

Capacity Allocation to Sales Agents in a Decentralized Logistics Network*

Ozgun Caliskan Demirag, Julie L. Swann

School of Industrial and Systems Engineering,
Georgia Institute of Technology
Atlanta, GA, 30332, USA
{ocaliska,jswann}@isye.gatech.edu

Accepted at *Naval Research Logistics* (June 13, 2007)

Abstract

Many logistics systems operate in a decentralized way, while most optimization models assume a centralized planner. One example of a decentralized system is in some sea cargo companies: sales agents, who share ship capacity on a network, independently accept cargo from their location and contribute to the revenue of the system. The central headquarters does not directly control the agents' decisions but can influence them through system design and incentives. In this paper, we model the firm's problem to determine the best capacity allocation to the agents such that system revenue is maximized. In the special case of a single-route, we formulate the problem as a mixed integer program incorporating the optimal agent behavior. For the NP-hard multiple-route case, we propose several heuristics for the problem. Computational experiments show that the decentralized system generally performs worse when network capacity is tight and that the heuristics perform reasonably well. We show that the decentralized system may perform arbitrarily worse than the centralized system when the number of locations goes to infinity, although the choice of sales incentive impacts the performance. We develop an upper bound for the decentralized system that gives insight on the performance of the heuristics in large systems.

1 Introduction

1.1 Motivation

The optimization of logistics systems has resulted in many improvements such as reduced costs, shorter lead-times, and better customer service. However, most optimization models have been

*This research was supported by NSF grant DMI-0348532. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

designed to be used by a centralized planner who makes system-wide decisions, while many organizations actually operate in a decentralized manner with agents making independent decisions. The individual agents may make locally optimal decisions for the part of the system that they manage, but in situations where they share resources, their decisions may have negative consequences for the overall organization.

In practice, many examples of decentralized systems exist due to several reasons. For example, decentralized decision-making can provide responsiveness and flexibility in handling uncertainty in environments such as military operations, where individual units may be authorized to make independent decisions in real-time [21]. Legal barriers to centralization may also exist: franchise laws in the US prohibit auto manufacturers from selling vehicles directly to consumers, leading to decentralized distribution through independent dealerships. For some environments, centralization may be prohibitively expensive, very complex, or the coordination may be too much of a burden. This is especially true for large systems, which would require substantial computational power to store and process large amounts of information for centralized decision-making.

Decentralized systems are sometimes less efficient in terms of the system-wide performance. For example, in a decentralized supply chain, the entire system may incur a revenue loss of 25% due to a phenomenon called “double marginalization” [27]. Simply applying centralized optimization models to decentralized systems may not be appropriate, but it may be possible to design and optimize the system to achieve good performance. A classic example in the economics literature that analyzes decentralized decision-making is the class of problems called *Principal-Agent*, where the principal contracts with the agent for performing certain acts. (For further details on Principal-Agent problems, see for example [28] or [29].) In Principal-Agent problems, the utility of the agent may not be directly aligned with that of the principal, thus the principal designs incentives or mechanisms to achieve the desired performance in the decentralized system. The Principal-Agent framework can be applied to decentralized logistics systems by developing optimization models that incorporate individual behaviors and by designing incentives or mechanisms to improve the system performance.

The motivation for our study comes from a decentralized booking practice in some sea cargo companies that transport containerized cargo and provide *liner* service. Liner is one mode of shipping operation in which the companies provide service according to regular schedules and fixed

itineraries or service routes [5]. In systems we study, a sales agent is located at each port on a network of service routes, and he handles cargo bookings that originate from his port. The sales agent earns revenue according to the incentive determined by the central headquarters, e.g., a fixed proportion of the total revenue generated by the agent’s bookings. The central headquarters assigns aggregate capacity to each sales agent for total bookings out of the agent’s port, where the firm’s objective is to maximize total revenue. This decentralized booking practice can be thought of as a Principal-Agent problem where the principal is the central headquarters and the agents are the sales offices that book cargo. The mechanisms are capacity allocation and sales incentives, where the performance is measured by total system revenue.

In this paper, we develop models for the central headquarters to analyze and optimize capacity allocation and sales incentives to improve the performance of the decentralized system. We use network flow problems to incorporate agent behavior in our models, and we link these individual problems through an overall optimization problem that determines the capacity limits to the agents and maximizes system revenue. For the special case of a single route, we formulate a comprehensive model including allocation decisions and agent behavior to solve the problem, and in the NP-hard general case, we develop several heuristics that consider agent behavior. We analyze the worst-case performance of the decentralized system and develop an upper bound on the optimal revenue that provides insight on the performance of the heuristics. Our computational results indicate that decentralized booking control is generally worse when capacity is tight, and the heuristics perform reasonably well and fast. The models that we develop can be used in rough-cut capacity allocation planning and evaluating “what-if” scenarios for the system design. For example, the central headquarters can use these models to determine allocation of capacity to agents, to choose sales incentives, to evaluate investment in an expensive centralized booking system, or to assess alternate designs of the network.

1.2 Literature Review

Research relevant to our work has been done in areas such as Principal-Agent analysis, revenue management, sea cargo routing and scheduling, decentralized organizational design, and network equilibria.

Principal-Agent problems are well-studied in the economics and operations management litera-

ture. (See, for example, [6], [14], [28], and [29].) These problems arise in the field of non-cooperative game theory and include applications to compensation of executives, contracting of workers, and management of supply chains. Our work presents analysis of a Principal-Agent problem in a non-traditional area, with agents booking cargo on a network of service routes.

Revenue management, which is a practice used in industries such as airline, car rental, hotel and entertainment, is concerned with achieving maximum revenue from the sales of perishable assets. The field of revenue management is relevant to the problem that we study since price-differentiated cargo is transported using perishable ship capacity, and the objective is to maximize revenue. For some examples of research in airline and air cargo revenue management see [2], [8], [11], [16], [17], and [18], or see [22] for a review. More recently, there has been interest in applying revenue management techniques in the sea cargo industry; Lee et al. [20] analyze a revenue management problem where the carrier decides which demands to satisfy or postpone as they are requested by contract and non-contract shipping customers. While most research in revenue management takes a centralized perspective on booking control, we focus on a decentralized system.

There is also a body of literature that relates specifically to sea cargo beyond the last example, most of which deals with the routing and scheduling of ships or moving empty containers to match supply and demand. (See [5] for a review.) Different than the research in this stream, we take the ship routes and schedules as given, and we focus on maximizing revenue in a decentralized system.

Some research related to decentralized systems has been done in the allocation of shared resources in multi-level organizations. The most relevant ones to our study include Burton and Obel [3] and Gazis [10]. In both papers, the authors analyze a problem where a central organizing body distributes the total capacity of common resources to sub-units who make individual decisions regarding these resources. In the formulations, feasibility for each resource is ensured with a constraint that limits its overall usage to the available capacity. Although the problem that we study is similar in concept, our problem is complicated by the fact that the sales agents receive aggregate capacity limits for multiple network resources. An effective distribution of the common resources that achieves high revenue requires us to know the agents' booking decisions, which is difficult to characterize on the network.

A number of papers have studied the loss of efficiency (i.e., the loss in a decentralized system compared to a centralized one) in the context of Nash equilibria or network models. The term "price

of anarchy” was first used in [19] and [25] to quantify the degree of loss in a restricted network. Some examples of research analyzing the price of anarchy in capacitated transportation networks and competitive network environments include [7] and [15]. Perakis [26] generalizes the work in this stream by considering systems with non-separable, asymmetric and nonlinear costs. The author finds that the loss due to decentralization can be unbounded in the worst case. Although the decentralized problem we study is different, we show a similar worst-case result.

This paper is organized as follows. We describe the problems that we address in Section 2 with a focus on capacity allocation in the decentralized system. In Section 3, we analyze a special case, present our heuristics and provide further theoretical analysis. We present some observations from our computational study in Section 4 and conclude with a summary of our findings in Section 5.

2 Models

In this section, we introduce some definitions, state our assumptions, and formulate the problems that we analyze.

We are interested in networks that are composed of directed cycles (not necessarily simple), which we call *routes*. A route is a sequence of ports that begins and ends at a specific location in the network. We present an example in Figure 1 where the network consists of a single route with revisitations that originates and terminates at Port 1. A *leg* is identified by a pair of locations on a route that are visited consecutively by the asset, or ship. Demand is in the form of *Origin-Destination (O-D)* pair requests. Ship capacity refers to the number of containers a ship can carry on a particular route. Since liner operators have regularly departing schedules (e.g., weekly) and we are interested in a system with time-stationary parameters, thus we can view a snapshot of the network in time, with all demand forecasts and capacities defined for the time period being studied.¹

FIGURE 1 APPROXIMATELY HERE

Associated with each port on the network is a *sales agent*. In the problem we study, an agent

¹In practice, the models presented in the paper can be solved in a rolling horizon, where reallocation of capacities to the agents accounts for already confirmed bookings and new information about O-D pair demand.

at a particular port only books freight leaving from that port; in practice that agent has the most information about the booked cargo from his port and the feeder services into and out of the port. Note that this is different than booking practices for most passenger air travel, where travel agents can book demand to and from any location. The sales agent optimizes his individual objective (independently of the other agents) based on the sales incentive as determined by the central headquarters. Unless otherwise noted, we assume that the sales incentive is based on total revenue, and we do not model sales effort or cost as a function of the incentives. We use the term “agent” to represent the sales office at a port and we assume that the sales office acts as one entity towards the common objective.

