Using Decomposition to Solve Facility Location/Fleet Management Problems

by

Mohammad Mehdi Fazel Zarandi

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

Abstract

Using Decomposition to Solve Facility Location/Fleet Management Problems

Mohammad Mehdi Fazel Zarandi
Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto
2010

The central thesis of this dissertation is that logic-based Benders decomposition can be effective in solving deterministic and stochastic facility location and fleet management problems. We demonstrate this thesis by developing logic-based Benders decomposition models for a facility location/fleet management problem from the literature and for a stochastic extension of the problem. We experimentally show the effectiveness of logic-based Benders in solving such problems. To our knowledge, this is the first work on solving deterministic and stochastic facility location/fleet management problems using logic-based Benders decomposition.

Acknowledgements

There are several people that I would like to thank for helping me to write this dissertation.

I would first like to thanks my supervisor, Professor Chris Beck, for his insight, support, and guidance for the past two years.

I would also like to thank my co-supervisor, Professor Oded Berman, for all his support, expertise, and comments, which contributed to a significant portion of the thesis.

I would also like to thank Professor Timothy Chan for his cogent comments.

Thanks to all the people in my lab. In particular, I need to say thank you to Daria Terekhov for the many helpful discussions and your help with the editing, and to Lei Duan, for answering all my questions about Linux, ILOG and the cluster.

I would also like to thanks my parents and my small sister, Mahya, for years of support and love.

Last and most, thanks to my sister, Maryam, for always inspiring and supporting me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The central thesis of this dissertation is that logic-based Benders decomposition can be effective in solving deterministic and stochastic facility location and fleet management problems. In particular, in this dissertation:

- We present a logic-based Benders decomposition approach for solving facility location and fleet management optimization problems.
- We develop and compare two logic-based Benders decomposition models for a two-stage stochastic facility location and fleet management optimization problem.

To our knowledge, this is the first work which attempts to solve deterministic and stochastic facility location and fleet management problems using logic-based Benders decomposition. Thus, this dissertation expands the scope of problems that can be solved by using logic-based Benders decomposition and also broadens the range of approaches used to solve facility location/fleet management and stochastic programming problems.

1.1. Motivations

Logistics is concerned with planning, managing, and controlling the flow and storage of raw materials, finished goods and information throughout the supply chain for the purpose of conforming to customer requirements. Two important problems in logistics are decisions about facility location and fleet management. Since these strategic decisions are related, many researchers have worked on problems that combine them. However, combining these two sets of decisions results in very challenging optimization problems. The success of hybrid approaches in solving a variety of optimization problems [12, 57, 96] provides the motivation to apply such techniques to deterministic and stochastic logistics problems. In more detail, the motivations for the work presented in this dissertation are:

1. **The Application of Hybrid Techniques to Facility Location/Fleet Management Problems**

   Combined facility location and fleet management problems are very complex and difficult to solve. Due to their complexity and importance, such problems require the use

of sophisticated mathematical approaches. Hybrid methods such as logic-based Benders decomposition, which combine methods from the fields of artificial intelligence and operations research, might be useful for solving such problems. In this dissertation, we apply a logic-based Benders decomposition approach, which combines integer programming and constraint programming, to a facility location and fleet management problem. To our knowledge, this is a first attempt to solve such problems with logic-based Benders decomposition.

2. **The Application of Hybrid Techniques to Two-Stage Stochastic Facility Location/Fleet Management Problems**

   Facility location and fleet management decisions are very costly, and their impact spans a long time horizon. During the time when design decisions are in effect, any of the parameters of the problem may fluctuate. Thus, it is important to take these uncertain parameters into account when we are modeling such problems. However, by considering uncertainty, the models become significantly more difficult to solve. A goal of this dissertation is to address a two-stage stochastic facility location/fleet management problem with logic-based Benders decomposition in order to investigate the use of such hybrid techniques for two-stage stochastic optimization problems.

3. **Two-Level versus Three-Level logic-based Benders Decomposition**

   It is not necessarily true that the more we decompose a problem, the faster we can find the optimal solution. In this dissertation we compare the performance of a two-level logic-based Benders decomposition to that of a three-level logic-based Benders decomposition. A goal of this dissertation is to investigate the behavior of the decomposition model as the number of decomposition levels increase.

1.2. Outline

The outline of the thesis is as follows:

Chapter 2 provides background information for the dissertation and looks at the literature relevant to our research. Since the main contribution of this dissertation is the application of logic-based Benders decomposition to facility location and fleet management problems, we present a review of deterministic and stochastic facility location and fleet management problems, as well as logic-based Benders decomposition.

In Chapter 3, we consider a facility location/fleet management problem that requires deciding the location of a set of facilities, the allocation of customers to those facilities under facility capacity constraints, and the allocation of the customers to trucks at those facilities under per truck travel-distance constraints. In order to solve the problem, a hybrid approach that combines integer programming and constraint programming using logic-based Benders decomposition is proposed. Computational experiments demonstrate that the Benders model is able to find and prove optimal solutions up to three orders-of-magnitude faster than an existing

integer programming approach, while also finding better feasible solutions in less time when compared to an existing tabu search algorithm.

In Chapter 4, we address a stochastic facility location and vehicle assignment problem which consists of simultaneously locating a set of facilities, determining the vehicle fleet size at each facility, and allocating customers to facilities and vehicles in the presence of random travel times. Non-deterministic travel times can arise, for example, due to daily traffic patterns or weather-related disturbances. We consider the different travel-time conditions as different scenarios with known probabilities. We present a stochastic programming model based on a bounded penalty approach [66] in which the expected recourse cost cannot exceed a given threshold. This problem is an extension of the problem considered in Chapter 3. To solve the problem, an integer programming, a two-level and a three-level logic-based Benders decomposition models are proposed. Experimental results demonstrate the strong performance of the two Benders models.

Chapter 5 concludes this dissertation by re-stating its main contributions and suggesting some areas for future work.

1.3. Summary of Contributions

The contributions of the thesis are as follows:

- We develop a logic-based Benders decomposition approach for a facility location/fleet management problem. To our knowledge, this is a first attempt to solve such problems with logic-based Benders decomposition. We show that not only is the logic-based Benders decomposition model significantly better than an integer programming model in terms of finding and proving an optimal solution, it can also be used for finding good feasible solutions in cases where the problem is too large to find the optimal solution.

- We propose a new stochastic facility location/fleet management problem and develop a stochastic programming model for it. Furthermore, we provide an integer programming, and two logic-based Benders decomposition models for solving it. The results demonstrate that logic-based Benders decomposition can be effective in solving stochastic programming problems.

- We compare the performance of a two-level and a three-level logic-based Benders decomposition approaches proposed for the stochastic facility location/fleet management problem. The results show that it is not always true that the more we decompose the problem the faster we can find and prove the optimal solution.

Chapter 2

Literature Review

As the main contribution of this dissertation is the application of logic-based Benders decomposition to facility location and fleet management problems, in this chapter we present a review of deterministic and stochastic facility location and fleet management problems, as well as logic-based Benders decomposition. In the first section, facility location and fleet management problems are introduced. Then, a review of facility location and fleet management problems under uncertainty is given. Finally, an overview of Benders decomposition and logic-based Benders decomposition is presented.

2.1. Facility Location and Fleet Management Problems

Facility location and fleet management are important strategic decisions made by organizations [36]. Facility location is concerned with modeling and solving problems about the optimal placement of facilities in order to optimize an objective such as facility opening cost or long-term transportation cost [40, 26]. Fleet management addresses the purchase, placement, and operation of the organization's fleet of vehicles [19, 70]. Since these strategic decisions are related, many researchers have worked on problems that combine them [67, 77, 80].

Figure 2.1 demonstrates the interrelation between facility location and fleet management decisions. As can be seen from the diagram, customers can either receive service at the facility, as in the case of a supermarket, or receive service at their door, which happens for example, with postal service or fire trucks. In the former case, the decision maker seeks to find the best locations from among those available to locate the facilities so as to optimize an overall objective. In this case, since the service is given at the site, no additional decisions regarding the purchase, placement, and operation of the fleet of vehicles needs to made. The $p-$median problem (Section 2.1.1.1), the uncapacitated facility location problem (Section 2.1.1.2), and the single source capacitated location problem (Section 2.1.1.3) are examples of such problems. In the latter case, since customers are being served at their doors, additional decisions concerning the management of the fleet which provide the service must be made. In such cases, the vehicles can give service via full return trips from the facility, or visit multiple customers in a single tour. A comprehensive review of combined facility location and fleet management problems is given in Sections 2.1.2 and 2.1.3.

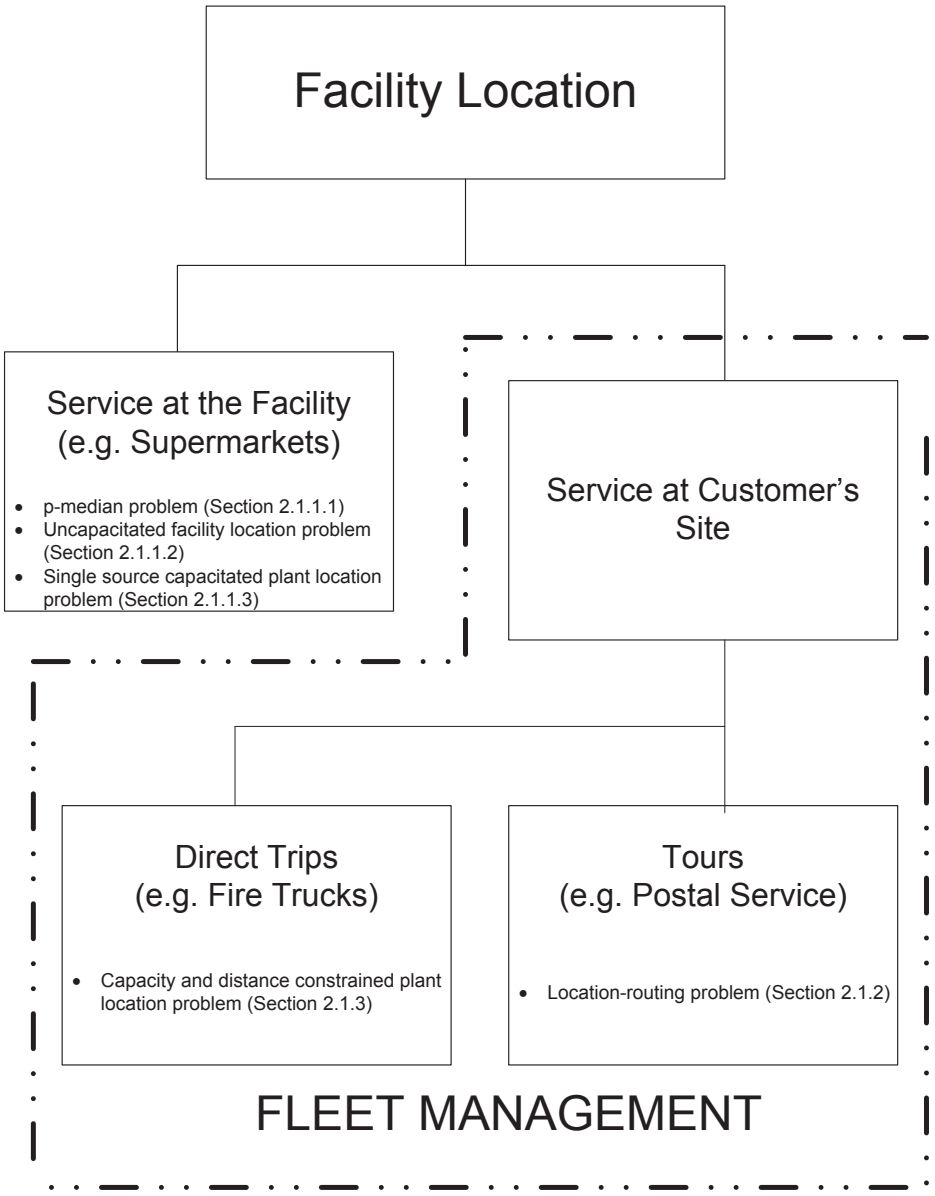

Figure 2.1: A schematic representation on how facility location and fleet management decision are interrelated.

2.1.1 Facility Location Problems

In classical facility location problems, the objective is to select the best locations for the facilities from among those available, and to assign customers to open facilities in an optimal manner. Most of these problems are known to be NP-hard [81]. The main difficulty in solving these problems arises from the non-convexity of the objective function and the existence of multiple local minima [33]. In this section, we will give an introduction to three primary problems in facility location: the $p-$median problem, the uncapacitated plant location problem, and

the single source capacitated plant location problem.

2.1.1.1 The $p-$median Problem

The simplest of facility location problems is the $p-$median problem introduced by Hakimi [49], but studied by many other researchers [61, 11, 23, 50]. In the $p-$median problem, p facilities are to be selected, such that the sum of demand-weighted distances between the customers, and the facility located nearest is minimized. In these problems, it is assumed that facilities have equal setup cost, and that every facility has enough capacity to serve all of the demand assigned to it.

The p-median problem is usually formulated as an integer program (IP). In order to formulate the problem the following notation is used:

$i \in I$: index of demand node,
$j \in J$: index of facility node,
d_i: demand at node i,
c_{ij}: distance between nodes i and j,
p: number of facilities to be located.

The decision variables of the model are:

$$p_j = \begin{cases} 1, & \text{if facility } j \text{ is open} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if demand node } i \text{ is served by facility } j \\ 0, & \text{otherwise} \end{cases}$$

By using the above notation, the IP formulation of the $p-$median problem can be stated as:

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

$$\text{s.t.} \quad \sum_{j \in J} p_j = p \qquad (1)$$

$$\sum_{j \in J} x_{ij} = 1 \qquad i \in I \qquad (2)$$

$$x_{ij} \leq p_j \qquad i \in I, i \in J \qquad (3)$$

$$x_{ij}, p_j, \in \{0, 1\} \qquad i \in I, j \in J \qquad (4)$$

The objective function minimizes the total demand-weighted distances. Constraint (1) indicates that exactly p facilities are to be located. Constraint (2) ensures that every demand is assigned to exactly one facility, while constraint (3) allows assignment only to sites at which facilities have been located. Constraint (4) is the integrality constraint.

While for fixed values of p (i.e. we want locate exactly p facilities) the $p-$median problem can be solved in polynomial time, for variable values of p the problem is NP-hard [44]. Thus, for problems with variable p values, meta-heuristics and approximation algorithms such as: tabu search [88, 89], simulated annealing [87, 71], variable neighborhood search [50, 32], and genetic algorithm [5] have been the predominant solution techniques. Exact algorithm such as branch-and-bound [61, 43] and branch-and-price [23] have also been proposed for such problems. For more details, see the surveys by Daskin [35] and Reese [85].

2.1.1.2 The Uncapacitated Facility Location Problem

The most important extension of the $p-$median problem is the uncapacitated facility location problem (UFLP). As in the $p-$median problem, the UFLP involves locating facilities to minimize the total distances, however, the problems differ in two ways. First, in the $p-$median problem it was assumed that all candidate sites are equivalent in terms of the setup cost for locating a new facility. In the UFLP, however, each potential facility may have its own setup cost, therefore, the objective function needs to be extended to take into account the fixed facility location costs. Second, unlike the $p-$median problem, the UFLP does not have a constraint on the maximum number of facilities, thus, the number of facilities to be established becomes a decision variable.

Let f_j be the fixed cost of locating a facility at potential site j, the IP formulation of the UFLP is:

$$\text{minimize} \quad \sum_{j \in J} f_j p_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$\text{s.t.} \quad (2), (3), (4)$$

The objective function minimizes the sum of the fixed facility location costs and the total travel costs (distances) for demand to be served. Constraints (2), (3), (4) are the same as before.

To solve the UFLP, exact algorithms such as branch-and-bound [62, 63], dual-based approach [42], and primal-dual approach [64] have been proposed. Since the UFLP is NP-hard [28], exact algorithms may not be effective in solving practical size problems. Thus, in order to solve practical size UFLPs, researchers have proposed heuristic and meta-heuristic approaches

such as tabu search [76, 95], simulated annealing [8], genetic algorithm [60], and neural networks [98].

2.1.1.3 The Single Source Capacitated Plant Location Problem

One of the most important extensions of the UFLP is the single source capacitated plant location problem (SSCPLP). In the SSCPLP, specific values are considered for the maximum demand that can be supplied from each potential site. Thus, the closest-assignment property is no longer valid. The overall goal is to choose a subset of the facilities to open and assign each customer to one of the chosen plants, such that the total cost is minimized and plant capacities are not exceeded. Figure 2.2 shows an example of the SSCPLP with four potential facilities sites, A, B, C, D, and five customers, 1, 2, 3, 4, 5. Customers 1, 2, 5 are served by facility A, while customers 3 and 4 are served by facility B.

Let b_j be the capacity of a facility at potential site j, the SSCPLP can be formulated as follows:

$$\text{minimize} \quad \sum_{j \in J} f_j p_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$\text{s.t.} \quad (2), (3), (4)$$

$$\sum_{i \in I} d_i x_{ij} \leq b_j p_j \quad j \in J \quad (5)$$

Constraints (2)-(4) are the same as before. The plant capacity limits are defined by constraint (5).

Researchers have developed both exact algorithms and heuristics to solve the SSCPLP. Holmberg et al. [53] propose a branch-and-bound method based on a Lagrangian heuristic for solving SSCPLP. They combine a Lagrangian dual approach which generates lower bounds with a strong primal approach which produces feasible solutions and upper bounds. The branch-and-bound procedure uses information obtained from the Lagrangian relaxation to speed up the bounding process. The performance of the approach is compared with the state-of-the-art commercial IP optimization software, CPLEX, based on 71 problem instances with sizes ranging from 20×50 (i.e. 20 possible facilities and 50 customers) to 30×200. The results show that the proposed method is faster than CPLEX in 90% of the cases, and can find the solution up to three orders-of-magnitude faster. Diaz and Fernández [39] develop an exact algorithm for the SSCPLP in which a column generation procedure is incorporated within a branch-and-price framework. To investigate the efficiency of their approach, they compared their approach with the Lagrangian heuristic approach proposed in Hindi and Pienkosz [52]. The results indicate that the column generation procedure finds better bounds, while also finding the optimal solution with smaller CPU times for the majority of the instances.

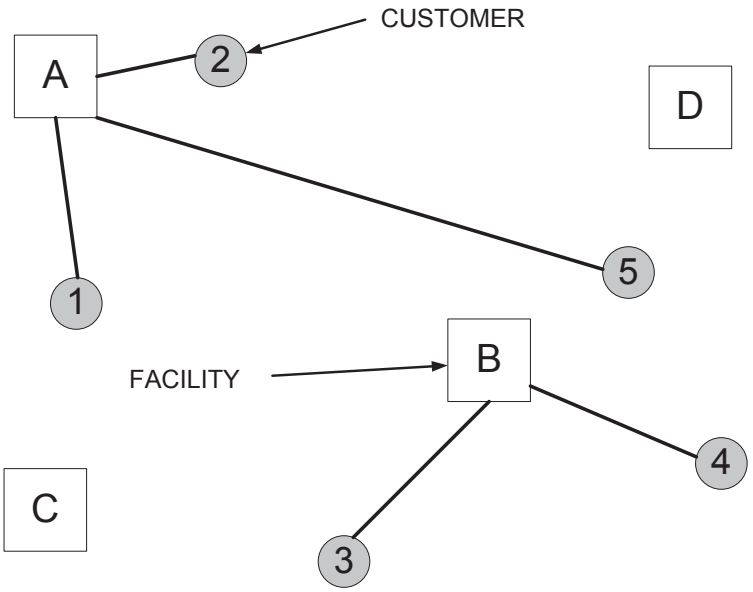

Figure 2.2: A schematic representation of the single source capacitated plant location problem.

Due to the intractability of SSCPLP, a great deal of research has focused on developing heuristics such as genetic algorithms [31], Lagrangian heuristic [9, 10, 52, 30], and very large-scale neighborhood search [1]. Delmaire et al. [38] also develop several heuristics for SSCPLP, each based on one or more of the following approaches: evolutionary algorithms, greedy randomized adaptive search (GRASP), simulated annealing and tabu search. In the very large-scale neighborhood search (VLSNS) algorithm proposed by Ahuja et al. [1], the neighborhood structures are induced by customer multi-exchanges and by facility moves. In order to evaluate the performance of the algorithm, they applied it to two varieties of benchmark instances available in the literature, and to a real-world instance. The first set of benchmark instances where taken from [38]. The results for these instances indicate that VLSNS can solve problems of size 30×200 in less than a minute with a mean gap to the best known solution of only 0.028%, while CPLEX needs around two and half hours to solve the same problems. The results for the second set of instances, taken from [10], demonstrate that the VLSNS is able to find solutions for problems of size 100×1000, with a mean gap of less than 0.04%. Finally, the VLSNS was tested using real data from an Italian cookie maker with 23 sites and 104 customers. The results show that the VLSNS computed a very good approximations in less than 50 seconds with a gap of 0.048% from the CPLEX solution (CPLEX was terminated after 2 hours).

Correia and Captivo [30] develop a Lagrangian heuristic to address a SSCPLP with several possible capacity levels for the facility that can be opened at each potential location. They use a two-phase heuristic which consists of a constructive phase and an improvement phase to compute upper bounds for the Lagrangian heuristic. The results indicate that this heuristic method finds solutions with a mean optimality gap of less than 3% (when the optimal solution is known), with mean execution times of 400 seconds for problem sizes ranging from 10×100

to 500×1000.

2.1.2 Location-Routing Problems

In many situations, the application requires that some sort of service is provided at the customer sites. For example, in the postal industry, the mail or packages need to be delivered to the customer. Such situations require making additional decisions concerning the management of the fleet which provides the service, giving rise to location-routing problems (LRP). Figure 2.3 shows an example of LRP with four potential facilities sites, A, B, C, D, and five customers, 1, 2, 3, 4, 5. Customers 1, 2 are served by the fleet of facility A via the route indicated by the arrows, while customers 3, 4, 5 are served by the fleet of facility B. The LRP aims at determining the location of facilities and the client-assignment decisions while taking into account the routing requirements; however, the location and routing decisions are often solved separately, sacrificing optimality guarantees. Salhi and Rand [90] investigated LRP using such a two-stage process. In the first stage, they ignore routing when locating depots and allocating customers. The second stage consists of making routing decisions based on the solution found in stage one. The second phase is solved based on the first phase solution, without feedback.

Many scholars have presented models and methods to integrate and solve the two levels of decisions simultaneously. Since LRPs merge two NP-hard problems, most of these methods have focused on heuristics: simulated annealing [100], tabu search [97, 79, 3], and Lagrangian heuristics [82, 2]. Exact methods such as cutting planes [69], branch-and-bound [66], and branch-and-cut [46, 65] have also been proposed for solving LRPs. Laporte [67], Mina et al. [77], and Nagy and Salhi [80] present extensive surveys on LRPs.

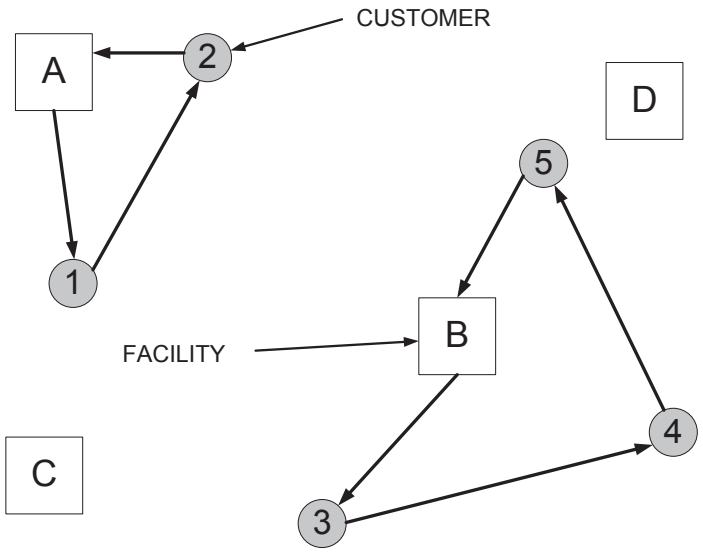

Figure 2.3: a schematic representation of the location-routing problem.

2.1.3 The Capacity and Distance Constrained Plant Location Problem

Given the difficulty of LRPs, Albareda-Sambola et al. [4] introduced the capacity and distance constrained plant location problem (CDCPLP), which captures some of the intricacies of routing decisions in location problems while avoiding some of the sources of complexity of the classical LRP. They assume that instead of requiring multi-customer routes to be found, vehicles serve customers by full return trips from the facility. A vehicle can serve multiple customers if its total travel distance is less than a given maximum. Albareda-Sambola et al. [4] propose an IP and a three-level nested tabu search to solve the problem. In the latter approach, the first level selects an appropriate set of plants to open using three neighborhoods based on opening a plant, closing a plant, and swapping an open plant with a closed one. The second level focuses on determining a good assignment of customers to the set of plants opened in level one. The neighborhoods are based on reassigning a customer to another open plant, swapping two customers, and transferring a subset of customers to another plant. The third and final level is the assignment of customers to vehicles based on the decisions made in the previous levels. The neighborhoods for this level are based on reassigning a customer to another truck, splitting a route into two, and merging two routes. Search then returns (i.e., as in a nested-loop) to the client assignments and eventually back to the facility openings. Computational results showed strong performance for the tabu search: it was able to find close-to-optimal solutions within a few minutes of CPU time. A more detailed description of the problem and the solution methods is presented in Chapter 3.

Although the main motivation for the definition of the CDCPLP was to simplify a very challenging location-routing problem [4], it is also the case that some real problems use full truck load logistics [20, 91, 47, 6]. For example, in the forest industry a full truck transports the logs from the forest to the mill [48].

It should be noted that the focus of this dissertation is the application of logic-based Benders decomposition (Section 2.3.2) to the CDCPLP and a stochastic extension of it.

2.2. Facility Location and Fleet Management Problems Under Uncertainty

In all the problems mentioned in the previous section, it was assumed that all important aspects of the problem are known with certainty or are controllable by the organization. In practice, however, such assumptions are unrealistic. For example, the demand of customers as well as travel times may not be certain, and are affected by factors such as seasonal demand patterns, weather conditions, and daily traffic patterns. In this section, we discuss how these uncertain parameters have been incorporated into facility location and fleet management problems.

