements specify that packages must be in particular locations at particular times. Primary requirements are explicitly supplied. Secondary requirements are inferred from primary requirements by establishing dependencies between requirements. For instance, there might be a primary requirement that an infantry unit be at a particular place at a particular time, and a secondary requirement that there be a portable mess hall at that same location shortly thereafter.

- a set of primary requirements corresponding to a subset of $P \times L \times T$
- a set of secondary requirements corresponding to a mapping from primary requirements to subsets of $P \times L \times T$

In an uncertain world, demanding that requirements be strictly satisfied is unrealistic; there are always extenuating circumstances that warrant sacrificing requirements in order to satisfy other, implicit constraints on performance. To capture this more general metric for performance, we introduce the following measure.

- an objective function, $g : H \rightarrow \mathbf{R}$

The above objective function subsumes the notion of requirement. Requirements might continue to have some heuristic function in scheduling transportation assets; for instance, it may tend to be the case that the objective function penalizes histories that fail to satisfy requirements.

Unlike the transition function, for which there currently exists an approximation in the form of a simulator, there currently is no authorized realization of the objective function, approximate or otherwise. However, the literature on scheduling and sequencing discusses many objective functions (*e.g.,* minimize cost, minimize maximum tardiness, minimize weighted tardiness, minimize expected cost). It is assumed that researchers will choose their own functions after consultations with domain experts.

The objective function is not intended to account for performance measures that pertain to the solution process. In particular, issues of scalability and integrability are not addressed here.

2.2 Decision-Making Structure

The decision-making structure describes the decision-making agents, their access to information and to each other, and their authority with regard to controlling assets. The agents and the communication channels linking them define a graph. The notions concerning the static structure of the communication network are depicted graphically in Figure 2 and formalized in the following mathematical objects.

- a set of agents, V
- a set of communication channels, E
- (V, E) corresponds to a graph with vertices, V, and edges, E

Figure 3 depicts the flow of information between the decision-making agents and the system that they seek to control. Agents attempt to influence and discern particular state variables of the dynamical system through communications and cooperation. The introduction of delays and uncertainty complicates the control process giving rise to the need for continuous monitoring, and replanning.

In our formulation, the set of agents are disjoint from the set of assets to distinguish physical capabilities from decision-making capabilities. In certain instances of the general problem, it may prove useful to build abstractions for closely coupled asset/decision-making units allowing the decision-making agent to exert direct control over its coupled asset.

We introduce additional structure to handle access to information and authority to exert control over assets. Information access determines whether or not an agent has immediate access to information regarding a particular property at a particular time. It may be convenient to map agents to locations and require that agents have immediate access only to information regarding properties of that and nearby locations, the assets in

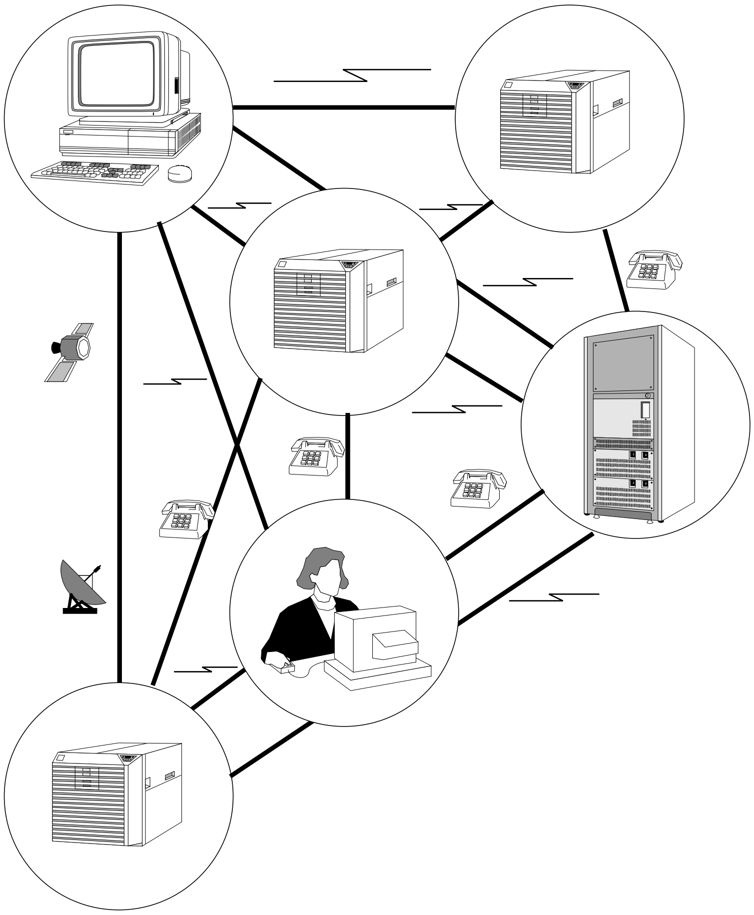


Figure 2: Decision-making structure

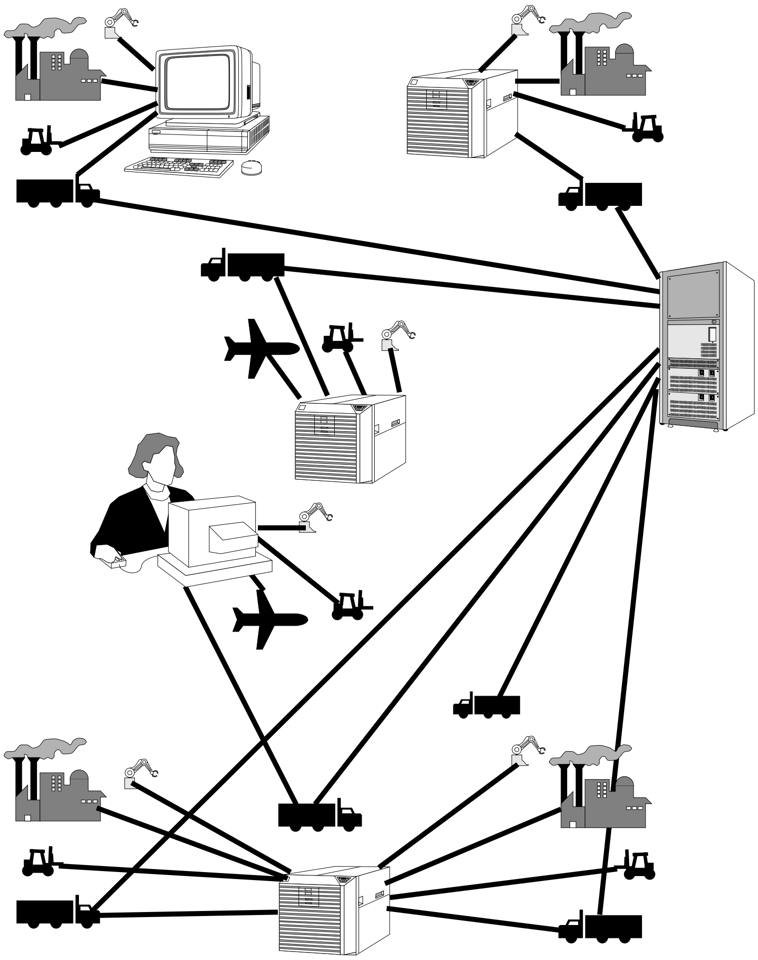


Figure 3: Interaction between assets and decision-making agents

those locations, and the packages in those assets. It is reasonable that agents only have immediate access to information concerning the current time; data logging can be performed to keep track of historical information. Communication access determines whether or not one agent can communicate directly with a second agent at a particular time. Finally, control authorization determines whether an agent is authorized to change the program of an asset to a particular program at a particular time.