The central headquarters assigns capacity on a given ship and route to each agent on that route; the capacity is an aggregate allocation that limits the total demand bookings out of the agent’s port rather than a capacity limit for each leg or O-D pair. We assume that the central headquarters has full information on prices and that O-D pair demand is known by both sales agents and central headquarters. In practice, demand is often forecasted by sales agents who give the information to central headquarters.

We assume there is no penalty associated with rejection of demand besides the lost revenue. Since ship operating cost is largely fixed and does not depend on accepted demand, we ignore this cost. We assume that customers are path indifferent, i.e., an O-D pair request may be transported on any ship travelling between its origin and destination ports. We do not allow transshipment and multiple loading/unloading of cargo. Finally, we assume that each unit of demand is equal to one container of the same size.

We identify the following problems:

- ***Capacity Allocation Problem (CAP)***: *CAP* determines the optimal set of capacity limits to allocate to each sales agent on each route. The central headquarters solves *CAP* to maximize total revenue, while ensuring that the overall solution determined from the agents’ booking decisions is feasible.
- ***Agent Problem (AP)***: *AP* determines the optimal set of accepted O-D pair demands for an agent. An individual sales agent solves *AP* to maximize his objective based on the sales incentive, while guaranteeing that the number of accepted demands is within his capacity limit as assigned by central headquarters.

- **Central Problem (CP):** *CP* determines the optimal set of O-D pair demands to accept for the system to maximize total revenue, while ensuring overall feasibility.

Our focus is primarily on *CAP*. In order to model *CAP*, we need to understand an agent's behavior, therefore we analyze *AP*. In *CP*, all decisions regarding demand acceptance are made in a centralized manner, therefore *CP* provides benchmark results for *CAP*.

Table 1 summarizes the main notation used in the paper. For notational convenience, the price per container for an O-D pair is the same regardless of the route chosen and the type of cargo, and the capacity on each leg of a route is equal to the ship capacity on the route.

TABLE 1 APPROXIMATELY HERE

We start by analyzing *AP*. An agent at port p receives the capacity limits \vec{a}_p and solves his individual problem, which we call AP_p . We show in the Appendix that AP_p can be formulated as a minimum cost flow problem. We formulate AP_p following the representation in [1], and we construct the underlying graph and choose the problem parameters according to our setting. Minimum cost flow problems are polynomially solvable and can be solved efficiently by specialized network flow algorithms.

We denote the optimal objective function value of AP_p with $Z_p^*(\vec{a}_p)$. From the optimal solution of AP_p , we obtain $y_k^{r*}(\vec{a}_p), \forall k \in OD_p, \forall r \in RR_k$, which denotes the number of demands accepted for O-D pair k on route r when agent p receives \vec{a}_p as capacity allocation.

Given the *AP* models, we present a mathematical formulation for *CAP*, where the decision variables are the capacity limits to allocate to each agent.

$$(CAP) \max_{\vec{a} \in \mathbb{Z}^+} \left\{ \sum_{p \in P} Z_p^*(\vec{a}_p) \text{ s.t. : } \sum_{p \in P} \sum_{k \in (OD_p \cap S_l^r)} y_k^{r*}(\vec{a}_p) \leq cap_r, \forall r \in R, \forall l \in L_r \right\}.$$

The objective is to maximize the total revenue generated from the booking decisions of all agents, while ensuring that the total accepted O-D pair demand on each leg is within the capacity limits. We can show that the multiple-route *CAP* is NP-hard [4], since a special case of it is equivalent to the directed multi-commodity integral flow problem, which is NP-hard [9]. However, in general, *CAP* is not equivalent to a multi-commodity flow problem, which assumes centralized control and does not incorporate the behavior of the individual agents.

The multiple-route *CAP* problem incorporates agent behavior through the individual *AP* solutions for each set of capacity allocation. Suppose instead, that one formulated an integer program without the *AP* solutions, where the decision variables are the capacity limits allocated to each agent and the number of demands accepted; constraints limit the bookings on each leg by ship capacity and each agent's bookings by his capacity allocation. Although the capacity limits are added as decision variables, the solution of this program is equivalent to the *CP* solution, and it does not model actual agent behavior in a decentralized system. In the decentralized system, a different approach is needed to incorporate agent behavior, ensure feasibility and maximize revenue. If the agent behavior can be explicitly characterized, then it would be possible to solve *CAP* in one comprehensive model that directly incorporates agent decisions. We show that this is possible for the special case of a single-route.

An alternative allocation scheme to aggregate capacity allocation on each route is *leg-based allocation*, where each agent receives separate capacity limits on each of the legs he can use. In this case, the feasibility of *CAP* is easily ensured in an integer programming model. In spite of this modelling convenience, a leg-based allocation scheme may not be appealing in practice for large systems. For instance, consider the size of a typical network for a sea cargo carrier operating in Intra-Asia (e.g., [23]). The sales agent in Hong Kong is located on a port that is serviced by 8 different routes. The sales agent would require 52 leg-based capacity limits or 41 O-D pair-based capacity limits compared to 8 aggregate capacity limits out of Hong Kong. Some sea cargo companies prefer the simplicity of assigning aggregate capacity despite the inefficiencies that may result. Moreover, assigning aggregate capacity limits rather than leg-based or O-D pair-based limits may be better as a planning tool since aggregate forecasts of demand are more accurate than forecasted demand for each O-D pair.

The last problem that we model is *CP*, where the central headquarters and the sales agents act as a single unit and jointly determine the best set of demands to accept in order to maximize the overall system revenue. *CP* can be formulated as an integer multi-commodity flow problem with bundling constraints on each leg for ship capacity; the commodities are the O-D pairs that differ by their origins and destinations. (See [4] for the formulation.)

3 Solution Approaches

3.1 Special Case: Single Route

In this section, we analyze capacity allocation in a decentralized network with a single-route, and we show several results using the special structure of *CAP*. These results also give us insight into the multiple-route problem and could also apply to other transportation areas, such as the railroad industry.

Similar to the analysis in Section 2, we first model the *Single-route Agent Problem (SAP)* to understand the behavior of the agents. Given an agent p , we call the agent's problem SAP_p and show the formulation below.

$$(SAP_p) \max_{y_k \in \mathbb{Z}^+} \left\{ \sum_{k \in OD_p} p_k y_k \text{ s.t. : } \sum_{k \in OD_p} y_k \leq a_p, y_k \leq d_k, \forall k \in OD_p \right\}.$$

The objective function is to maximize the revenue of the agent. The first set of constraints ensures that the total number of demands accepted is limited by the agent's capacity allocation. The second set of constraints restricts the number of accepted bookings for each O-D pair by the corresponding demand. SAP_p is equivalent to a continuous knapsack problem with integer data, which is optimized by a greedy algorithm that accepts the O-D pair demands in the order of non-increasing prices. This characterization of the agents' optimal solutions allows us to directly model agent behavior in the single-route case, which we incorporate in a mixed integer linear program for the *Single-route Capacity Allocation Problem (SCAP)*.

To formulate *SCAP*, we order the O-D pairs of each agent from highest to lowest in prices, and we obtain the set $\overline{OD}_p, \forall p \in P$. We use an index $l(k)$ to denote the order of O-D pair k in \overline{OD}_p , where $l(k) = 1$ denotes the O-D pair k with the highest price, and $l(k) = |OD_p|$ denotes the O-D pair k with the lowest price in OD_p . (Since O-D pair k is exclusively serviced by one agent, $l(k)$ uniquely identifies each O-D pair k .) We use the binary decision variables z_k^p ($p \in P, k \in OD_p$) to incorporate the precedence relations among O-D pairs of each agent. We formulate *SCAP* as

follows:

$$\begin{aligned}
(SCAP) \quad & \max \quad \sum_{k \in OD} p_k y_k \\
& \text{subject to} \quad \sum_{k \in S_l} y_k \leq cap \quad \text{for all } l \in L, \\
& \quad d_k z_k^p \leq y_k \leq d_k \quad \text{for all } p \in P, k \in OD_p : l(k) = 1, \\
& \quad d_k z_k^p \leq y_k \leq d_k z_{k'}^p \quad \text{for all } p \in P, k, k' \in OD_p : l(k) = l(k') + 1, l(k) > 1, \\
& \quad y_k \geq 0 \quad \text{for all } k \in OD, \\
& \quad z_k^p = \{0, 1\} \quad \text{for all } p \in P, k \in OD_p.
\end{aligned}$$

The first set of constraints ensures that leg capacities are not exceeded. The second set of constraints ensures that the demands accepted for the O-D pairs satisfy each agent's optimal behavior as characterized by the *SAP* solution. We calculate the capacity limits to agent p from the solution of *SCAP* as $a_p = \sum_{k \in OD_p} y_k, \forall p \in P$.

Finally, we model the *Single-route Central Problem (SCP)* as an integer program with a constraint matrix that has the circular 1's property in its columns. This is best seen when the matrix is formed with rows denoting the legs (in the order that they are traversed according to the sequence of ports) and columns with O-D pairs. Then, the matrix has an entry of 1 when the leg is traversed by the O-D pair in the corresponding column. For a definition of the circular 1's property and results on complexity status of related problems, see [12]. Since *SCP* is equivalent to *SCAP* without the precedence constraints, *SCAP* is at least as difficult as *SCP*, the complexity status of which is unknown to the best of our knowledge.

3.2 Heuristics for Multiple-Route CAP

As mentioned in Section 2, the multiple-route *CAP* is NP-hard. Therefore, we focus on developing efficient heuristics that incorporate agent behavior.

