Solving a Supply Chain Optimization Problem Collaboratively

Hoong Chuin Lau and **Andrew Lim**
School of Computing
National University of Singapore
3 Science Drive 2, Singapore 117543

Qi Zhang Liu
Department of Decision Sciences
National University of Singapore
Lower Kent Ridge Road, Singapore 119260

Abstract

We propose a novel algorithmic framework to solve an integrated planning and scheduling problem in supply chain management. This problem involves the integration of an inventory management problem and the vehicle routing problem with time windows, both of which are known to be NP-hard. Under this framework, algorithms that solve the underlying sub-problems collaborate rigorously yet in a computationally efficient manner to arrive at a good solution. We will then present two algorithms to solve the inventory management problem: a complete mathematical model integrating integer programming with constraint programming, and an incomplete algorithm based on tabu search. We present experimental results based on extended Solomon benchmark vehicle routing problems.

Introduction

Supply chain optimization involves the integration of decision-making processes to manage the production and flow of products and services from the source to the customers. An emerging industry trend in supply chain management is the formation of logistics operators that provide a one-stop point-to-point service for the entire distribution operation in the supply chain. Under this system, retailers need not manage their own inventory to ensure timely re-supply while the logistics operator achieves economies of scale by being able to co-ordinate deliveries to multiple retailers. Hence, a system-wide optimization can be achieved through an algorithmic integration of inventory and transportation.

More precisely, consider a distribution system with multiple suppliers, capacitated warehouses, capacitated retailers, identical capacitated vehicles and unit-sized items. The items are to be transported from the suppliers to the warehouses, and subsequently delivered to the retailers by vehicles. Vehicles can combine deliveries to multiple retailers, provided that the items are delivered within stipulated time windows. Given the retailers' time-varying demand forecast over a finite planning horizon, the goal is to find a distribution plan so as to minimize the total operating cost, which comprises the inventory cost (for amounts exceeding demand), backlogging cost (for amounts falling short of demand) and transportation cost.

We call this problem the *Inventory Routing Problem with Time Windows* (IRPTW). Clearly, IRPTW is a complex problem, since it involves the integration of two classical optimization problems: the dynamic capacitated lot-sizing problem and vehicle routing problem with time-windows. Both are proved to be NP-hard even for very simple instances. (See Florian, Lenstra and Rinnooy Kan (1980) and Savelsbergh (1986).)

Many inventory models have been proposed by the OR community in the past. The one-warehouse multi-retailers problem under constant demand has been extensively studied in the past few decades, and the structure of the (near) optimal policies under this system is already well-understood. The reader may refer to Graves, Kan and Zipkin (1993) for a comprehensive review. ¿From the Constraint Programming perspective, an interesting inventory management problem for reusable resources has been recently investigated by Caseau and Kokeny (1998).

As far as integrating inventory and transportation, one of the earliest work in this area is by Federgruen and Zipkin (1984) who considered a *single-period* IRP with stochastic demands at the retailers' end. Their model aims to determine the optimal allocation of inventory to retailers while minimizing routing and inventory costs. This problem in itself already leads to a huge mixed integer programming problem that can only be solved by heuristics. Chan, Federgruen and Simchi-Levi (1998) recently modelled a *single-item, constant demand* distribution system and presented worst case as well as probabilistic bounds for their models. Unfortunately, due to the unrealistic assumption on demand, it is doubtful that any of the asymptotically optimal heuristic proposed will perform well for realistic problems with time-varying demand.

To our knowledge, a supply chain problem as extensive as IRPTW has not been carefully studied in the literature. A notable exception is Carter *et al.* (1996), who proposed a Lagrangean heuristic to solve a single-supplier, single-warehouse IRPTW. Unfortunately, their approach cannot guarantee feasibility

(even if a solution exists), and the algorithm is sensitive to the values of several parameters where there are no good heuristics for setting them.