- an information access function, $i : V \times Q\star \rightarrow \{\text{true}, \text{false}\}$
- a communication access function, $d : V \times V \rightarrow \{\text{true}, \text{false}\}$
- a control authorization function, $a : V \times A \times U \rightarrow \{\text{true}, \text{false}\}$

The communication channels are used by agents to obtain indirect access to information. In order for agents to transmit information via communications channels it will be necessary to establish protocols. These protocols will cover requests for information, commitments to send information, actual transmission and acknowledgement that the information has been received. Message protocols, packet sizes, etc. are all part of the solution and not the problem description; however, we assume that one agent can request information from another agent. If security becomes an important issue, it will be necessary to add an information access authorization function that determines what information each agent is allowed access to and when.

To complete the picture, the state transition function described in Section 2.1 has to be extended to account for the dynamics of access and authorization. In particular, the state transition function may account for delays in communication, transmission noise, and changes in the topology of the communication network. These extensions involve additional state variables and, therefore, the state space may be augmented to account for the additional variables. It may be convenient to think of the state transition function as a composition of simpler state transition functions, each operating independently and asynchronously. In many cases this abstraction is physically valid.

3 Two Special Cases of the Transportation Problem

In this section, we demonstrate two natural transportation problems that can be expressed within our formulation. They are the *subscriber dial-a-ride problem* and the *greyhound* *problem*. These special cases of the general transportation problem give us an opportunity to map concrete problems to the general formalization provided in Section 2 by focusing on the static sources of complexity (*i.e.,* the basic entities). In Section 4, we more thoroughly explore the process of reformulating the general transportation problem by focusing on certain sources of complexity and neglecting others. In Section 5, we explore how to deal with dynamic sources of complexity.

3.1 Subscriber Dial-a-Ride Problem

One reformulation of the general transportation problem that has been studied in the literature is the *Subscriber Dial-a-Ride Problem* [38, 7]. This problem consists of a set of customers (packages) who each need to be delivered from a source location to a destination location. In addition, the pickup and/or delivery time for each customer is specified. Given a set of vehicles (assets), the problem is to construct a set of schedules such that the capacities of the vehicles are not exceeded. The performance criteria for this problem is to minimize customer dissatisfaction. This equates to minimizing the maximum for any customer of the deviation of the actual pickup time from the desired pickup time plus the deviation of the actual delivery time from the desired delivery time.

Using the formalization techniques of Section 2 we have:

- a finite, fixed set of locations, L;
- a set of location properties, QL; all locations have identical properties, including infinite storage capacity;
- a finite, fixed set of direct routes, R, between locations in L;
- a set of direct-route properties, QR; direct-routes have fixed, finite length;
- a graph (L, R) with vertices, L, and edges, R;
- a set of packages, P (customers);
- a set of package properties, QP; packages have size, release date (desired pickup time), delivery date (desired delivery time), source location (pickup location) and destination location (delivery location) properties;

- a finite set of transportation assets (vehicles), A;
- a set of asset properties, QA; assets have finite capacity and infinite range; and
- a set of asset programs, U; an asset program consists of a route that begins and ends at a specified location (depot) and cannot be preempted.

The performance criteria is to find a set of routes for transportation assets and one load and unload per package such that the package is loaded at source and unloaded at destination such that the customer dissatisfaction is minimized subject to the capacity constraints of the assets. Formally, we have

$$\min \max_{p \in P}(|arelease(p) - drelease(p)| + |adelivery(p) - ddelivery(p)|)$$

where $arelease(p)$ is the actual time that package p is picked up at its source, $drelease(p)$ is the desired time of pickup for p at its source, $adelivery(p)$ is the actual time that package p is delivered to its destination and $ddelivery(p)$ is the desired delivery time for p at its destination. In cases where only pickup or delivery times are given the corresponding term can be dropped.

The dial-a-ride problem can be expressed and solved as a linear program. A mathematical formalization for this problem is found in [38].

3.2 Greyhound Package Scheduling Problem

A second reformulation of the general transportation problem is the *Greyhound Package Scheduling Problem*. This problem is characterized by fixed asset routes resembling bus schedules. Consider a situation in which a bus company has created a set of schedules for its buses that it has optimized for some performance criteria based on its passenger operations. For example, it has scheduled its buses so that a bus is available at every location every fifteen minutes and no bus makes more than three stops between any two locations; or it has minimized the time a passenger has to wait before the next bus arrives given that passengers arrive at fixed times. The company then realizes that, although the schedules are optimal for its desired performance criteria, there is unused capacity available on some buses. The bus company would like to make use of this capacity by managing a package delivery business on the side.

The bus company sets up package drop-off points at each of its locations. Since the bus schedules are already fixed, the problem reduces to that of determining a sequence of loads and unloads for each package so that each package is transported from its source to its destination. The sequence of loads and unloads determines the route, or flow, of the package. Not only is the problem constrained to follow the fixed bus routes but, the company must be assured that the packages assigned to any bus at any time will not take up more space than the capacity available. This capacity is fixed in advance by the bus company's solution to the passenger routing problem.[1] Finally, given these constraints, the bus company would like to minimize the total transit time for all packages so that it can make some claims about average delivery time. Alternately, the company could choose to minimize the maximum delivery time for any package and make claims about "overnight" delivery, etc.

Formally, we have:

- a finite, fixed set of locations, L;
- a set of location properties, QL; all locations have identical properties, including infinite storage capacity;
- a finite, fixed set of direct routes, R, between locations in L;
- a set of direct-route properties, QR; direct-routes have fixed, finite length;
- a graph (L, R) with vertices, L, and edges, R;
- a set of packages, P;
- a set of package properties, QP; packages have size, release date and source location and destination location properties;
- a finite set of transportation assets, A;
- a set of asset properties, QA; assets have finite capacity and infinite range; and

---

[1] In practice, cargo is not stored in the passenger area but, rather, in the baggage area. The unused capacity at any time will vary though we have assumed it is fixed. A more convincing version of this problem can be framed in terms of containerized freight.

- a set of asset programs, U; an asset program consists of a fixed, possibly repeated route that cannot be preempted.

The basic entities of the greyhound problem are depicted in Figure 4. The performance criteria that we consider here is to find a set of schedules for packages (*i.e.,* for each package, a series of loads/unloads onto assets such that the package begins at its source and ends at its destination) such that the total time between release date and delivery date (at destination) for all packages subject to the capacity constraints of the assets is minimized. Formally, we have

$$\min \sum_{p \in P} delivery(p) - release(p)$$

where $delivery(p)$ is the time at which package p is delivered at its destination and $release(p)$ is the time at which package p is released at its source.