3.2.1 Marginal Revenue (MR) Heuristic

In this section, we describe the *Marginal Revenue (MR)* heuristic to solve *CAP*. Given current allocated capacities, we build a feasible solution by successively assigning fixed increments of capacity to the agent that brings the highest current additional revenue (or highest marginal revenue) without violating system feasibility. The latter is determined by finding the capacity used on each

leg considering bookings by all agents. Although designing an allocation mechanism based on marginal revenue is a simple idea, the revenue can be quite close to that of the optimal *CAP* solution. We summarize the main steps of the *MR* heuristic below and provide a formal description of the algorithm in the Appendix.

1. Initialize the capacity limits to all agents at zero.
2. For each agent p on each route $r \in R_p$, temporarily increase the agent's capacity limit on route r by a *stepsize* and solve AP_p .
3. Find the agent and route pair (p_{max}, r_{max}) that brings the largest increase in the objective function value and satisfies overall system feasibility.
4. Increase the capacity limits to agent p_{max} by *stepsize*.
5. Repeat Steps 2-4 while there exists an agent whose capacity limit can feasibly be increased on a route.

The most time consuming part of the algorithm is finding (p_{max}, r_{max}) , since this requires solving multiple *APs* for each new capacity increment and checking the feasibility of the overall system. An upper bound on the total number of minimum cost flow problems solved is $\max_{p \in P} \{ |R_p| \sum_{r \in R_p} (cap_r \cdot nv_p^r) \}$. The use of a scalar, nv_p^r , is needed in order to capture the increased capacity resulting from multiple visitations to a port in a particular route. The parameter *stepsize*, which is the size of the capacity increment in the allocation vector, may be increased to reduce runtime, although this can result in a decrease in the solution quality.

3.2.2 Priority O-D (POD) Based Allocation Heuristic

The *Priority O-D (POD)* based allocation heuristic is partly motivated by the single route results in Section 3.1, where the optimal solution of *AP* is such that the agents satisfy demands in a greedy fashion according to the ranked prices of the O-D pairs. Thus, for the *POD* heuristic, we form \overline{OD} by ordering the O-D pairs in the order of decreasing prices. We allocate capacity to the agents according to this priority list, where allocation increments are as large as possible without violating system feasibility. We summarize the main steps of the *POD* heuristic below and provide a formal description of the algorithm in the Appendix.

1. Start with the O-D pair k that has the highest price and find a route $r \in RR_k$ that may accommodate the largest amount of its demand.
2. Temporarily increase the allocation of the agent at the origin port of O-D pair k (op_k) on route r by the demand or the available capacity on that route, whichever is smaller.
3. Solve AP_{op_k} . Make the agent's allocation increment permanent if the new allocation does not violate the system feasibility. Go to Step 5 if O-D pair k is entirely booked.
4. Continue with $\bar{r} \in RR_k$, $\bar{r} \neq r$ and repeat Steps 2 and 3.
5. Continue with the next O-D pair in \overline{OD} until all O-D pairs have been considered or no capacity is remaining.

An alternative we consider to giving priority to the routes with higher allowable space (Step 1) is to solve the linear relaxation of CP and use the dual variables corresponding to the legs of the routes for ranking the routes. Neither of the two approaches to route selection is outperformed by the other in our computational experiments. The runtime of the algorithm is directly proportional to the total number of O-D pairs. For each O-D pair k , at most $|RR_k|$ number of minimum cost flow problems are solved.

While agent behavior is greedy and relatively easy to characterize in the single-route problem, this is not true in the multiple-route case. There may be several types of behaviors where the agent uses capacity differently than as estimated in the POD heuristic. In Figure 2, we illustrate a situation, which we call "O-D replacement", where the agent may use the (temporarily) allocated capacity in his own self interest and may cause system infeasibility. In this case, we can identify the behavior and do not increase the agent's capacity limit so that feasibility is maintained.

FIGURE 2 APPROXIMATELY HERE

Other kinds of agent behaviors may also occur, such as "O-D swapping", where the agent may swap routes used for previously considered O-D pairs and book an O-D pair with higher price using the new capacity; thus causing system infeasibility. This behavior is more difficult to characterize since it may result in multiple swaps of O-D pairs among routes. Therefore, we ensure the feasibility of the overall solution by solving AP in Step 3 of the heuristic. In this step, if

we detect that the optimal *AP* solution causes system infeasibility, we do not assign the capacity increment permanently to the agent.

The *POD* heuristic assigns capacities to the sales agents based on estimated agent behavior according to the priority list of O-D pairs. In contrast, the *MR* heuristic does not make any assumptions about the structure of agent behavior and solves a new *AP* to capture behavior for each potential capacity allocation. The *POD* heuristic allocates as much capacity as possible at each iteration of the algorithm, while the *MR* heuristic allocates fixed capacity increments determined by the *stepsize*. We show in our computational experiments that the *POD* heuristic is much faster than the *MR* heuristic with a small *stepsize*.

3.2.3 Benchmark Heuristics

We analyze two simple heuristics, *Equal Allocation (EA)* and *Conservative Allocation (CA)*, which are easy to implement and can be solved very quickly. In general, we do not expect that these heuristics perform well for *CAP*, but they provide another benchmark against which to compare the *MR* and *POD* heuristics. The *EA* heuristic equally distributes ship capacity on a route among the agents that can book demand on the route. The *CA* heuristic accounts for the O-D pair prices by solving a minimum cost flow problem, where the total capacity allocated to all agents on a route is restricted by the ship capacity. In contrast, the sum of the capacity limits in the optimal *CAP* solution may be greater than the ship capacity based on the agents' booking decisions.

The benchmark heuristics guarantee system feasibility independently of how the agents solve their own problems, although this is achieved at the expense of revenues. The *MR* and *POD* heuristics achieve higher revenues by incorporating how agents use their allocated capacities, thus accommodating higher total demand; however, they require more extensive effort to ensure overall feasibility.

3.3 Further Analysis of CAP

3.3.1 Alternative Agent Incentives

In all the models described so far, we assume that the agents maximize total revenue generated by their O-D pair bookings. This corresponds with the incentives that sales agents currently receive in some sea cargo companies, i.e., a proportion of the total revenue generated by their bookings.

The total revenue incentive might be disadvantageous from a system perspective when O-D pairs occupying a high number of legs bring higher revenue. We propose an alternative sales incentive, which we call *revenue per leg (rev/leg)*. Under the rev/leg incentive, an agent receives revenue for an O-D pair based on the route he chooses to transport that demand, where one unit of demand accepted for O-D pair k on a route r improves the agent's revenue by $p_k/|LL_k^r|$. Consequently, the agent's objective is to maximize the sum of the revenues generated per leg rather than the total revenue, while the central headquarters' objective is to maximize total system revenue under both incentives.

In our computational analysis we show that simply changing the agents' objectives in this way can be effective in moving the decentralized solution towards the centralized solution. One reason for its effectiveness is the following. If an O-D pair traverses over a large number of legs, then it typically shares capacity with a large number of other O-D pairs. The rev/leg incentive essentially penalizes O-D pairs that use a large number of legs, and therefore, a large amount of system capacity.

In some systems it is possible that the price to transport cargo already incorporates the number of legs traversed between its origin and destination ports. In that case, another incentive may be appropriate, such as *revenue/container-mile* if the price is not based on the distance the cargo is transported. Other incentives that may be appropriate in some situations include: i) the total number of demands booked by a sales agent ii) a proportion of the total system revenue. The latter aligns agent's revenue functions with the centralized system, although it may be difficult to implement since the agents may not have full knowledge of demand in other locations.

3.3.2 Theoretical Results for CAP

In this section we investigate the performance of the optimal decentralized system under the total revenue and rev/leg incentives. We show through examples that the optimal revenue of the decentralized system may be asymptotically worse than that of the centralized system. This is consistent with theoretical results on the price of anarchy in other kinds of decentralized networks [26]. This theoretical result also has practical implications as it indicates that investment in a centralized system may be cost effective. One interesting result is that when the decentralized system performed poorly under one incentive, the other incentive led it to achieve the optimal *CP* solution.

Theorem 1 *As the number of ports goes to infinity, the revenue obtained by the optimal CAP solution with the total revenue incentive may be arbitrarily worse than the revenue obtained by the optimal CP solution.*

To show the theoretical inefficiency of CAP, we focus on the special case of the single route, SCAP. We construct an example with a network of one simple cycle. That is, port visitations are in the form of $\{1, 2, 3, \dots, n, 1\}$, where n (the number of ports) ≥ 3 . We present the formal proof of this theorem and other key results in the Appendix.

Remark 1 *For the example in Theorem 1, the revenue obtained by the optimal SCAP solution with the rev/leg incentive is the same as the revenue obtained by the optimal SCP solution.*

Remark 1 shows that the rev/leg incentive achieves the best performance for the decentralized system in the example. Unfortunately, this incentive does not always give a good decentralized system outcome. The following is a result on the worst case performance of the rev/leg incentive.

Theorem 2 *As the number of ports goes to infinity, the revenue obtained by the optimal CAP solution with the rev/leg incentive may be arbitrarily worse than the revenue obtained by the optimal CP solution.*

To prove Theorem 2, we construct an example as we did for Theorem 1; the example has a similar network structure but has different prices for O-D pairs, and it is provided in the Appendix.

Remark 2 *For the example in Theorem 2, the revenue obtained by the optimal SCAP solution with the total revenue incentive is the same as the revenue obtained by the optimal SCP solution.*

Theorems 1 and 2 indicate that even if the decentralized system operates optimally, its performance may be very poor compared to the optimal centralized system, i.e., the price of anarchy may be high. Remarks 1 and 2 show that one incentive may perform better than another in a particular system, therefore careful choice of incentives is important to improve the decentralized system performance.