2.2.1 Stochastic Location Problems

Facility location decisions are very costly, and their impacts are long term. During the time when design decisions are in effect, any of the parameters of the problem, such as the location of customers, the level of customer demand, and the travel cost, may fluctuate. Given that these parameters are subject to change, researchers have proposed stochastic location problems

(SLPs). In SLPs, location decisions must be taken before the actual value of the uncertain parameters are known. In many SLP models the objective is to determine facility locations such that the expected cost is minimized or the expected profit is maximized [27, 74, 37]. In other models, the objective is to maximize the probability that the solution meets or exceeds some quality threshold [21, 17, 41].

In the location literature, the three main approaches used to incorporate uncertainty in location decisions are:

1. stochastic programming,
2. queueing theory,
3. scenario planning.

The first approach used to incorporate uncertainty is stochastic programming. Stochastic programming models are extensions of linear and non-linear programming models, where the uncertainty of the data are explicitly taken into account [34]. Two frequently used stochastic programming approaches are chance-constrained programming [24], and two-stage stochastic programming with recourse [34].

Chance-Constrained Programming In chance-constrained programming, the constraints are required to hold with at least a specified level of probability, but not necessarily with probability one. Assume we want to find the values of the decision variables x_j, which minimize the objective function $\sum_j c_j x_j$, subject to linear constraints $\sum_j a_{ij} x_j \leq b_i, i \in I$. Assume further that the parameters a_{ij}, b_i are random variables. The chance-constrained programming model is [84],

$$\text{minimize} \quad \sum_{j \in J} c_j x_j$$

$$\text{s.t.} \quad P\{\textstyle\sum_j a_{ij} x_j \leq b_i\} \geq \alpha_i \quad i \in I$$

where α_i are specific probabilities. This means that the constraint can be violated for at most $100 \times (1 - a_i)$ of the time.

Carbone [21] used chance-constrained programming in order to incorporate uncertainty into a $p-$median problem, where the demand is stochastic and has a normal distribution. In order to solve the problem, the author transforms it into a nonlinear deterministic equivalent, and solved it using nonlinear programming.

Two-Stage Stochastic Programming with Recourse In two-stage stochastic programming with recourse models, in the first stage, before the stochastic parameters are known, a decision is made. A recourse decision is then made in the second stage which compensates for any bad effects that might have been experienced as a result of the first-stage decision.

Let x be a vector of our first stage decision variables, and ξ a vector of random variables. Two-stage stochastic programming with recourse can be modeled as [18]:

$$\text{minimize} \quad c^T x + E_\xi Q(x, \xi)$$

$$\text{s.t.} \quad Ax = b \quad i \in I$$

$$x \geq 0$$

where $Q(x, \xi)$, referred to as the second stage problem, determines the recourse action associated with the solution x and the outcome ξ of the random parameter. Its expected value, $E_\xi Q(x, \xi)$, is called the recourse function. The objective is choose some initial decision, x, in order to minimizes current costs, $c^T x$, plus the expected value of future recourse actions, $E_\xi Q(x, \xi)$.

Louveaux [74] studies how the p-median problem (Section 2.1.1.1), and the UFLP (Section 2.1.1.2) are transformed into a two-stage stochastic program with recourse when there is uncertainty in demands, production costs and transportation costs. Since both the production and transportation costs and the demands become stochastic, it is no longer possible to define the size of a facility as the sum of the demands it serves. Also for some realizations of the production and transportation costs, it might be more appropriate not to serve all demands. Thus, the demands to be served and the size of the facilities become a part of the decision process. In Louveaux's recourse model, the first-stage decisions determine the location and the size of the facilities to be built, while the second-stage decisions determine the allocation of the available production to the most profitable demand points. The author studies the relations between the two models, but solution methods are not discussed. A common approach in solving two-stage stochastic programming with recourse models is the L-shaped method [68], which is based on classical Benders decomposition (see Section 2.3.1). In this method, the problem is partitioned into a deterministic master problem and a stochastic sub-problem, and the Benders cuts (Section 2.3.1) are generated from the LP dual solution of the sub-problem.

The second approach used to take uncertainty into account is to incorporate the stochastic parameters within a queueing framework. Since it is out of the scope of this thesis to look at queueing-location problems, the reader is refereed to the surveys by Owen and Daskin [81] and Snyder [94] for thorough discussions of these models.

The third approach is scenario planning. Scenario planning is a method in which decision makers capture uncertainty by specifying a number of possible future states. Each scenario represents possible values for the parameters that may fluctuate over the planning horizon. In the location literature, scenario planning has been incorporated in location models based on

three approaches [81]: optimizing the expected performance over all scenarios, optimizing the worst-case performance, and minimizing the expected or worst-case regret across all scenarios. Serra and Marianov [92] use scenario planning to locate fire stations in Barcelona, where the demand and the travel times vary over the course of the day. Scenarios are used to capture different demand patterns and/or travel times (a scenario for a each specific period of the day). Over these scenarios, facilities are located with the objectives of minimizing the maximum average travel time and minimizing the maximum unserved demand. Carson and Batta [22] use a similar scenario planning approach for an ambulance location problem. The authors use four scenarios to capture demand conditions during 24 hours. To minimize system-wide average response times, they formulate a model for determining the optimal ambulance position under each scenario. By comparing the resultant optimal strategy with historical data, the model predicted a 30% reduction in average response time.

In Chapter 4, we extend the CDCPLP by considering random travel-times, in which scenarios with known probabilities are used to capture different travel-time conditions. We present a two-stage stochastic programming with recourse for the stochastic problem.

2.2.2 Stochastic Location-Routing Problems

The majority of the LRP literature has focused on deterministic models [80]. In practice, however, the number, demand, and location of customers as well as travel times of vehicles may not be known a priori and consequently should be treated as random variables. Given these random variables, researchers have proposed stochastic location-routing problems (SLRPs). Both exact methods [66, 75, 7] and heuristics [78, 72, 3] have been used to solve SLRPs. Berman et al. [16] and Nagy and Salhi [80] provide comprehensive surveys on SLRPs.

Laporte et al. [66] considered an SLRP where both depot locations and a vehicle routes must be planned before the exact demand is known. Since planning is done beforehand, a route may exceed the vehicle capacity. In this situation, the vehicle must return to the depot prematurely to unload, then must resume service to the remaining customers. The cost of this additional journey can be viewed as a penalty cost. They study two variants of the problem: (a) minimize location and routing costs (first stage costs) so that the expected penalty of any route does not exceed a fraction of its planned cost; (b) minimize first stage cost so that the probability of a route exceeding its capacity does not exceed a given threshold. The problems are modeled as integer linear programs and solved using a branch-and-bound algorithm. The results show that the algorithm is able solve problems with up to 30 customers and 3 potential sites to optimality within reasonable CPU time.

Albareda-Sambola et al. [3] also look at a problem where both facility locations and routes are designed before demand is known. In their model, after the demands are known, if the total demand is such that the vehicle capacity is exceeded, some of the customers on the route are omitted. Unserved customers result in a penalty. They use a two-stage stochastic programming with recourse approach to model their problem. In a first stage, the set of plants and a family of routes are determined; in a second stage, once the demands are known, a recourse action is applied to adapt these routes to the actual set of customers to visit. They propose a two-phase heuristic to solve the SLRP. An initial solution is generated in the construction phase, which selects the set of plants to open, determines the allocation of customers to open plants, and designs a route for each open plant. The solution is then iteratively improved in a local search

phase. The results show that the improvement phase can improve the quality of the constructive phase by up to 30%.

2.3. Benders Decomposition

Since the main contribution of this dissertation is the application of logic-based Benders decomposition to facility location/fleet management problems, in this section we present a review of classical and logic-based Benders decompositions.

2.3.1 Classical Benders Decomposition

Classical Benders Decomposition [13, 45, 86] is a mathematical programming approach for solving hard optimization problems. In classical Benders decomposition, a problem is partitioned into a mixed-integer master problem and linear sub-problems. The solution procedure iterates between solving the master problem, a relaxation of the original model, and solving each sub-problem until the optimal solution is found. When the sub-problem is infeasible based on the master solution, its LP dual is used to generate a Benders cut. The cut, when added to the master problem, eliminates at least the current master solution.

Benders decomposition can be applied to problems of the form [54]:

$$\begin{aligned} \min \quad & z = c^T x + f(y) \\ \text{s.t.} \quad & Ax + g(y) \geq b \\ & x \geq 0, y \in Y \subseteq \mathbb{R}^q \end{aligned}$$

Given a fixed value $y = \overline{y}$, a lower bound in the objective function can be found by solving the linear programming sub-problem:

$$\begin{aligned} \min \quad & c^T x + f(\overline{y}) \\ \text{s.t.} \quad & Ax \geq b - g(\overline{y}) \\ & x \geq 0 \end{aligned}$$

However, in Benders decomposition instead of solving the sub-problem, one solves its dual:

$$\begin{aligned} \min \quad & u(b - g(\overline{y})) + f(\overline{y}) \\ \text{s.t.} \quad & A^T u \leq c \\ & u \geq 0 \end{aligned}$$

Cuts can be generated by using the dual solution:

$$z \geq u^*(b - g(y)) + f(y)$$

The cuts are then added to the master problem with the form:

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & z \geq u^k(b - g(y)) + f(y) \quad k = 1, ..., K \\ & y \in Y \end{aligned}$$

where $u^1, ..., u^K$ are the solutions of the first K sub-problem dual solutions.

2.3.2 Logic-Based Benders Decomposition

Classical Benders decomposition has been generalized to logic-based Benders decomposition by removing the restriction that the master problem be mixed-integer and sub-problems be linear [54, 58]. Figure 2.4 gives a schematic representation of logic-based Benders decomposition. As can be seen, logic-based Benders decomposition is based on the idea of defining a master problem and a set of sub-problems. The master problem is solved to optimality, inducing sub-problems based on the master problem solution. Each sub-problem is then solved. Solving the sub-problem may result in one or more Benders cuts which are obtained by solving the inference dual of the sub-problem [54]. The inference dual infers from the constraints and the current master problem solution the tightest possible bound on the master objective function value. If there are no such cuts to be derived, the master solution combined with the solutions to the sub-problems is a globally optimal solution. If there are cuts, these are added to the master problem and it is re-solved, inducing another set of sub-problems and potentially more Benders cuts.

In logic-based Benders decomposition the usual challenges are:

- It is crucial to get a tight relaxation of the sub-problems represented in the master problem in order to reduce the search space.
- The cuts should be as tight as possible, ruling out the maximum possible number of master solutions.

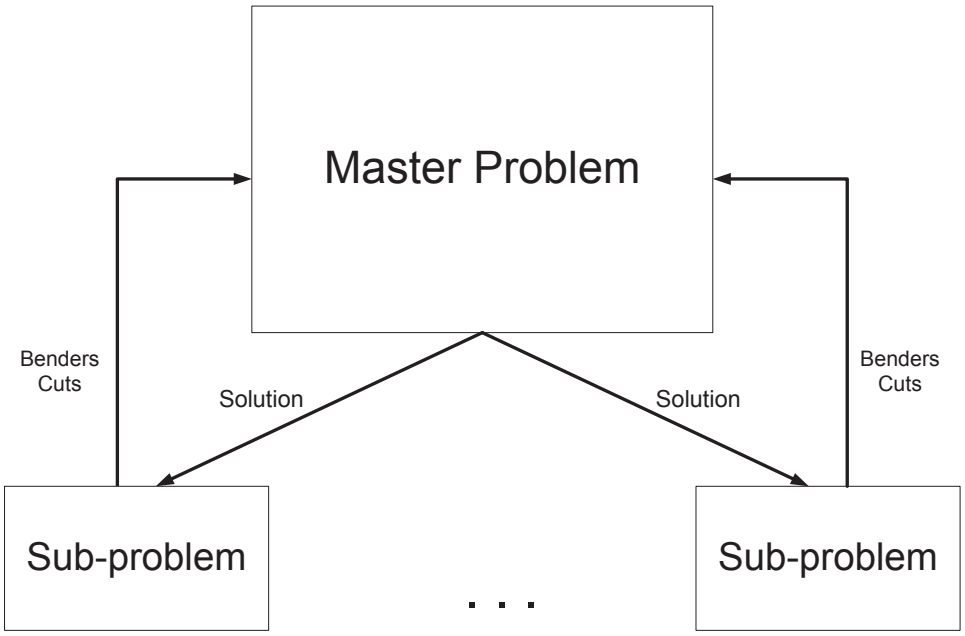

Figure 2.4: A schematic representation of logic-based Benders decomposition

Example In this example, we demonstrate how logic-based Benders decomposition is applied to a planning and scheduling problem [54]. Assume that a set of jobs, J, each with individual release dates, R_j, and due dates, S_j, must be scheduled on a set of unary capacity resources, I. Each job, $j \in \{1, .., m\}$, can be assigned to any resource, $i \in \{1, .., n\}$, where it consumes processing time, p_{ij}, and has an associated cost, f_{ij}. The goal is to assign the jobs to resources in the most cost-efficient manner. The assignments must be feasible with respect to the resource unary capacity constraint.

Hooker [54] developed a logic-based Benders decomposition approach for the above problem which decomposes it into a job assignment master problem, and a set of single-machine scheduling sub-problems. He used integer programming for the master problem and constraint programming for the sub-problems.

The job assignment master problem can be defined as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{ij} f_{ij} y_{ij} \\
\text{s.t.} \quad & \sum_{i} y_{ij} = 1, && \text{all } j && (6)\\
& \sum_{j} p_{ij} y_{ij} \leq \max_{j}\{S_j\} - \min_{j}\{R_j\}, && \text{all } i && (7)\\
& \sum_{j \in J_{hi}} (1 - y_{ij}) \geq 1, && i \in I_h, h = 1, \ldots H-1 && (8)
\end{aligned}$$

where y_{ij} is a binary variable which indicates whether job j is assigned to resource i, J_{hi} is the set of jobs assigned to resource i in iteration h and that led to an infeasibility in the sub-

problem, and I_h is the set of resources for which the sub-problem is infeasible in iteration h.

The objective function minimizes the cost of assigning jobs to resources. Constraint (6) ensures that all jobs are assigned to exactly one resource. Constraint (7) is the sub-problem relaxation. This constraint ensures that the sum of the durations of the jobs assigned to a given resource is less than the time between the minimum release date and maximum due date. Constraint (8) is the Benders cuts. The cuts ensures that the same set of jobs (or a superset of them) that led to an infeasibility in the sub-problem is not re-assigned to the same resource.

Given the set of jobs assigned to each resource, $J_i = \{j \mid y_{ij} = 1\}$, the goal of each subproblem is to assign start times to jobs, t_j, such that a feasible schedule is found. Therefore, the sub-problems are single machine scheduling problems with release dates and due dates. The constraint programming formulation of the sub-problem for resource i is:

$$\text{cumulative}(t_j, p_{ij}, [1, ..., 1], 1) \quad (9)$$

$$t_j \geq R_j \quad (10)$$

$$t_j + p_{ij} \leq S_j \quad (11)$$

The cumulative global constraint (9) represents a single-machine scheduling problem to assign values to all start times, t_j, taking into account the durations of each job, p_{ij}, and the resource capacity. Since this is a unary capacity problem, both the rate of resource consumption of each job and the capacity of the resource are 1.

Logic-based Benders decomposition has been applied to a variety of problems. Jain and Grossmann [59] applied it to minimum-cost planning and scheduling problems in which the sub-problems are one-machine disjunctive scheduling problems. They achieved two to three orders of magnitude reduction in CPU time compared to a pure mixed integer linear programming (MILP) model and a pure CP model. In related work, Hooker [55, 56] solves minimum-cost, minimum-makespan and minimum-tardiness planning and scheduling problems where tasks are allocated to machines using MILP and scheduled using CP. They obtain a speedup of a few orders of magnitude when compared to the state-of-the-art pure MILP and pure CP models. Logic-based Benders decomposition has also been used to solve call center scheduling [15], steel production scheduling [51], minimal dispatching of automated guided vehicles [29], multicore architecture optimization [14], traffic diversion [99], transportation network design problems [83], and queue design and control [96].

2.4. Conclusions

In this chapter, we presented a review of the literature on deterministic and stochastic facility location and fleet management problems. In addition, we reviewed the literature on Benders and logic-based Benders decomposition.

In the next chapter we apply for the first time, a logic-based Benders decomposition approach to the CDCPLP discussed in Section 2.1.3. The logic-based Benders approach decomposes the problem into a facility location master problem and a set of fleet management

sub-problems. The approach uses integer programming to model the master problem and constraint programming to model the sub-problems. In Chapter 4, we apply logic-based Benders decomposition to a stochastic location-allocation problem.

Chapter 3

Solving the Capacity and Distance Constrained Plant Location Problem with Logic-Based Benders Decomposition

As indicated in the preceding chapter, combined facility location and fleet management problems are well-studied and very challenging problems in the area of logistics. Given the difficulty of these problems, Albareda-Sambola et al. [4] recently introduced the capacity and distance constrained plant location problem (CDCPLP) which simplifies the routing aspect of the problem. In this chapter, we first provide a description of the CDCPLP and discuss the different methods that have been used to solve it. Then we describe our logic-based Benders decomposition approach for the problem. In our Benders model, we combine integer programming and constraint programming, taking advantage of the strengths of both.

This chapter is organized as follows: in Section 3.1 we present a detailed description of the problem along with an integer programming formulation. A tabu search proposed in Albareda-Sambola et al. [4] for the problem is presented in Section 3.2. The details of our logic-based Benders decomposition approach are described in Section 3.3. Computational results are presented in Section 3.4. Section 3.5 concludes this chapter.

3.1. Problem Definition

The capacity and distance constrained plant location problem (CDCPLP) considers a set of capacitated facilities, each housing a number of identical vehicles for serving clients. Clients are served by full return trips from one facility. The same vehicle can be used to serve several clients as long as its distance traveled does not exceed a given maximum. The goal is to select the set of facilities to open, determine the number of vehicles required at each opened site, and assign clients to facilities and vehicles in the most cost-efficient manner. The assignments must be feasible with respect to the facilities' capacities and the maximum distance a vehicle can travel. Recall from Section 2.1.3, that while the main motivation for the definition of the CDCPLP was to simplify a very challenging location-routing problem [4], it is also the case that some real problems use full truck load logistics [20, 91, 47, 6].

Formally, let J be the set of potential facilities (or sites) and I be the set of clients. Each

facility, $j \in J$, is associated with a fixed opening cost, f_j, and a capacity, b_j (e.g., a measure of the volume of material that a facility can process). Clients are served by open facilities with a homogeneous set of vehicles. Each vehicle has a fixed utilization cost, u, and a maximum total driving distance, l. Serving client i from site j generates a driving distance, t_{ij}, for the vehicle performing the service, consumes a quantity, d_i, of the capacity of the site, and has an associated cost, c_{ij}. The available vehicles are indexed in set K with parameter $\overline{k} \geq |K|$ being the maximum number of vehicles at any site. Albareda-Sambola et al. [4] formulate an IP model of the problem, where the decision variables are:

$$p_j = \begin{cases} 1, & \text{if facility } j \text{ is open} \\ 0, & \text{otherwise} \end{cases}$$

$$z_{jk} = \begin{cases} 1, & \text{if a } k\text{th vehicle is assigned to site } j \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ijk} = \begin{cases} 1, & \text{if client } i \text{ is served by the } k\text{th vehicle of site } j \\ 0, & \text{otherwise} \end{cases}$$

The IP formulation is as follows:

$$\text{minimize} \quad \sum_{j \in J} f_j p_j + u \sum_{j \in J} \sum_{k \in K} z_{jk} + \sum_{i \in I} \sum_{j \in J} c_{ij} \sum_{k \in k} x_{ijk}$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{j \in J} \sum_{k \in K} x_{ijk} = 1 && i \in I && (1) \\
& \sum_{i \in I} t_{ij} x_{ijk} \leq l \cdot z_{jk} && j \in J, k \in K && (2) \\
& \sum_{i \in I} \sum_{k \in K} d_i x_{ijk} \leq b_j p_j && j \in J && (3) \\
& z_{jk} \leq p_j && j \in J, k \in K && (4) \\
& x_{ijk} \leq z_{jk} && i \in I, j \in J, k \in K && (5) \\
& z_{jk} \leq z_{jk-1} && j \in J, k \in K \backslash \{1\} && (6) \\
& x_{ijk}, p_j, z_{jk} \in \{0, 1\} && i \in I, j \in J, k \in K && (7)
\end{aligned}$$

The objective function minimizes the sum of the costs of opening the facilities, using the vehicles, and the travel. Constraint (1) ensures that each client is served by exactly one facility. The driving distance limits are defined by constraint (2). This constraint limits the sum of the distances for all clients assigned to a facility, while also putting an upper bound on the distance of a single client from the facility to which it is assigned. Constraint (3) limits the demand allocated to facility j. Constraints (4) and (5) ensure that a client cannot be served from a site that has not been opened nor by a vehicle that has not been allocated. Constraint (6) states that at a site, vehicle k can only be used if vehicle $k-1$ is used.

3.2. Tabu Search

Albareda-Sambola et al. proposed a three-level nested tabu search to solve the CDCPLP. Figure 3.1 is a schematic representation of the three-level nested tabu search.

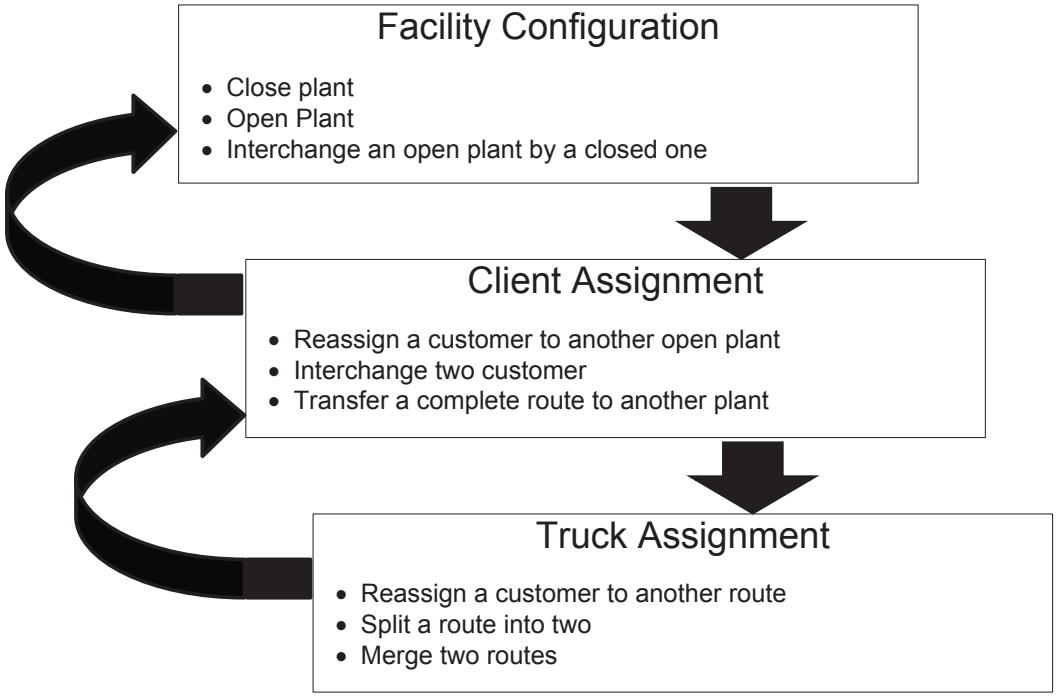

Figure 3.1: A schematic representation of the three-level nested tabu search proposed by Albarada-Sambola [4].

In this approach, we start with an initial solution, which might not be feasible with respect to the distance and the capacity constraints. The initial solution is found using a two-phase constructive heuristic. In the first phase, the set of facilities to be opened is chosen based on the total demand and the facility opening costs. In the second phase, once the set of open facilities is fixed, customers are assigned one at a time to the open facility with sufficient residual capacity that has the smallest $c_{ij} \times t_{ij}$. If no feasible assignment exists, then the customer is assigned to the facility with the largest residual capacity, and to the vehicle with the largest actual load among those that could serve it without violating their total driving distance constraint. If no such vehicle exists, then a new vehicle is allocated to the facility and the customer is assigned to it. If the facility already has $\overline{k}$ vehicles, the customer is assigned to the vehicle with the smallest load.

The initial solution is then sent to the first level, the facility configuration level. This level tries to improve the initial solution by using three neighborhoods based on closing a facility (N_{ep}), opening a new facility (N_{op}) and heuristically allocate some customers to it, and swapping an open facility with a closed one (N_{oc}). The second level, the client assignment level, focuses on improving the customer assignment decisions based on the solution sent by the first level. The neighborhoods are based on reassigning a customer to another open facility (N_{cp}), swapping two customers (N_{ic}), and transferring a complete route to another facility (N_{tr}). The third and final level tries to improve the vehicle assignment decisions based on the solutions of the previous levels. The neighborhoods for this level are based on reassigning a customer to another route (N_{cr}), splitting a route into two (N_{sr}), and merging two routes (N_{mr}). Search then returns (i.e., as in a nested-loop) to the client assignments and eventually back to the facility openings.

At each level, the best known solution is updated, and throughout the search, infeasible solutions with respect to capacity and distance constraints are allowed, but are penalized. A new solution is considered better than its incumbent if: (a) when the two solution are both feasible and the new one has a smaller cost, (b) when the incumbent solution is infeasible and the new solution is feasible, or (c) in the case where both are infeasible, both distance and the capacity violations are smaller in the new solution. Algorithm 1 shows the structure of the tabu search.