In this paper, we present a novel approach based on decomposition into two sub-problems (distribution and routing) plus an interface mechanism to allow the two algorithms to collaborate in a *master-slave* fashion, with the distribution algorithm (A1) driving the routing algorithm (A2). Intuitively, the procedure is as follows. A1 will determine the flow amounts between the suppliers, warehouse and the retailers so as to minimize inventory and backlogging costs subject to warehouse and retailers' capacities. A2, based on the given flow amounts (or customer demands, in Vehicle Routing terminology), sequences the deliveries into routes so as to minimize transportation cost, subject to vehicle capacities and time windows. These routes in turn induce a re-partitioning of the retailers for A1, and the process is repeated. The novelty hinges on the definition of a good interface between A1 and A2 so as to ensure *convergence*, since the objective functions are conflicting when taken separately. (To reduce inventory and backlogging, it is necessary to make more frequent deliveries, but this will increase the transportation cost). To achieve this, we complicate A1 with *vehicle capacity constraints* and impose penalty for flow amounts that have high aggregated transportation cost. This will be explained in greater detail later.

We see several advantages of our approach. First, from the planning and scheduling perspective, it offers an efficient and readily implementable way to tackle the intricacy of integrating two processes along a supply chain. Second, from the computational perspective, our framework is agent-oriented in the sense that the algorithms are agents trying to generate an overall plan collaboratively while guarding their individual interests. Finally, from the software engineering viewpoint, it supports the paradigm of reusability and plug-and-play in the sense that it allows either one of the modules to be replaced without affecting the other. We believe our approach can be adapted to provide decision support for a host of integrated supply chain optimization problems we face today.

Preliminaries

IRPTW is defined as, given the following input:

S: set of suppliers;

R: set of retailers;

J: set of items;

T: consecutive days in the planning period $\{1, 2, \cdots, n\}$;

D_{ijt}: demand of retailer i for item j on day t;

Q_V: vehicle capacity;

Q_W: warehouse storage capacity;

Q_i: storage capacity of retailer i;

W_i: time window of retailer i

C_j: inventory holding cost per unit item j per day at the warehouse;

C_{ij}: inventory holding cost per unit item j per day at retailer i;

B_{ij}: backlogging cost per unit item j per day at retailer i;

T_{ik}: transportation cost incurred by visiting retailer i followed by k on the same route;

output the following:

(1) the distribution plan, which is denoted by:

x_{sjt}: integral flow amount of item j from supplier s to the warehouse on day t; and

x_{ijt}: integral flow amount of item j from the warehouse to retailer i on day t; and

(2) the set of daily transportation routes Φ, which carry the flow amounts in (1) from the warehouse to the retailers such that the sum of the following linear costs is minimized: a) inventory cost at the warehouse (C_j), b) inventory cost at the retailers (C_{ij}), c) backlogging cost (B_{ij}); and d) transportation cost from the warehouse to the retailers (T_{ik}).

We will use indices i, s, j, t for retailers, suppliers, items and days respectively.

The distribution plan must obey the demands and storage capacity constraints. We further assume that items arriving at the warehouse on day t can only be delivered to retailers from day $t+1$ onwards. The transportation routes must obey the standard routing, vehicle capacities and time windows constraints. For notational convenience, we let Φ_t denote the set of routes for day t. Each route is an ordered list of retailers representing the delivery sequence performed by one particular vehicle per day.

Algorithmic Framework

Our algorithmic framework is an iterative approach between 2 sub-problems, namely the distribution problem (DP) and the vehicle routing problem with time-windows (VRPTW).

DP is a *constrained* version of the dynamic lotsizing problem, since it has to iterate with VRPTW in a manner that guarantees convergence. It receives a set of transportation routes Φ as part of the input, and returns a solution that has to be *consistent with* Φ (see definition below). Moreover, its objective function has an additional transportation cost component, which serves as a heuristic in order to generate a distribution plan such that VRPTW will in turn generate low-cost routes subsequently. In this way, the iterative improvement will be sustainable and hence effective.