3.3.3 Upper Bound on Performance of CAP

Since we use heuristics to solve *CAP* for large problem sizes, additional analysis is needed to evaluate the performance of the heuristics. This is particularly important when the heuristics perform badly compared to *CP*; in that case, it is not clear if the gap between the heuristic solution and the *CP* solution is due to the nature of the heuristics or the inefficiency of the decentralized system. In the following, we introduce an upper bound on the performance of *CAP* that can be efficiently computed when it is not practical to solve *CAP* optimally for large problem sizes.

We compute the upper bound by solving the independent *APs* for all agents where the agents' capacity limits are obtained from the accepted demands in the optimal *CP* solution.

Lemma 1 *The procedure outlined in Steps 1-4 below provides an upper bound on CAP.*

1. Solve *CP* and obtain the solution vectors of accepted demand out of each port.
2. For each agent on each route, sum the accepted O-D pair demand originating from the agent's port in the *CP* solution, and let this be the agent's capacity allocation on that route.
3. Solve $AP_p, \forall p \in P$, with the agents' capacity limits as found in Step 2.
4. Sum the objective function values (total revenue) of the *APs* to obtain the upper bound on the optimal *CAP* revenue.

Given capacity limits from the *CP* solution, the agents can achieve a greater revenue by accepting O-D pairs with higher prices since overall system feasibility is not required. Thus, the total revenue generated by all agents is guaranteed to be at least as high as the objective function value of the *CP* solution. We also know that the *CP* solution is an upper bound on the *CAP* solution; therefore the result holds.

Introducing an upper bound for *CAP* that is weaker than the *CP* solution may seem ineffectual. However, we motivate this strategy by noting that the performance gap between the *CAP* and *CP* solutions may relate to the performance gap between the upper bound and *CP* solutions. For example, we intuitively expect *CAP* to perform badly compared to *CP* when the *CP* solution accepts many short O-D pairs but the agents prefer longer O-D pairs with higher prices. In this case, to ensure feasibility the *CAP* solution assigns relatively small capacity limits to the agents,

thus resulting in significantly smaller revenue than the *CP* solution. As a result, the revenue generated in the computation of the upper bound is significantly greater than the *CP* solution, since the agents receive capacity limits from the *CP* solution and they accept longer O-D pairs with higher prices.

Further, when capacity is very large relative to demand, both *CAP* and the upper bound find the *CP* solution. We conjecture that the gap between the upper bound and the *CP* solution is similar to the gap between the *CP* and *CAP* solutions. This is further supported by our numerical experiments. Consequently, the upper bound may give insight on the performance of the heuristics when the optimal *CAP* solution is not known.

4 Computational Experiments

The goals in our computational study are twofold. The first is to test the decentralized system performance under different sales incentives in order to understand factors that influence its performance relative to the centralized system. The second is to examine the performance of our heuristics.

4.1 Data Generation

The network characteristics in our computational study are intended to be similar to the industry practice of sea cargo logistics companies. These carriers provide service throughout the globe on trade lanes/routes (e.g., Trans-Pacific, Trans-Atlantic, Asia-North America, Asia-Europe, and Intra-Asia.), where each route is further divided into sub-routes. For example, OOCL’s Trans-Pacific service area includes 11 routes visiting a total of 29 different ports, and all routes are non-simple cycles with approximately 25% of the ports being revisited in the port sequence [24].

In data generation, we first specify the number of ports and the number of routes. Next, we randomly assign the ports to the routes and determine a port rotation (or sequence) within each route, allowing ports to be visited multiple times. We limit the number of revisitations to 1/3 of the total number of ports.

The price of an O-D pair is a uniform random variable taking values between 0 and 2000, and the demand of an O-D pair is a uniform integer random variable taking values between 0 and

Maxdemand. In our experiments, we choose the values for *Maxdemand* as 30, 100 and 500. We calculate the ship capacity on a route as a fraction (*Capacity/demand ratio*) of the total demand for the most highly requested leg on the route. We choose the values of *Capacity/demand ratio* as 0.3, 0.5, and 0.8, representing low, medium and high capacity levels. In the computational experiments, we generate 30 random instances and average over the iterations. We conduct all computational experiments on a computer with a 2.66 GHz Intel processor and 1 Gb of memory. We implement the formulations for the *APs* and *CP* using ILOG’s OPL Studio 3.7, and we use the callable component library feature (C++ API) to access ILOG’s CPLEX 9.0 solver when solving *CAP*.

4.2 Enumerative Method

The optimal solution of *CAP* can be found by an enumerative algorithm that checks all possible combinations of capacity limits for agents to find the set with the highest system revenue. However, such an algorithm is impractical for reasonable problem sizes since the number of possible combinations is very large. In an instance with $|P| = 10, |R| = 5, |R_p| = 2, cap_r = 1000, nv_p^r = 1, \forall p \in P, \forall r \in R$, the total number of combinations is $\prod_{p \in P} (\prod_{r \in R_p} (cap_r \cdot nv_p^r)) = 10^{60}$. Thus, to evaluate the heuristics, we solve the *CAP* to optimality for small instances. The optimal *CAP* solution also gives us insight on the performance of the upper bound that we introduced in Section 3.3.3, which we calculate for all problems.

Note that for a given vector of allocated capacities, there may exist alternative optimal solutions of *APs*. In such cases, system feasibility may be violated if the agents use capacity differently than predicted. The solution methods that we propose for *CAP* are guaranteed to contain a set of optimal *AP* solutions that constitutes a system feasible solution. In cases where the optimal *AP* solution can uniquely be found, system feasibility is always ensured. In practice, this can be achieved by the use of an optimization software throughout the company that has the same deterministic solver engine [13]. Perturbing the O-D pair prices such that it is less likely for multiple combinations of O-D pairs to have the same revenue can also be effective in alleviating the existence of alternative optimal solutions.

4.3 Experimental Results

In Figure 3, we show the upper bound and the performance of various solution methods for *CAP* as a percentage of the optimal *CP* revenue. In Figure 3(a), which is based on small-sized instances, we see that both the optimal and heuristic decentralized system solutions are able to achieve at least 88% of the optimal *CP* revenue. In the worst case, the heuristics achieve 98% of their respective optimal decentralized system revenues in the *CAP* solution. The upper bound also has a similar performance to that of the decentralized optimal solution, performing the worst when capacity is tight.

In Figure 3(b), we analyze the performance of heuristics using larger-sized instances. As expected, the *MR* and *POD* heuristics perform significantly better than the benchmark heuristics. For example, the former achieve 80% of the centralized revenue under the tightest capacity, while the latter attain no more than 43% in that case. Therefore, it is valuable to improve performance by explicitly solving *APs* as in the *MR* and *POD* heuristics. The *MR* heuristic with rev/leg incentive has the best performance of the heuristics (within 5% of the optimal *CP* revenue). For the large problems, the upper bound may be used as a proxy for the performance of the decentralized optimal solution; note that the gap between the upper bound and the optimal *CP* revenue as a function of capacity is about the same in both graphs.

One observation that is common to Figures 3(a) and 3(b) is that the performance of all heuristic solutions improves relative to the optimal *CP* solution with an increasing *Capacity/demand ratio*. We can explain this observation as follows. As capacity becomes tight compared to demand, the revenue loss for not accepting the most profitable demands is more important since any loss may have a significant impact.

FIGURE 3 APPROXIMATELY HERE

In Figure 4, we show examples on additional networks with different sizes. Note that, the upper bound and the optimal *CAP* solution behave similarly relative to the optimal *CP* solution, which is consistent with our conjecture in Section 3.3.3.

FIGURE 4 APPROXIMATELY HERE

Two performance measures that are of interest to the cargo carriers are the service levels of the system and the utilizations of the ships. We calculate the service level for each agent as the ratio of total demand accepted by the agent to the total demand out of the agent’s home port. In Figure 5(a), we depict the average service levels across all agents as a function of *Capacity/demand ratio*. One interesting observation is that the *MR* heuristic solution with the rev/leg incentive achieves higher service levels than the optimal *CP* solution. The ship utilization on each route is equal to the average leg occupancy, which is defined as the ratio of total flow on a leg to the route capacity. In Figure 5(b), we show the ship utilizations averaged over all routes. We see that ship utilizations are significantly higher in the optimal *CP* solutions compared to the *MR* heuristics because the *CP* solution is able to use the ship capacities more efficiently without considering agent behavior.

FIGURE 5 APPROXIMATELY HERE

The average runtimes of the algorithms for the decentralized system are given in Table 2 for the instances corresponding to Figure 3. In Table 2(a), which shows the small networks, the runtimes of all algorithms increase as the *Capacity/demand ratio* increases. As we expect, the enumerative algorithm that finds the optimal *CAP* solution increases at a much faster rate than the runtimes of the heuristics. Table 2(b) shows the average runtimes of the heuristics for the larger network. Since *CAP* is solved quarterly or weekly rather than in real-time, these runtimes are reasonable. Also note that the runtime of the *POD* heuristic is significantly smaller than the *MR* heuristic, while its performance in these instances is at most 10% worse than the latter. We may reduce the runtimes of the *MR* heuristics by choosing larger *stepsize* values, although note that this can decrease the quality of the solution. In the instances with medium capacity in Table 2(b), the *MR* heuristic with the total revenue incentive achieves 93% of the centralized optimal revenue with a *stepsize* of 1, as compared to 87% with a *stepsize* of 10 and a runtime of 35 CPU seconds.