Given the speed of each asset (bus) and the sequence of locations it must visit, along with the distance between locations, we can construct the fixed asset routes required by the problem. A example fixed asset route table that corresponds to the example problem in Figure 4 is:

	first stop on route	second stop on route	third stop on route	fourth stop on route	fifth stop on route	sixth stop on route	cycle time*
Bus a	12:00 (A)	12:09 (C)	12:18 (B)	repeat			30.6 min
Bus b	12:00 (B)	12:24 (F)	12:39 (E)	12:54 (F)	repeat		78 min
Bus c	12:00 (E)	12:13 (D)	12:28 (C)	12:41 (B)	1:00 (A)	repeat	91 min

*Cycle times assume 3 minutes per stop for loading/unloading. In this simplification of the problem, we neglect any variance in the travel times due to traffic conditions, etc., and assume that the fixed routes (bus schedules) are guaranteed. In Section 5, we describe how to extend a static problem to one with dynamic complexity such as uncertainty in fixed routes.

Upon closer inspection it is apparent that in the greyhound problem the bus schedules and problem graph can be collapsed into a single structure, since the bus routes are fixed there are a fixed number of ways for a package to travel between nodes. Using this fact we can take an instance of the greyhound problem including graph, fixed asset routes and packages and induce a new graph as follows:

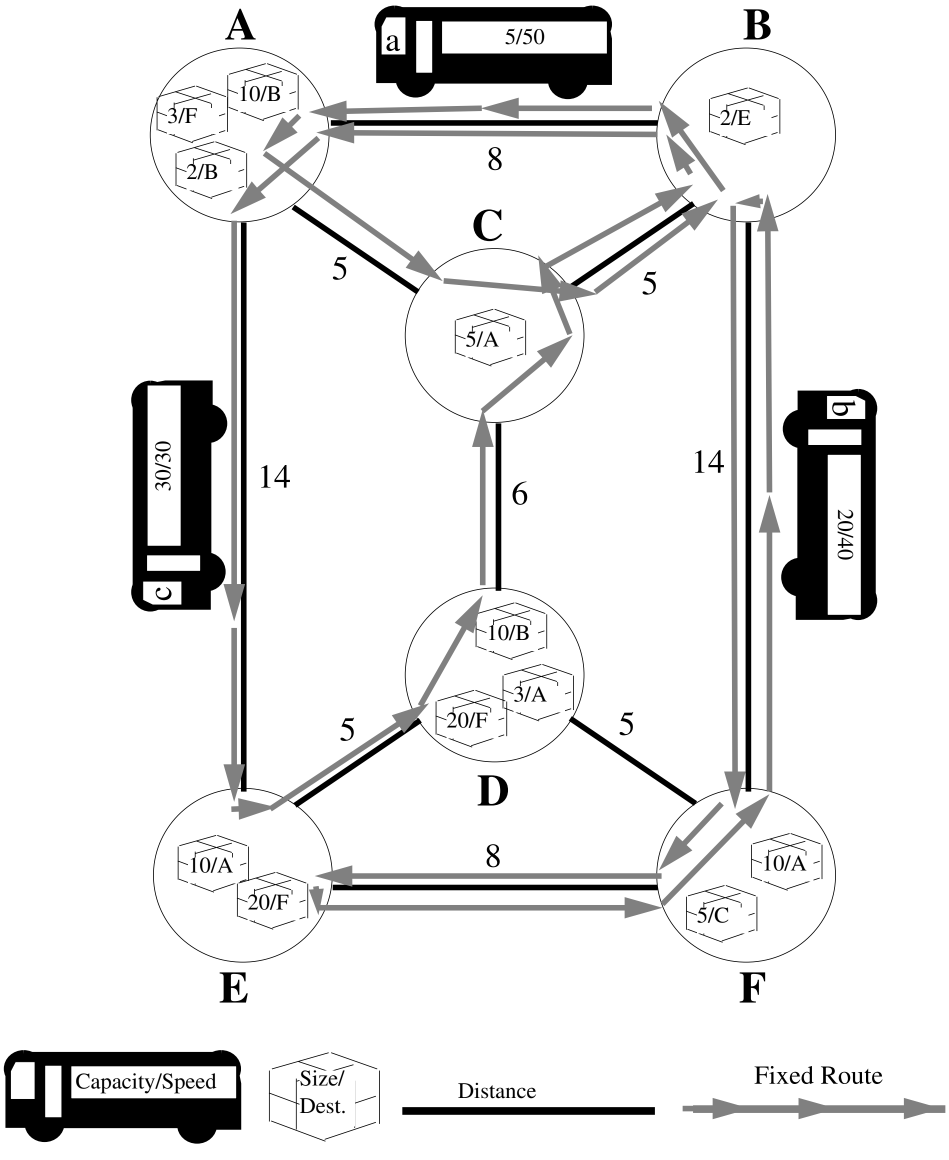


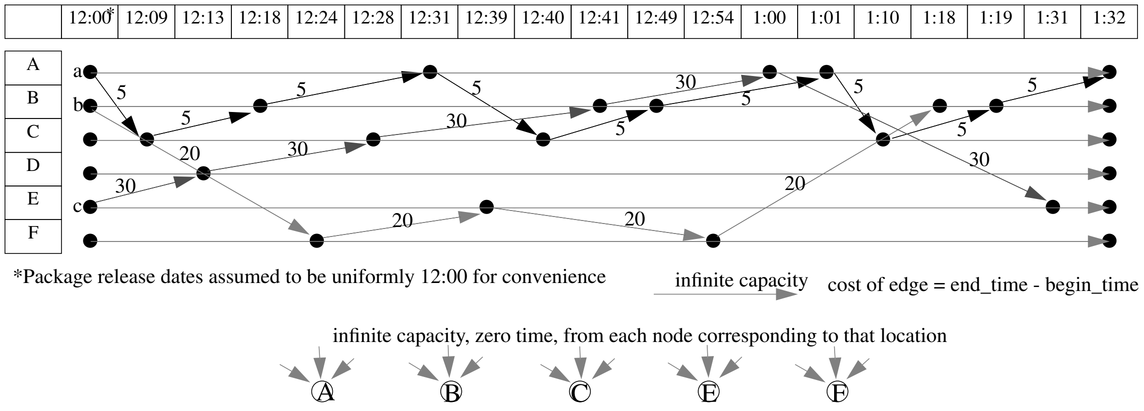
Figure 4: Greyhound Problem – Basic Entities

- pick some time interval;
- for each asset $a \in A$:
    - for each route stop i within the time interval:
        - create a node $(l_{a,i}, t_{a,i})$ such that asset a arrives at location $l_{a,i}$ at time $t_{a,i} \in T$
        - create directed edge $((l_{a,i}, t_{a,i}), (l_{a,i+1}, t_{a,i+1}))$ with capacity $cap(a)$ and cost $t_{a,i+1} - t_{a,i}$;
- for each package $p \in P$:
    - create a node (s_p, t_p) such that package p is released at source node $s_p \in L$ at release date $t_p \in T$;
- for each time step t_l within the time interval, if (l, t_l) corresponds to a node created above then create a directed edge $((l, t_l), (l, t_{l'}))$ with infinite capacity and cost $t_{l'} - t_l$; where $t_{l'}$ is minimum time step greater than t_l corresponding to node created above for location l;
- let $DEST = \bigcup_{p \in P} dest(p)$; for each $dt \in DEST$ create node dt and for each node (l, t) created above, if $l = dt$ then create a directed edge $((l, t), dt)$ with infinite capacity and zero cost.