Definition 3.1. We say that a distribution plan x is *consistent with* a set of transportation routes Φ iff Φ can fulfill the transport needs of x without violating any vehicle capacity constraints. More specially, for all days $t \in T$ and routes $\phi \in \Phi_t$, $\sum_{i \in \phi} \sum_j x_{ijt} \leq Q_V$.

VRPTW is a well-studied NP-hard problem. Several variants of VRPTW have been studied, and many efficient optimal as well as heuristic approaches have been

developed to solve them. For a comprehensive review on these algorithms, see Desrosiers *et al.* (1995). Our algorithm framework works for all kinds of VRPTW and any algorithm solving VRPTW. For the convenience of expression, we assume that there is no limit on number of vehicles. However, each vehicle is charged a high penalty so that the number of vehicles used will be minimized. This is to avoid the difficulty of finding feasible solutions for original VRPTW. (Even if there is some limit on number of vehicles, as the iteration between VRPTW and DP progresses, some retailers won't be visited on certain day. Hence the problem instances of VRPTW needed to be solve in the next iteration will decrease. Then the limit may become easy to meet.) With this assumption, we will assume the availability of one such efficient algorithm that returns to us a near-optimal feasible solution when given a VRPTW instance.

Let the algorithms for solving DP and VRPTW be denoted A1 and A2, and let the objective functions be denoted f_1 and f_2, respectively. Let **lowestSoFar** be the objective value of the best DP solution found so far, initialized to ∞. Procedurally, we propose the following:

Algorithm A:

(1) call A2 to generate an initial set of transportation routes Φ

(2) call A1 to generate a distribution plan x that is consistent with Φ

(3) if $f_1(x) =$ **lowestSoFar** return (x, Φ) and stop else set **lowestSoFar** $= f_1(x)$

(4) call A2 to generate a new set of routes Φ' based on x s.t. $f_2(\Phi') < f_2(\Phi)$

(5) if no such Φ' can be found, return (x, Φ) and stop

(6) set $\Phi = \Phi'$; goto Step (2)

It suffices to say now that, by optimality, Step (2) will never return a solution whose objective value is worse than the previous solution. In the next section, we will present details of A1 and A2, and prove that the above algorithm is correct and converges to a solution. More precisely, we will prove that under this framework, the overall objective function of IRPTW decreases monotonically from one iteration to the next.

Integrated IP/CP Model

In this section, we present an exact IP/CP model for solving DP.

Basically, we model DP as a multi-commodity flow problem complicated by side constraints. This model is a time-expanded 3-layer network for suppliers, warehouse and retailers respectively. The warehouse node and each retailer node are replicated n times for the n-day planning period. Arcs between nodes of different layers represent the flow amounts (suppliers to warehouse, warehouse to retailers), while arcs between adjacent replicated nodes represent either inventory carrying over to the next day, or backlogging from the previous day. This model is complicated by the consistency condition and additional transportation component, which can be handled by adding artificial nodes and concave costs on the incident arcs. The resulting model is a fixed-charge (or concave-cost) layered network.

In this section, we present a complete formulation with redundant logic constraints to model interesting relationships between inventory and demands. In the next section, we will present a tabu search strategy with strategic oscillation.

We explain further notations used.

Recall that by A2 (i.e. the VRPTW algorithm) outputs a set of routes Φ to DP. The following variables can be derived directly from Φ:

h_{irt}: 1 if retailer i is served by route $r \in \Phi$ on day t, and 0 otherwise;

c_r: cost of route r, defined as the sum of transportation costs T_{ik} over all adjacent retailers i and k on route r.