TABLE 2 APPROXIMATELY HERE

5 Conclusions

In this paper we study a Principal-Agent problem on a network motivated by practices in some sea cargo companies. In such companies, the central headquarters, acting as the principal, assigns capacity limits and incentives to sales agents, and the sales agents individually book O-D pair demands originating from their home port. This is an example of a system that operates in a decentralized fashion, and it is sometimes less efficient than a centralized system where all decisions are made by a central organizing body. However, decentralized systems are desirable when centralization requires high investment and operating costs. Our research focuses on managing the decentralized system efficiently by developing optimization tools to determine the capacity limits and incentives for the sales agents.

We formulate a capacity allocation problem for the central headquarters that determines the capacity limits by explicitly incorporating agent behavior with network flow models. For the special case of the single route, we show that the optimal agent behavior can be included in a mixed integer program for the capacity allocation problem. In the general case of the multiple-route, we show that the capacity allocation problem is NP-hard, and we develop several heuristics that integrate agent behavior. We show an asymptotic result that the decentralized system performance can be arbitrarily worse than the centralized system performance. On the other hand, for more typical networks, we use computational experiments to show that the decentralized system performance can be close to the centralized system performance, and its performance improves as capacity increases. We further show that the choice of sales incentive is a factor that can improve the performance of the decentralized system. Our models can be useful in rough-cut capacity planning and for the managers to evaluate “what-if” scenarios.

There are several limitations to our work. First, we do not explicitly model the agents’ decisions in choosing sales effort. Therefore, the models are expected to be useful when the effect of incentives on effort levels is limited. Next, we do not include competition among agents or sea cargo companies. We also study a deterministic system as a first step towards studying the more complex real system that is stochastic and dynamic.

Our work develops optimization models to manage a system with individual decision makers. Analyzing decentralized decision-making is important since there are many systems in practice that operate in a decentralized manner, and most optimization models in literature assume a centralized

planner. It would be interesting to extend this research by modelling the decentralized booking system in a stochastic setting, where capacity can be re-allocated to the sales agents as demand bookings are realized. Further research also includes the analysis of other decentralized systems that could benefit from the design of optimization models to explicitly capture agent behavior and mechanism design.

6 Appendix

6.1 AP Formulation

The underlying graph consists of nodes corresponding to the O-D pairs $(n_k, k \in OD_p)$, routes $(m_r, r \in R_p)$ and a sink node (t) for feasibility. We denote the network for agent p with $H_p = (N_p, A_p)$. The problem parameters corresponding to the net supply at node i , (b^i) , the per unit cost of sending flow on arc (i, j) , (c_{ij}) , and the upper bound on arc (i, j) , (u_{ij}) , are defined below.

$$N_p = \{t \cup (\bigcup_{k \in OD_p} n_k) \cup (\bigcup_{r \in R_p} m_r)\}$$

$$A_p = \{(\bigcup_{k \in OD_p, r \in RR_k} (n_k, m_r)) \cup (\bigcup_{k \in OD_p} (n_k, t)) \cup (\bigcup_{r \in R_p} (m_r, t))\}$$

$$b^i = \begin{cases} d_k & \text{if } i = n_k \\ -\sum_{k \in OD_p} d_k & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}; c_{ij} = \begin{cases} -p_k & \text{if } (i, j) = (n_k, m_r) \\ 0 & \text{otherwise} \end{cases}; u_{ij} = \begin{cases} a_p^r & \text{if } (i, j) = (m_r, t) \\ d_k & \text{otherwise.} \end{cases}$$

Given port p and the vector of allocated capacities \vec{a}_p , the formulation for AP_p with the decision variables x_{ij} denoting the flow on arc (i, j) is:

$$(AP_p) \quad Z_p^*(\vec{a}_p) = \min \quad \sum_{(i,j) \in A_p} c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{(i,j) \in A_p} x_{ij} - \sum_{(j,i) \in A_p} x_{ji} = b^i, \quad \forall i \in N_p$$

$$0 \leq x_{ij} \leq u_{ij}.$$

From the optimal solution of AP_p , we obtain $y_k^{r*}(\vec{a}_p), \forall k \in OD_p, \forall r \in RR_k$, using the following relation: $y_k^{r*}(\vec{a}_p) = x_{(n_k, m_r)}^*$. When the rev/leg incentive is used, we modify the AP_p formulation as follows:

$$c_{ij} = \begin{cases} -p_k/|LL_k^r| & \forall (i, j) = (n_k, m_r), \forall k \in OD_p \\ 0 & \text{otherwise.} \end{cases}$$

6.2 MR Heuristic

We formally describe the algorithm in Table 3. In implementation, we store the optimal AP solutions of all agents, based on their most recent allocated capacities in a *solutions array*.

TABLE 3 APPROXIMATELY HERE

6.3 POD Heuristic

In order to formally describe the algorithm in Table 4, we introduce the following notation: r_l denotes the booked capacity on leg l , \bar{d}_k denotes the remaining demand for O-D pair k , w_k^r denotes the available space on route r that may be devoted to booking of O-D pair k . As defined in Table 1, \overline{OD} denotes the ordered set of all O-D pairs with non-increasing prices, where the first O-D pair has the highest price. For easy representation, we consider the routes with no particular order in the description of the algorithm. See discussion on the *POD* heuristic in Section 3.2.2 for the choice of the order of routes.

TABLE 4 APPROXIMATELY HERE

6.4 Proofs of Theorems and Remarks

For each O-D pair k , let ij be an alternative index where $i = op_k$ and $j = dp_k$. Parallel to Table 1, \overline{OD}_i is the ordered set of O-D pairs of the agent at port i , representing the priority list of the agent under the total revenue incentive. Let \overline{OD}'_i represent the priority list under the rev/leg incentive. We let $F_{SCAP}^*(n)$ and $\bar{F}_{SCAP}^*(n)$ denote the optimal revenue of the *SCAP* solution with the total revenue and rev/leg incentive, respectively. We denote the optimal revenue of the *SCP* solution by $F_{SCP}^*(n)$.

Proof. (Theorem 1)

In the example with travel sequence $\{1, 2, 3, \dots, n, 1\}$, let $cap = 1$ and $d_{ij} = 1 \forall i, j : 1 \leq i < j \leq n$

and $\forall i \neq 1, j = 1$. Assume no demand exists for the other O-D pairs. We let $0 < \gamma < \frac{p}{n}, 0 < \epsilon < \frac{p}{n}$, and we define the parameters as follows:

$$p_{1n} = p; p_{n1} = p - \epsilon;$$

$$p_{i(i+1)} = p - \epsilon, \forall i = 1, \dots, (n - 1);$$

$$p_{ij} = p - \epsilon + \gamma(|LL_{ij}| - 1), \forall (i, j) \text{ s.t.: } |LL_{ij}| > 1, (i, j) \neq \{(1, n), (n, 1)\}.$$

In this example, we show that $\lim_{n \rightarrow \infty} \frac{F_{SCAP}^*(n)}{F_{SCP}^*(n)} = \frac{2p - \epsilon}{n(p - \epsilon)} = 0$. First, note that: $\overline{OD}_i = \{OD_{i1}, OD_{in}, OD_{i(n-1)}, \dots, OD_{i(i+1)}\}, \forall i : 1 < i \leq (n - 1)$. Since capacity is 1, each agent may at most receive an allocation of 1, which he will use to accept the highest priority O-D pair of the agent, i.e., OD_{1n} for agent 1, OD_{n1} for agent n, and OD_{i1} for the other agents. However, since the highest priority O-D pairs of all agents except 1 and n share at least one leg, these agents cannot be assigned a positive allocation at the same time in a feasible solution. Therefore, the set of feasible solutions of *SCAP* with the total revenue incentive is:

1. $a_1 = 1, a_n = 1, a_i = 0, \forall i : 1 < i \leq (n - 1)$. In this case, $F_{SCAP}(n) = 2p - \epsilon$ with accepted demand $y_{1n} = y_{n1} = 1$ and $y_{ij} = 0$ for all other O-D pairs.
2. For each agent $i : 2 \leq i \leq (n - 1)$, a feasible solution of the form $a_i = 1$ with solution $y_{i1} = 1$ and $F_{SCAP}(n) = p_{i1}$. The solution $y_{21} = 1$ and $y_{ij} = 0$, for all other O-D pairs with objective value $F_{SCAP}(n) = p - \epsilon + (n - 2)\gamma$ has the highest revenue in this set of solutions.

Since $\gamma < \frac{p}{n} < \frac{p}{(n-2)} \Rightarrow 2p - \epsilon > p - \epsilon + (n - 2)\gamma$, the optimal solution value is $F_{SCAP}^*(n) = 2p - \epsilon$ corresponding to the solution in 1 above.