An induced graph corresponding to our example is given in Figure 5. As we point out in Section 4.3, this graph can then be used to solve the greyhound problem as a multicommodity flow problem. For a more detailed treatment of the greyhound problem see [13].

3.3 Comparing the Two Special Cases

In the general transportation formulation of Section 2, properties on basic entities are left unspecified. This lack of structure can lead to combinatorial problems. Both the dial-a-ride and greyhound problems address this issue by further constraining the problem. They both are concerned solely with static sources of complexity and neglect dynamic sources of complexity. The two problems are similar in that all the basic entities are the same. They



*Package release dates assumed to be uniformly 12:00 for convenience

cost of edge = end_time - begin_time

infinite capacity, zero time, from each node corresponding to that location

Figure 5: Greyhound Problem – Induced Graph

differ from each other in the properties of the packages and programs of the assets and their performance criteria. Specifically, they have the following differences:

1. package properties
    - dial-a-ride: release date and delivery date properties of packages correspond to a time window in which the package must be delivered
    - greyhound: delivery date is not necessarily specified
2. package program (routing property)
    - dial-a-ride: single load/unload per package
    - greyhound: multiple loads/unloads per package
3. asset program
    - dial-a-ride: asset routes begin and end in the same location but are otherwise unconstrained
    - greyhound: asset routes are completely fixed in advance
4. performance criteria
    - dial-a-ride: minimize maximum time window fluctuation for any package

- greyhound: minimize total travel time for all packages

The key differences in terms of complexity are the fixed versus scheduled asset routes and the single versus multiple loads per package. The complexity issues caused by these differences are explored in more detail in Section 4. Linear programming and integer programming formularization have been obtained for both the dial-a-ride and greyhound problems. As will be seen in Section 4.3, linear programming is an important tool for solving transportation problems.

4 Static Complexity Issues and Problem Reformulation

In Section 2, we present a formalization of the transportation problem. This formalization is sufficiently general to address the wide range of issues within the transportation domain. However, many sources of complexity are combined in the problem description as presented. In Section 3, we describe two special cases of the general transportation problem. The subscriber dial-a-ride problem was found in the literature and then shown to be a special case of the general problem. In the greyhound problem, we began with the formalization of the general problem described in Section 2 and simplified the formalization into a more tractable yet very natural problem. In this section we describe this process of reformulating the general problem into simplified problems based on complexity issues in more detail and discuss the usefulness of the general formalization for constructing new, interesting problems.

The two special cases above serve to isolate certain complexity issues from others. For example, in the dial-a-ride problem the complexity due to multiple loads/unloads of packages is removed and in the greyhound problem the complexity due to unconstrained asset routes is removed. In both problems, the complexity due to finite storage capacity at sites and all dynamic sources of complexity are removed. This process of abstracting from the general transportation problem features based on the complexity sources can be explored further.

We proceed by explicitly pointing out the static sources of complexity in the problem formulation. Once we are aware of the sources of complexity, we can reformulate the problem so that the sources of complexity that are critical to a particular natural problem or set of research interests are included and those that are peripheral are excluded. Each reformulation of the problem may emphasize different sources of complexity. In Section 5, we discuss dynamic sources of complexity.

We define a source of complexity as a problem feature whose inclusion has tended to make similar problems NP-hard, all other features remaining equal. These features move a problem across the threshold from tractable to NP-hard. We must emphasize that these categorizations only provide perspective on the problem, based on seemingly similar problems in related lines of research. Any actual complexity determination depends upon the combination of all features in a particular problem description. We take advantage of the wealth of research in sequencing and scheduling, vehicle routing and operations research to make our preliminary categorizations.

4.1 Static Sources of Complexity from Related Research

Included here are summaries of the relevant complexity of the related research. It is beyond the scope of this article to go into detail about the specific concepts of these related lines of research and mappings from their terminology to the transportation domain. For a more detailed discussions see [25, 6, 38, 10, 19, 16].

There is a well-developed body of research in sequencing and scheduling that explores some of the questions with which we are concerned [25]. This research is divided into single machine deterministic problems, parallel machine deterministic problems, stochastic problems and job shop problems. If we are to take advantage of results in the scheduling literature, we must describe our problem in terms of machines, jobs, operations, etc. The fit is not exact and, therefore, there are many different ways to map the problem into this language. Though we do not go into detail about how to map transportation problems to traditional scheduling problems, these choices can greatly affect the complexity of the solutions.

Below is a summary of related results in the scheduling literature. Each of these results point to a specific combination of features that have either been shown NP-hard or polynomial in single machine (unless noted) deterministic machine scheduling. Deterministic machine scheduling literature is most closely related to the static complexity issues in the transportation problem. It is important to reiterate here that these results are not cut and dry. These categorizations only provide perspective on the problem. Determining whether a specific transportation problem is NP-hard or polynomial depends upon an analysis of the combination of all features in that particular problem.

NP-hard:

- arbitrary precedence constraints among jobs
- jobs with deadlines and weighted minsum objective function
- variable release dates among jobs (as opposed to all jobs being released at the same time)
- no job preemption, parallel machine model and minmax objective function
- job preemption, parallel machine model and weighted minsum objective function
- job preemption, single machine model and weighted minsum tardiness objective function
- arbitrary processing requirements among jobs and unweighted minsum objective function

Polynomial:

- special cases of precedence among jobs (*e.g.,* tree, series-parallel)
- jobs with deadlines and unweighted minsum objective function
- variable release dates among jobs and job preemption
- variable release dates among jobs and similar ordering of release and due dates among jobs (*i.e.,* if job a has release date preceding job b then job a also has due date before job b)
- unit length jobs

The general transportation problem can be best described as involving unrelated parallel machines (machine properties vary with both machine and job) that cannot be preempted once a job is started, jobs with arbitrary deadlines and release dates and a weighted minsum objective function. The results above indicate that this combination of features may be NP-hard. The greyhound problem can be best described as involving uniform parallel machines (machine properties vary with machine only) that cannot be preempted once a job is started, jobs with variable release dates and processing requirements, and a weighted minsum objective function. This simplified problem also appears to be NP-hard, based on related scheduling results above. However, when uncertainty is introduced into the problem and we are looking for expected performance rather than exact performance, it is often the case that NP-hard problems can be solved in polynomial time [25]. Similar optimality relaxation results for approximation algorithms are discussed in Section 4.3.