The following intermediate variables are used:

z_{ijt}: integral amount of item j held in retailer i on day t;

z_{jt}: integral amount of item j held in the warehouse on day t;

b_{ijt}: integral amount of item j backlog for retailer i on day t;

y_r: $(0,1)$ variable, whether route $r \in \Phi$ is used;

y_{it}: $(0,1)$ variable, whether retailer i is served on day t.

The integer programming formulation of DP is given as follows:

$\min \sum_j \sum_t (z_{jt}C_j + \sum_i z_{ijt}C_{ij} + \sum_i b_{ijt}B_{ij}) + \sum_r y_r c_r$

subject to the following linear constraints:

$$\sum_j z_{jt} \le Q_W, \text{ for } t \in T \quad (1)$$

$$\sum_j x_{ijt} + \sum_j z_{ijt} \le Q_i, \text{ for } i \in R \text{ and } t \in T \quad (2)$$

$$\sum_i x_{ijt} \le z_{jt}, \text{ for } j \in J \text{ and } t \in T \quad (3)$$

$$\sum_s x_{sjt} + z_{jt} - z_{j,t+1} - \sum_i x_{ijt} = 0, \text{ for } j \in J \text{ and } t < n \quad (4)$$

$$x_{ijt} + z_{ijt} - z_{ij,t+1} - b_{ijt} + b_{ij,t+1} = D_{ijt}, \text{ for } i \in R, j \in J \text{ and } t < n \quad (5)$$

$$x_{ijn} + z_{ijn} - b_{ijn} = D_{ijn}, \text{ for } i \in R, j \in J \quad (6)$$

$$\sum_j x_{ijt} \le y_{it} Q_i, \text{ for } i \in R \text{ and } t \in T \quad (7)$$

$$y_{it} \le y_r, \text{ for } i \in R, r \in \Phi, t \in T \text{ with } h_{irt} = 1 \quad (8)$$

$$\sum_i \sum_j \sum_t x_{ijt} h_{irt} \le y_r Q_V, \text{ for } r \in \Phi \quad (9)$$

The objective function comprises 4 components: the warehouse inventory cost, retailers' inventory cost, backlogging cost, and a specially designed function to reflect transportation cost.

Constraint (1) is the warehouse capacity constraint; (2) is the retailers' capacity constraint; (3) means the inventory in the warehouse must exceed daily delivery requirement; (4) is the inventory balance constraint on the warehouse; (5) & (6) are the inventory balance constraints on retailers; (7) defines whether each retailer

is served on each day; (8) means a route is used iff at least 1 retailer on this route is served; (9) is the vehicle capacity constraints that enforce the consistency condition on used routes.

The highlight of this model is that, when used cooperatively with the VRPTW algorithm, guarantees convergence. Particularly:

(a) The last component of the objective penalizes usage of *expensive* routes, i.e. it discourages the generation of a distribution plan that will incur a high transportation cost.

(b) Constraint (9) introduces bundle (or knapsack) constraints on the flows to enforce the consistency condition (see Definition 3.1).

Obviously, the above model by itself is much harder to solve than the standard multi-commodity flow problem, due to the following reasons:

(a) The last component of the objective introduces *disjunction* (on y_r) into the problem; and

(b) Typically, a problem with tight bundle constraints takes much longer to solve than one of comparable size without, as shown in Ho and Loute (1983). In fact, it has been shown in Garey and Johnson (1979) that the min-cost network flow problem with bundle constraints is NP-complete, even if all capacities are 1 and all bundles have 2 arcs.