Next we show that $F_{SCP}^*(n) = n(p - \epsilon)$, which corresponds to the optimal solution $y_{i(i+1)}^* = 1, \forall i : 1 \leq i < n$ and $y_{n1}^* = 1$. For this instance, any accepted demand from *SCAP* that is an O-D pair with more than one leg may be replaced by the 1-leg O-D pairs occupying the same set of legs but with higher total revenue. This is possible in the centralized system because there is no need to incorporate agent behavior. In a solution, if OD_{ij} demand is accepted, we can improve this solution by rejecting OD_{ij} demand and instead accepting the $|LL_{ij}|$ 1-leg OD demands since $p - \epsilon + (|LL_{ij}| - 1)\gamma < |LL_{ij}|(p - \epsilon)$ for any $(i, j) : |LL_{ij}| > 1, (i, j) \neq (1, n)$. This is also true for $(1, n)$ since $(2p - \epsilon) < n(p - \epsilon)$. Since any solution that accepts an OD_{ij} with $|LL_{ij}| > 1$ may be improved by such replacements, $F_{SCP}^*(n) = n(p - \epsilon)$. Therefore, we have that $\frac{F_{SCAP}^*(n)}{F_{SCP}^*(n)} = \frac{2p - \epsilon}{n(p - \epsilon)}$, and the limit of this worst case ratio approaches 0 as the number of ports grows. ■

Proof. (Remark 1)

Corresponding to the example in the proof of Theorem 1, $\overline{OD}'_i = \{OD_{i(i+1)}, OD_{i(i+2)}, \dots, OD_{in}\}$, since $\frac{p_{ij}}{|LL_{ij}|}$, $\forall i : 1 < i \leq (n-1)$ is decreasing. Since shorter O-D pairs have higher priorities, the optimal centralized solution that accepts all of the 1-leg O-D pairs is feasible for the optimal decentralized problem and is the best that is possible for the decentralized problem. So, we can conclude that under the rev/leg incentive, in this problem $\overline{F}_{SCAP}^*(n) = F_{SCP}^*(n)$. ■

Proof. (Theorem 2)

We use the same network as in the proof of Theorem 1. We let $\epsilon \leq \frac{p}{n}$, and we modify the price parameters as follows:

$$p_{1i} = (i-1)p - (i-2)\epsilon, \forall i = 2, \dots, n;$$

$$p_{n1} = p;$$

$$p_{ij} = (j-i)\left(\frac{p}{n}\right) - (|LL_{ij}| - 1)\epsilon, \forall (ij) : |LL_{ij}| \geq 1 \text{ and } j > i \neq 1;$$

$$p_{i1} = (n - (i-1))\left(\frac{p}{n}\right) - (n-i)\epsilon, \forall i = 2, \dots, (n-1).$$

In this example,, we show that $\lim_{n \rightarrow \infty} \frac{\overline{F}_{SCAP}^*(n)}{F_{SCP}^*(n)} = \frac{(\frac{3n-2}{n})p}{np - (n-2)\epsilon} = 0$. Note that the priority lists of the agents for the rev/leg incentive are the same as those in the proof of Theorem 1 since $\frac{p_{ij}}{|LL_{ij}|}$ is decreasing. If we give an allocation of 1 to each agent, each agent will satisfy the demand for the O-D pair with the shortest travel.

With a similar argument as used for Remark 1, we may conclude that the optimal *SCAP* solution is $y_{i(i+1)}^* = 1, \forall i : 0 \leq i \leq (n-1)$ and $y_{n1}^* = 1$, and has objective value $\overline{F}_{SCAP}^*(n) = p_{12} + \sum_{i=2}^{n-1} p_{i(i+1)} + p_{n1} = (\frac{3n-2}{n})p$.

We next show that *SCP* has the optimal solution $y_{1n}^* = y_{n1}^* = 1, y_{ij}^* = 0$ for all other O-D pairs, with objective value $F_{SCP}^*(n) = p_{1n} + p_{n1} = np - (n-2)\epsilon$. This result follows from Remarks 3 and 4.

Remark 3 *In the optimal solution of SCP for this example, $y_{n1} = 1 \Rightarrow y_{1n} = 1$.*

Proof. Assume first that OD_{n1} is accepted, then we have to find the best way of using capacity from 1 to n. In total, there are 2^{n-2} combinations, but due to the structure of the price parameters we can consider a limited set. We first show that the best use of ship capacity between all ports $(i, j) : 1 < i < (n-1), i < j \leq n$ is to accept all of the 1-leg O-D pairs between i and j . We prove this by induction. There is only one choice of demand or path from port $(n-1)$ to

n . Since there is only one unit of capacity, the choice of which demand to accept is equivalent to finding the longest path where distance is determined by the price of the O-D pairs. First assume that the best capacity use from $(n-i) \geq 2$ to n is to accept the i 1-leg O-D pairs. i.e., $\{OD_{(n-i)(n-i+1)}, OD_{(n-i+1)(n-i+2)}, \dots, OD_{(n-1)n}\}$. We need to show that the best path from $n-i-1$ to n is to use the $(i+1)$ 1-leg O-D pairs, i.e., $\{OD_{(n-i-1)(n-i)}, OD_{(n-i)(n-i+1)}, \dots, OD_{(n-1)n}\}$. If we choose to go from $n-i-1$ to $n-i+j$, $\forall j: 0 \leq j \leq i$, then by the induction hypothesis, the best path to n from $n-i+j$ is to choose the $(i-j)$ 1-leg O-D pairs between those two ports. The path from $n-i-1$ directly to $n-i+j$ and from $n-i+j$ in the best way to n has a total revenue of $(j+1)\frac{p}{n} - j\epsilon + (i-j)\frac{p}{n} = (i+1)\frac{p}{n} - j\epsilon$, which is maximized at $j=0$. This means that the best choice from $n-i-1$ to n is to use all of the 1-leg O-D pairs between the two ports.

Now we find the best path from port 1 to port n . We consider a path that visits j . We have shown that the best choice from j to n is to use all of the 1-leg O-D pairs between the ports j and n . Then, the value of this path is $(j-1)p - (j-2)\epsilon + (n-j)\frac{p}{n}$, which is maximized at $j=n$. Therefore, $y_{n1} = 1 \Rightarrow y_{1n} = 1$. ■

Remark 4 Any solution of SCP with $y_{n1} = 0$ can be improved by modifying the solution so that $y_{n1} = 1$.

Proof. If $y_{n1} = 0$, then exactly one of OD_{j1} , $j = 2, \dots, n-1$ must have been accepted. Otherwise we can accept OD_{n1} without violating feasibility. Consider the path from j directly to 1, which has a value of $(n-j+1)\frac{p}{n} - (n-j)\epsilon$. We can modify this path such that it is composed of two partial paths: the first from j directly to n , and the second from n to 1. The modified path has a value of $(n-j)\frac{p}{n} + p$, which is greater than $(n-j+1)\frac{p}{n} - (n-j)\epsilon$; therefore the SCP solution improves. ■

Based on Remarks 3 and 4, we have that $\lim_{n \rightarrow \infty} \frac{\bar{F}_{SCP}^*(n)}{F_{SCP}^*(n)} = \frac{(\frac{3n-2}{n})p}{np - (n-2)\epsilon} = 0$. ■

Proof. (Remark 2) The result can be proved with a similar argument to that used for Remark 1. ■

References

- [1] P.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows, Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, 1993.

- [2] D. Bertsimas and I. Popescu. Revenue management in a dynamic network environment. *Transportation Science*, 37(3):257–277, 2003.
- [3] R.M. Burton and B. Obel. Mathematical contingency modelling for organizational design: taking stock. In R.M. Burton and B. Obel, editors, *Design models for hierarchical organizations: computation, information, and decentralization*. Kluwer Academic Publishers, Boston, 1995.
- [4] O. Caliskan Demirag. Demand management in decentralized logistics systems and supply chains. PhD Thesis, In preparation, Georgia Institute of Technology, 2007.
- [5] M. Christiansen, K. Fagerholt, and D. Ronen. Ship routing and scheduling: Status and perspectives. *Transportation Science*, 38(1):1–18, 2004.
- [6] C. Corbett and X. de Groot. A supplier’s optimal quantity discount policy under asymmetric information. *Management Science*, 46(3):444–450, 2000.
- [7] J. R. Correa, A. S. Schulz, and N. E. Stier Moses. Selfish routing in capacitated networks. *Mathematics of Operations Research*, 29(4):961–976, 2004.
- [8] M. Dror, P. Trudeau, and S.P. Ladany. Network models for seat allocation on flights. *Transportation Research Part B-Methodological*, 22(4):239–250, 1988.
- [9] M.R. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Freeman, 1979.
- [10] D.C. Gazis. Resource allocation in a large decentralized enterprise. *European Journal of Operational Research*, 30(3):339–343, 1987.
- [11] F. Glover, R. Glover, J. Lorenzo, and C. McMillan. The passenger-mix problem in the scheduled airlines. *Interfaces*, 12(3):73–80, 1982.
- [12] D.S. Hochbaum and P.A. Tucker. Minimax problems with bitonic matrices. *Networks*, 40(3):113–124, 2002.
- [13] ILOG Customer Support. Private communication, March 2005.

- [14] A.V. Iyer, L.B. Schwarz, and S.A. Zenios. A principal-agent model for product specification and production. *Management Science*, 51(1):106–119, 2005.
- [15] R. Johari and J.N. Tsitsiklis. Efficiency loss in a network resource allocation game. *Mathematics of Operations Research*, 29(3):407–435, 2004.
- [16] I. Karaesmen and G. van Ryzin. Coordinating overbooking and capacity control decisions on a network. Working Paper, University of Maryland and Columbia University, June 2004.
- [17] R.G. Kasilingam. Air cargo revenue management: Characteristics and complexities. *European Journal of Operational Research*, 96(1):36–44, 1997.
- [18] R.G. Kasilingam. An economic model for air cargo overbooking under stochastic capacity. *Computers and Industrial Engineering*, 32(1):221–226, 1997.
- [19] E. Koutsoupias and C.H. Papadimitriou. Worst-case equilibria. *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, 1563:404–413, 1999.
- [20] L.H. Lee, E.P. Chew, and M.S. Sim. A heuristic to solve a sea cargo revenue management problem. *OR Spectrum*, 29.
- [21] G.Y. Lin, E.R. Luby, and K.Y. Wang. New model for military operations. *OR/MS Today*, December 2004.
- [22] J.I. McGill and G.J. van Ryzin. Revenue management: Research overview and prospects. *Transportation Science*, 33(2):233–256, 1999.
- [23] Orient Overseas Container Line (OOCL). <http://www.oocl.com/eng/ourservices/serviceroutes/iat/>.
- [24] Orient Overseas Container Line (OOCL). <http://www.oocl.com/eng/ourservices/serviceroutes/tpt/>.
- [25] C.H. Papadimitriou. Algorithms, games, and the internet. *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 749–753, 2001.
- [26] G. Perakis. The “price of anarchy” when costs are non-separable and asymmetric. Working Paper, MIT, 2006.