A second body of research that relates to features in the transportation problem are shortest path, traveling salesman (TSP) and multicommodity flow problems [15, 20, 10, 36]. In the shortest weight-constrained path problem [16], we are given: a graph $G = (V, E)$; length $l(e) \in Z^+$, for each $e \in E$; weight $w(e) \in Z^+$, for each $e \in E$; specified vertices $s, t \in V$; and positive integers K, W; and ask the question: Is there a simple path (all vertices in the path are distinct) in G from s to t with total weight W or less and total length K or less? This problem is NP-complete for directed and undirected graphs, but solvable in polynomial time if all weights are equal or all lengths are equal [16]. This problem indicates that an optimal assignment of assets to packages, even for a single package, is NP-complete.

Another closely related problem from [16] is the path constrained network flow problem. In this problem: given a directed graph $G = (V, A)$; specified vertices s and t; capacity $c(a) \in Z^+$, for each $a \in A$; a collection P of directed paths in G; and a requirement $R \in Z^+$; we ask the question: Is there a function $g : P \rightarrow Z_0^+$ such that if $f : A \rightarrow Z_0^+$ is the flow function defined by $f(a) = \sum_{p \in P(a)} g(p)$, where $P(a) \subseteq P$ is the set of all paths in P containing the arc a, then f is such that (1) $f(a) \leq c(a)$ for all $a \in A$, (2) for each $v \in V - \{s, t\}$, flow is conserved at v, and (3) the net flow into t is at least R?

This problem remains NP-compete even if all $c(a) = 1$. It is similar to the static transportation problem in that it specifies a set of paths, each with a unique cost (size). However, it does not specify schedules since paths do not include any timing information.

A third related problem is the directed two-commodity integral flow problem [16]. In this problem: given a directed graph $G = (V, A)$; specified vertices s_1, s_2, t_1, and t_2; capacity $c(a) \in Z^+$, for each $a \in A$; and requirements $R_1, R_2 \in Z^+$; we ask the question: Are there two flow functions $f_1, f_2 : A \rightarrow Z_0^+$ such that (1) for each $a \in A$, $f_1(a) + f_2(a) \leq c(a)$, (2) for each $v \in V - \{s, t\}$ and $i \in \{1, 2\}$, flow f_i is conserved at v, and (3) for $i \in \{1, 2\}$, the net flow into t_i is at least R_i?

This problem remains NP-complete even if $c(a) = 1$ for all $a \in A$ and $R_1 = 1$. The corresponding M-commodity (multicommodity flow) problem is polynomially equivalent to integer programming and with non-integral flows allowed is polynomially equivalent to linear programming for all $M \geq 2$. One consequence of this result is that, if packages are small enough, we can think of them as continuous variables and possibly generate usable approximation algorithms. In Section 4.3, we discuss approximation algorithms in more detail, including some based on multicommodity flow.

4.2 Static Sources of Complexity in the General Transportation Problem

From the above discussions, we can pinpoint some sources of complexity in the formulation of the transportation problem. It can be seen that the following are static sources of complexity:

- Finite, mobile transportation assets
- Transportation assets with finite capacity constraints (even if uniform)
- Nonpreemption of transportation assets (once in transit)
- Nonidentical release dates, nonidentical package processing requirements (route size), nonidentical deadlines
- Nonidentical packages, assets, sites, edges
- Finite capacity storage and edges
- Finite range of transportation assets
- Unconstrained routing of transportation assets
- Arbitrary precedence constraints in routes
- Transit and package handling time dependent on location and asset

In the presentation of the subscriber dial-a-ride and greyhound problems in Section 3, we isolate some sources of complexity from others. In general, we can choose certain sources of complexity as being critical to a natural problem and then neglect those that are peripheral. In essence, this process takes the general problem formalization and filters it through related complexity observations and a particular natural problem or set of research interests and arrives at a problem reformulation.

4.3 Algorithms and Performance Guarantees that Address Static Complexity Issues

In Section 4, we cite problems related to the general transportation problem and isolate the static sources of complexity in the transportation problem. Given that these related problems are known to be NP-hard, it would seem unlikely that we would be able to find polynomial time optimal algorithms to solve the static transportation problem or reformulations that contain comparable sources of complexity. However, there are ways to approach the transportation problem if we are willing to relax our desire for optimal solutions.

In this section, we discuss two classes of algorithms that can be used to overcome the difficulty of the static transportation problem. They are *approximation algorithms* and *randomized algorithms.* In short, approximation algorithms give up the notion of an "optimal solution" to an optimization problem but, in turn, guarantee that the quality of the solution of the resulting algorithm will not exceed a performance bound for any problem instance; whereas randomized algorithms allow for better bounds on algorithms by bounding the probability that a worst-case (*i.e., bad*) event occurs and thereby guaranteeing that a good event will occur with high-probability.

We can obtain provable performance guarantees for certain problems by constructing approximation algorithms. Approximation algorithms apply to optimization problems in which we can be content with solutions that are provably *near-optimal* using fast (polynomial) algorithms. Good discussions of approximation algorithms can be found in [16, 11].

When analyzing an approximation algorithm, we compare the *value* of the solution with the optimal value for the optimization problem. Since an optimization problem is concerned with either minimization or maximization of some objective function, the value that we are concerned with is the value of that objective function. In general, an approximation algorithm run on different instances of a problem will yield different values, just as the optimal value will be different for different instances. It may also be the case that the relation between the value given by the approximation algorithm and the optimal value will be different for different instances of the problem. It is that relation or ratio that we would like to provably guarantee.

Let $OPT(I)$ be the optimal value for instance I of a problem and let $AA(I)$ be the value returned by a given approximation algorithm for instance I of a problem. For a minimization problem, we are interested in the ratio

$$\frac{AA(I)}{OPT(I)}$$

and for a maximization problem we are interested in the inverse ratio. More succinctly, we are interested in the value $\max\{\frac{AA(I)}{OPT(I)}, \frac{OPT(I)}{AA(I)}\}$. Since the optimal value is a lower bound on the value that any algorithm will return for an instance, this ratio is lower bounded by one. There are further details concerning the difference between *absolute* performance bounds and *asymptotic* performance bounds that we will not go into here (see [16]).

In the analysis of a given approximation algorithm, we would like to upper and/or lower bound this ratio. By providing an upper bound on this ratio, we are guaranteeing that for any instance of the problem (or any instance large enough in size) the ratio of the value of the algorithm to the optimal value is bounded from above. This bound can either be a constant bound or a bound that is a function of the size of the instance. Ideally, we would like constant bounds that are as close as possible to one. Non-constant bounds imply that the value of the approximation algorithm can be infinitely worse than the optimal value. However, in cases when constant bounds are not possible, we may be satisfied with polylogarithmic bounds.