3.3. A Logic-Based Benders Decomposition Approach

Following the logic-based Benders decomposition approach (see Section 2.3.2), we decompose the CDCPLP into a location-allocation master problem (LAMP) and a set of truck assignment subproblems (TASPs). The LAMP is concerned with choosing the open facilities, allocating clients to these sites, and deciding on the number of trucks at each site. It is similar to the single source plant location problem (see Section 2.1.1.3) with an aggregated distance constraint involving the number of vehicles. The TASPs assign clients to specific vehicles and can be modeled as a set of independent bin-packing problems, one for each open facility. Clients are

Algorithm 1 Tabu Search heuristic search structure (taken from [4])

```
Initialize solution x {might violate length and/or capacity constraints}
x* = x {best known solution}
while not stop criterion 1 do
  if x is facility feasible then
    Explore non-tabu(1) moves in N_op, N_ep and N_oc
  else
    Explore non-tabu(1) moves in N_op and N_oc
  end if
  Perform selected move
  Update tabu list 1 and x*, if applicable
  while not stop criterion 2 do
    Explore non-tabu(2) moves in N_cp, N_ic and N_tr
    Perform the chosen move. Let j ∈ J be the affected facility
    Update tabu list 2, and x*, if applicable
    while not stop criterion 3 do
      if x is vehicle-feasible at facility j then
        if merge is not tabu(3) then
          Explore N_mr
        else
          Explore non-tabu(3) moves in N_cr
        end if
      else
        if split is non-tabu(3) then
          Explore N_sr
        else
          Explore non-tabe(3) moves in N_cr
        end if
        Perform chosen move
        Update tabu list 3 and x*, if applicable
      end if
      Update TS-3 parameters
    end while
    Update TS-2 parameters
  end while
  Update TS-1 parameters
end while
```

allocated to the trucks so that the maximum-distance constraint on each truck is satisfied. We use integer programming (IP) for the master problem and constraint programming (CP) for the subproblems. Figure 3.2 is a schematic representation of our logic-based Benders decomposition approach for the CDCPLP.

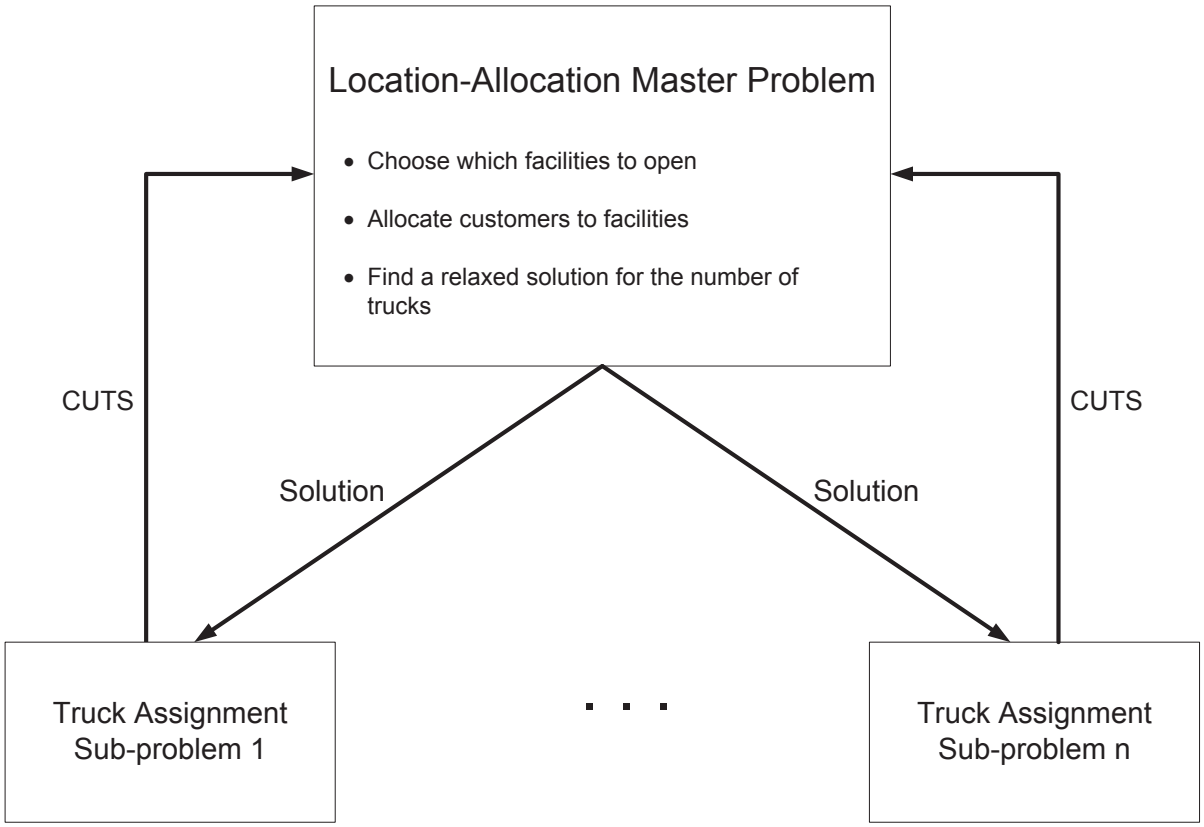

Figure 3.2: A Logic-Based Benders Decomposition Approach for the CDCPLP

3.3.1 The Location-Allocation Master Problem

An IP formulation of LAMP is as follows:

$$\text{minimize} \quad \sum_{j \in J} f_j p_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + u \sum_{j \in J} numVeh_j$$

$$\text{s.t.} \quad \sum_{j \in J} x_{ij} = 1 \qquad i \in I \qquad (8)$$

$$\sum_{i \in I} t_{ij} x_{ij} \leq l \cdot \overline{k} \qquad j \in J \qquad (9)$$

$$t_{ij} x_{ij} \leq l \qquad i \in I, j \in J \qquad (10)$$

$$\sum_{i \in I} d_i x_{ij} \leq b_j p_j \qquad j \in J \qquad (11)$$

$$numVeh_j \geq \left\lceil \frac{\sum_{i \in I} t_{ij} x_{ij}}{l} \right\rceil \qquad j \in J \qquad (12)$$

$$cuts \qquad (13)$$

$$x_{ij} \leq p_j \qquad i \in I, j \in J \qquad (14)$$

$$x_{ij}, p_j \in \{0,1\}, numVeh_j \in \{0,..,\overline{k}\} \qquad i \in I, j \in J \qquad (15)$$

where p_j is as defined in Section 3.1, x_{ij} is a binary variable which indicates whether client i is assigned to site j, and $numVeh_j$ is an integer variable indicating the number of vehicles assigned to facility j.

Constraint (8) ensures that all clients are served by exactly one facility. In (9), $\overline{k}$ is the upper bound on the number of trucks at a facility; equivalent to $|K|$ in the original IP formulation. So, constraint (9) states the upper bound on the sum of the distances for all clients assigned to a facility, while (10) is an upper bound on the distance of a single client from the facility to which it is assigned. Constraint (11) limits the demand assigned to facility j. Constraint (12) is the relaxation of the subproblem which defines the minimum number of vehicles assigned to each site. $cuts$ are constraints that are added to the master problem each time one of the subproblems does not have a feasible solution. A detailed description of the cuts is presented in Section 3.3.3.

3.3.2 The Truck Assignment Subproblem

Given the set of clients allocated (I_j) and the number of vehicles assigned to an open facility ($numVeh_j$), the goal of the TASP is to assign clients to the vehicles of the site such that the vehicle travel-distance constraints are satisfied. The TASP for a facility can be modeled as a bin-packing problem.

A CP formulation of the TASP is as follows:

$$\min \quad numVehBinPacking_j$$

$$\text{s.t.} \quad pack(load, truck, dist) \tag{16}$$

$$numVeh_j \leq numVehBinPacking_j \leq numVehFFD_j \tag{17}$$

where $load$ is an array of variables such that $load[k] \in \{0, ..., l\}$ is the total distance assigned to vehicle $k \in \{1, ..., numVehBinPacking_j\}$, $truck$ is an array of decision variables, one for each client $i \in I_j$, such that $truck[i] \in \{1, ..., numVeh_j\}$ is the index of the truck assigned to client i, and $dist$ is the vector of distances between site j and client $i \in I_j$. The pack global constraint (16) maintains the load of the vehicles given the distances and assignments of clients to vehicles [93]. The upper and lower bound on the number of vehicles is represented by constraint (17).

Figure 3.3 and Algorithm 2 show how we solve the subproblems in practice. We first use the first-fit decreasing (FFD) heuristic (line 3) to find $numVehFFD_j$, a heuristic solution to the subproblem. If this value is equal to the value assigned by the LAMP solution, $numVeh_j$, then the subproblem has been solved. Otherwise, in line 5 we solve a series of satisfaction problems using the CP formulation, setting $numVehBinPacking_j$ to each value in the interval $[numVeh_j..numVehFFD_j - 1]$ in increasing order. Informally, we try to pack the customers in $numVeh_j$ trucks; if a feasible solution is found we are done, otherwise we increase the number of trucks by one and again try to pack the customers into the trucks. We continue adding trucks and packing customers until a feasible solution is found or until $numVehBinPacking_j = numVehFFD_j$. The FFD heuristic is used because it is faster than CP and often finds a solution equal to $numVeh_j$. When FFD does not find the same value as the master problem, it provides a good upper bound for the CP model.

Algorithm 2 Algorithm for solving the TASP

SolveTASP():

```
1 cuts = ∅
2 for each facility do
3 |   numVehFFD = runFFD()
4 |   if numVehFFD > numVeh_j then
5 |   |   numVehBinPacking = runCPBinPacking()
6 |   |   if numVehBinPacking > numVeh_j then
7 |   |   |   cuts ← cuts + new cut
8 return cuts
```

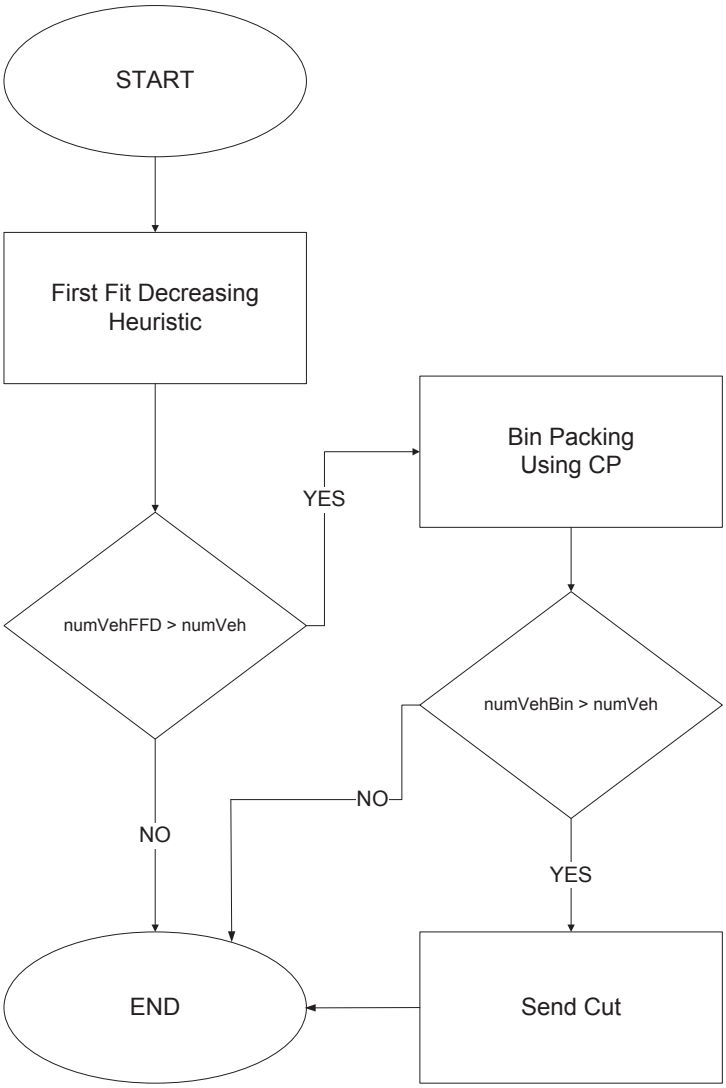

Figure 3.3: The algorithmic flowchart of how the TASPs are solved in practice.

3.3.3 Benders Cuts

The generation of Benders cuts is an essential part of logic-based Benders decomposition [58]. Benders cuts are constraints that are added to the master problem each time one of the sub-problems is not able to find a feasible solution. Cuts ensure that all future solutions to the master problem are closer to being feasible solutions to the global problem. In our approach, the Benders cuts that are added to the LAMP remove the current optimal solution of the LAMP because it does not result in a feasible solution to one or more of the TASPs.

Assume that in one particular iteration, the solution to the LAMP assigns a set, Q, of clients to a facility, j, and specifies the number of trucks needed at j, $numVeh_j^Q$. Assume further that the TASP solution indicates that $numVeh_{jh}^*$ vehicles are needed to serve the Q customers, and

$numVeh^*_{jh} > numVeh^Q_j$. The cut that arises as a result specifies that if Q or a superset of Q is again assigned to facility j, then the number of trucks must be greater than or equal to $numVeh^*_{jh}$.

Formally, the cuts after iteration h are:

$$numVeh_j \geq numVeh^*_{jh} - \sum_{i \in I_{jh}} (1 - x_{ij}), \quad j \in J_h,$$

where $I_{jh} = \{i \mid x^h_{ij} = 1\}$ is the set of clients assigned to facility j in iteration h, J_h is the set of sites for which the TASP is infeasible in iteration h, and $numVeh^*_{jh}$ is the number of vehicles needed at site j to serve the clients that were assigned. The summation term in this constraint is the maximal decrease in the number of trucks needed, given that some of the clients may be reassigned to other facilities: the largest possible reduction in the number of trucks in reassigning one client is one. The form of this cut is directly inspired by the Benders cut for scheduling with makespan minimization formulated by Hooker [57].

Chu and Xia [25] define a $valid$ Benders cut as a logical expression that has two properties:

$Property$ 1: the cut must exclude the current master problem solution if it is not globally feasible,

$Property$ 2: the cut must not remove any globally feasible solutions.

They show that property 1 guarantees finite convergence if the master problem variables have finite domains and property 2 guarantees optimality since the cut never removes globally feasible solutions.

Theorem 1. The proposed Benders cut is a valid cut. Thus, our logic-based Benders approach will converge to optimality in a finite number of steps.

Proof. In order to prove the validity of our proposed cut we need to show that both properties 1 and 2 are satisfied.

We first show property 1. Recall that I_{jh} is the set of clients assigned to facility j in iteration h, and $numVeh^*_{jh}$ is the minimum number of vehicles needed at facility j as found by TASP j in iteration h. We define $numVeh_{jh}$ to be the number vehicles assigned to facility j by iteration h of the LAMP model. If the LAMP solution is infeasible in the TASP, then:

$$numVeh_{jh} < numVeh^*_{jh} \quad (18)$$

which will result in the cut:

$$numVeh_j \geq numVeh^*_{jh} - \textstyle\sum_{i \in I_{jh}} (1 - x_{ij}) \quad (19)$$

If the same set of customers is again assigned to facility j, $\sum_{i \in I_{jh}} (1 - x_{ij}) = 0$. Then from (18) and (19),

$$numVeh_j > numVeh_{jh} \quad (20)$$

Therefore property 1 is satisfied: the same assignment of clients must result in a larger number of vehicles assigned in the master problem and conversely the same number of vehicles assigned in the master problem must result in a different set of clients. In (20), we have shown that the cut excludes the current master problem solution from all subsequent master problem solutions.

We now show property 2, that is, that our proposed cut does not remove any globally feasible solutions. Let S be a globally feasible solution found in iteration $s > h$. We present a proof by contradiction and so assume that S does not satisfy the cut, therefore:

$$numVeh_{js} < numVeh^*_{jh} - \sum_{i \in I_{jh}} (1 - x_{ij}) \quad (21)$$

Let I_{js} be the set of clients assigned to facility j in iteration s. We define three sets:

$\theta_1 = \{I_{jh} \backslash I_{js}\}$: customers in I_{jh} not in I_{js},

$\theta_2 = \{I_{jh} \cap I_{js}\}$: customers in both I_{jh} and I_{js},

$\theta_3 = \{I_{js} \backslash I_{jh}\}$: customers in I_{js} not in I_{jh}.

We can ignore θ_2 and θ_3 in (21) since they do not make any contribution: the customers in θ_2 will result in $\sum_{i \in \Theta_2} (1 - x_{ij}) = 0$, since in solution S, $x_{ij} = 1, \forall i \in \theta_2$, and the customers in θ_3 are not in the summation term of (21) as they are not in the set I_{jh}. Thus, (21) can be written as:

$$numVeh_{js} < numVeh^*_{jh} - \sum_{i \in \theta_1} (1 - x_{ij}) \quad (22)$$

Let $|\theta_1| = p$, then (22) becomes:

$$numVeh_{js} < numVeh^*_{jh} - p \quad (23)$$

Now we consider $I_{jh} \cup I_{js}$. Let $numVeh_j^{(I_{jh} \cup I_{js})}$ be the minimum number of vehicles needed for $I_{jh} \cup I_{js}$. Since $numVeh^*_{jh}$ is the minimum number of vehicles needed for I_{jh}, thus:

$$numVeh_j^{(I_{jh} \cup I_{js})} \geq numVeh^*_{jh} \quad (24)$$

We now assign the customers in $I_{jh} \cup I_{js}$ to vehicles as follows: assign I_{js} to $numVeh_{js}$ vehicles and assign each $i \in I_{jh} \backslash I_{js}$ to its own vehicle. As $I_{jh} \backslash I_{js} = \theta_1$ and $|\theta_1| = p$, we have:

$$numVeh_j^{(I_{jh} \cup I_{js})} \leq numVeh_{js} + p \quad (25)$$

and from (24) and (25):

$$numVeh_{js} \geq numVeh^*_{jh} - p \quad (26)$$

which contradicts (23) and thus, our assumption that S is a globally feasible solution that does not satisfy the cut. Therefore, the cut does not remove any globally feasible solutions (property 2).

Since properties 1 and 2 are satisfied and $numVeh_j$ has a finite domain, the proposed cut results in a finite convergence to optimality. □

3.4. Computational Results

We compare our Benders approach to the IP and tabu search models due to Albareda-Sambola et al. presented above. Unless otherwise noted, the tests were performed on a Dual Core AMD 270 CPU with 1 MB cache, 4 GB of main memory, running Red Hat Enterprise Linux 4. The IP model was implemented in ILOG CPLEX 11.0. The Benders IP/CP approach was implemented in ILOG CPLEX 11.0 and ILOG Solver 6.5.

3.4.1 IP vs. Benders

We first present the problem sets used to compare the IP model and our Benders approach. Then an in-depth analysis of the results is presented.

3.4.1.1 Problem Sets

We evaluate our Benders approach on two problem sets. In the first problem set, we generate problems following the same method as Albareda-Sambola et al. [4]. Since these instances were created by extending the random instances proposed by Barceló et al. [9] for SSCPLP, we have also generated our own problem instances which we believe are more realistic as they are designed specifically for CDCPLP.

Problem Set I We generated problems following exactly the same method as Albareda-Sambola et al. [4]. We start with the 25 instances of Barceló et al. [9]. These instances consist of 6 instances of size 20×10 (i.e., 20 clients, 10 possible facility sites), 11 instances of size 30×15, and 8 instances of size 40×20. The fixed facility opening cost, f_j, demands for each client, d_i, assignment costs, c_{ij}, and facility capacities, b_j, are as in the instances of Barceló et al. [9] with the exception that the facility capacities are multiplied by 1.5 as they were considered by Albareda-Sambola et al. to be too tight. Six different pairs of truck distance limit, l, and truck usage cost, u, values are then used to create different problem sets (l, u): $(40, 50), (40, 100), (50, 80), (50, 150), (100, 150), (100, 300)$. Finally, the travel distances, t_{ij}, are randomly generated based on the costs, c_{ij}, in two different ways. In the *correlated* case: $t_{ij} = scale(c_{ij}, [15, 45]) + rand[-5, 5]$. The first term is a uniform scaling of c_{ij} to the integer interval $[15, 45]$ while the second term is a random integer uniformly generated on the interval $[-5, 5]$. In the *uncorrelated* case, $t_{ij} = rand[10, 50]$. Overall, therefore, there are 300 problems instances: 25 original instances times 6 (l, u) conditions times 2 correlated/uncorrelated conditions. The only difference with the instances in Albareda-Sambola et al. arises from the random term in the generation of the travel distances.

Problem Set II The instances in this problem set consist of three sizes: 15 instances of size 20×10, 15 instances of size 30×15, and 15 instances of size 40×20. The facility capacities are randomly drawn from an integer interval $b_j \in [50, 200]$; the fixed facility opening costs, f_j, are randomly generated based on the capacities, b_j, and are calculated as: $f_j = (b_j \times (10 + rand[1, 15]))$, where 10 is the per unit capacity cost. We have added a random multiplier from [1,15] to take into account the difference in property cost. The maximum number of vehicles at a facility takes the value $\overline{k} = |I|/4$, where I is the set of all clients. The travel

distances are randomly drawn from an integer interval $t_{ij} \in [10, 60]$. The costs, c_{ij}, are randomly generated based on the travel distances t_{ij} in two different conditions. In the *correlated* condition: $c_{ij} = scale(t_{ij}, [10, 90]) + rand[-5, 5]$. The first term is a uniform scaling of t_{ij} to the integer interval $[10, 90]$, while the second term is a random integer uniformly generated on the interval $[-5, 5]$. In the *uncorrelated* condition, $c_{ij} = rand[5, 95]$. We have also generated six different pairs of truck distance limit, l, and truck usage cost, u, to create different problem sets: $(50, 50), (50, 100), (70, 100), (70, 150), (100, 150), (100, 300)$. In problem set II, therefore, there are 540 problem instances: 45 original instances times 6 (l, u) conditions times 2 correlated/uncorrelated conditions.

3.4.2 Results

Figures 3.4 and 3.5 show two scatter-plots of the run-times of problem set I and problem set II for both IP and the Benders approach. Both axes are log-scale and the points below the $x = y$ line indicate a lower run-time for the Benders approach. In problem set I, the Benders approach achieves an equivalent or better run-time on all but 26 of the 300 instances. In problem set II, on all but 21 of the 540 instances, the Benders approach achieves equivalent or better run-time.

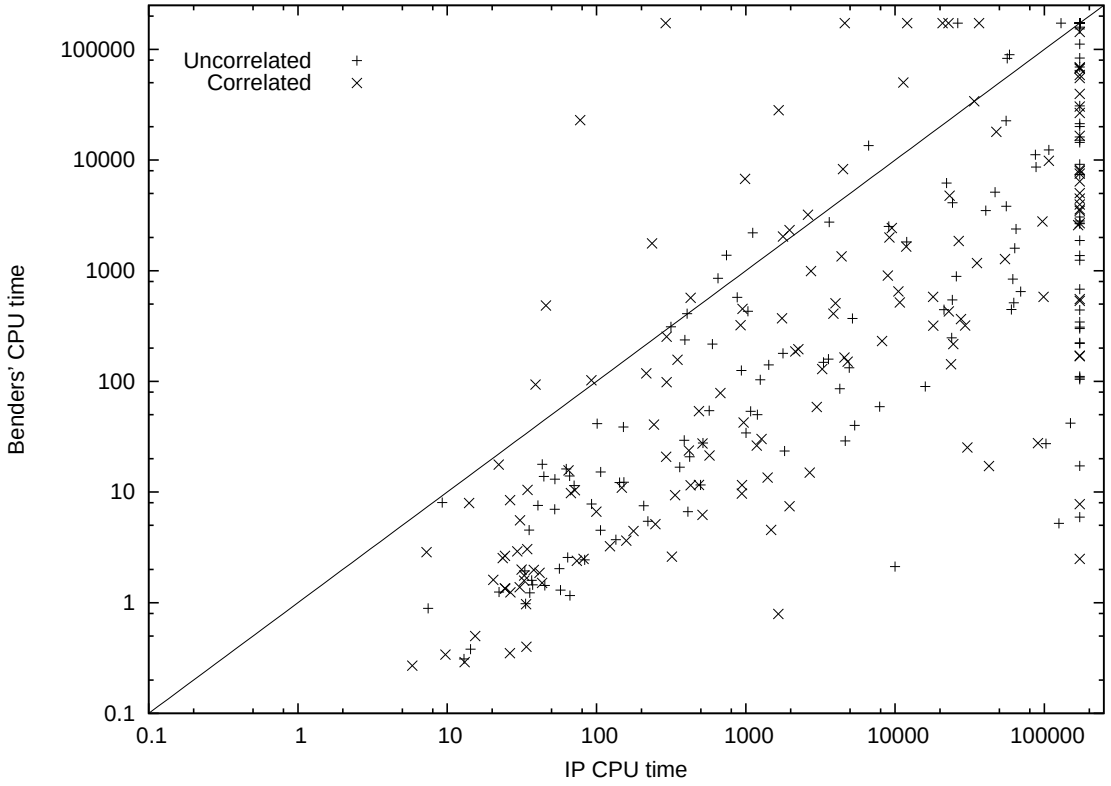

Figure 3.4: Run-time of IP model (x-axis, log-scale) vs. Benders IP/CP model (y-axis, log-scale) for problem set I. Points below the $x = y$ line indicate lower run-time for the Benders model.