- [27] J. Spengler. Vertical integration and anti-trust policy. *Journal of Political Economy*, 58:347–352, 1950.
- [28] J. Tirole. *The Theory of Industrial Organization*. The MIT Press, Cambridge, Massachusetts, 1988.
- [29] H.R. Varian. *Microeconomic Analysis*. W. W. Norton and Company, New York, 1992.

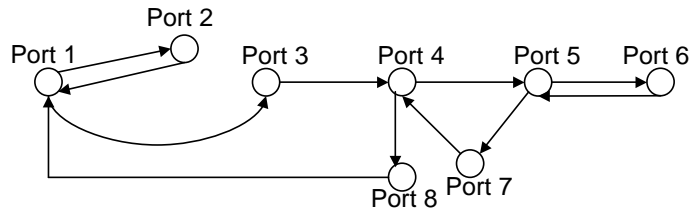


Figure 1: A network with a single route and revisitations.

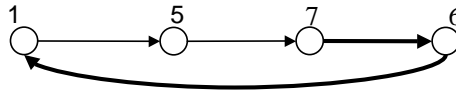
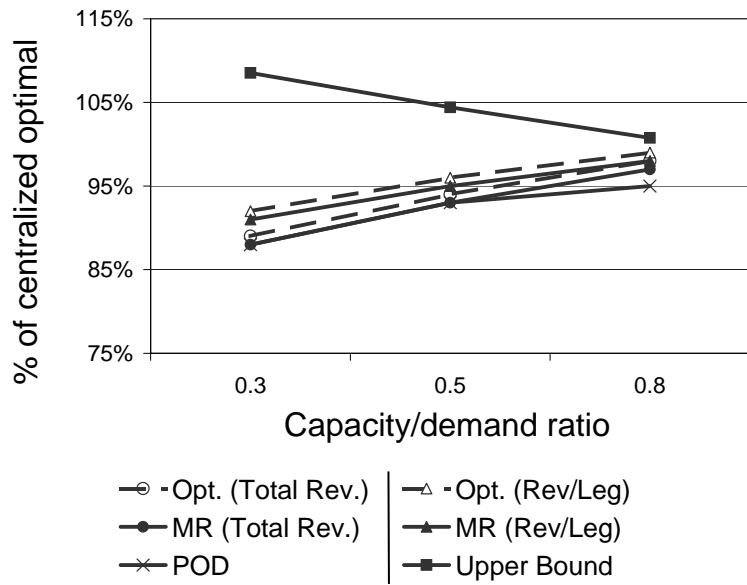
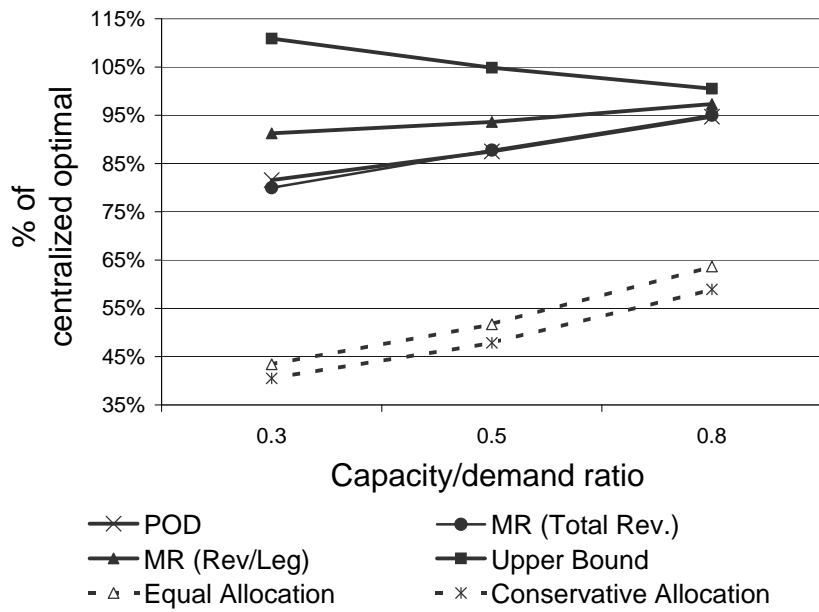


Figure 2: Example illustrating O-D replacement by agent at port 5. Legs $(7, 6)$ and $(6, 1)$ are totally occupied (represented by bold arcs with heavy lines), and the prices of O-D pairs 5-1 and 5-7 are such that $p_{5-1} > p_{5-7}$. Note that O-D pair 5-1 needs space on $(7, 6)$ and $(6, 1)$, but O-D pair 5-7 does not. Consequently, the *POD* heuristic does not allocate any capacity to the agent when considering O-D pair 5-1, but may attempt to allocate him capacity for O-D pair 5-7 in order to improve the current solution. However, the agent can use this capacity for O-D pair 5-1 since $p_{5-1} > p_{5-7}$. This violates capacity limits on legs $(7, 6)$ and $(6, 1)$. *POD* avoids future O-D replacements by agent 5 by not allocating him more space on this route.

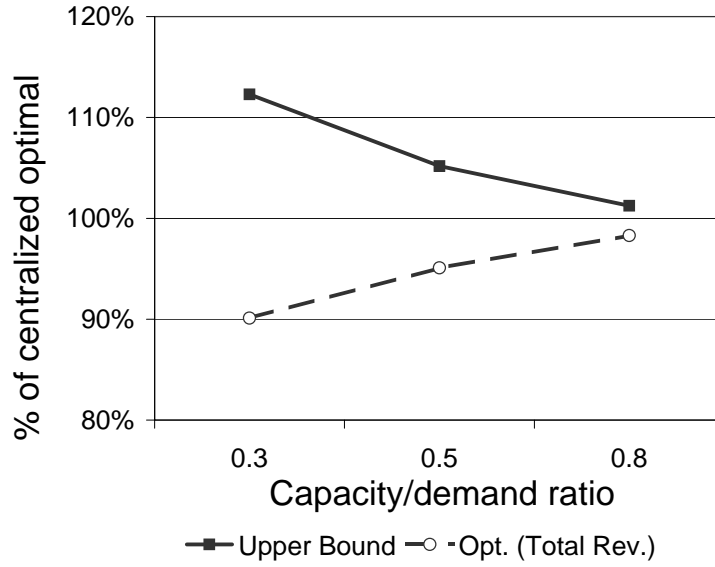


(a) #Routes=2, #Ports=6, Maxdemand=30

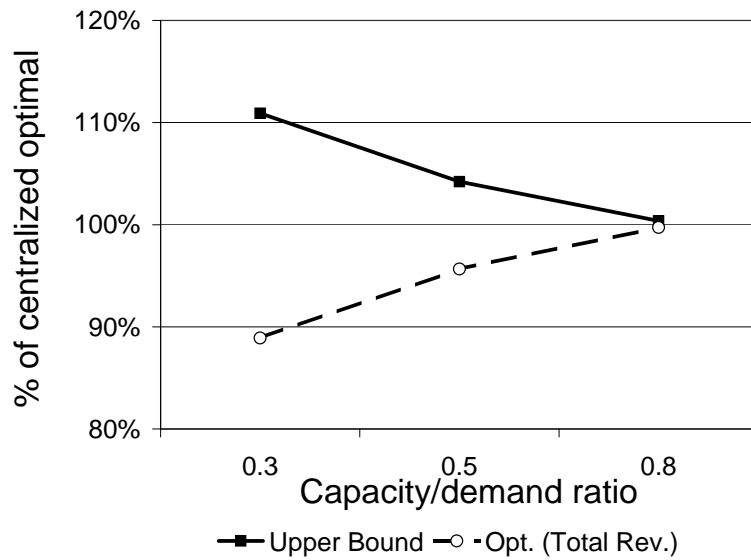


(b) #Routes=5, #Ports=10, Maxdemand=500

Figure 3: Decentralized system performance with increasing *Capacity/demand ratio*