As an example of an approximation algorithm related to the transportation problem, consider the traveling salesman problem (TSP). The TSP problem is a single-vehicle problem in which, given a finite set of cities and a distance between any two cities, we would like to find a tour of minimum distance for the vehicle such that each city is visited exactly once. This problem has been extensively studied [24] and has been shown NP-complete. However Christofides [9] has demonstrated an approximation algorithm with ratio 3/2 based on minimum spanning tree methods for instances of the TSP problem that satisfy the triangle inequality (*i.e.,* for every 3 cities $\{a, b, c\}$, $dist(a, c) \leq dist(a, b) + dist(b, c)$). So, for any instance of the TSP problem satisfying this condition, a solution can be found quickly (in polynomial time) that provides a tour in which the vehicle travels no more than 50% more than the optimal tour. It is interesting to note that it has been shown [37] that a good approximation algorithm to solve the TSP problem without the triangle inequality constraint cannot be found in polynomial time unless $P = NP$.

As another example of an approximation algorithm related to the transportation problem, consider the bin packing problem. In the bin packing problem, we are given a finite set of items of rational size $\in [0,1]$ and we must partition the items into the fewest number of sets such that no set contains items that sum to more than one. While this problem is also NP-hard, an approximation algorithm exists [16] that can find a solution in which the number of sets in the partition is within 22% of the optimal number of sets.

There is an extension on the approximation algorithm idea into what is called an *approximation scheme*. In an approximation scheme we are given an "accuracy requirement" $\epsilon > 0$. Given this ϵ, the approximation scheme is an approximation algorithm that will solve the problem instance with a ratio less than or equal to $1 + \epsilon$. Generally, the lower the ϵ the slower the algorithm (in terms of computation time). If the algorithm used is polynomial in the size of the instance for each fixed $\epsilon > 0$ then we have a *polynomial time approximation scheme*. If, additionally, the process of finding the particular algorithm for the fixed ϵ (the scheme) is polynomial in terms of the size of the instance and $1/\epsilon$, then we have a *fully polynomial time approximation scheme* (FPAS). In trying to solve an NP-hard problem, an FPAS is, in some sense, the best possible result.

Randomized algorithms incorporate random choices into otherwise deterministic procedures. Whereas approximation algorithms provide provable performance guarantees for solving an instance of a problem near-optimally, algorithms constructed using the randomization technique provide performance guarantees that will be correct for every instance of the problem *with high-probability*. In other words, there is some non-zero probability that the performance guarantee will be violated in some cases. However, these cases are provably very rare. A good discussion of randomized algorithms can be found in [32].

One example of a randomized algorithm is a variant of quicksort called *random quicksort*. The major difference from deterministic quicksort is that, in random quicksort, the splitter element is picked randomly. Recall that deterministic quicksort has an expected running time of $O(n \log n)$ and a worst-case running time of $O(n^2)$. Therefore, our performance guarantee for deterministic quicksort is $O(n^2)$ for any input. By taking the randomness out of the input and putting it into the algorithm, randomized quicksort is able to provide a performance bound of $O(n \log n)$ for any input, with high probability. The probability in this case is $1 - O(n^{-6})$. In other words, the probability that random quicksort will terminate in $O(n \log n)$ steps is very close to one [32].

Randomization techniques can also be used to construct approximation algorithms that are provably close to optimal with high-probability. The relation of multicommodity flow problems to transportation problems was mentioned in Section 4.1. In [26] the authors demonstrate an approximate k-commodity flow algorithm that runs in $O(knm \log^4 n)$ time with high probability, where n is the number of nodes and m is the number of edges in the network. Given any multicommodity flow problem, this algorithm is guaranteed with high probability to provide a feasible solution, if one exists, in which all capacities are increased by a $(1+\epsilon)$-factor or to provide a proof that no feasible solution exists.

Another use of randomization techniques to construct approximation algorithms for multicommodity flow and other transportation related problems can be found in [34, 33]. This work introduces the concept of *randomized rounding* in which high-probability algorithms are obtained for integer problems by first solving a relaxed linear program and then using the resulting real-number values to determine integer values for the variables. (See [2] for recent results in using linear programming for solving complex transportation problems).

In [13] we apply this randomized rounding technique to the greyhound problem. We demonstrate the benefits of this method for restrictions of the greyhound problem by providing algorithmic solutions. We also demonstrate the limitations of applying this method to more general transportation problems. Algorithmic solutions to the dial-a-ride problem are discussed in [31, 39, 8, 41].

It should be noted that approximation and randomization techniques can be applied to the dynamic problems discussed in Section 5 as well as the static ones discussed here. A good reference for surveys and results for combinatorial optimization problems is [30].

5 Dynamic Complexity Issues and Solution Concepts

In Section 4.1, we explore research that points to static sources of complexity that tend to make a problem hard. In Section 4.3, we discuss approximation algorithms and how to provide performance guarantees for the corresponding problem reformulations. This is only half the story. We have thus far completely neglected the additional complexity due to dynamic sources. In particular, the general transportation problem introduced in Section 2 allows for uncertain and changing basic entities in the problem definition, partial and uncertain information in the communications structure and distributed control in the solutions. This additional complexity is in part due to the fact that a transportation problem naturally consists of the concurrent operation of dynamical systems in an uncertain environment. While related research provides much insight into static complexity, there is a relative paucity of related research considering these dynamic sources of complexity. In the following we describe some related research and discuss ways to extend solutions to static problems into solutions addressing the dynamic sources of complexity as well.

5.1 Dynamic Sources of Complexity from Related Research

One line of related research provides insights into the dynamic sources of complexity in the transportation problem is the field of discrete event systems (DES). This research is particularly applicable to the distributed, decentralized aspects of our problem in that there are some noteworthy results concerning decentralized supervisory control in the DES literature. Results from this area may also be applicable to the dynamic nature of the transportation problem, including their consideration of the problem of partial information.

The results of Ramadge, Wonham and Lin [35, 27] are encouraging in that they are the first results we have come upon that enable us to formally analyze the feasibility of a distributed solution. They provide conditions under which distributed solutions are provably feasible.

In the DES context, the complexity of synthesizing supervisors is the relevant measure. A *supervisor* is a control program that restricts the dynamical system from entering certain states. Ramadge and Wonham [35] give polynomial algorithms for solving some supervisor synthesis problems. However, they show that it is unlikely that techniques to realize minimal supervisors will be feasible in large problems. Also, these algorithms are polynomial in the size of the state space, which may not be a useful measure in the transportation problem since the state space is exponential in the number of state variables. Ramadge and Wonham also mention results related to the use of the shuffle product (forming DES from the concurrent actions of two or more asynchronous and independent machines) in modeling the concurrent operation of several discrete dynamical systems. In these systems the number of states increases exponentially with the number of components.

Tsitsiklis [42] provides necessary and sufficient conditions for the existence of certain classes of supervisors and then demonstrates polynomial-time algorithms for showing whether or not those conditions are met for specific DES. However, this only shows the existence of a supervisor in polynomial-time. There is not an efficient method for actually synthesizing the supervisor. It is not necessarily the case that a supervisor with polynomial-state space may always be found. As mentioned above, shuffle product systems have a state-space that is exponential in the size of the state space of a single component. Tsitsiklis demonstrates polynomial algorithms for some control problems, but also shows that some control problems of interest are computationally difficult. He has also observed that problems with partial information are often algorithmically intractable.