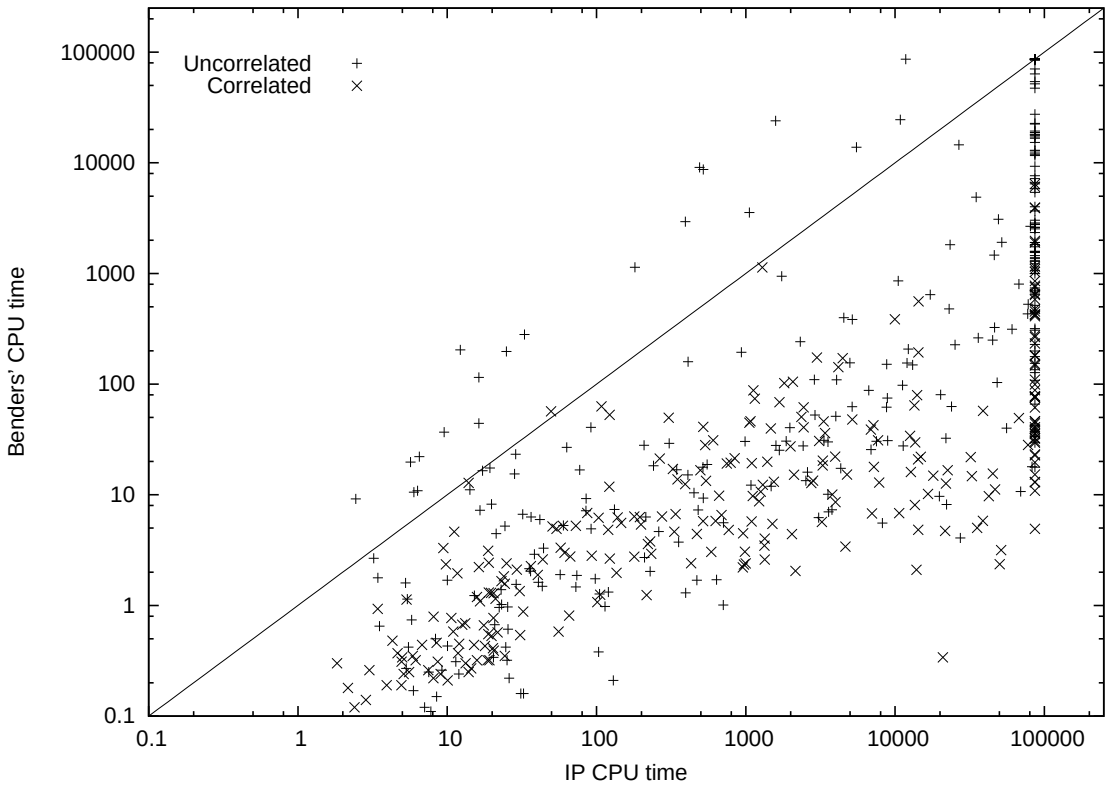

Figure 3.5: Run-time of IP model (x-axis, log-scale) vs. Benders IP/CP model (y-axis, log-scale) for problem set II. Points below the $x = y$ line indicate lower run-time for the Benders model.

Table 3.1 compares the mean CPU time in seconds required to solve each problem instance for each size in problem set I. In all cases, 24 hours was used as a maximum time, and for unsolved instances, the 24-hour time limit was used as their run-times in the calculations. The columns labeled "% Uns." indicate the percentage of unsolved problems. The Benders approach is unable to find the optimal solution within the time limit in 10.5% of instances while the IP model is unable to find the optimal solution in 30% of the instances. As can be seen, as the problem size increases, both the percentage of unsolved problems and the number of iterations increase. The "Time Ratio" for a given instance is calculated as the IP run-time

Problem Set	Uncorrelated						Correlated					
	IP		Benders			Time	IP		Benders			Time
	Time	% Uns.	Time	% Uns.	Iter	Ratio	Time	% Uns.	Time	% Uns.	Iter	Ratio
20×10	252	0	33	0	3.8	22.0	65	0	24	0	4.1	18.4
30×15	33850	29	11925	9	16	422.7	17451	17	5882	3	13.1	79.4
40×20	78107	81	39095	39	26.2	514.5	46591	42	19221	10	22.8	1125
Overall	39949	39	17766	16	16.3	355.9	22604	21	8745	5	14.0	399.4

Table 3.1: The mean CPU time (seconds) and percentage of unsolved problem instances (% Uns.) for the IP and Benders approaches. For the Benders approach, we also present the mean number of iterations. "Overall" indicates the mean results over all problem instances–recall that each subset has a different number of instances.

divided by the Benders run-time. The mean over each instance in each subset was then calculated. It can be seen that on average, Benders is over 300 times faster than the IP model.

Table 3.2 compares the mean CPU time in seconds required to solve each problem instance for each size in problem set II. In all cases a 24-hour time-limit was used. It can be seen that the Benders approach is unable to find the optimal solution in only 2.5% of the instances while the IP model is unable to find the optimal solution in 30% of the instances. The time ratios show that on average the Benders approach has a run-time of about 630 times faster than IP.

Problem Set	Uncorrelated						Correlated					
	IP		Benders			Time	IP		Benders			Time
	Time	% Uns.	Time	% Uns.	Iter	Ratio	Time	% Uns.	Time	% Uns.	Iter	Ratio
20×10	159	0	15	0	5.9	62.4	103	0	3	0	2.7	42.8
30×15	30819	24	3020	1	11.5	532.5	23325	18	170	0	3	2274.4
40×20	73366	78	20238	14	6.1	128.3	40933	43.3	294	0	2.3	746.8
Overall	34781	34	7757	5	7.9	241.1	21453	20.4	156	0	2.6	1021.3

Table 3.2: The mean CPU time (seconds) and percentage of unsolved problem instances (% Uns.) for the IP and Benders approaches. For the Benders approach, we also present the mean number of iterations. "Overall" indicates the mean results over all problem instances.

As can be seen from both problem sets, correlated instances are easier to solve, likely due to the fact that in the uncorrelated instances assignment costs and vehicle utilization act as conflicting criteria when making the customer assignment decisions [4].

The performance of the Benders approach is better in the second problem set. This is due to the fact that in the second problem set, the ratio of average capacities to average demand is smaller, and the facility capacities and costs are correlated, while in the first set this is not the case. The correlation between the capacities and costs cause the LAMP to find the solution faster, and using a smaller capacity to demand ratio makes the subproblem relaxation tighter, thus, the optimal solution is found in fewer iterations.

3.4.3 Tabu search vs. Benders

In the Benders decomposition approach the first globally feasible solution found is the optimal solution. Therefore, cutting off runs due to a time-limit will not result in any feasible solutions. Typically, then, for problems too large for a Benders approach to find optimality, another algorithm is needed to find a good, but not necessarily optimal solution. Metaheuristic techniques, such as tabu search, are widely used in location and routing problems for this purpose [67].

With our logic-based Benders formulation, however, we can derive a globally feasible, suboptimal solution at each iteration. In generating a cut, we find the minimum number of trucks needed at each facility. This number of trucks constitutes a feasible solution even though fewer trucks were assigned in the master solution. While this method will not generate a feasible solution if the minimal number of trucks required at a facility is greater than $\overline{k}$, we did not encounter this case in any of our problem instances. Thus, at the end of each iteration, we have a globally feasible solution, and therefore, our Benders approach may be useful even when the goal is to find as good a solution as possible in a limited amount of time.

Albareda-Sambola et al. used 72 small (20×10) problem instances (36 correlated and 36 uncorrelated), 11 medium (30×15) problem instances, and 8 large (40×20) problems instances generated as in problem set I to evaluate the performance of their tabu search approach. The

medium and large problem instances are correlated instances with (l, u) values of $(50, 80)$. We received these instances from the authors.[1] We believe that the run-time for the first iteration of our Benders model and therefore the time to find the *first* feasible solution provides some basis for comparison with the tabu search.

Table 3.3 presents the mean and the median time for the first iteration and the mean percentage gap from optimal for the Benders approach. This is compared to the run-time on a 2.4GHz Pentium IV and mean percentage gap from the optimal solution reported by Albareda-Sambola et al. [4]. They presented the cost of their best solution and bounds on the percentage gap. As we found the optimal solutions, we were able to calculate the exact gap from optimality for the tabu search. The columns labeled "% Opt." indicate the percentage of problems in each set for which the optimal solution was obtained. The columns labeled "# Dom." indicate the number of problems in each set for which one approach was dominant, where dominance is defined as finding a better solution in less time. The median and # Dom of the small instances for the tabu search are not reported since we were not able to calculate them for the results provided in Albareda-Sambola et al. [4].

For medium and large instances, logic-based Benders exhibits three run-time outliers that obscure the results. For the medium and large instances, Benders finds a better solution faster than tabu search on 10 out of the 19 instances while tabu search is not able to find a better solution faster than Benders for any instance. As can be seen, on average the Benders approach finds feasible solutions 1.4 times faster than tabu search and finds solutions with substantially smaller optimality gaps. It can also be seen that in 46.2% of the instances, Benders finds the optimal solution in the first iteration, while the tabu search finds the optimal solution over *all* iterations in only 16% of the instances.

Problem Set	Benders					Tabu				
	Time		% Gap	% Opt.	# Dom.	Time		% Gap	% Opt.	# Dom.
	Mean	Median	Mean			Mean	Median	Mean		
20×10	2.2	1.6	1.6	53	-	19	-	2.15	20.8	-
30×15	60.4	13.5	2.07	27	4	60.5	66.1	4.12	0	0
40×20	185.4	86.2	1.82	12	6	148.7	163.0	10.41	0	0
Overall	25.3	10.5	1.67	46	10	35.18	-	3.11	16	0

Table 3.3: The mean and median CPU time (seconds), the mean percentage gap from optimal, percentage of instances for which the optimal solution was obtained, and the number of instances for which each approach dominated the other. The tabu search results are taken from Albareda-Sambola et al. [4].

3.5. Discussion

In this chapter we presented the first application of logic-based Benders decomposition to a facility location/fleet management problem. We demonstrated that the proposed Benders approach is the current state-of-the-art technique for finding both exact and good solutions for such a problem. Although, other techniques such as a dedicated branch-and-bound and branch-and-price have been applied to location problems, such techniques have not been applied to our

[1] We would like to thank Maria Albareda-Sambola for providing these instances.

facility location/fleet management problem. Thus, there is no evidence that our approach could outperform such techniques. Therefore, as non-trivial future work, it would be interesting to implement these techniques to the CDCPLP and compare their performance with logic-based Benders decomposition.

It should be noted that the reason why we compared our Benders model to an IP model formulated in and solved with CPLEX, the state-of-the-art MIP solver containing many sophisticated heuristics and techniques, is that MIP is the default approach for hard combinatorial problems. Furthermore, in all the work that we are aware of on logic-based Benders decomposition, MIP performance was one of the main comparanda.

3.6. Conclusion

In this chapter, we presented a novel logic-based Benders decomposition approach to the capacity and distance constrained plant location problem. Our approach was able to substantially out-perform an existing IP model by finding and proving optimality between 240 to 1000 times faster depending on the problem set and size. Our approach also performed better than an existing tabu search in finding good, feasible solutions in a short time. On over half of the test instances the Benders approach finds a better solution in less time than the tabu search.

At least two directions for future research are suggested by this study. Firstly, it would be interesting to add complete vehicle routes to the problem. In this case, vehicles would visit multiple clients in a single tour and so the subproblems would become vehicle routing problems. Secondly, in real world situations some of the components of the problem, such as travel-times, might not be known with certainty. Therefore, in the next chapter we extend the CDCPLP by considering stochastic travel-times.

Chapter 4

Using Decomposition to Solve a Stochastic Facility Location/Fleet Management Problem

In the previous chapter, we showed that logic-based Benders decomposition can be effective in solving combined facility location and fleet management problems. However, it was assumed that all important aspects of the problem are known with certainty or are controllable by the organization. Such an assumption is unrealistic. For example, travel-times are not certain in practice but rather are affected by factors such as weather conditions, daily traffic patterns, and collisions.

In this chapter, we extend the problem addressed in Chapter 3 by considering random travel-times. We introduce the stochastic facility location and vehicle assignment problem (SFLVAP) and present a stochastic programming model based on a bounded penalty approach in which the expected recourse cost cannot exceed a given threshold. In order to solve the problem, we propose an integer programming (IP) model and a two-level and a three-level logic-based Benders decomposition approaches.

This chapter is organized as follows: Section 4.1 presents a detailed definition of the problem along with an IP formulation. The details of the two-level logic-based Benders decomposition approach are described in Section 4.2. The three-level logic-based Benders decomposition is presented in Section 4.3. Computational results are presented in Section 4.4. In Section 4.5, a discussion on the main contributions of the chapter is presented. Section 4.6 concludes the chapter.

4.1. Problem Definition and Formulation

The stochastic facility location and vehicle assignment problem (SFLVAP) is a combined facility location/fleet management problem which aims at selecting the optimal set of facilities to open, determining the vehicle fleet size at each opened site, and assigning clients to facilities and vehicles in the presence of random travel-times. Therefore, the SFLVAP is a network design problem with travel-time uncertainty (i.e., non-deterministic link lengths). We consider the different travel-time conditions as different *scenarios* with known probabilities, where a

scenario is a snapshot of the network with regard to link lengths. Recall from Section 2.2.1 that the scenario analysis approach is one of the common ways to deal with uncertainty in location problems.

The SFLVAP can be modeled as a two-stage stochastic program with recourse, where the expected recourse cost is required to be less than a threshold value. Specifically, we use the penalty bounded approach [66]. In the first stage, before the travel-times are known, we select the set of facilities to open, determine the number of vehicles required at each opened site, and assign clients to facilities and vehicles. In the second stage, when the travel-times are observed, a failure occurs when a vehicle travel-time limit is exceeded. In this situation, the vehicle continues serving its assigned customers, however, a penalty cost is paid to the driver for every extra unit of time he/she has to operate. The penalty cost is the recourse cost of the vehicle. The goal is to minimize the first stage cost, which consists of facility opening, vehicles utilization, and client assignment costs, under the condition that the expected recourse cost of each vehicle does not exceed a specified fraction of its utilization cost.

Formally, let J be the set of potential facilities, each housing a number of identical vehicles, and I, the set of clients. Clients are served by full return trips from the facility. The same vehicle can be used to serve several clients as long as its travel-time does not exceed a given total driving duration. For each site j there is an opening cost, f_j, and a capacity, b_j. The sum of demands of the clients, d_i, assigned to a site must be less than or equal to the site capacity. Each vehicle has a fixed utilization cost, u, and a maximum total daily travel-time, l. We assume that travel-times will occur according to one of $s \in S$ scenarios with probability P_s (S is the set of all scenarios). Serving client i from site j in scenario s generates a travel-time, t_{ij}^s, for the vehicle performing the service. Providing service to client i from site j has an associated cost, $c_{ij} = f(\sum_s P_s t_{ij}^s)$, where c_{ij} is a deterministic value, fixed for every i and j, and is a function of the expected travel-times. The available vehicles at a site are indexed in set K with parameter $\overline{k} \geq |K|$ being the maximum number of vehicles at any site. We define Π_{jk}^s to be the recourse cost of vehicle k of facility j in scenario s, pen to be the per time-unit penalty cost and αu to be the maximum allowable penalty (a fraction of vehicle utilization cost u).

In order to formulate the problem, building on the non-stochastic model proposed by Albareda-Sambola et al. [4], we use the following decision variables:

$$p_j = \begin{cases} 1, & \text{if facility } j \text{ is open} \\ 0, & \text{otherwise} \end{cases}$$

$$z_{jk} = \begin{cases} 1, & \text{if a } k\text{th vehicle is assigned to site } j \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ijk} = \begin{cases} 1, & \text{if client } i \text{ is served by the } k\text{th vehicle of site } j \\ 0, & \text{otherwise} \end{cases}$$

The integer programming (IP) formulation is as follows:

$$\min \quad \sum_{j\in J} f_j p_j + u \sum_{j\in J}\sum_{k\in K} z_{jk} + \sum_{i\in I}\sum_{j\in J} c_{ij} \sum_{k\in k} x_{ijk}$$

$$\text{s.t.} \quad \sum_{j\in J}\sum_{k\in K} x_{ijk} = 1 \qquad i \in I \qquad (1)$$

$$\sum_{i\in I}\sum_{k\in K} d_i x_{ijk} \leq b_j p_j \quad j \in J \qquad (2)$$

$$E(\Pi_{jk}) \leq \alpha u \qquad j \in J, k \in K \qquad (3)$$

$$z_{jk} \leq p_j \qquad j \in J, k \in K \qquad (4)$$

$$x_{ijk} \leq z_{jk} \qquad i \in I, j \in J, k \in K \qquad (5)$$

$$z_{jk} \leq z_{jk-1} \qquad j \in J, k \in K\backslash\{1\} \qquad (6)$$

$$x_{ijk}, p_j, z_{jk} \in \{0,1\} \qquad i \in I, j \in J, k \in K \qquad (7)$$

The objective function minimizes the sum of the costs of opening the facilities, driving the vehicles, and serving the customers. Constraint (1) ensures that each client is served by exactly one facility. Constraint (2) limits the client demand allocated to facility j. Constraints (4) and (5) ensure that a vehicle cannot be assigned to a site that has not been opened, and a client cannot be served by a vehicle that has not been allocated. Constraint (6) states that at a site, vehicle k will not be used before vehicle $k-1$.

The recourse cost constraints are defined by (3), where the expected recourse cost of vehicle k of facility j is

$$E(\Pi_{jk}) = \textstyle\sum_S (P_s \times \Pi^s_{jk})$$

and,

$$\Pi^s_{jk} = [pen \times (\textstyle\sum_i (t^s_{ij} x_{ijk}) - l.z_{jk})]^+.$$

The $+$ sign represents $max(0, pen \times (\sum_i (t^s_{ij} x_{ijk}) - l.z_{jk}))$. We take the maximum because "negative penalties" (i.e., rewards) are not given in scenarios where the vehicle is able to serve its customers without exceeding its travel-time limit. Thus constraint (3) can be rewritten as:

$$\textstyle\sum_S (P_s \times [pen \times (\sum_i (t^s_{ij} x_{ijk}) - l.z_{jk})]^+) \leq \alpha u$$

4.2. A Two-Level Logic-Based Benders Decomposition Approach

As shown in Figure 4.1, the SFLVAP can be decomposed into an expected value master problem (EVMP) and a set of scenario sub-problems (SSPs). In the EVMP, the stochastic variables are replaced with their expected values. Thus, instead of having multiple scenarios of the network, we only have one scenario, where each travel-time is equal to the probability-weighted mean of the travel-times over all the scenarios. Each SSP, one for every scenario, calculates the recourse cost for the corresponding scenario given the EVMP solution. Thus, for each SSP we plug-in the EVMP solution into the recourse cost equations and check to see whether the expected recourse cost constraints are violated.

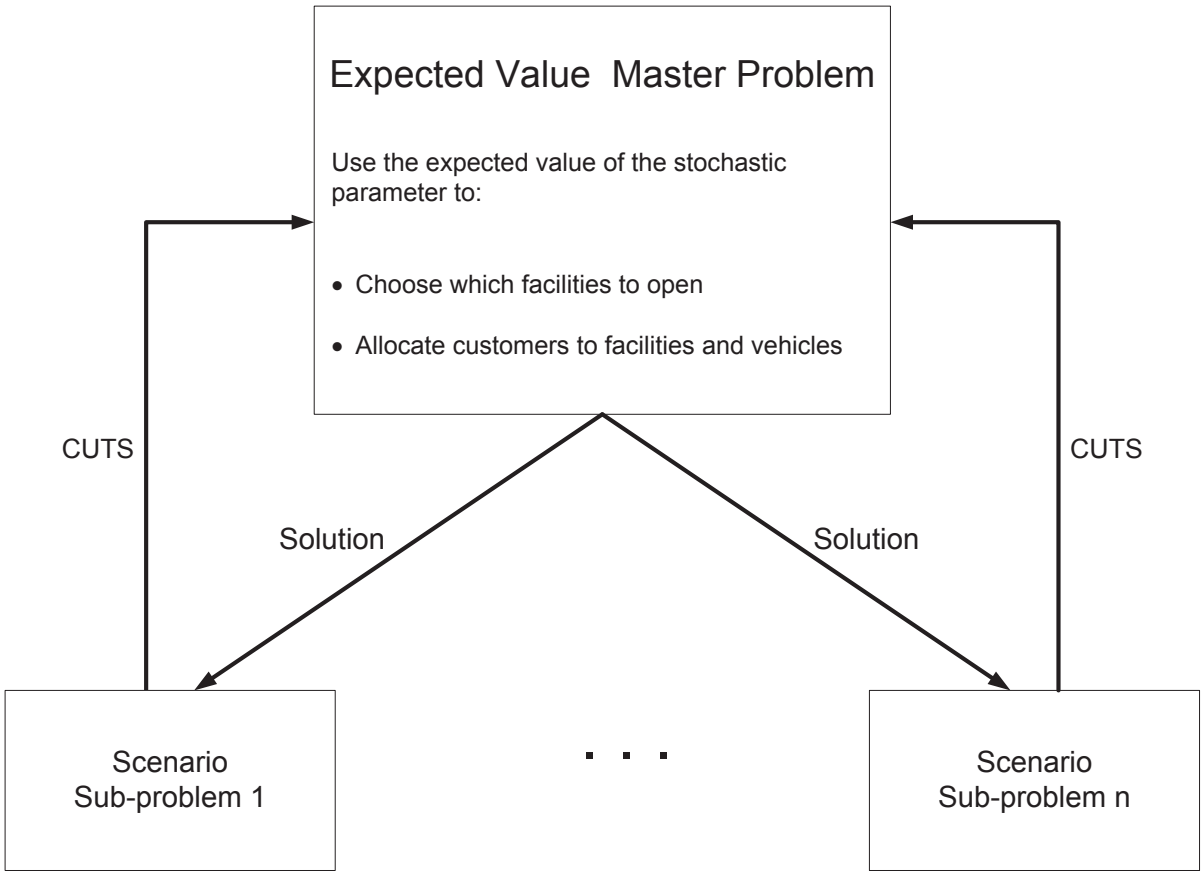

Figure 4.1: A Logic-Based Benders Decomposition Approach for the SFLVAP

4.2.1 The Expected Value Master Problem

Let $\overline{t_{ij}}$ be the expected travel-time between client i and facility j over all scenarios, where

$$\overline{t_{ij}} = \sum_S P_s \times t_{ij}^s.$$

An integer programming (IP) formulation of EVMP is as follows:

$$\min \quad \sum_{j\in J} f_j p_j + u \sum_{j\in J}\sum_{k\in K} z_{jk} + \sum_{i\in I}\sum_{j\in J} c_{ij} \sum_{k\in k} x_{ijk}$$

$$\text{s.t.} \quad \sum_{j\in J}\sum_{k\in K} x_{ijk} = 1 \qquad i \in I \qquad (8)$$

$$\sum_{i\in I}\sum_{k\in K} d_i x_{ijk} \leq b_j p_j \qquad j \in J \qquad (9)$$

$$pen \times (\sum_i \overline{t_{ij}} x_{ijk} - l \times z_{jk}) \leq \alpha u \quad j \in J, k \in K \qquad (10)$$

$$cuts \qquad (11)$$

$$z_{jk} \leq p_j \qquad j \in J, k \in K \qquad (12)$$

$$x_{ijk} \leq z_{jk} \qquad i \in I, j \in J, k \in K \qquad (13)$$

$$z_{jk} \leq z_{jk-1} \qquad j \in J, k \in K\backslash\{1\} \qquad (14)$$

$$x_{ijk}, p_j, z_{jk} \in \{0,1\} \qquad i \in I, j \in J, k \in K \qquad (15)$$

The objective function and constraints (8)-(9) and (12)-(15) are the same as before. The recourse cost limits for the expected travel-times are defined by constraint (10). It should be noted that constraint (10) imposes limits on the recourse cost of only one network structure (the network where the travel-times equal the mean travel-time over all the scenarios) while constraint (3) limits the recourse cost over different scenarios of the network. $Cuts$ are constraints that are added to the master problem each time one of the sub-problems is not able to find a feasible solution. A detailed description of the cuts is presented in Section 4.3.

In Benders decomposition, the master problem must be a relaxation of the original model. Therefore, in order to justify our use of the expected travel-times in constraint (10), we must prove the following theorem.

Theorem 1. The solution of EVMP is a lower bound for SFLVAP.

$$EVMP \leq SFLVAP$$

Proof. In the EVMP and the SFLVAP, constraints (3) and (10) apply to each vehicle respectively. Therefore, here we focus on a single vehicle, k', at open facility j'. Let I' represent the set of customers assigned to k'. We need to show that for a given set of $x_{ij'k'}$ and $z_{j'k'}$ assignments, the left hand side of constraint (10) is always smaller than or equal to the left hand side of constraint (3):

$$pen \times [\sum_{i\in I'}(\sum_S P_s t^s_{ij'}) - l] \leq pen \times \sum_S P_s[(\sum_{i\in I'} t^s_{ij'}) - l]^+ \quad (16)$$

Assume that for $N \subseteq S$, the total travel-time of the vehicle is less than or equal to the travel-time limit, l, thus,

$$[(\sum_{i\in I'} t^s_{ij'}) - l] \leq 0 \quad s \in N \quad (17)$$

We need to show that equation (16) holds for three cases: $N = \emptyset$, $N = S$, and $N \subset S$.

Case 1: If $N = \emptyset$, then the recourse costs of all scenarios are positive, thus the max operator can be removed from constraint (3). In this case the left hand sides of constraints (10) and (3) are identical, satisfying (16).

Case 2: If $N = S$, then the recourse cost for all scenarios is zero, therefore $[(\sum_{i\in I'} t^s_{ij'}) - l]^+ = 0$, making the right hand side of (16) zero:

$$pen \times [\sum_{i\in I'}(\sum_S P_s t^s_{ij'}) - l] \leq 0 \quad (18)$$

By rearranging the sums and dividing by the positive number pen, (18) can be rewritten as:

$$(\sum_S P_s \sum_{i\in I'} t^s_{ij'}) \leq l \quad (19)$$

which is true since from (17) the total travel-time of the vehicle in all scenarios is less than or equal to the travel limit, l.

Case 3: Finally we need to show that equation (16) holds if $N \subset S$. In this case, removing pen, the left hand side of (16) can be rewritten as:

$$[\sum_{i\in I'}(\sum_{s\in N} P_s t^s_{ij'})] + [\sum_{i\in I'}(\sum_{s\in S\setminus N} P_s t^s_{ij'})] - l \quad (20)$$

Since for the scenarios in N the recourse cost is zero, the right hand side of (16) is:

$$[\sum_{s\in S\setminus N} P_s(\sum_{i\in I'} t^s_{ij'})] - [\sum_{s\in S\setminus N} P_s l] \quad (21)$$

By replacing the left and right hand sides of (16) by (20) and (21) we have,

$$[\sum_{s\in N} P_s(\sum_{i\in I'} t^s_{ij'})] - l \leq -\sum_{s\in S\setminus N} P_s l \quad (22)$$

Since $\sum_{s\in S\setminus N} P_s = 1 - \sum_{s\in N} P_s$, from (22) we have,

$$[\sum_{s\in N} P_s(\sum_{i\in I'} t^s_{ij'})] \leq l[\sum_{s\in N} P_s] \quad (23)$$

Finally, from (23),

$$(\sum_{s\in N} P_s \sum_{i\in I'} t^s_{ij'}) - l \leq 0 \quad s \in N \quad (24)$$

which is true, since in N the recourse cost is zero.