One way to speed up search is to introduce redundant constraints that will trigger constraint propagation. We add the following logical (non-linear) constraints into the IP formulation:

$$x_{ijt} > 0 \Rightarrow y_{it} = 1, \text{ for all } i \in R, j \in J \text{ and } t \in T \quad (10)$$

$$(z_{ijt} > 0) + (b_{ijt} > 0) < 2, \text{ for all } i \in R, j \in J \text{ and } t \in T \quad (11)$$

$$z_{ijt} > z_{ij,t-1} \Leftrightarrow x_{ijt} > D_{ijt} + b_{ij,t-1} \text{ for all } i \in R, j \in J \text{ and } t \in T \quad (12)$$

$$b_{ijt} > b_{ij,t-1} \Leftrightarrow x_{ijt} + z_{ij,t-1} < D_{ijt} \text{ for all } i \in R, j \in J \text{ and } t \in T \quad (13)$$

Constraint (10) says retailer i is served on day t only if there is a positive flow to i that day. (11) is the mutual-exclusivity constraint on inventory holding and backlogging. (12) relates the rise and fall of retailers' inventories between 2 consecutive days with the demand-supply situation. (13) does likewise for retailers' backlogs.

(10) plays the role in forcing early instantiation of the variables y_{it}, which, by Constraint (8), cause an early instantiation of the disjunctive variables y_r. Similarly, (11) to (13) serve to trigger constraint propagation on the variables z_{ijt} and b_{ijt}, by Constraints (2) to (6), cause an early instantiation of the variables x_{ijt}.

The above formulation cannot be solved by traditional MIP solvers which do not support logic constraints, but can be efficiently solved by ILOG Planner (version 3.0) which integrates the ILOG Solver (a constraint propagation search engine) and CPLEX MIP solver.

Having presented enough details, we now proceed to give the convergence proof of algorithm A.

Convergence Proof

Let A1 be an algorithm that solves DP (represented by the above IP/CP model) to optimality, and A2 be any algorithm that returns a feasible solution for a given feasible VRPTW instance. We now prove the convergence of Algorithm A.

Let $f_1(x, \Phi) = \sum_j \sum_t (z_{jt}C_j + \sum_i z_{ijt}C_{ij} + \sum_i b_{ijt}B_{ij}) + \sum_{r \in \Phi} y_r c_r$ denote the objective function of DP and $f_2(\Phi) = \sum_{r \in \Phi} c_r$ denote the objective function of VRPTW. The key argument is that between two consecutive calls to A2, the objective value of f_1 on the same distribution plan must decrease, shown as follows.

Write $f_1(x, \Phi)$ as $f_1(x) + f_1(\Phi)$, where $f_1(x) = \sum_j \sum_t (z_{jt}C_j + \sum_i z_{ijt}C_{ij} + \sum_i b_{ijt}B_{ij})$ and $f_1(\Phi) = \sum_{r \in \Phi} y_r c_r$. Split Φ into $\Phi_1 = \{r \in \Phi | y_r = 1\}$ and $\Phi_2 = \{r \in \Phi | y_r = 0\}$. Then $f_1(\Phi) = f_1(\Phi_1) = f_2(\Phi_1)$. Suppose A2 generates a new set of routes $\Phi' = \Phi_1' \cup \Phi_2'$, where Φ_1' (resp. Φ_2') covers the retailers in Φ_1 (resp. Φ_2). Since $f_2(\Phi') < f_2(\Phi)$, it follows necessarily that $f_2(\Phi_1') < f_2(\Phi_1)$. Hence, $f_1(\Phi') = f_2(\Phi_1') < f_2(\Phi_1) = f_1(\Phi)$. This implies, $f_1(x, \Phi') = f_1(x) + f_1(\Phi') < f_1(x) + f_1(\Phi) = f_1(x, \Phi)$. We can therefore easily get the following lemma.

Lemma 1 *Given a feasible instance of IRPTW, Algorithm A converges to a solution.*

Tabu Search

In this section, we present a tabu search (TS) algorithm for solving DP.

TS is a form of local search augmented with adaptive memory. In TS, a *move* operator defines the neighborhood $N(s)$ of the current solution s. Starting with an initial solution, TS proceeds iteratively by replacing current solution s with a *best* neighbor $s' \in N(s)$ among all possible moves. One crucial feature of TS is the notion of a *tabu list*, which is a short-term memory that helps the search avoid cycling as well as escape from local optimality. Another interesting feature is the notion of *strategic oscillation*, which is a long-term memory that achieves an effective interplay between intensification and diversification of search. For a comprehensive description of the tabu search methodology, the reader may refer to the text of Glover and Laguna (1997). In the following, we assume that the reader is familiar with standard TS terminology.