5.2 Solution Concepts

In the problem description of Section 2, we discuss the basic entities and decision-making structure that comprise a transportation problem. There is no explicit formula for how to deal with the dynamic, distributed and uncertain properties that many natural transportation problems tend toward. These issues may best be dealt with not as part of the problem formalization but as part of the solution. In this section, we sketch this approach to dealing with dynamic complexity.

In some sense, the decision-making structure defined in a particular transportation problem formulation dictates the form of the solutions. For example, a decision-making structure that is defined to require multiple, distributed agents may lead to vastly different solution techniques than a problem in which the decision-making structure can be designed to best suit the problem. An unrestricted decision-making structure can lead to more flexibility in constructing solutions, whereas a problem in which the decision-making structure is fixed in advance has a more restricted choice of solution techniques.

The structures through which the decision-making structure dictates the form of the solution include:

- number of decision-making agents,
- information access function,
- communication access function, and
- control authorization function.

For example, if the decision-making structure includes multiple agents and constrains the control authorization function such that every agent is only authorized to change the program of a specific subset of assets at any time, we must derive a solution that includes multiple distributed controllers.

We define some solution concepts that can be combined to form a complete solution as follows:

Control :

Centralized: A single agent has exclusive access to all programmable assets.

Distributed: Multiple agents have individual access to programmable assets.

Coordination :

Centralized: A single agent determines the coordination of all agents.

Distributed: Multiple agents, each independently (or through communication) determine the coordination of a subset of agents.

Data Access :

Global: All data (state information) is equally accessible to any agent.

Local: Access to data (state information) is restricted by location and/or other factors.

Predictive Model :

Perfect: The deterministic predictive model of the underlying dynamical system is known and allows perfect extrapolation.

Imperfect: The system model is incomplete, stochastic, or error prone and, therefore, extrapolation is imperfect.

Schedule Generation :

Static: Entire schedule is generated at $t = 0$ and then executed without feedback.

Dynamic: Schedule is generated and modified on-line as conditions change.

The five solution concept categories can be combined in numerous ways, subject to the restrictions in the decision-making structure of the problem. These concepts form a space of complete solution points from which to search. When searching for techniques to employ in solving the transportation problem, we can restrict our search to one point in this space at a time. For example, the combination of distributed control, centralized coordination, global data access, perfect model and dynamic schedule generation form one point in this space.

The solution space concept can be best used to direct the search for solutions within a particular formulation of the transportation planning problem. For any single transportation problem, a series of solution points may be explored. Conversely, for any single solution point there may be many alternative problem formulations to explore. We discuss alternate problem formulations in Section 4. Therefore, a research program may move simultaneously in two spaces: the problem space and the solution space, alternatively reformulating the problem and considering different points in the solution space until a satisfactory solution to a natural problem is derived.

It may be the case that the class of interesting problems are such that the decision-making structure completely restricts the solution space from within the problem description. However, maintaining the distinction between problem concepts and solution concepts allows us to better generalize solution techniques for different problem formulations and, conversely, move between solution concepts within a single problem formulation.

Although this section is concerned with extending static algorithms to algorithms that support dynamic complexity as well, it should be noted that exploring the static algorithms has merit of its own. In specific, in cases where fast medium bandwidth communication exists, static, centralized scheduling algorithms can be the major part of a solution to a particular transportation problem. The transportation problem consists of both high-level scheduling and low-level control issues. While the low-level control is inherently distributed, the high-level scheduling need not be.

5.3 Algorithms and Performance Guarantees that Address Dynamic Complexity

In addition to the approximation algorithm and randomization techniques discussed in Section 4.3, there are other algorithmic approaches to solving problems with dynamic complexities similar to those in transportation problems. In this section, we discuss two of these techniques that provide performance guarantees in the face of dynamic sources of complexity. They are *anytime algorithms* and *competitive algorithms.*

A classic military maxim is "Good plans offered now are always better than perfect plans offered later" [12]. This maxim points to the dynamic complexities involved in operating in a changing environment. Anytime algorithms address this issue by providing iterative algorithms that achieve better solutions as more time for computation is available, but are able to provide some solution at any time [3]. In other words, the quality of the solution (*i.e.*, performance guarantee) is a function of the computation time. This notion differs from the traditional concept in which algorithms provide only one solution at the completion of processing.

Anytime algorithms have many applications to transportation problems. In [4] anytime vehicle routing solutions are provided based on the edge-exchange algorithm of [28]. Anytime algorithms are useful for both dynamic and uncertain environments because they allow the decision-making agent to make effective use of the time available for decision-making while at the same time allowing for unexpected changes to interrupt the decision-making at any moment.

In Section 5.2, we consider the difference between the off-line (static) construction of schedules and the on-line (dynamic) construction of schedules. In [40] a method of analysis is developed that compares the performance of an on-line algorithm on any sequence of operations to the performance of the optimum off-line algorithm (*i.e.,* one in which the entire sequence is known in advance) on the same sequence of operations. This analysis is similar to that of approximation algorithms in that we are interested in the ratio of the best on-line to the best off-line solution,

$$\frac{online(I)}{offline(I)}.$$

An algorithm is termed *k-competitive* [23] if the ratio is within a factor k. An algorithm is considered *strongly competitive* if the factor obtained is provably the best possible. An example of a strongly competitive algorithm is given by [29].

In [1] the authors use the notion of *polylog(n)*-competitive to describe an on-line scheduling problem related to transportation problems. They describe an on-line algorithm to balance the job load in a network of processors that is provably *polylog(n)*-competitive with the best off-line algorithm. This scheduling problem is similar to the parallel machine scheduling problems mentioned in Section 4.1 and described fully in [25]. This demonstrates the use of competitive algorithms to bound the performance of an on-line scheduling problem.

In the past, researchers have recognized the tradeoffs inherent in the extremes characterized by anticipatory control strategies involving complex forms of inference and responsive control strategies involving much simpler forms of inference. There appear to be similar tradeoffs inherent in the extremes characterized by, on the one hand, distributed planning systems involving complex negotiation and cooperation and, on the other hand, fault tolerant systems involving much simpler protocols to effect coordination and guarantee behavior. In distributed planning, issues of time criticality are exacerbated by delays in communication. Many forms of inference can be protracted due to communications delays. Research on the transportation problem promises to redirect work in planning, scheduling and combinatorial optimization to address the dynamic problems faced by real-world, time-pressured embedded control systems.

6 Conclusion

This paper is intended to serve as a starting point for coordinating the work of computer scientists and operations researchers toward addressing the difficult issues inherent in the transportation problem. By providing a standard formalization in which to discuss the transportation problem, we hope to bring forth the commonalities of existing research in the two areas. We hope that this description facilitates technical discussion among participants in the Transportation Planning and Scheduling Initiative and within the larger community of researchers in planning, scheduling and control.

Acknowledgments

The authors would like to thank Jon Doyle, Paris Kanellakis, and Mike Wellman for their suggestions and feedback on an earlier draft of this paper. We would also like to thank Philip Klein and R. Ravi for their assistance in exploring approximation algorithms for the greyhound problem.