Thus, we have shown that the solution of EVMP is a lower bound for SFLVAP.

□

It is obviously true that the deterministic problem where each link is assigned to its minimum over all scenarios is also a lower bound on SFLVAP. Therefore, such a formulation could be used in place of the EVMP. However, this approach would not be as tight as the expected value. As noted in Section 2.3.2, finding a tight sub-problem relaxation is the usual challenge with logic-based Benders.

4.2.2 The Scenario Sub-problem

Given the set of facilities opened (J_o), the set of clients allocated to vehicle k of facility j ($I_{jk} = i|x_{ijk} = 1$), and the set of vehicles allocated to facility j ($K_j = k|k_{jk} = 1$), the SSP calculates the recourse cost of each vehicle in scenario s.

Formally, the recourse cost of scenario s is:

$$\Pi^s_{jk} = [pen \times (\sum_{i \in I_{jk}} (t^s_{ij}) - l)]^+, \quad j \in J_o, k \in K_j$$

After all the SSPs have been solved, the expected recourse cost of each vehicle over all the scenarios is calculated,

$$E(\Pi_{jk}) = \sum_{s \in S} P_s \Pi^s_{jk}, \quad j \in J_o, k \in K_j$$

Then a set of feasibility checks, one for every active vehicle, is done to see whether any of the expected recourse cost constraints are violated, thus for each active vehicle we check the following condition:

$$E(\Pi_{jk}) \leq \alpha u, \quad \forall j \in J_o, \forall k \in K_j$$

If, for all vehicles, the condition is satisfied, the EVMP solution does not violate any of the expected recourse cost constraints (constraints (3)). Since the EVMP solution is a lower bound on the SFLVAP solution (Theorem 1), we have found the optimal solution to SFLVAP. If one or more vehicles does not satisfy the condition, the EVMP solution has violated at least one of the expected recourse cost constraints, and, thus, a cut is sent back to the EVMP which prohibits at least the current optimal solution of the EVMP.

4.2.3 Benders Cuts

An essential part of logic-based Benders decomposition is the generation of Benders cuts. Each time one of the sub-problems is not feasible with respect to the master problem solution, a Benders cut is added to the master problem to ensure that all future solutions are closer to being feasible.

Assume that in one particular iteration, h, the solution to the EVMP assigns a set, I^h_{jk}, of clients to vehicle k of facility j. Assume further that there is no feasible solution with regards to the expected recourse cost for vehicle k. The cut generated specifies a new lower bound on the expected recourse cost of vehicle k. At the least, this new lower bound prevents the clients in I^h_{jk} or a super-set of them from being assigned to vehicle k in future iterations.

Formally, the cuts after iteration h are:

$$\sum_S (P_s \times [\Pi^{sh}_{jk} - pen \times (\sum_{i \in I^h_{jk}} (t^s_{ij}(1 - x_{ijk})))]) \leq \alpha u, \quad j \in J_h, k \in K_{jh}, \forall \Pi^{sh}_{jk} > 0 \quad (25)$$

where $I^h_{jk} = \{i \mid x^h_{ijk} = 1\}$ is the set of clients assigned to vehicle k of facility j in iteration h, J_h is the set of sites for which vehicle $k \in K_{jh} = \{k \mid z^h_{jk} = 1\}$ results in infeasible sub-problems in iteration h, and Π^{sh}_{jk} is the recourse cost of vehicle k of facility j in scenario s in iteration h (only the Π^{sh}_{jk} that are greater than zero are considered in the cuts to make them tighter). The inner summation term is an upper bound on the decrease in the recourse cost of the corresponding vehicle, given that the clients are reassigned to other vehicles: the maximum amount of reduction of the recourse cost of scenario s in reassigning one client is pen times the travel-time of that client in scenario s.

Since each facility houses a homogeneous set of vehicles, a cut for every vehicle up to $\overline{K}$ is added to the EVMP.

Example Consider the network with the two scenarios shown in Figure 4.2. The network consists of one facility and three customers. Each scenario has probability 0.5. The maximum total driving-time, $l = 35$, and vehicle utilization cost $u = 40$, the per unit penalty cost, $pen = 5$, and $\alpha = 0.1$. From the figure, the expected travel-time for the customers over all scenarios are $\overline{t_{11}} = 15, \overline{t_{21}} = 15, \overline{t_{31}} = 2$. Since the sum of the expected travel-times is 32, which is smaller than the maximum travel-limit, 35, in the first iteration the master assigns all the customers to one vehicle, thus $x_{111} = 1, x_{211} = 1, x_{311} = 1$. This solution is sent to the SSPs to test feasibility with respect to the expected recourse cost constraints. From SSP1 the recourse cost of the vehicle in scenario 1 is equal to $\Pi^1_{11} = 0$ since the total travel is only 25. From SSP2, the recourse cost of the vehicle in scenario 2 is $\Pi^2_{11} = 20$. The expected recourse cost of the vehicle in iteration one is $E[\Pi_{11}] = 10$ which is greater than the $\alpha u = 4$. Thus, the following cut is added to the master problem:

$$\begin{aligned} 0.5 \times [\Pi^{21}_{11} - 5 \times (\textstyle\sum_{i=1..3} t^1_{ij}(1 - x_{ijk}))] \leq 4 \qquad & j = 1, \ k = 1 \\ = 0.5 \times [20 - 5 \times (t^1_{11}(1 - x_{111}) + t^1_{21}(1 - x_{211}) + t^1_{31}(1 - x_{311}))] \leq 4 \quad (26) & \end{aligned}$$

The cut prohibits the reassignment of the same set of customers to one vehicle, since by reassigning the customers to the same truck, $x_{111} = 1, x_{211} = 1, x_{311} = 1$, (26) is reduced to $E(\Pi_{11})$, which we know from the SSPs is greater than αu. The cut also prohibits the assignment of customers 1 and 2 to the same vehicle since by reassigning them to one vehicle and only assigning customer 3 to another vehicle, the expected recourse cost of the vehicle would

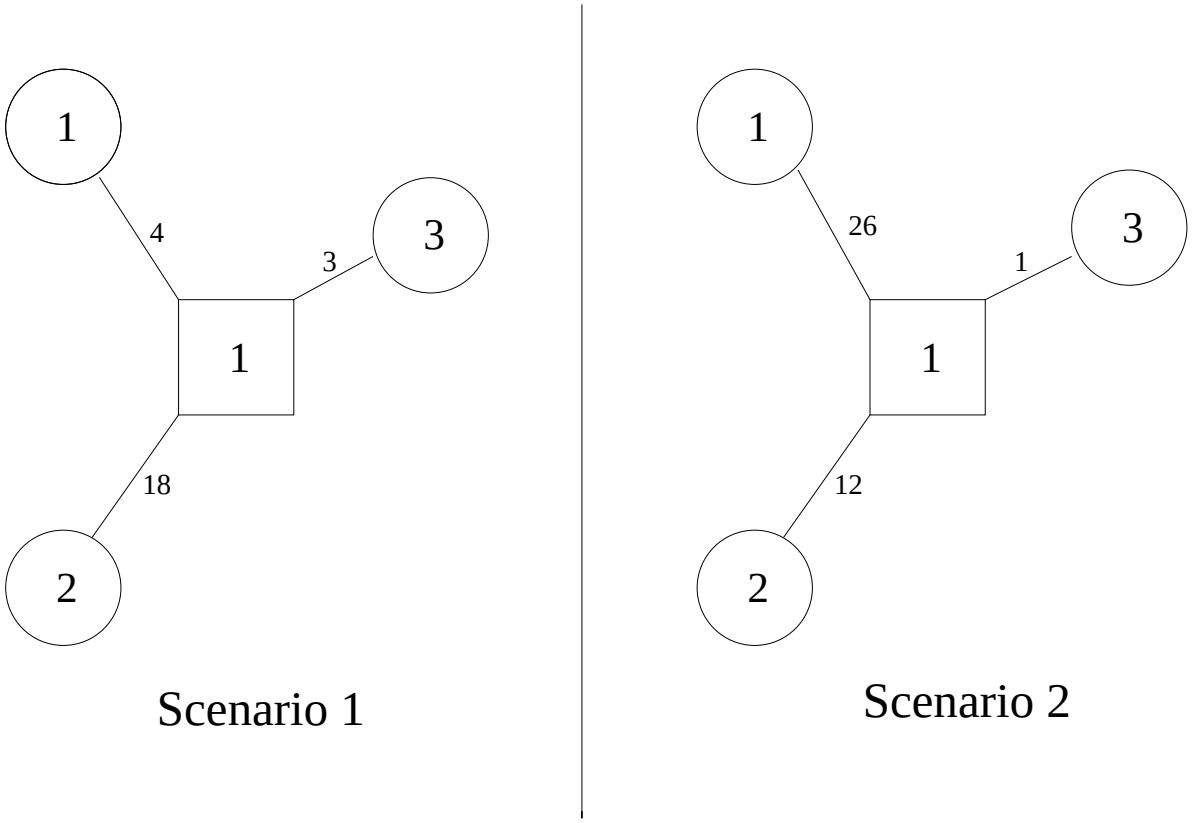

Figure 4.2: Example

be reduced to 7.5, which is still greater than 4. Thus, in any future iteration customer 1 and 2 must be assigned to different vehicles.

Recall from Section 3.3.3 that Chu and Xia [25] define a $valid$ Benders cut as a logical expression that has two properties:

$Property\ 1$: the cut must exclude the current master problem solution if it is not globally feasible,

$Property\ 2$: the cut must not remove any globally feasible solutions.

They show that property 1 guarantees finite convergence if the master problem variables have finite domains and property 2 guarantees global optimality.

Theorem 2. The proposed Benders cut is a valid cut. Thus, our logic-based Benders approach will converge to optimality in a finite number of steps.

Proof. In order to prove the validity of our proposed cut we need to show that both properties

1 and 2 are satisfied.

We first show property 1. Recall that I_{jk}^h is the set of clients assigned to vehicle k of facility j in iteration h, and $E(\Pi_{jk})^h$ is the expected recourse cost of vehicle k at facility j in iteration h of the EVMP model. If the EVMP solution is infeasible, then:

$$E(\Pi_{jk})^h > \alpha u \quad (27)$$

which will result in the cut:

$$\sum_S (P_s \times [\Pi_{jk}^{sh} - pen \times (\sum_{i \in I_{jk}^h} (t_{ij}^s (1 - x_{ijk})))]) \leq \alpha u \quad (28)$$

If the same set of customers is re-assigned to facility j, $\sum_{i \in I_{jk}^h} (1 - x_{ijk}) = 0$. This will result in the left hand side of (28) to be equal to $E(\Pi_{jk})^h$, which we know from (27) is greater than αu. Thus, the cut excludes the current master problem solution from all subsequent master problem solutions (property 1).

We now show property 2, that is, our proposed cut does not remove any globally feasible solutions. Let G be a globally feasible solution found in iteration $g > h$, thus:

$$E(\Pi_{jk})^g \leq \alpha u \quad (29)$$

We present a proof by contradiction and so assume that G does not satisfy the cut. Let I_{jk}^g be the set of clients assigned to vehicle k of facility j in iteration g. We define three sets:

$\theta_1 = \{I_{jk}^h \backslash I_{jk}^g\}$: customers in I_{jk}^h not in I_{jk}^g.
$\theta_2 = \{I_{jk}^g \backslash I_{jk}^h\}$: customers in I_{jk}^g not in I_{jk}^h.
$\theta_3 = \{I_{jk}^h \cap I_{jk}^g\}$: customers in both I_{jk}^h and I_{jk}^g.

If G does not satisfy the cut, then:

$$\sum_S (P_s \times [\Pi_{jk}^{sh} - pen \times (\sum_{i \in \theta_1} t_{ij}^s)]) > \alpha u \quad (30)$$

From (25) we know that only the Π_{jk}^{sh} that are greater than zero are considered in the cut, thus in (30), Π_{jk}^{sh} can be replaced by $pen \times [\sum_{i \in I_{jk}^h} t_{ij}^s - l]$. Therefore,

$$\sum_S (P_s \times [pen \times (\sum_{i \in I_{jk}^h} t_{ij}^s - l - \sum_{i \in \theta_1} t_{ij}^s)]) > \alpha u \quad (31)$$

Since $\sum_{i \in I_{jk}^h} t_{ij}^s = \sum_{i \in \theta_1} t_{ij}^s + \sum_{i \in \theta_3} t_{ij}^s$, (31) is reduced to,

$$\sum_S (P_s \times [pen \times (\sum_{i \in \theta_3} t_{ij}^s - l)]) > \alpha u \quad (32)$$

We know that $\theta_3 \subseteq I_{jk}^g$, so,

$$\sum_S (P_s \times [pen \times (\sum_{i \in I_{jk}^g} t_{ij}^s - l)]) \geq \sum_S (P_s \times [pen \times (\sum_{i \in \theta_3} t_{ij}^s - l)]) \quad (33)$$

and from Theorem 1 we know that,

$$\sum_S (P_s \times [pen \times (\sum_{i \in I_{jk}^g} t_{ij}^s - l)]^+) \geq \sum_S (P_s \times [pen \times (\sum_{i \in I_{jk}^g} t_{ij}^s - l)]) \quad (34)$$

Finally, from (32), (33), (34) we have:

$$\sum_S (P_s \times [pen \times \sum_{i \in I^g_{jk}} t^s_{ij}]^+) > \alpha u \quad (35)$$

The left hand side is $E(\Pi_{jk})^g$, which contradicts (29) and thus, our assumption that G is a globally feasible solution that does not satisfy the cut. Therefore, the cut does not remove any globally feasible solutions (property 2).

Since properties 1 and 2 are satisfied, and x_{ijk} decision variables have finite domains, the proposed cut results in a finite convergence to optimality. □

4.3. A Three-Level Logic-Based Benders Decomposition Approach

In the previous section, we demonstrated that the SFLVAP can be decomposed into a deterministic master problem and a set of stochastic sub-problems. Since the master problem is deterministic, with little modification we can use the logic-based Benders approach proposed in Chapter 3 for the CDCPLP to develop a three-level Benders decomposition for the SFLVAP. Thus, we can decompose the SFLVAP into an expected value location-allocation master problem (EVLAMP), a set of expected value truck assignment sub-problems (EVTASPs), and a set of scenario sub-problems (SSPs) for each EVTASP. A schematic representation of our three-level logic-based Benders decomposition approach is shown in Figure 4.3. The EVLAMP is

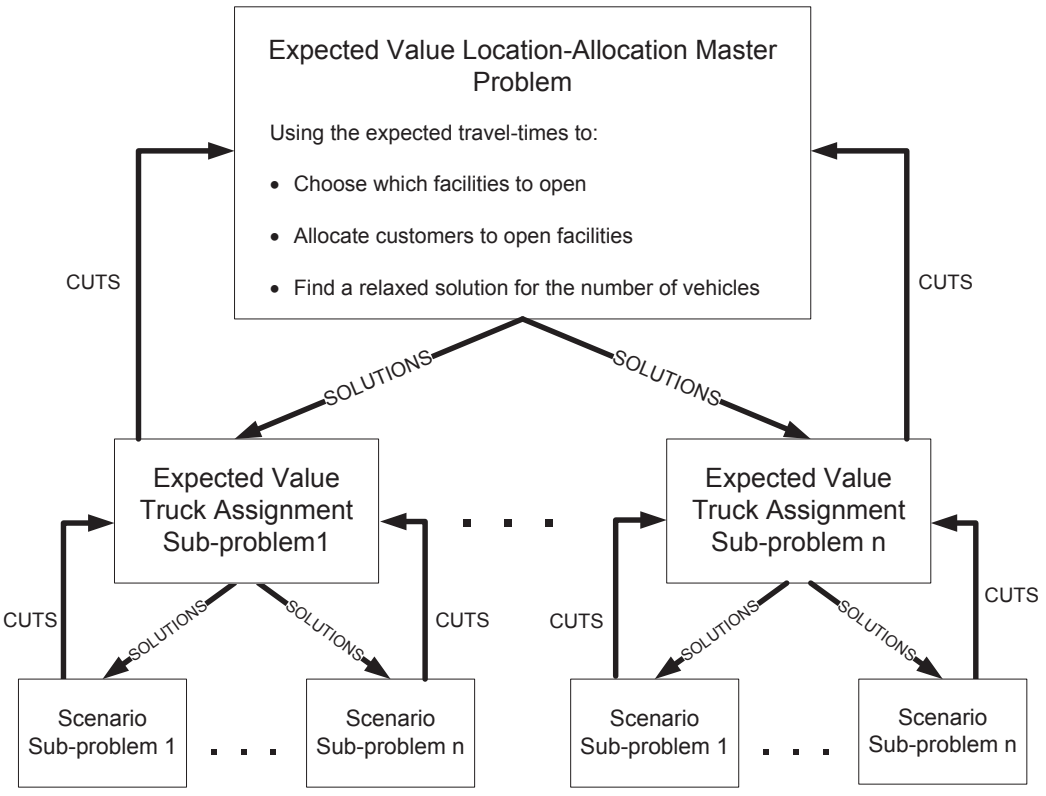

Figure 4.3: A Three-Level Logic-Based Benders Decomposition Approach for the SFLVAP

concerned with choosing the open facilities, allocating clients to these sites, and deciding on a relaxed number of trucks at each site. In the EVLAMP, the expected travel-times are used. The EVTASPs use the expected travel-times of each client to assign them to specific vehicles and can be modeled as a set of independent bin-packing problems, one for each open facility. Clients are allocated to the trucks so that the expected travel-time limit on each truck is satisfied. Each SSP, one for every scenario, calculates the recourse cost of each truck for the corresponding scenario given the EVTASP solution. Thus, for each SSP we plug-in the EVTASP solution into the recourse cost equations of each truck and check to see whether its expected recourse cost is violated. We use IP for the EVLAMP and CP for the EVTASPs.

The algorithmic flow chart of the three-level logic-based Benders decomposition approach is shown Figure 4.4. As the flow chart shows, the solution of the EVLAMP is sent to the EVTASPs (one for each open facility). Using the expected travel-times, the EVTASP assigns the clients to trucks. This solution is then sent to the SSPs to see whether the recourse cost limit of the truck is violated. If it is violated, a cut is add to the corresponding EVTASP, and the EVTASP is re-solved. If no recourse limit is violated, we check to see whether the number

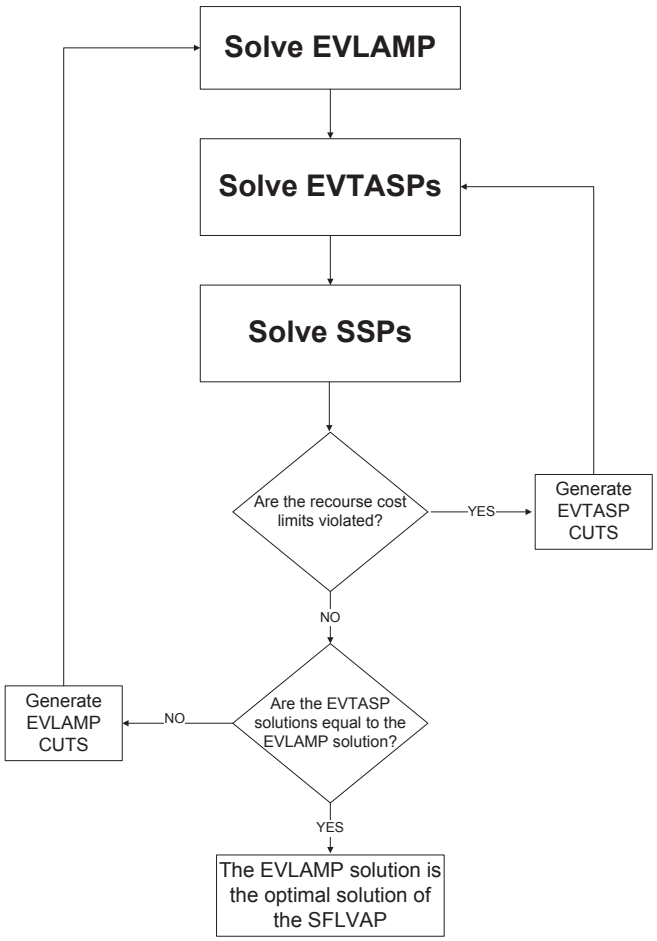

Figure 4.4: The algorithmic flowchart of the three-level logic-based Benders approach

of trucks indicated by the EVLAMP is equal to the number indicated by the EVTASP. If the two values are not equal, a cut is generated and added to the EVLAMP. In this case, after all the EVTASPs have been solved, the EVLAMP is re-solved with the new cuts. Otherwise, if for all open facilities, EVLAMP and EVTASP have equal solutions, the EVLAMP solution is globally optimal.

4.3.1 The Expected Value Location-Allocation Master Problem

An IP formulation of EVLAMP is as follows:

$$\text{minimize} \quad \sum_{j\in J} f_j p_j + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + u\sum_{j\in J} numVeh_j$$

$$\text{s.t.} \quad \sum_{j\in J} x_{ij} = 1 \qquad i \in I \qquad (36)$$

$$\sum_{i\in I} d_i x_{ij} \le b_j p_j \qquad j \in J \qquad (37)$$

$$\sum_{i\in I} \overline{t_{ij}} x_{ij} \le (l + \frac{\alpha u}{pen}) \cdot \overline{k} \qquad j \in J \qquad (38)$$

$$\overline{t_{ij}} x_{ij} \le (l + \frac{\alpha u}{pen}) \qquad i \in I, j \in J \qquad (39)$$

$$numVeh_j \ge \left\lceil \frac{\sum_{i\in I} t_{ij}x_{ij}}{l + \frac{\alpha u}{pen}} \right\rceil \qquad j \in J \qquad (40)$$

$$cuts \qquad (41)$$

$$x_{ij} \le p_j \qquad i \in I, j \in J \qquad (42)$$

$$x_{ij}, p_j \in \{0,1\}, numVeh_j \in \{0,..,\overline{k}\} \qquad i \in I, j \in J \qquad (43)$$

where p_j is a binary variable which indicates whether facility j is open, x_{ij} is a binary variable which indicates whether client i is assigned to site j, and $numVeh_j$ is an integer variable indicating the number of vehicles assigned to facility j.

Constraint (36) ensures that all clients are served by exactly one facility. Constraint (37) limits the demand assigned to facility j. In (38), $\overline{k}$ is the upper bound on the number of trucks at a facility. So, constraint (38) states the upper bound on the sum of the travel-times for all clients assigned to a facility, while (39) is an upper bound on the travel-time of a single client from the facility to which it is assigned. Constraint (40) is the relaxation of the sub-problem which defines the minimum number of vehicles assigned to each site. In constraints (38)-(40) the maximum travel-time of the each vehicle has been increased by $\frac{\alpha u}{pen}$, since, if the maximum allowable penalty of a vehicle is αu, and the per time unit penalty is pen, then the truck is

allowed to travel an extra $\frac{\alpha u}{pen}$ time units. A detailed description of the cuts is presented in Section 4.4.4.1.

4.3.2 The Expected Value Truck Assignment Sub-problems

Given the set of clients allocated (I_j) and the number of vehicles assigned to an open facility ($numVeh_j$), the goal of the EVTASP is to use the expected value of the travel-times to assign clients to the vehicles of each site such that the vehicle travel-time constraints are satisfied. The EVTASP for each facility can be modeled as a bin-packing problem. It should be noted that the EVTASP is an extensions of the TASP presented in Section 3.3.2.

A CP formulation of the EVTASP is as follows:

$$\begin{aligned}
\min \quad & numVehBinPacking_j \\
\text{s.t.} \quad & pack(load, truck, \lfloor \overline{t_{ij}} \rfloor) && (44) \\
& numVeh_j \leq numVehBinPacking_j && (45) \\
& cuts && (46)
\end{aligned}$$

where $load$ is an array of variables such that $load_k \in \{0, ..., (l + \frac{\alpha u}{pen})\}$ is the total travel-time assigned to vehicle $k \in \{1, ..., numVehBinPacking_j\}$, $truck$ is an array of decision variables, one for each client $i \in I_j$, such that $truck_i \in \{1, ..., numVeh_j\}$ is the index of the truck assigned to client i, and $\lfloor \overline{t_{ij}} \rfloor$ is the vector of the truncated expected travel-times between site j and client $i \in I_j$. The truncated expected travel-times are used since in the pack global constraint the weights can only be integer. Use of the truncated expected travel-times instead of the expected travel-times does not affect the correctness of the model, since, the expected travel-times are used as a relaxation, and using the truncated expected travel-times relaxes the problem further. The pack global constraint (44) maintains the load of the vehicles given the expected travel-times and assignments of clients to vehicles [93]. The lower bound on the number of vehicles is represented by constraint (45). The cuts for the EVTASPs will be described in Section 4.4.2.

In practice, we solve a series of satisfaction problems using the CP formulation, setting $numVehBinPacking_j$ to each value in the interval $[numVeh_j..\overline{k}]$ in increasing order. Informally, we try to pack the customers in $numVeh_j$ trucks; if a feasible solution is found, we send the solution to the scenario sub-problems, otherwise we increase the number of trucks by one and again try to pack the customers into the trucks. We continue adding trucks and packing customers until a feasible solution is found or until $numVehBinPacking_j = \overline{k}$. It should be noted that unlike the TASP solution procedure (see Figure 3.3), we can no longer use the FFD heuristic due to the EVTASP cuts.

4.3.3 The Scenario Sub-problems

Given the set of clients allocated to vehicle k, $I_k = \{i \mid truck_i = k\}$, and the set of vehicles allocated to facility j, $K_j = \{k \mid load_k > 0\}$, the SSP calculates the recourse cost of each vehicle in scenario s.