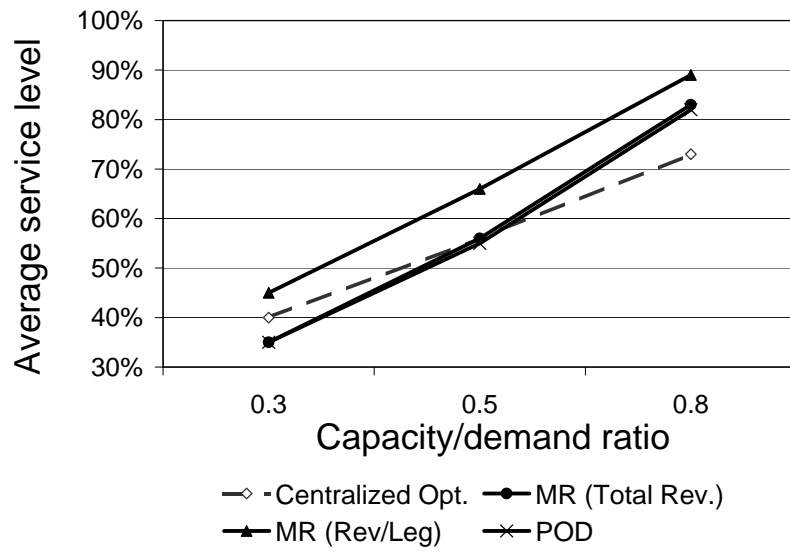


(a) #Routes=1, #Ports=6, Maxdemand=30

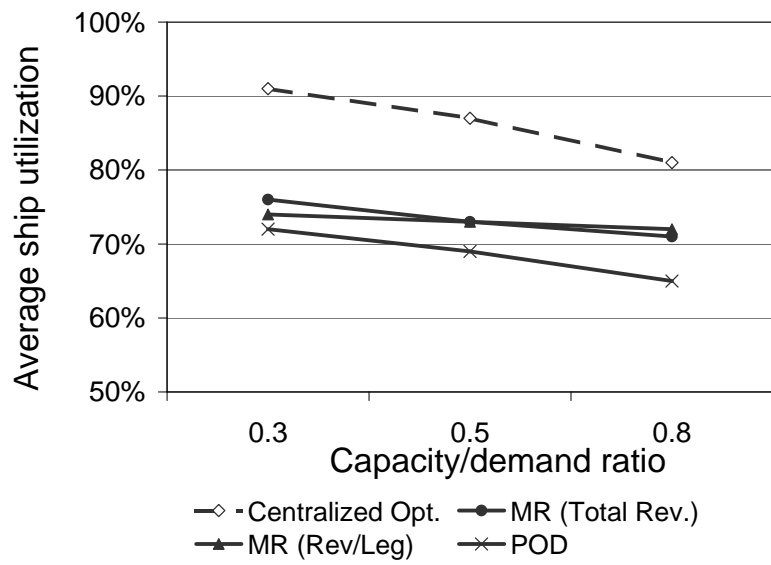


(b) #Routes=3, #Ports=6, Maxdemand=100

Figure 4: Upper bound and optimal *CAP* solution versus optimal *CP* solution



(a) #Routes=5, #Ports=10, Maxdemand=500



(b) #Routes=5, #Ports=10, Maxdemand=500

Figure 5: Average service level and ship utilization with increasing *Capacity/demand ratio*

Table 1: Main notation

P	: Set of ports or agents
R_p	: Set of routes that visit port p ($\cup_{p \in P} R_p = R$, where R denotes the set of all routes)
OD_p	: Set of O-D pairs that originate at port p ($\cup_{p \in P} OD_p = OD$, where OD denotes the set of all O-D pairs)
\overline{OD}_p	: Set of O-D pairs for port p ordered by non-increasing prices
\overline{OD}	: Set of all O-D pairs ordered by non-increasing prices
op_k, dp_k	: Origin, destination ports of O-D pair k
RR_k	: Set of routes such that a directed path exists from op_k to dp_k
L_r	: Set of legs on route r
LL_k^r	: Set of legs O-D pair k traverses on route r (denoted as LL_k if $ R = 1$, where $ R $ is the cardinality of the set R)
S_l^r	: Set of O-D pairs on route r using leg $l \in L_r$ (denoted as S_l if $ R = 1$)
cap_r	: Ship capacity on route r (denoted as cap if $ R = 1$)
p_k	: Price per unit demand shipped for O-D pair k
d_k	: Total amount of demand for O-D pair k
nv_p^r	: Number of times port p is visited on route r
a_p^r	: Capacity allocated to agent at port p on route r (denoted as a_p if $ R = 1$)
\vec{a}_p	: The vector of allocated capacities on all the routes of agent at port p , where $\vec{a}_p = (a_p^1, a_p^2, \dots, a_p^{ R_p })$
\vec{a}	: The vector of allocated capacities of all agents, where $\vec{a} = \{\vec{a}_1 \vec{a}_2 \dots \vec{a}_{ P }\}$
e_p^r	: Unit vector for agent p with number of entries = $ R_p $, entry of 1 for route r and 0 otherwise
$y_k^r(\vec{a}_p)$: Number of demands accepted for O-D pair k on route r when the agent at port p receives \vec{a}_p
y_k	: Number of demands accepted for O-D pair k (defined for the special case of a single-route)
z_k^p	: Binary decision variables that equal one if the entire demand of O-D pair $k \in OD_p$ is satisfied and zero otherwise
$Z_p(\vec{a}_p)$: The objective function value of AP corresponding to the agent at port p
\mathbb{Z}^+	: The set of positive integers

Table 2: Runtimes of the algorithms in CPU seconds

(a) $\#Routes = 2, \#Ports = 6, Maxdemand = 30$

	<i>Capacity/demand ratio</i>		
	0.3	0.5	0.8
<i>CAP</i> Optimal (total revenue)	7.04	50.59	509.49
<i>CAP</i> Optimal (rev/leg)	7.07	51.92	540.47
<i>MR</i> (total revenue)*	1.23	2.04	3.19
<i>MR</i> (rev/leg)	1.39	2.25	3.28
<i>POD</i>	0.12	0.14	0.17

(b) $\#Routes = 5, \#Ports = 10, Maxdemand = 500$

	<i>Capacity/demand ratio</i>		
	0.3	0.5	0.8
<i>MR</i> (total revenue)	204.18	326.44	489.75
<i>MR</i> (rev/leg)	309.38	434.14	588.87
<i>POD</i>	0.65	0.81	0.95

* *stepsize* = 1 for the *MR* heuristic

Table 3: Marginal revenue heuristic

<p>1. Initialize the allocation vectors at zero ($a_p^r = 0, \forall p \in P, \forall r \in R_p$).</p> <p>2. For all $p \in P, r \in R_p$ do</p> <p style="padding-left: 40px;">solve $AP_p(\vec{a}_p + \text{stepsize} \cdot e_p^r)$. Record the solutions in <i>solutions array</i>¹.</p> <p>3. Do begin</p> <p style="padding-left: 40px;"> $(p_{max}, r_{max}) = \underset{p \in P, r \in R_p}{\operatorname{argmax}} \{ Z_p^*(\vec{a}_p + \text{stepsize} \cdot e_p^r) - Z_p^*(\vec{a}_p) : \sum_{q \in P \setminus \{p\}} \sum_{\{k \in OD_q \cap S_l^r\}} y_k^{r'*}(\vec{a}_q) + \sum_{\{k \in OD_p \cap S_l^r\}} y_k^{r'*}(\vec{a}_p + \text{stepsize} \cdot e_p^r) \leq \text{cap}_{r'}, \forall r' \in R, l \in L_{r'} \}$ </p> <p style="padding-left: 40px;"> $\text{change} = Z_{p_{max}}^*(\vec{a}_{p_{max}} + \text{stepsize} \cdot e_{p_{max}}^{r_{max}}) - Z_{p_{max}}^*(\vec{a}_{p_{max}})$ </p> <p style="padding-left: 40px;">if $\text{change} > 0$ then begin</p> <p style="padding-left: 80px;"> $\vec{a}_{p_{max}} = \vec{a}_{p_{max}} + \text{stepsize} \cdot e_{p_{max}}^{r_{max}}$ </p> <p style="padding-left: 80px;">for all $r \in R_{p_{max}}$ do begin</p> <p style="padding-left: 120px;">if ($a_{p_{max}}^r + \text{stepsize} \leq \text{cap}_r \cdot \text{nv}_{p_{max}}^{r_{max}}$) then</p> <p style="padding-left: 160px;">solve $AP_{p_{max}}(\vec{a}_{p_{max}} + \text{stepsize} \cdot e_{p_{max}}^{r_{max}})$. Update <i>solutions array</i> for p_{max}.</p> <p style="padding-left: 120px;">end for</p> <p style="padding-left: 80px;">end if</p> <p style="padding-left: 40px;">end while ($\text{change} > 0$)</p>
--

Table 4: Priority OD (*POD*) based heuristic

<p>1. Initialization: $a_p^r = 0, \forall p \in P, \forall r \in R_p; r_l = 0, \forall l \in L; \bar{d}_k = d_k, \forall k \in OD; w_k^r = \text{cap}_r, \forall r \in R, \forall k \in RR_k$</p> <p>2. Starting with the first O-D pair in \overline{OD}, for all $k \in \overline{OD}$ do begin</p> <p style="padding-left: 40px;">for all $r \in RR_k$ do begin</p> <p style="padding-left: 80px;">if ($\bar{d}_k > 0$) then begin</p> <p style="padding-left: 120px;"> $w_k^r = \min_{l \in LL_k^r} (\text{cap}_r - r_l)$ </p> <p style="padding-left: 120px;">solve $AP_{op_k}(\vec{a}_{op_k} + \min(w_k^r, \bar{d}_k) \cdot e_{op_k}^r)$</p> <p style="padding-left: 120px;">if ($\sum_{\{m \in OD_{op_k} \cap S_l^r\}} y_m^{r*}(\vec{a}_{op_k} + \min(w_k^r, \bar{d}_k) \cdot e_{op_k}^r) + \sum_{q \in P \setminus \{op_k\}} \sum_{\{m \in OD_q \cap S_l^r\}} y_m^{r*}(\vec{a}_q) \leq \text{cap}_r, \forall r \in R, l \in L_r$) begin</p> <p style="padding-left: 160px;"> $\vec{a}_{op_k} = \vec{a}_{op_k} + \min(w_k^r, \bar{d}_k) \cdot e_{op_k}^r$ </p> <p style="padding-left: 160px;"> $\bar{d}_k = \bar{d}_k - \min(w_k^r, \bar{d}_k)$ </p> <p style="padding-left: 160px;"> $r_l = r_l + \sum_{\{m \in OD_{op_k} \cap S_l^r\}} y_m^{r*}(\vec{a}_{op_k}), \forall r \in R, l \in L_r$ </p> <p style="padding-left: 120px;">end if</p> <p style="padding-left: 80px;">end if</p> <p style="padding-left: 40px;">end for</p> <p style="padding-left: 40px;">end for.</p>