References

[1] Awerbuch, Baruch, Kutten, Shay and Peleg, David, Competitive Distributed Job Scheduling, in preparation.

[2] Bixby, R., Gregory, J., Lustig, I., Marsten, R. and Shanno, D., Very Large-Scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods, *Rutcor Research Report 34-91*, June, 1991.

[3] Boddy, Mark, Solving Time-Dependent Problems: A Decision-Theoretic Approach to Planning in Dynamic Environments, Ph.D. Thesis, Brown University Department of Computer Science, (Providence, RI 1991).

[4] Boddy, Mark and Dean, Thomas, Solving Time-Dependent Planning Problems, *Proceedings IJCAI 11*, (Detroit, Michigan 1989) 979–984.

[5] Boddy, Mark and Dean, Thomas, Approximation Algorithms for Planning and Control, *Proceedings of the NASA Conference on Telerobotics*, 1989.

[6] Bodin, Lawrence D., Twenty Years of Routing and Scheduling, *Operations Research*, (38:4) July–August 1990, 571–579.

[7] Bodin, Lawrence and Golden, Bruce, Classification in Vehicle Routing and Scheduling, *Networks*, (11) 1981, 97–108.

[8] Bodin, L., and Sexton, T., The Subscriber Dial-a-Ride Problem, Final Report, U.S. Department of Transportation, Urban Mass Transportation Administration, Washington, D.C. 1979.

[9] Christofides, N., Worst-case Analysis of a New Heuristic for the Traveling Salesman Problem, *Technical Report, Graduate School of Industrial Administration*, Carnegie-Mellon University (Pittsburgh, PA 1976).

[10] Christofides, N., 'Vehicle Routing', in *Combinatorial Optimization: Annotated Bibliographies*, ed. O'hEigeartaigh, M. and Lenstra, J.K. and Rinnooy Kan, A., (John Wiley & Sons, New York 1985).

[11] Cormen, Thomas, Leiserson, Charles and Rivest, Ronald, *Introduction to Algorithms*, (MIT Press, 1990).

[12] Cross, Major Stephen, A Proposed Initiative in Crisis Action Planning, Information Science and Technology Office, DARPA, (Arlington, VA 1990).

[13] Dean, Thomas and Greenwald, Lloyd, Provable Performance Guarantees for Package Routing in Transportation Networks with Fixed Vehicle Schedules, in preparation.

[14] Dean, Thomas and Wellman, Michael, *Planning and Control*, (Morgan Kaufmann Publishers, San Mateo, CA 1991).

[15] Even, S., Itai, A. and Shamir, A., On the Complexity of Timetable and Multi-Commodity Flow PRoblems, *SIAM Journal on Computing*, (5:4) December, 1976, 691–703.

[16] Garey, M. and Johnson, D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, (New York, W. H. Freeman and Company, 1979).

[17] Global Transportation Network (GTN), Trans Plan Concept of Operations, (1991).

[18] Gopal, M., *Modern Control System Theory*, (Halstead Press, New York 1985).

[19] Greenwald, L., Distributed Transportation Scheduling: Preliminary Complexity Perspective, *Brown University Computer Science Department, Unpublished Notes*, (Providence, 1991).

[20] Itai, Alon, Two-Commodity Flow, *J. ACM*, (25:4) October, 1978, 596–611.

[21] The Joint Staff Officer's Guide 1988, National Defense University, Armed Forces Staff College, (Norfolk, VA 1988).

[22] Kalman, R. E., Falb, P. L., and Arbib, M. A., *Topics in Mathematical System Theory*, (McGraw-Hill, New York, 1969).

[23] Karlin, A., Manasse, M., Rudolph, L. and Sleator, D., Competitive Snoopy Caching, *Algorithmica*, (3:1) 1988, 79–119.

[24] Lawler, E., Lenstra, J., Rinnoy Kan, A. and Shmoys, D., *The Traveling Salesman Problem*, (John Wiley & Sons, 1985).

[25] Lawler, E., Lenstra, J., Rinnooy Kan, A. and Shmoys, D., Sequencing and Scheduling: Algorithms and Complexity, *Report BS-R8909, Centre for Mathematics and Computer Science*, (Amsterdam, 1990).

[26] Leighton, Tom, Makedon, Fillia, Plotkin, Serge, Stein, Clifford, Tardos, Éva and Tragoudas, Spyros, Fast Approximation Algorithms for Multicommodity Flow Problems, *ACM*, (1991) 101–111.

[27] Lin, F., and Wonham, W. M., Decentralized Supervisory Control of Discrete-Event Systems, *Inform Sci.*, **44** (1988) 199–224.

[28] Lin, S. and Kernighan, B. W., An Effective Heuristic for the Travelling Salesman Problem, *Operations Research*, (21), 1973 498–516.

[29] McGeoch, L. and Sleator, D., A Strongly Competitive Randomized Algorithm for Paging, 1989.

[30] O'hEigeartaigh, M., Lenstra, J.K. and Rinnooy Kan, A., eds. *Combinatorial Optimization: Annotated Bibliographies*, (John Wiley & Sons, New York 1985).

[31] Psarafitis, H., Dynamic Programming Algorithms for Specially Structured Sequencing and Routing Problems in Transportation, Ph.D. Thesis, MIT, (Cambridge, MA 1978).

[32] Raghavan, Prabhakar, *Lecture Notes on Randomized Algorithms*, (Yorktown Heights, 1989).

[33] Raghavan, Prabhakar, Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs, *Computer and Systems Sciences*, (37) (1988) 130–143.

[34] Raghavan, Prabhakar and Thompson, Clark, Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs, *Combinatorica*, (7:4) (1987) 365–374.

[35] Ramadge, Peter and Wonham, Murray, The Control of Discrete Event Systems, *Proceedings of the IEEE*, **77**(1) (1989) 81–98.

[36] Sahni, Sartaj, Computationally Related Problems, *SIAM J. Comput.*, (3:4) December, 1974, 262–279.

[37] Sahni S. and Gonzalez T., P-complete approximation problems, *Journal of the ACM*, (23) 1976, 555–565.

[38] Schank, John, Mattock, Michael, Sumner, Gerald, Greenberg, Irwin, Rothenberg, Jeff and Stucker, James, A Review of Strategic Mobility Models and Analysis, *Rand Report R-3926-JS.*

[39] Sexton, T., The Single Vehicle Many to Many Routing and Scheduling Problem, Ph.D. Thesis, State University of New York at Stony Brook, (Stony Brook, NY 1979).

[40] Sleator, D., and Tarjan, R. Amortized Efficiency of List Update and Paging Rules, *Communication of the ACM*, (28:2) 1985, 202–208.

[41] Stein, D., Scheduling Dial-a-Ride Transportation Systems, *Trans. Sci.*, (12) 1978 232–249.

[42] Tsitsiklis, J. N., On the Control of Discrete Event Dynamical Systems, *Math. Contr., Signals, and Syst.*, **2**(1) (1989) 95–107.

[43] United States Transportation Command: Command, Control, Communications, and Computer Systems Master Plan (1989).