Formally, the recourse cost of scenario s is:

$$\Pi_k^s = [pen \times (\sum_{i \in I_k} (t_{ij}^s) - l)]^+, \quad k \in K_j$$

After all the SSPs have been solved, the expected recourse cost of each vehicle over all the scenarios is calculated,

$$E(\Pi_k^s) = \sum_{s \in S} P_s \Pi_k^s, \quad k \in K_j$$

Then a set of feasibility checks, one for every active vehicle, is done to see whether any of the expected recourse cost constraints are violated, thus for each active vehicle we check the following condition:

$$E(\Pi_k^s) \leq \alpha u, \quad \forall k \in K_j$$

If, for all vehicles, the condition is satisfied, the EVTASP solution does not violate any of the expected recourse cost constraints. If one or more vehicles does not satisfy the condition, the EVTASP solution has violated at least one of the expected recourse cost constraints, thus, a cut is sent back to the EVTASP which prohibits at least the current optimal solution of the EVTASP.

4.3.4 Benders Cuts

In this section we describe the cuts for the EVLAMP and the EVTASP.

4.3.4.1 The EVLAMP Cuts

The Benders cuts added to the EVLAMP, remove the current optimal solution of the EVLAMP because it does not result in a feasible solution to one or more of the EVTASPs. These cuts are identical to the cuts proposed for CDCPLP in Section 3.3.3. The cuts after iteration h are:

$$numVeh_j \geq numVeh_{jh}^* - \sum_{i \in I_{jh}} (1 - x_{ij}), \quad j \in J_h,$$

where $I_{jh} = \{i \mid x_{ij}^h = 1\}$ is the set of customers assigned to facility j in iteration h, J_h is the set of sites for which the EVTASP is infeasible in iteration h, and $numVeh_{jh}^*$ is the minimum number of vehicles needed at site j to serve the clients that were assigned. The cut specifies that if the same set or a superset of customers are again assigned to facility j, then the number of trucks must be greater than or equal to $numVeh_{jh}^*$. For a more detailed explanation of the cut, the reader is referred to Section 3.3.3.

Theorem 3. The EVLAMP cut is a valid cut.

Proof. Since these cuts are identical to the cuts presented Section 3.3.3., the same proof applies here.

□

4.3.4.2 The EVTASP Cuts

Assume that in one particular iteration, the solution to the EVTASP assigns a set, Q, of clients to vehicle k of facility j. Assume further that there is no feasible solution with regards to the expected recourse cost for vehicle k. The cut generated specifies a new lower bound on the expected recourse cost of vehicle k. At the least, this new lower bound prevents the clients in Q or a superset of them from being assigned to vehicle k in future iterations.

The cuts after iteration h of the EVTASP are:

$$\sum_{s}(P_s \times [\Pi_k^{sh} - pen \times \sum_{i \in I_k^h} t_{ij}^s \times (truck_i \neq truck_i^h)]) \leq \alpha u, \quad k \in K_{jh}, \forall \Pi_k^{sh} > 0,$$

where $truck_i^h$ is the truck that customer i was assigned to in iteration h, $I_k^h = \{i \mid truck_i^h = k\}$ is the set of clients assigned to vehicle k in iteration h, K_{jh} is the set of vehicles $k \in K_{jh} = \{k \mid load_k > 0\}$ is the set of vehicles used at facility j in iteration h, and Π_k^{sh} is the recourse cost of vehicle k in scenario s in iteration h. The term $(truck_i \neq truck_i^h)$ is a boolean expression equal to 1, if client i is assigned to $truck_i^h$, and 0 otherwise. The inner summation term is an upper bound on the decrease in the recourse cost of the corresponding vehicle, given that the clients are reassigned to other facilities: the maximum amount of reduction of the recourse cost of scenario s in reassigning one client is pen times the travel-time of that client in scenario s.

Since each facility houses a homogeneous set of vehicles, a cut for every vehicle up to $\overline{K}$, $truck_i^h = k, k \in \overline{K}$, is added to the EVTASP. It should be noted that the EVTASP cuts are stored and used again in future iterations.

Example 2 Consider the network with the two scenarios shown in Figure 4.5. The network consists of one facility and four customers. Each scenario has probability 0.5. The maximum total driving-time, $l = 50$, and vehicle utilization cost $u = 50$, the per unit penalty cost, $pen =$

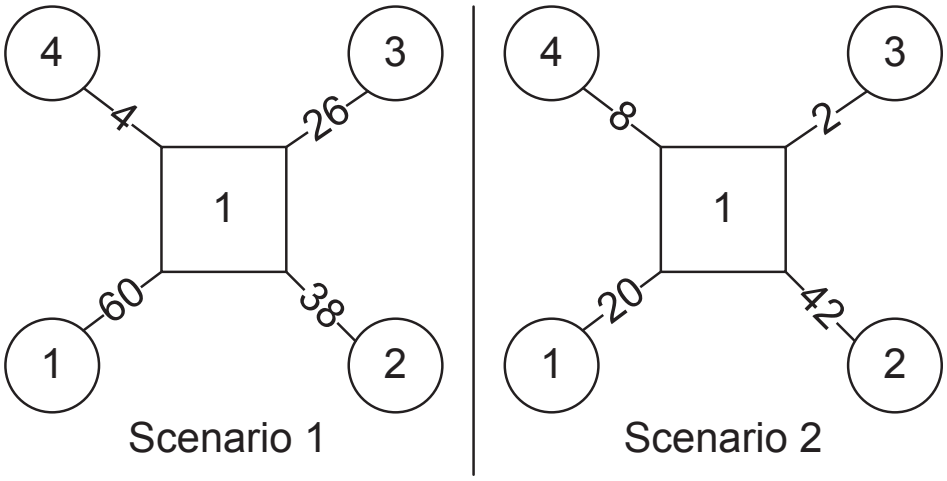

Figure 4.5: Example 2

2, and $\alpha = 0.1$. From the figure, the expected travel-time for the customers over all scenarios are $\overline{t_{11}} = 40, \overline{t_{21}} = 40, \overline{t_{31}} = 14, \overline{t_{41}} = 6$. The EVLAMP solution indicates that all the customers are assigned to facility one, thus, $x_{11} = 1, x_{21} = 1, x_{31} = 1, x_{41} = 1$, and that two trucks are needed at the facility, $numVeh_1 = 2$. This solution is then sent to $EVTASP_1$ to assign customers to trucks. $EVTASP_1$ indicates that 3 trucks are needed to serve the assigned customers, and assigns the customers to truck as follows: customer 1 and 4 to the first truck, $truck_1 = 1$ & $truck_4 = 1$, customer 2 to the second truck, $truck_2 = 2$, and customer 3 to third truck, $truck_3 = 3$. This solution is then sent to the SSPs to test the feasibility with respect to the expected recourse cost constraints. The SSP solutions indicate: $E(\Pi_{11})^1 = 7, E(\Pi_{21})^1 = 0, E(\Pi_{31})^1 = 0$. Since $E(\Pi_{11})^1 > \alpha u$, the following cut is added to the EVTASP:

$$\begin{aligned} 0.5 \times [\Pi_{11}^{11} - 2 \times (\textstyle\sum_{i=1,4} t_{ij}^1 \times (truck_i \neq 1))] \leq 5 \qquad j = 1 \\ = 0.5 \times [14 - 2 \times (t_{11}^1 \times (truck_1 \neq 1) + t_{41}^1 \times (truck_4 \neq 1))] \leq 5 \quad (47) \end{aligned}$$

The cut prohibits the reassignment of customers 1 and 4 to the same vehicle, since by reassigning them to the same vehicle, $truck_1 = truck_2$, (47) is reduced to $E(\Pi_{11})^1$, which we know from the SSPs is greater than αu. The EVTASP is then re-solved, which results in the following solution: $truck_1 = 1, truck_2 = 2, truck_3 = 3, truck_4 = 2$. This solution is send to the SSPs, and the expected recourse cost are calculated, $E(\Pi_{11})^2 = 10, E(\Pi_{21})^2 = 0, E(\Pi_{31})^2 = 0$, which indicates that no recourse constraint is violated. Thus, the minimum number of trucks needed at the facility is 3, $numVeh_{11}^* = 3$. Since this solution is not equal to the EVLAMP solution, $numVeh_1 = 2$, the following cut is added to the EVLAMP and the EVLAMP is re-solved,

$$numVeh_j \geq 3 - \textstyle\sum_{i=1\ldots4}(1 - x_{ij}), \quad j = 1 \quad (48)$$

The same procedure is repeated until all EVTASPs are feasible.

Theorem 4. The EVTASP cut is a valid cut.

Proof. We need to show that the two properties presented in Section 4.2.3 are satisfied.

Recall that I_k^h is the set of clients assigned to vehicle k in iteration h, and $E(\Pi_k)^h$ is the expected recourse cost of vehicle k in iteration h of the EVTASP model. If the EVTASP solution is infeasible, then:

$$E(\Pi_k)^h > \alpha u \quad (49)$$

which will result in the cut:

$$\sum_S (P_s \times [\Pi_k^{sh} - pen \times (\sum_{i \in I_k^h} (t_{ij}^s \times (truck_i \neq truck_i^h)))]) \leq \alpha u \quad (50)$$

If the same set of customers is re-assigned to vehicle k, $\sum_{i \in I_k^h} (truck_i \neq truck_i^h) = 0$. This will result in the left hand side of (50) to be equal to $E(\Pi_k)^h$, which we know from (49) is greater than αu. Thus, the cut satisfies property 1.

In order to show property 2, let F be a feasible solution of the EVTASP found in iteration $f > h$ of the corresponding EVTASP. Since F is a feasible:

$$E(\Pi_k)^f \leq \alpha u \quad (51)$$

We present a proof by contradiction and so assume that F does not satisfy the cut. Consider I_k^f to be the set of clients assigned to vehicle k in iteration f. We define three sets:

$\theta_1 = \{I_k^h \backslash I_k^f\}$: customers in I_k^h not in I_k^f.
$\theta_2 = \{I_k^f \backslash I_k^h\}$: customers in I_k^f not in I_k^h.
$\theta_3 = \{I_k^h \cap I_k^f\}$: customers in both I_k^h and I_k^f.

Since we assumed that F does not satisfy the cut, then:

$$\sum_S (P_s \times [\Pi_k^{sh} - pen \times (\sum_{i \in \theta_1} t_{ij}^s)]) > \alpha u \quad (52)$$

Expressions (51) and (52) match expressions (29) and (30), therefore, the proof of Theorem 2 applies here. Thus, the cut satisfies property 2.

□

4.4. Computational Experiments

We compare the performance of the three model based on two experiments: scaling with scenarios and scaling with size. We first present the experimental set up. Then an in-depth analysis of the results for both experiments is presented.

4.4.1 Experimental Set Up

The problem instances for both experiments were generated as follows: the facility capacities are randomly drawn from an integer interval $b_j \in [50, 200]$; the fixed facility opening costs, f_j, are randomly generated based on the capacities b_j and are calculated as: $f_j = (b_j \times (10 + rand[1, 15]))$, where 10 is the per unit capacity cost. We have added a random multiplier from [1,15] to take into account difference in property cost. The maximum number of vehicles takes the value $\overline{k} = |I|/2$. The travel-times for each scenario are randomly drawn from an integer interval $t^s_{ij} \in [10, 60]$. The probability of each scenario is generated one by one, but we put an upper and a lower bound on the interval of the probability that each scenario can have so that the probability of a single scenario cannot either be too big nor too small. For 2 scenarios the probabilities are chosen from a random interval [0.01,0.99], for 4 scenarios from [0.01,0.4], for 8 scenarios from [0.01,0.25], for 16 scenarios from [0.01,0.09] and for 32 scenarios [0.01,0.05]. In all cases, the probability of the last scenario is one minus the sum of all the others and typically falls within the same range. The costs, c_{ij}, are randomly generated based on the expected travel-time over all scenarios, $c_{ij} = scale(\overline{t_{ij}}, [10, 90]) + rand[-5, 5]$. The first term is a uniform scaling of the expected travel-times to the integer interval $[10, 90]$, while the second term is a random integer uniformly generated on the interval $[-5, 5]$.

The tests were performed on a Dual Core AMD 270 CPU with 1 MB cache, 4 GB of main memory, running Red Hat Enterprise Linux 4. The IP, the two-level Benders, and the EVLAMP models are implemented in ILOG CPLEX 11.0, and the EVTASP is implemented in ILOG Solver 6.5. In both experiments, four hours was used as a maximum time limit. For unsolved instances, that time-limit was used in the calculations.

4.4.2 Experiment I: Scaling with Scenarios

In Experiment I we start with the 5 instances of size 20×10 (i.e., 20 clients, 10 possible facility sites). Eight different pairs of truck distance limit, l, and truck usage cost, u, values are then used to create different problem sets (l, u): $(60, 60)$, $(80, 80)$, $(80, 150)$, $(100, 100)$, $(100, 200)$, $(120, 120)$, $(120, 240)$, $(140, 280)$. In this experiment the per unit penalty cost is 5% of the truck usage cost, $pen = 0.05 \times u$. Four different $\alpha = \{0.1, 0.2, 0.3, 0.4\}$ are used to evaluate the effects of α on performance of the approaches. The number of scenarios takes the values: 2, 4, 8, 16, and 32. Overall, therefore, there are 800 problems instances: 5 original instances times 8 (l, u) conditions times 4 α values times 5 different number of scenarios.

Figure 4.6 shows the mean run-times over all instances for the IP, the two-level and the three-level Benders models. The run time of the IP model increases significantly with the number of scenarios, flattening only when a large proportion of the instances time-out. For the two-level Benders model there is a large increase in the CPU time as we go from 2 to 4 scenarios, while from 4 to 16 scenarios the number of scenarios does not seem to have an affect on the run-time. An interesting observation is that for 32 scenarios the two-level run-time decreases. The run time of the three-level Benders model increases with the number of scenarios, however, this increase is substantially less than the IP models increase. It can be seen that the break even point of the IP and the two-level Benders curves is 8 scenarios, while the break even point of the two Benders curves is 16 scenarios. Thus, the results suggest that for problems with 16 scenarios or fewer the performance of the three-level Benders is better

than the other two models, while for problems with large number of scenarios the two-level Benders models performance is superior.

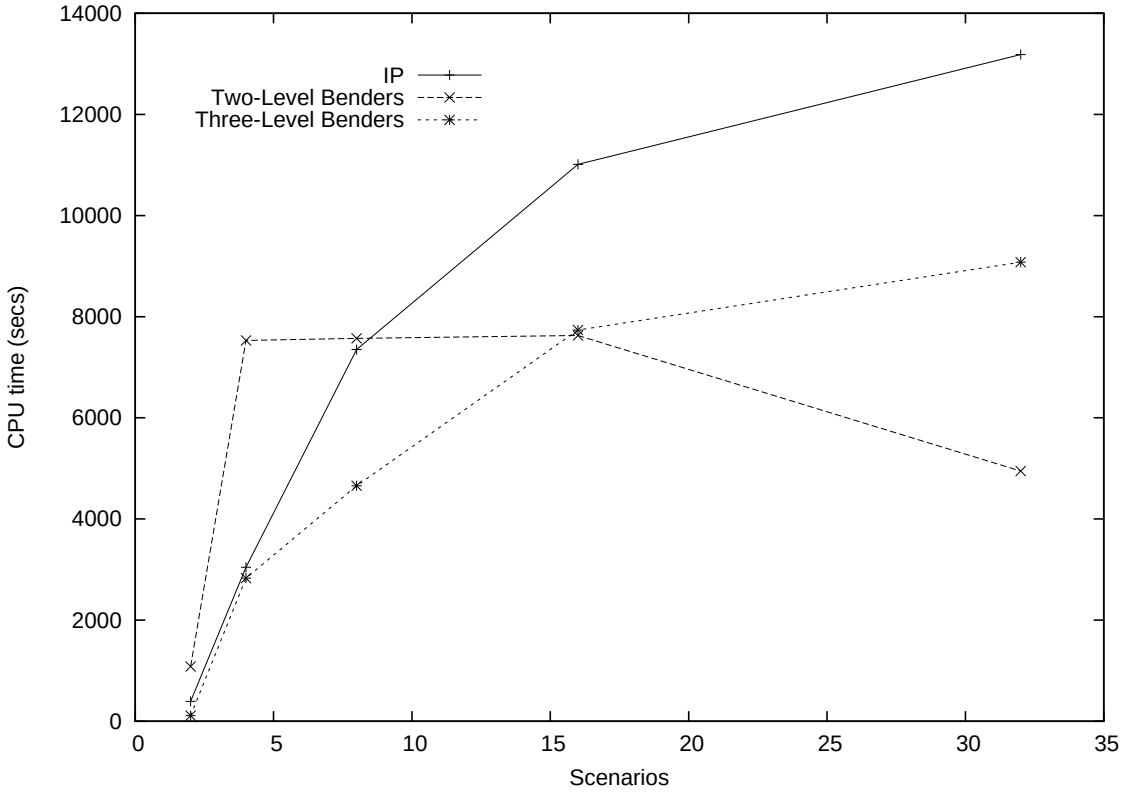

Figure 4.6: Run-time of IP model and the two Benders models

Table 4.1 compares the mean CPU time in seconds required to solve each problem instance for each scenario. The columns labeled "% Uns." indicate the percentage of unsolved problems. The three-level Benders approach is unable to find the optimal solution within the time limit in 30.5% of instances, while the two-level Benders model is unable to find the optimal solution in 32.4% of the instances, and the IP model is unable to find the optimal solution in 43.4% of the instances. The results show that the two-level Benders approach finds the optimal solution (or times-out) on average in 28.8 iterations, while the three-level Benders approach finds the optimal solution in 193.8 iterations (the mean number of iterations for the three-level model indicates the number of times the EVLAMP has been solved). Thus, three-level Benders iterates about 7 times more than the two-level model. This is due to the fact that the master problems relaxations are much looser, but much easier to solve. In other words, as we increase the number of levels of the decomposition approach, we push more information from the master problem to the sub-problems. Therefore, the more we decompose, more information is push out from the master problem to the sub-problems, making the master problems easier to solve but not as tight.

Number of Scenarios	IP		Two-Level Benders			Three-Level Benders		
	Time	% Uns.	Time	% Uns.	Iter	Time	% Uns.	Iter
2	393	1.3	1081	1.9	11.4	110	**0.6**	13.7
4	3044	13.8	7530	43.1	33.9	2929	**13.1**	205.2
8	7351	43.1	7570	43.8	31.1	4656	**28.1**	200.1
16	11014	70.0	7628	**43.8**	34.7	7735	49.4	271.8
32	13185	88.8	4946	**29.4**	32.9	9077	61.3	278.0
Overall	6997	43.4	5751	32.4	28.8	4881	**30.5**	193.8

Table 4.1: The mean CPU time (seconds) and percentage of unsolved problem instances (% Uns.) for the IP and the two Benders approaches, and the mean number of iterations the Benders models. Overall indicates the mean results over all problem instances. For each number of scenarios, the technique with the lowest % Uns is highlighted (bold).

Table 4.2 exhibits the mean and median Time Ratios of the three models, where time-ratio for a given instance is calculated as the IP run-time divided by the two-level Benders run-time, the IP run-time divided by the three-level Benders run-time, and the two-level Benders run-time divided by the three-level Benders run-time. The mean and median over each instance in each subset was then calculated. It can be seen that the three-level Benders method is on average 1306 times faster than IP, and 272.9 time faster than the two-level Benders model. The results also show that the two-level Benders model is 58 time faster than IP. The advantage of both Benders model over IP increases with the number of scenarios. The time-ratio of the three-level Benders model over the two-level model decreases as the number of scenarios increase. Although for problems with smaller number of scenarios (i.e. 2 or 4 scenarios) the mean time ratio of IP over two-level Benders shows that the two-level Bender approach is faster, the medians show that for the majority of these instance the IP is finding the solution faster.

Number of Scenarios	IP vs. Two-Level		IP vs. Three-Level		Two-Level vs. Three-Level	
	Mean	Median	Mean	Median	Mean	Median
2	1.4	0.7	195.8	102.3	322.4	111.9
4	1.8	0.18	114.5	9.7	227.9	43.2
8	7.3	1	608.8	8.8	528.5	13.7
16	16.7	1	575.8	1	173.5	1
32	262.2	27.7	5037.0	1	112.4	1
Overall	57.9	1	1306.4	10.0	272.9	18.5

Table 4.2: The time-ratio for the IP and the two Benders approaches.

Figures 4.7, 4.8, and 4.9 show three scatter-plots of the run-times of each problem instance for the three approaches. The x and y axes are log-scale and the points below the $x = y$ line indicate lower run-time for the approaches that are on the y-axes. Over all the instances, the three-level Benders performs better than the other two models in 61% of the problem instances, while the two-level Benders is best in 17% of the instances and the IP is best in only 9% of the instances. In Figures 4.7, 4.8, 4.9 the points (instances) in which the two-levels Benders is best mostly represent problems with 16 and 32 scenarios, while for the majority of the points that represent problems with 2, 4 and 8 scenarios the three-level Benders has the lowest CPU time. Table 4.3 shows the percentage of the problem instances for different α value for which one approach was dominant, where dominance is defined as finding the solution in less run time. Table 4.3 suggests that as α grows the percentage of dominant solutions of the IP model decreases, while the percentage of the three-level Benders approach increases. Since the two-level Benders approach is dominant for most of the problems with 16 and 32 scenarios, its percentage of dominant solution is steady and almost the same for the different values of α. The column labeled three-way tie shows the percentage of instances in which the three models had the same run-time. It should be noted that for the majority of the three-way tie instances, the three models timed-out. By observing Table 4.3, we can see that α has a significant affect on the performance of the approaches.

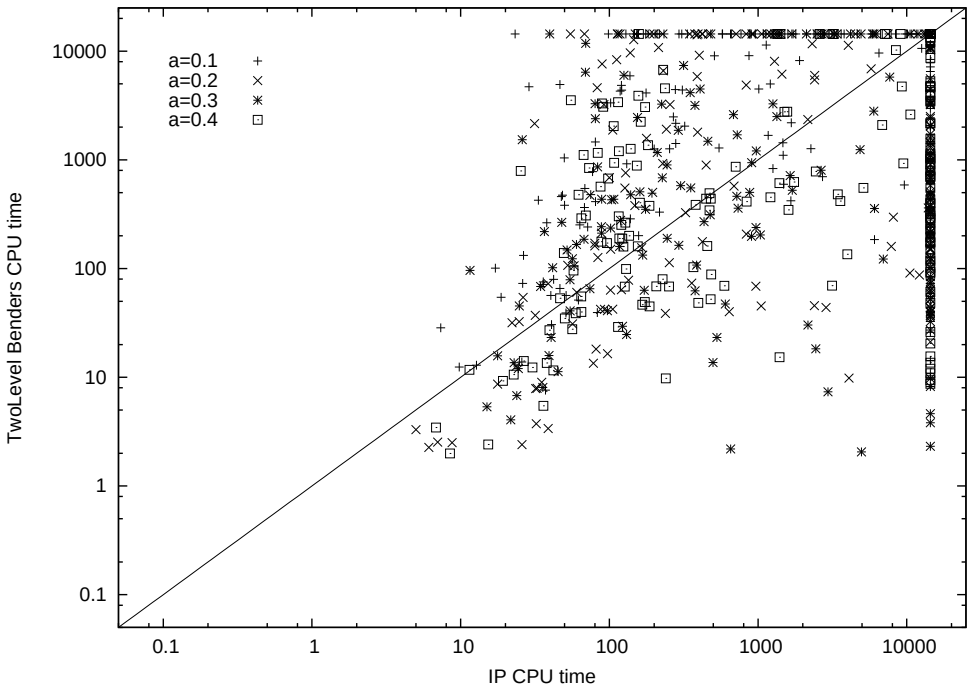

Figure 4.7: Run-time of IP model and two-level Benders model for different α values

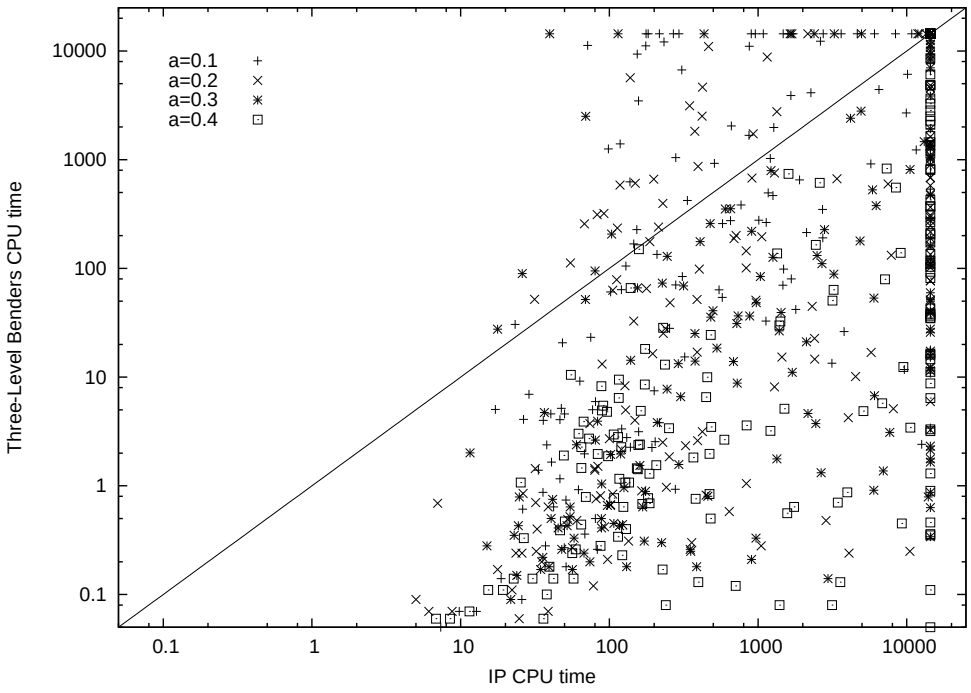

Figure 4.8: Run-time of IP model and Three-Level Benders model for different α values

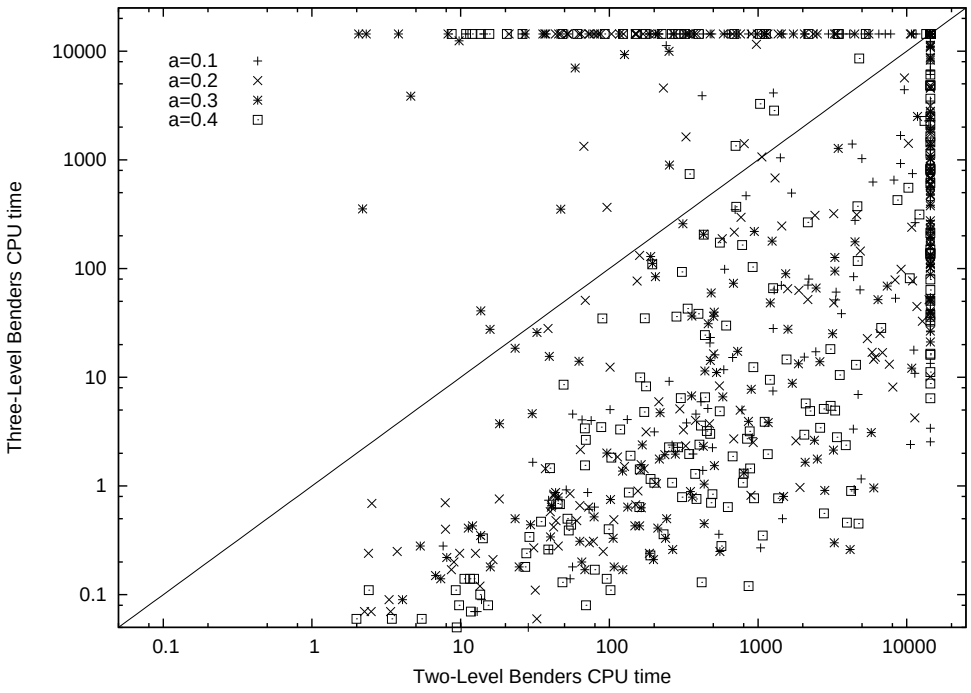

Figure 4.9: Run-time of two-level model and Three-Level Benders model for different α values

α	IP	Two-Level Benders	Three-Level Benders	Three-Way Tie
0.1	19	17	46	19
0.2	12	21	53	15
0.3	5	17	68	11
0.4	0	16	77	8
Overall	9	17	61	13

Table 4.3: The percentage of dominant problem instances for the IP, the two-level and the three-level Benders approaches, and the percentage of instances in which the three models had the same run-time.

The Impact of α In the previous section we observed that α had a significant affect on the performance of the different approaches. Thus, in this section we present an in-depth analysis of the impact of α. Recall from Section 4.1, that αu is the maximum allowable penalty of each truck. Figure 4.10 shows the mean run-time of the three models for different α values. The value of α does not seem have a major affect on the performance of the IP model with respect to the CPU time, while the CPU time of both Benders models decreases substantially as α increases.

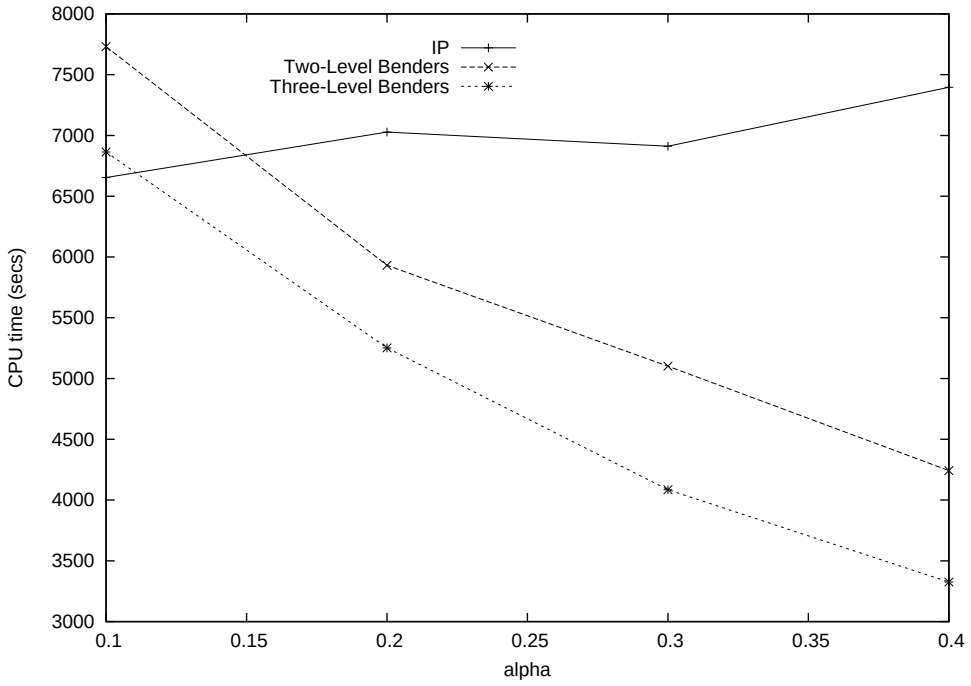

Figure 4.10: Run-time of the three model for different a values.

Table 4.4 compares the mean CPU time in seconds, and the percentage of unsolved problems for different α values. As can be seen the value of α does not have a major affect on the performance of the IP model. This is due to the fact that regardless of what value α might have, the problems with a large number of scenarios are too large for the IP model to find and prove the optimal solution within the time limit. The table also suggests that as α increases, the run times and percentage of unsolved instances of both Benders models decrease. We believe this decrease is due to the fact that as α increases, our relaxation using the expected travel-times in the EVMP and EVLAMP is tighter, thus, both approaches are able to find the optimal solution in fewer iterations. The reason why the relaxation is better as α increases is that by increasing α, the right hand side of the recourse cost constraints, (3), gets bigger, while the left hand side does not change. In other words, for bigger α values, each truck has more flexibility, therefore, there is a smaller chance that the solutions of the EVMP and EVLAMP violate the recourse cost constraints.

	IP		Two-Level Benders			Three-Level Benders		
α	Time	% Uns.	Time	% Uns.	Iter	Time	% Uns.	Iter
0.1	6653	40	7731	44	39.7	6864	43	192.6
0.2	7028	45	5931	33	26.6	5252	33	166.2
0.3	6912	42	5101	30	24.9	4084	25	267.3
0.4	7396	47	4242	23	23.9	3325	21	149.1

Table 4.4: The mean CPU time (seconds) and percentage of unsolved problem instances (% Uns.) for the IP and Benders approaches and for the Benders approach, the mean number of iterations.

Figure 4.11, 4.12, 4.13, and 4.14 show the mean run-time of the three models for different α values. The IP curve shows the same trend for the different α values. As mentioned above, the performance of both Benders models improve as α increases. The three-level Benders curve straightens as α increases, and for $\alpha = 0.4$ there seems to be almost a linear relationship between the CPU time and the number of scenarios. For large number of scenarios (i.e. 32 scenarios), and large α values, the two-level Benders model shows an absolute and relative speed-up. We believe this can be explained as follows: for problems with a large number of scenarios and large α values our relaxation using the expected travel-times in the master problem is tighter, thus, Benders is able to find the optimal solution in fewer iterations. The tighter relaxation is likely due to the fact that as the number of scenarios increases, the probability of each scenario decreases, thus, each scenario has less power to violate the expected recourse cost constraints on its own. The results for the two-level Benders also show that for smaller α value, the superiority with respect to the CPU time of problems with a large number of scenarios begins to disappear and for $\alpha = 0.1$ it disappears entirely: the CPU time seems to be almost the same for the different number of scenarios.

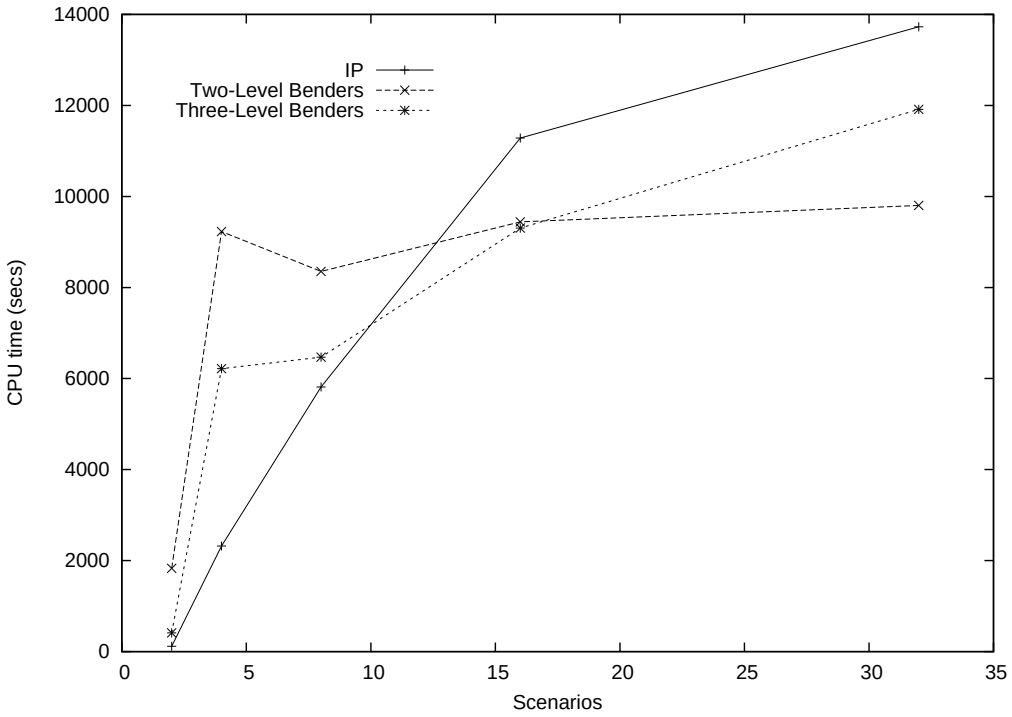

Figure 4.11: Run-time of the three model for different $\alpha = 0.1$.

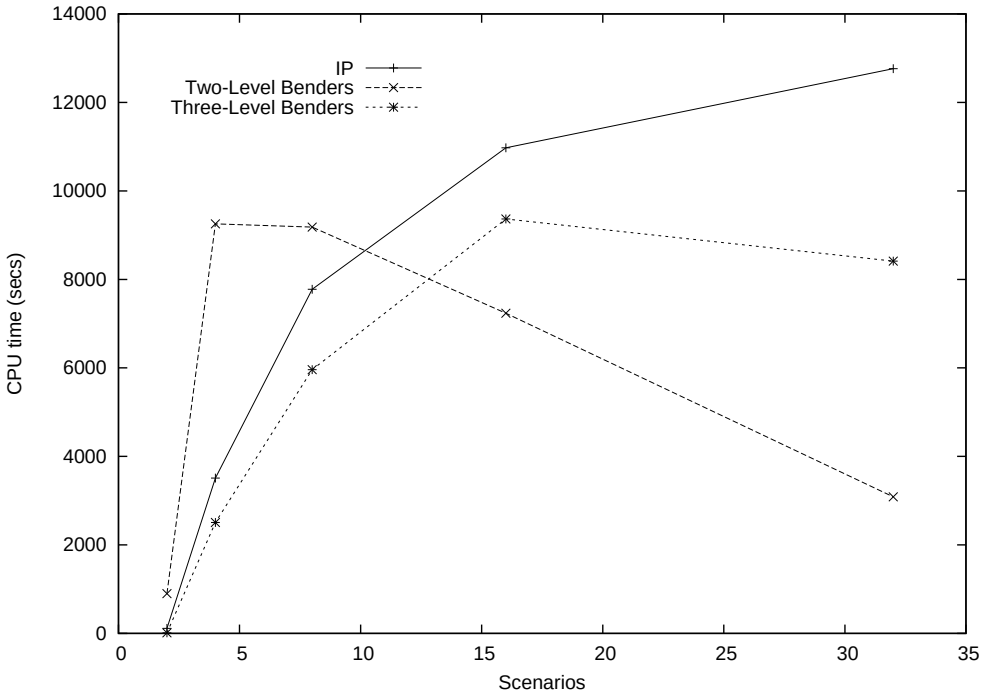

Figure 4.12: Run-time of the three model for different $\alpha = 0.2$.

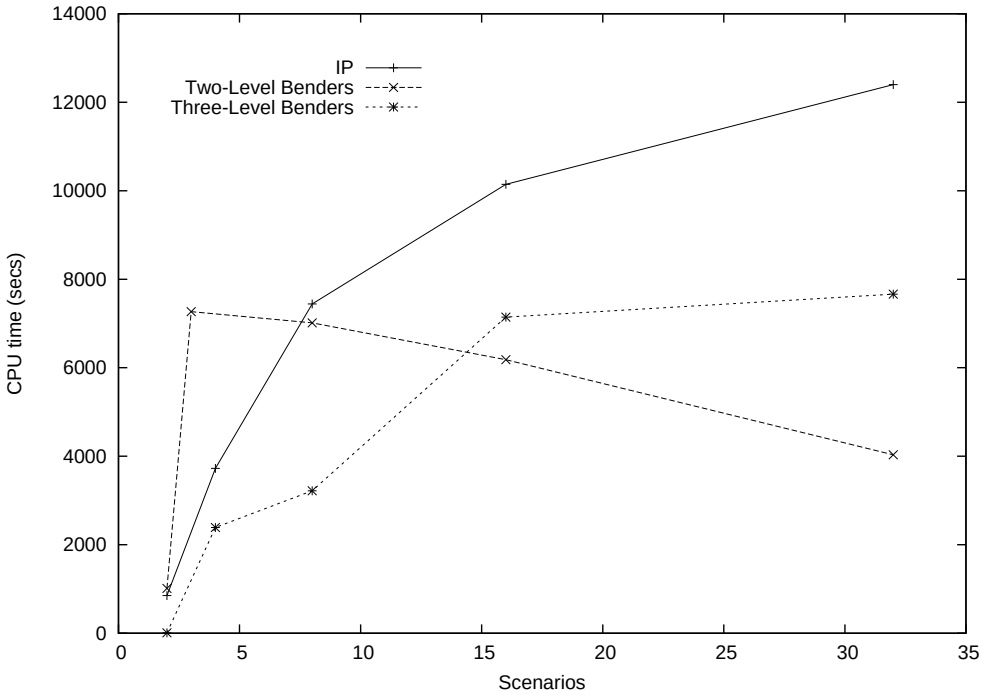

Figure 4.13: Run-time of the three model for different $\alpha = 0.3$.

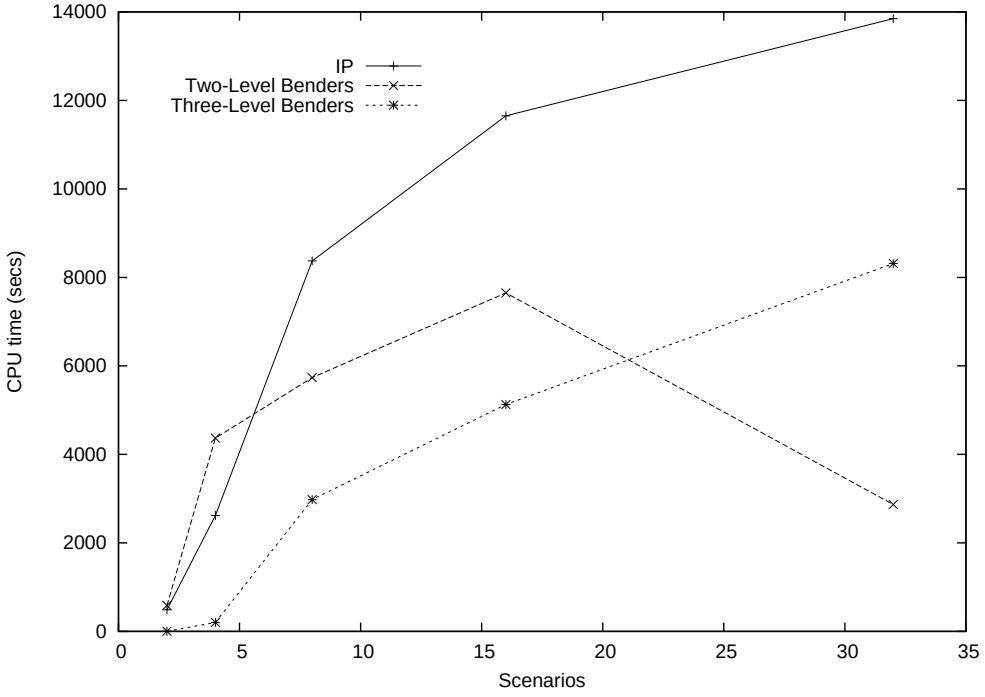

Figure 4.14: Run-time of the three model for different $\alpha = 0.4$.

In sum, it is both the increase in the number of scenarios, which reduces the power of each scenario, and the increase in α, which loosens the recourse constraints, that contributes to the improvement in both the two-level and the three-level Benders models.

The Impact of the Vehicle Travel-Times Limits Figure 4.15 exhibits the performance of the different models for different vehicle travel-times limits. The CPU time of both Benders models is quite sensitive to the characteristics of the vehicles. On one hand, the two-level Benders solves the instances with tight travel-time constraints on the vehicle much faster than those with very loose constraint. On the other hand, the IP and the three-level Benders model show opposite performance.

Table 4.5 demonstrates that as the travel-time limit on the vehicle increase, the percentage of unsolved instances of the IP and the three-level Benders models decreases, while, the percentage of unsolved instances increases for the two-level Benders model. For the two-level Benders the average number of iterations increases from 17 to 31, while for the three-level model the average number of iterations decreases from 145 to 96. It can also be seen that there is a positive correlation between l and the time-ratio of the two Benders model.

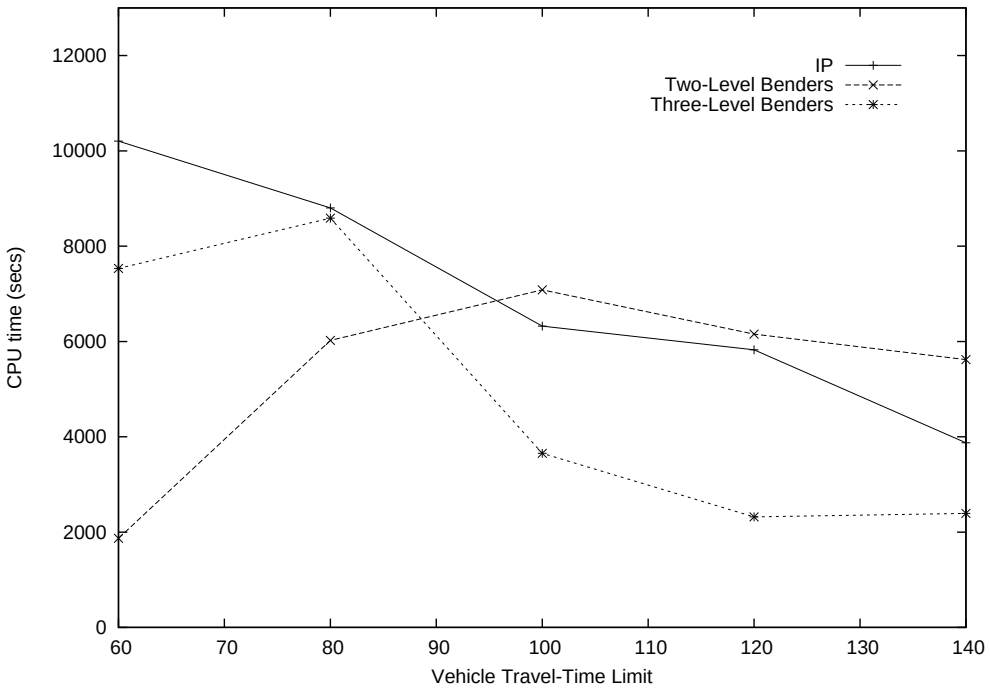

Figure 4.15: Run-time of the three model for different travel-limits.

Travel-Limit (l)	IP		Two-Level			Three-Level			IP vs. Two-Level	IP vs. Three-Level	Two-Level vs. Three-Level
	Time	% Uns.	Time	% Uns.	Iter	Time	% Uns.	Iter			
60	10203	68	1869	4	17	7533	49	145	67	76	47
80	8802	57	6023	35	17	8588	55	397	158	145	116
100	6323	39	7083	44	35	3655	23	151	20	2399	292
120	5824	33	6155	34	33	2320	12	107	18	1178	390
140	3874	23	5619	31	31	2394	15	96	4	2931	540

Table 4.5: The mean CPU time (seconds) and percentage of unsolved problem instances (% Uns.) for the IP and Benders approaches; the mean number of iterations for the Benders approaches; and the time-ratio of the three approaches (the last 3 columns).

The reason why we see this trend is that for the two-level Benders approach, as the travel-time limit decreases, fewer customers can be assigned to single vehicle, thus, there is a smaller chance that the recourse limit of the truck is violated. For the three-level Benders approach, however, as the travel limit decreases, the relaxations used in the EVLAMP (constraint (30)) become looser, therefore, more iterations are need to find the optimal solution.

4.4.3 Experiment II: Scaling with Size

Experiment II consists of 90 instances in nine sizes: ten instances each of size 10×5, 20×5, 30×5, 20×10, 30×10, 40×10, 20×15, 30×15, and 40×15. Each size contains 5 instances with 4 scenarios and 5 instances with 16 scenarios.

Figure 4.16 shows the percentage of unsolved instances for the different sizes. As can be seen the percentage of unsolved instances of the three models increases as the size of the problem increases. There are no bars for the percentage of unsolved instances of the three approaches for problem size 10×5, and the percentage of unsolved instances of the two-level

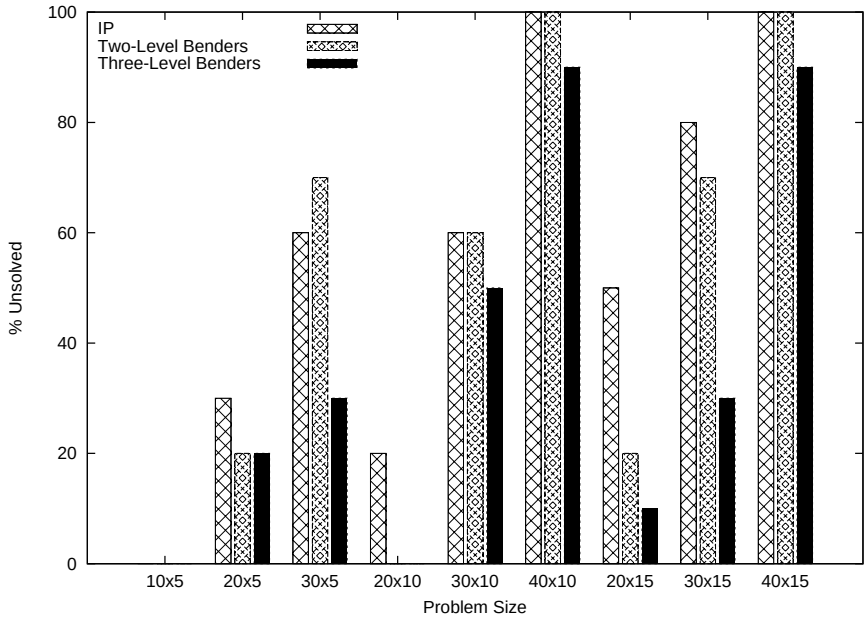

Figure 4.16: Percentage of unsolved problem instances of the three models for the different problem sizes.

and the three-level Benders approaches for problem size 20×10 in Figure 4.16, since their percentages are zero.

Table 4.6 compares the mean CPU time in seconds required to solve the problem instances for each size, and the percentage of unsolved problems. The overall results show that the percentage of unsolved instances for the three-level Benders is 35%, for the two-level Benders is 49%, and for the IP model is 56%. It can be seen that for both Benders approaches, the number of unsolved problems increase as the problem size increases. This trend is due to the fact that for larger instances, the master problems themselves become very challenging to solve. It can also be seen that as the problem size increases more iterations are needed to find the optimal solution, which suggests that for larger instances the solution of EVMP and EVLAMP are farther from the global optimum. The number of iterations decrease for 30×5, 40×10, and 40×15 instances because most/all of these instances timed-out, thus we used the number of iterations at the time limit in the calculations.

Problem Size	Number of Scenarios	IP		Two-Level Benders			Three-Level Benders		
		Time	% Uns.	Time	% Uns.	Iter	Time	% Uns.	Iter
10×5	4	0.8	0	2.1	0	9.0	0.2	0	6.2
	16	4.5	0	4.0	0	12.8	0.4	0	8.8
20×5	4	38.6	0	2654.9	0	50.6	2.8	0	17.0
	16	9086.6	60	6842.1	40	92.8	5801.1	40	786.0
30×5	4	6378.2	40	9474.5	60	26.2	3264.2	20	84.8
	16	11520.2	80	11520	80	27.2	7133.8	40	131.6
20×10	4	172.9	0	2304.9	0	13.0	34.7	0	13.0
	16	7467.5	40	3597.8	0	43.0	56.9	0	22.6
30×10	4	3848.6	20	8864.2	40	39.2	4305.8	20	63.0
	16	>14400	100	12755.6	80	42.0	11520.1	80	153.6
40×10	4	>14400	100	>14400	100	4.6	13797.5	80	59.0
	16	>14400	100	>14400	100	4.6	>14400	100	81.6
20×15	4	169.0	0	2215.9	0	12.6	6.0	0	5.0
	16	>14400	100	8166.0	40	63.8	5147.9	20	218.0
30×15	4	8844.1	60	8843.4	60	8.8	4638.3	20	63.6
	16	>14400	100	12337.6	80	41.0	8248.8	40	160.6
40×15	4	>14400	100	>14400	100	4.4	11533.7	80	56.8
	16	>14400	100	>14400	100	3.4	>14400	100	56.8
Overall		8240.6	56	8176.8	49	27.7	5794.0	35	112.0

Table 4.6: The mean CPU time (seconds) and percentage of unsolved problem instances (% Uns.) for the IP and Benders approaches; the mean number of iterations for the Benders approaches.

4.5. Discussion

The contributions of this chapter are as follows:

- While logic-based Benders decomposition has been applied to stochastic integer programs [73], there does not appear to have been published work that follows our approach of defining the decomposition such that each sub-problem corresponds to one scenario. However, the idea has been considered and, though not explicitly stated in the text, some of the empirical results in [58] are based on such an approach.[1]

[1] John Hooker, personal communication.

- We develop and compare the performance of a two-level and a three-level logic-based Benders decomposition approaches for a stochastic programming problem. The results demonstrate that it is not always true that the more we decompose the problem the faster we can find and prove the optimal solution. For example in the SFLVAP, for problems with a large number of scenarios, or problems with small travel-time limits, the three-level Benders model is worse than the two-level Benders model. This is due to the fact that as we decompose the problem, the master problem relaxations become looser, and the information is pushed out of the master problem and distributed among more sub-problems. This increase in the distribution of information may slow down the model as the number of levels increase.

In terms of generality of logic-based Benders decomposition for two-stage stochastic programming, in cases where the expected values of the random variables cannot be used in the master problem, the minimum (or maximum, depending on the objective function) of the random variables could be used.

4.6. Conclusion

In this chapter, we presented a two-level and a three-level logic-based Benders decomposition approach to a stochastic facility location and vehicle assignment problem. Our Benders approaches were able to substantially out-perform an IP model by finding and proving optimality up to three orders-of-magnitude faster.

Chapter 5

Conclusion

In this final chapter, we summarize the work presented in previous chapters, re-state the major contributions of this dissertation, and present some possible directions for future work.

5.1. Summary and Contributions

The Application of Logic-Based Benders Decomposition to Facility Location/Fleet Management Problems In Chapter 3, we addressed an optimization problem that requires deciding the location of a set of facilities, the allocation of customers to those facilities under capacity constraints, and the allocation of customers to trucks at those facilities under truck travel-distance constraints. We presented a hybrid approach that combines integer programming and constraint programming using logic-based Benders decomposition. Computational experiments demonstrated that the Benders model is able to find and prove optimal solutions up to three orders-of-magnitude faster than an existing integer programming approach, while also finding better feasible solutions in less time when compared to an existing tabu search algorithm.

To our knowledge, no previous work exists which attempts to solve facility location/fleet management problems with logic-based Benders decomposition. Therefore, this work broadens the range of approaches used to solve facility location/fleet management problems. We demonstrated that not only is logic-based Benders decomposition a useful approach for finding and proving optimal solutions, but it is also effective in finding good feasible solutions in cases where the problems is too large to prove optimality. We also helped to expand the scope of problems that can be solved by using logic-based Benders decomposition. More generally, we contribute to the recent work on the integration of operations research and constraint programming techniques.

The Application of Logic-Based Benders Decomposition to Two-Stage Stochastic Facility Location/Fleet Management Problems In Chapter 4, we extended the problem addressed in Chapter 3 by considering random travel times. Non-deterministic travel times can arise, for example, due to daily traffic patterns or weather-related disturbances. We considered the different travel time conditions as different scenarios with known probabilities. We presented

a stochastic programming model based on a bounded penalty approach [66] in which the expected recourse cost is constrained to not exceed a given threshold. In order to solve the problem, an integer programming and a two-level and a three-level logic-based Benders decomposition models were proposed. The two-level Benders model decomposes the problem into a deterministic master problem and a set of stochastic sub-problems, one for each scenario. Since in the two-level Benders approach the master problem is deterministic, we use a modified version of the logic-based Benders approach proposed in Chapter 3 to develop a novel three-level logic-based Benders model.

We compared the performance of the three models based on how they scaled with the number of scenarios and with the problem size. Computational experiments demonstrated the efficiency of the two Benders models over the integer programming model. The results suggested that for problems with small or medium number of scenarios the three-level logic-based Benders model outperforms the other two models with respect to the CPU time needed to find and prove the optimal solution, while for problems with a larger number of scenarios, the two-level logic-based Benders model is superior. This difference in performance is due to the fact that the more we decompose the problem, the looser the master problem relaxations become, thus pushing information out of the master problem and distributing it among the sub-problems. The increase in the distribution of information slows down the three-level model as the number of decomposition levels increase.

Overall, Chapter 4 illustrated that logic-based Benders decomposition is effective in solving two-stage stochastic facility location/fleet management problems, and significantly outperforms an integer programming approach. While logic-based Benders decomposition has been applied to stochastic integer programs [73], there does not appear to have been published work that follows our approach of defining the decomposition such that each sub-problem corresponds to one scenario. However, the idea has been considered and, though not explicitly stated in the text, some of the empirical results in [58] are based on such an approach.[1] By comparing the performance of the two-level and the three-level logic-based Benders models on a two-stage stochastic facility location/fleet management problems, we demonstrated that it is not always true that the more we decompose the problem the faster we can find and prove the optimal solution.

5.2. Future Work

Possible directions for future work include exploring extensions of the problems addressed in the preceding chapters, as well as further investigation and possible improvements of logic-based Benders decomposition.

5.2.1 Extensions of the Problems Addressed in Chapters 3 and 4

An extension of the problem addressed in Chapter 3 is the location-routing problem (see Section 2.1.2). In the location-routing problem it is assumed that vehicles serve multiple clients in a single tour. Due to the different levels of decisions present in such problems, they appear

[1] John Hooker, personal communication.

to be well-suited to a logic-based Benders decomposition approach. It would be interesting to compare such an approach to other exact methods from the literature such as branch-and-bound [66], and branch-and-cut [46, 65]. The logic-based Benders approach would decompose the problem into a location-allocation master problem and a set of vehicle-routing sub-problems. Since the sub-problems would be optimization problems, the convergence to optimality might be slow. Therefore, strong cuts (which remove more infeasible solutions) and tight sub-problem relaxations in the master (which shrink the search space) are needed to overcome this obstacle. It would also be interesting to apply a three-level Benders approach to the location-routing problem. In this approach, the location-allocation master problem would itself be decomposed into a facility location master problem and a customer allocation sub-problem.

In Chapter 4 we assumed that only the travel times were uncertain. One may attempt to take into account other stochastic aspects of the problem, such as uncertain demands. We believe that a modified version of our current Benders approach could be applied to such problems: the problem would be decomposed into a deterministic master problem which would use the expected or minimum value of the travel times and demands, and a set of sub-problems, one for each scenario. Figure 5.1 gives a schematic representation of the proposed logic-based Benders decomposition. In this approach, two different sub-problem relaxations would have to be used in the master problem, one for demands and the other for travel times. In other words, a set of constraints for the demands and another set for the travel times are used to represent the sub-problem relaxation in the master problem. Benders cuts would have to be developed for both demand violations and travel time violations

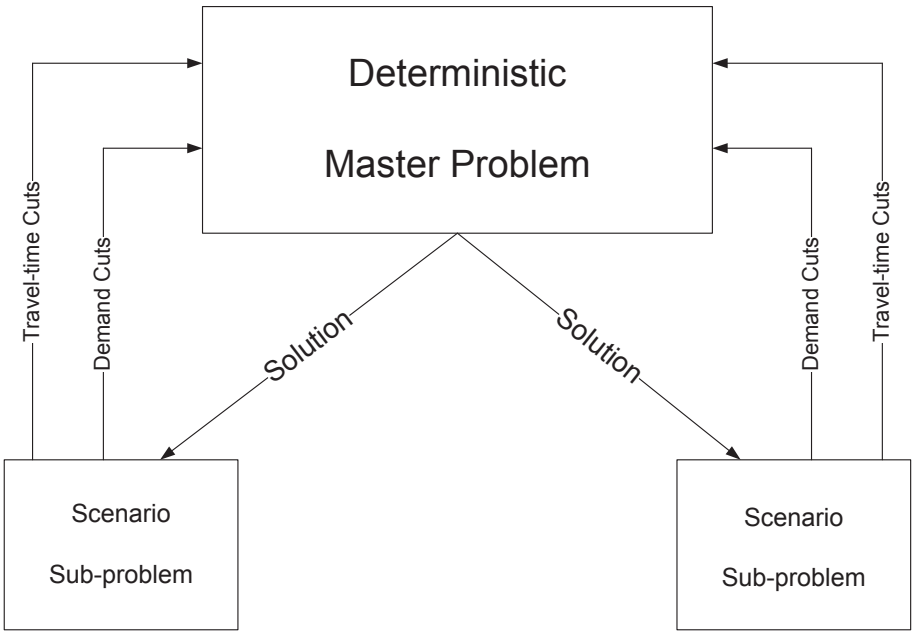

Figure 5.1: A schematic representation of the proposed logic-based Benders decomposition.

5.2.2 Further Investigation of Logic-Based Benders Decomposition.

Four important future research directions on logic-based Benders decomposition are:

1. **Improving the Performance of Logic-Based Benders Decomposition**

   Our experimental results demonstrated that the master problems were responsible for most of the computational burden (i.e., 95% of the CPU-time). Therefore, different techniques could be used to speed up the master problem. For instance, in every iteration of the Benders algorithm, heuristic approaches could be used to find good feasible solutions for the master problem, which can be added to the master problem as upper bounds in order to shrink its search space.

2. **Applying Three-Level Logic-Based Benders Decomposition to Other Problems**

   It would be interesting to investigate the performance of three-level logic-based Benders decomposition on other deterministic and stochastic problems. We conjecture that such an approach could be effective in solving problems with three or more levels of decision-making. For example, the facility location/fleet management problem addressed in Chapter 3 could be further decomposed into a facility location master problem, a customer allocation sub-problem, and a set of truck assignment sub-problems.

   As mentioned in Chapter 4, the main challenge one may face in developing a three-level Benders decomposition is the distribution of information among different levels, which may result in a slow convergence to optimality. Incorporation of a strong relaxation of the third (lowest) level of the problem into the first (highest) level would decrease the number of iterations and, thus, would increase the speed of convergence.

3. **Using Logic-Based Benders Decomposition to Find Good Feasible Solutions Quickly**

   For problems that are too large for a Benders approach to find and prove optimality, another algorithm is needed to find a good, but not necessarily optimal solution. Meta-heuristic techniques, such as tabu search [67], genetic algorithms [5] and simulated annealing [100] are widely used for this purpose. In Chapter 3, however, we showed that logic-based Benders decomposition can also be effective in finding good feasible solutions in cases where it cannot prove optimality. Therefore, an important question suggested by these results is: Can logic-based Benders decomposition be used as an effective decomposition method for finding good feasible solutions for large and complex problems? It would be interesting to evaluate the performance of logic-based Benders decomposition with respect to finding good feasible solutions on a variety of problems, and to identify problem characteristics that would allow such an approach to be effective.

4. **Logic-Based Benders Decomposition for Two-Stage Stochastic Programming Problems**

   Another important possible future research direction suggested by the results of Chapter 4 is: Can logic-based Benders decomposition be used as a general framework for solving two-stage stochastic programming problems? In such a framework, the stochastic problem is decomposed into a deterministic master problem, where the expected values or the minimum (maximum, depending on the objective function) of the random variables are used, and a set of stochastic sub-problems. Therefore, an important goal of future research is to apply logic-based Benders decomposition to a variety of two-stage stochastic programming problems.

5.3. Conclusion

The central thesis of this dissertation is that logic-based Benders decomposition can be effective in solving deterministic and stochastic facility location and fleet management problems. We demonstrated this thesis by developing logic-based Benders decomposition models for a facility location/fleet management problem from the literature and for a stochastic extension of the problem. We have experimentally shown the effectiveness of logic-based Benders in solving such problems. To our knowledge, this is the first work on solving deterministic and stochastic facility location/fleet management problems using logic-based Benders decomposition.

Bibliography

[1] R. Ahuja, J. Orlin, S. Pallottino, M. Scaparra, and M. Scutell. A multi-exchange heuristic for the single source capacitated facility location problem. *Management Science*, 50(6):749–760, 2004.

[2] D. Aksena and K. Altinkemer. A location-routing problem for the conversion to the click-and-mortar retailing: The static case. *European Journal of Operational Research*, 186(2):554–575, 2008.

[3] M. Albareda-Sambola, E. Fernández, and G. Laporte. Heuristic and lower bound for a stochastic location-routing problem. *European Journal of Operational Research*, 179(3):940–955, 2007.

[4] M. Albareda-Sambola, E. Fernández, and G. Laporte. The capacity and distance constrained plant location problem. *Computers & Operation Research*, 36(2):597–611, 2009.

[5] O. Alp, E. Erkut, and Z. Drezner. An efficient genetic algorithm for the p-median problem. *Annals of Operations Research*, 122:21–42, 2003.

[6] S. Arunapuram, K. Mathur, and D. Solow. Vehicle routing and scheduling with full truckloads. *Transportation Science*, 37:170 – 182, 2003.

[7] I. Averbakh, O. Berman, and D. Simchi-Levi. Probabilistic a priori routing-location problems. *Naval Research Logistics*, 41:973989, 1994.

[8] M. E. Aydin and T. C. Fogarty. A distributed evolutionary simulated annealing algorithm for combinatorial optimization problems. *Journal of Heuristics*, 10(3):269–292, 2004.

[9] J. Barceló, E. Fernández, and Jornsten K. Computational results from a new lagrangian relaxation algorithm for the capacitated plant locating problem. *European Journal of Operational Research*, 53:38–45, 1991.

[10] J . E. Beasley. Lagrangian heuristics for location problems. *European Journal of Operational Research*, 65:383–399, 1993.

[11] J. E. Beasley. A note on solving large p-median problems. *European Journal of Operational Research*, 21(2):270–273, 1985.

[12] J. C. Beck and P. Refalo. A hybrid approach to scheduling with earliness and tardiness costs. *Annals of Operations Research*, 118:49–71, 2003.

[13] J. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[14] L. Benini, M. Lombardi, M. Mantovani, M. Milano, and M. Ruggiero. Multi-stage Benders decomposition for optimizing multicore architectures. In L. Perron and M. Trick, editors, *In the Proceeding of the Fifth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'08)*, pages 36–50, 2008.

[15] T. Benoist, E. Gaudin, and B. Rottembourg. Constraint programming contribution to Benders decomposition: A case study. In Pascal Van Hentenryck, editor, *In the Proceeding of the Eighth International Conference on Principles and Practice of Constraint Programming (CP'2002)*, pages 603–617, 2002.

[16] O. Berman, P. Jaillet, and D. Simichi-Levi. Location-routing problems with uncertainty. In Z. Dezner, editor, *Facility location: a survey of applications and methods*, pages 427–453. New York: Springer Verlag, 1995.

[17] O. Berman, J. Wang, Z. Drezner, and C.O. Wesolowsky. A probabilistic minimax location problem on the plane. *Annals of Operations Research*, 122:59–70, 2003.

[18] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Ser. Oper. Res., Springer-Verlag, New York, NY, 1997.

[19] L. Bodin, B. Golden, A. Assad, and M. Ball. The state of the art in the routing and scheduling of vehicles and crews. *Computers and Operations Research.*, 10:63–211, 1983.

[20] G.G. Brown and G.W. Graves. Real-time dispatch of petroleum tank trucks. *Management Science*, 27(1):19–32, 1981.

[21] R. Carbone. Public facilities under stochastic demand. *Institute for operations research*, 12 (3):261–70, 1974.

[22] Y.M. Carson and B. Batta. Locating an ambulance on the Amherst campus of the State University of New York at Buffalo. *Interfaces*, 20(5):43–49, 1990.

[23] A. Ceselli and G. Righin. A branch-and-price algorithm for the capacitated p-median problem. *Networks*, 45(3):125–142, 2005.

[24] A. Charnes and W.W. Cooper. Chance-constrained programming. *Management Science*, 5:73–79, 1959.

[25] Y. Chu and Q. Xia. Generating Benders cuts for a general class of integer programming problems. In *In the Proceeding of the First International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'04)*, pages 127–136, 2004.

[26] L. Cooper. Location-allocation problems. *Operations Research*, 11:331–343, 1963.

[27] L. Cooper. A random locational equilibrium problem. *Journal of Regional Science*, 14(1):47–54, 1974.

[28] Cornujols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In *Discrete Location Theory,*. John Wiley & Sons, New York, NY, USA, 1990.

[29] A. I. Correa, A. Langevin, and L. M. Rousseau. Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. In *In the Proceeding of the First International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP'2004)*, pages 370–379, 2004.

[30] I. Correiaa and M. E. Captivo. Bounds for the single source modular capacitated plant location problem. *Computers & Operations Research*, 33:2991–3003, 2006.

[31] M. J. Cortinhal and M. E. Captivo. Genetic algorithms for the single source capacitated location problem. In *Metaheuristics: computer decision-making*, pages 187–216. 2004.

[32] T. G. Crainic, M. Gendreau, P. Hansen, and N. Mladenovic. Cooperative parallel variable neighborhood search for the p-median. *Journal of Heuristics*, 10(3):293–314, 2004.

[33] J. Current, M . Daskin, and D. Schilling. Discrete network location models. In *Facility location: applications and theory*. Springer Verlag, 2002.

[34] G.B. Dantzig. Linear programming under uncertainty. *Management Science*, 1:197–206, 1955.

[35] M. S. Daskin. *Network and Discrete Location: models, algorithms, and applications*. John Wiley & Sons, Inc., New York, 1995.

[36] M.S. Daskin. Logistics: An overview of the state of the art and perspectives on future research. *Transportation Research Part A: General*, 19(5-6):383–398, 1985.

[37] M.S. Daskin, C.R. Coullard, and Z.-J.M. Shen. An inventory-location model: formulation, solution algorithm and computational results. *Annals of Operations Research*, 110:83–106, 2002.

[38] H. Delmaire, J. A. Díaz, E. Fernández, and M. Ortega. Comparing new heuristics for the pure integer capacitated plant location problem. Technical Report DR97/10, Department of Statistics and Operations Research, Universitat Politecnica de Catalunya, Barcelona, Spain, 1997.

[39] J. A. Diaz and E. Fernández. A branch-and-price algorithm for the single source capacitated plant location problem. *Journal of the Operational Research Society*, 53:728–740, 2002.

[40] Z. Drezner. *Facility Location. A Survey of Applications and Methods*. Springer Series in Operations Research. Springer Verlag, New York, 1995.

[41] Z. Drezner and A. Suzuki. The big triangle small triangle method for the solution of nonconvex facility location problem. *Operations Research*, 52(1):128135, 2004.

[42] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009, 1978.

[43] R. D. Galvao and L. A. Raggi. A method for solving to optimality uncapacitated location problems. *Annals of Operations Research*, 18:225–244, 1989.

[44] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[45] A. M. Geoffrion and G. W. Graves. Multicommodity distribution system design by Benders decomposition. *Management Science*, 20(5):822–844, 1974.

[46] G. Ghiani and G. Laporte. Eulerian location problems. *Networks*, 34:291–302, 1999.

[47] M. Gronalt, R. Hartl, and M. Reimann. New savings based algorithms for time constrained pickup and delivery of full truckloads. *European Journal of Operational Research*, 151(3):520–535, 2003.

[48] M. Gronalt and P. Hirsch. Log-truck scheduling with a tabu search strategy. In *Metaheuristics*, pages 65–88. Springer US, 2007.

[49] S. L. Hakimi. Optimal locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12:450–459, 1964.

[50] P. Hansen and N. Mladenović. Variable neighborhood search for the p-median. *Location Science*, 5(4):207226, 1997.

[51] I. Harjunkoski and I. E. Grossmann. A decomposition approach for the scheduling. *Computers and Chemical Engineering*, 25:1647–1660, 2001.

[52] K. Hindi and K. Piénkosz. Efficient solution of large scale, single-source, capacitated plant location problem. *Journal of the Operational Research Society*, 50(3):268–274, 1999.

[53] K. Holmberg, M. Ronnqvist, and D. Yuan. An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research*, 113(3):544–559, 1999.

[54] J. N. Hooker. *Logic-based Methods for Optimization: Combining Optimization and Constraint Satisfaction,*. John Wiley & Sons, 2000.

[55] J. N. Hooker. A hybrid method for planning and scheduling. In *In the Proceeding of the Tenth International Conference on Principles and Practice of Constraint Programming (CP'2004)*, pages 305–316, 2004.

[56] J. N. Hooker. A hybrid method for the planning and scheduling. *Constraints*, 10(4):385–401, 2005.

[57] J. N. Hooker. Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55(3):588–602, 2007.

[58] J. N. Hooker and G. Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96:33–60, 2003.

[59] V. Jain and I. E. Grossmann. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing*, 13:2001, 2001.

[60] J. H. Jaramillo, J. Bhadury, and R. Batta. On the use of genetic algorithms to solve location problems. *Computers & Operations Research*, 29(6):761–779, 2002.

[61] P. Jarvinen, J. Rajala, and H. Sinervo. A branch-and-bound algorithm for seeking the p-median. *Operations Research*, 20(1):173–178, 1972.

[62] B. M. Khumawala. An efficient branch and bound algorithm for the warehouse location problem. *Management Science*, 18(12):718–733, 1978.

[63] A. Klose. A branch and bound algorithm for an UFLP with a side constraint. *International Transactions in Operational Research*, 5(2):155–168, 1998.

[64] M. Koerkel. On the exact solution of large-scale simple plant location problems. *European Journal of Operational Research*, 39(2):157–189, 1989.

[65] M. Labbé, I. Rodríguez-Martín, and J.J. Salazar-Gonzalez. A branch-and-cut algorithm for the plant-cycle location problem. *Journal of the Operational Research Society*, 55:513–520, 2004.

[66] F. Laporte, G.and Louveaux and H. Mercure. Models and exact solutions for a class of stochastic location-routing problems. *European Journal of Operational Research*, 39:71–78, 1989.

[67] G. Laporte. Location-routing problems. In B. L. Golden and A. A. Assad, editors, *Vehicle routing: methods and studies*, pages 163–197. Amsterdam : North-Holland, 1988.

[68] G. Laporte and F. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133–142, 1993.

[69] G. Laporte, Y. Nobert, and P. Pelletier. Hamiltonian location problems. *European Journal of Operational Research*, 12:82–89, 1983.

[70] G. Laporte and I. H. Osman. Routing problems: a bibliography. *Annals of Operations Research*, 61:227–262., 1995.

[71] T. V. Levanova and M. A. Loresh. Algorithms of ant system and simulated annealing for the p-median problem. *Automation and Remote Control*, 65(3):431–438, 2004.

[72] S.C. Liu and S.B. Lee. A two-phase heuristic method for the multi-depot location routing problem taking inventory control decisions into considerations. *International Journal of Advanced Manufacturing Technology*, 22:941–950, 2003.

[73] M. Lombardi and M. Milano. Stochastic allocation and scheduling for conditional task graphs in MPSoCs. In *Principles and Practice of Constraint Programming CP 2006*, volume 4204, pages 299–313. Springer, 2006.

[74] F.V. Louveaux. Discrete stochastic location models. *Annals of Operations Research*, 6:23–34, 1986.

[75] C. McDiarmid. Probability modelling and optimal location of a traveling salesman. *Journal of the Operational Research Society*, 43:533–538, 1992.

[76] L. Michel and P. Van Hentenryck. A simple tabu search for warehouse location. *European Journal of Operational Research*, 157(3):576–591, 2004.

[77] H. Mina, V. Jayaraman, and R. Srivastava. Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research*, 108(1):1–15, July 1998.

[78] G. Mosheiov. Pickup delivery location problem on networks. *Networks*, 26:243–251, 1995.

[79] G. Nagy and S. Salhi. Nested heuristic methods for the location-routing problem. *Journal of the Operational Research Society*, 47:1166–1174, 1996.

[80] G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, March 2007.

[81] S. H. Owen and M. S. Daskin. Strategic facility location: A review. *European Journal of Operational Research*, 111(3):423–447, 1998.

[82] Z. Ozyurt and D. Aksen. Solving the multi-depot location-routing problem with lagrangian relaxation. In *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*. Springer US, 2007.

[83] B. Peterson and M. A. Trick. A Benders approach to a transportation network design problem. In *In the Proceeding of the Sixth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'09)*, pages 326–327, 2009.

[84] A. Prekopa. On probabilistic constrained programming. In *Proceedings of the Princeton Symposium on Mathematical Programming*, 1970.

[85] J. Reese. Methods for solving the p-median problem: An annotated bibliography. Technical report, Department of Mathematics, Trinity University, 2005.

[86] R. Richardson. An optimization approach to routing aircraft. *Transportation Science*, 10:52–71, 1976.

[87] G. Righini. A double annealing algorithm for discrete location/allocation problems. *European Journal of Operational Research*, 86(3):452–468, 1995.

[88] E. Rolland, D.A. Schilling, and J.R. Current. An efficient tabu search heuristic for the p-median problem. *European Journal of Operational Research*, 96:329–342, 1997.

[89] K. E. Rosing, C. S. ReVelle, E. Rolland, D. A. Schilling, and J. R. Current. Heuristic concentration and tabu search: A head to head comparison. *European Journal of Operational Research*, 104(1):93–99, 1998.

[90] S. Salhi and G. K. Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156, 1989.

[91] M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[92] D Serra and V. Marianov. The p-median problem in a changing network: the case of Barcelona. *Location Science*, 6:383–394, 1999.

[93] P. Shaw. A constraint for bin packing. In *In the Proceeding of the Tenth International Conference on Principles and Practice of Constraint Programming (CP'2004)*, pages 648–662, 2004.

[94] L.V. Snyder. Facility location under uncertainty: a review. *IIE Transactions*, 38:537–554, 2006.

[95] M. Sun. Solving the uncapacitated facility location problem using tabu search. *Computers & Operations Research*, 33(9):2563–2589, 2006.

[96] D. Terekhov, J. C. Beck, and K. N. Brown. A constraint programming approach for solving a queueing design and control problem. *INFORMS Journal on Computing*, 21(4):549–561, 2009.

[97] D. Tuzun and L.I. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99, 1999.

[98] S. Vaithyanathan, L. I. Burke, and M. A. Magent. Massively parallel analog tabu search using neural networks applied to simple plant location problems. *European Journal of Operational Research*, 93(2):317–330, 1996.

[99] Q. Xia, A. Eremin, and M. Wallace. Problem decomposition for traffic diversions. In *In the Proceeding of the First International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, pages 349–363, 2004.

[100] V.F. Yu, S.W. Lin, W. Lee, and C.J. Ting. A simulated annealing heuristic for the capacitated location routing problem. *Computers and Operations Research*, 58(2):288–299, 2010.