Move Operators

The key move operator is the *transfer move*, which transfer flow amount from one flow variable to another having the same retailer and item (i.e. transfer flow across *different* days). We define two ways to determine the units of amount to be transferred. The first is based on a *variable scaling* strategy where each x_{ijt} is scaled

to some discrete units and at each iteration, only 1 unit is transferred. The scaling factor differs over different iterations. We begin with large factors (i.e. coarse-granular flows) and gradually decrease them. When no refinement can be made to obtain better solution, the procedure reverts to a large factor and the process repeats. The second is *greedy feeding* strategy, where we transfer as many units as possible between two flow variables without violating any capacity constraint.

Two other move operators are introduced to speed up the search for better solutions and to escape from local optimality: (1) the *free move*, which frees a retailer on a certain day by transferring all items to other days via the greedy feeding strategy; and (2) the *empty move*, which empties a route by freeing all retailers on the route via free moves.

Note that the moves can result in infeasible solutions (see sub-section on Strategic Oscillation).

Tabu List

The tabu search procedure uses a *tabu list* to store the time (i.e. iteration number) when tabu-active[1] status of each variable ends. Let $\Theta_x = (\tau_{ijt})$, 3D-array of size $|R| \cdot |J| \cdot |T|$, be the tabu list associated with the variables x_{ijt}. We define τ_{ijt} as follows:

$$\tau_{ijt} = start_{ijt} + tenure_{ijt},$$

where $start_{ijt}$ is the iteration number immediately before x_{ijt} was changed, and $tenure_{ijt}$ is the tabu tenure, which is a fraction of the size of Θ_x. Experimentally, TS is effective when the tenure is a random value in the range $[|R||J||T|/5, |R||J||T|/4]$ generated at iteration $start_{ijt}$.

The tabu-active status of other variables, such as retailer coverage y_r and route usage y_{it} are defined likewise.

Candidate List

In general, the neighborhood associated with each move operator can be extremely large. For instance, the transfer move neighborhood has $|R||J||T||T-1|$ elements and hence an exhaustive search of the entire neighborhood at every iteration is too expensive. Instead, we achieve an effective tradeoff between the quality of the best move and the effort expended to find it by determining a much-smaller *candidate list* for each move operator via a greedy strategy.

For the transfer move, we maintain a sorted list of relative inventory costs for each retailer i and item j:

$$w_{ij} = (\sum_t z_{ijt} + \sum_t b_{ijt}) / \sum_t D_{ijt}$$

and those variables x_{ijt} whose corresponding costs are within the top c% (where c is a pre-defined constant) are considered for move.

[1] When a variable is *tabu-active*, its value is not allowed to change during that iteration.

In the same vein, for *free moves* and *empty moves*, we consider retailers and routes (respectively) whose *relative loads* (defined below) are either very high or very low. Low loads can potentially save cost, while high loads may help escape from local optimality. Hence, for free moves, we maintain a sorted list of weights for each retailer i on each day t:

$$w_{it} = max(\sum_j x_{ijt} / \sum_{s,j} D_{ijs}, 1 - \sum_i x_{ijt} / \sum_{s,j} D_{ijs});$$

and for empty moves, we maintain:

$$w_r = max(\sum_{i,t} (h_{irt} \cdot \sum_j x_{ijt}) / Q_V, 1 - \sum_{i,t} (h_{irt} \cdot \sum_j x_{ijt}) / Q_V);$$

those retailers and routes whose corresponding weights are within the top c% (where c is a pre-defined constant) are considered for move.

Strategic Oscillation

Strategy oscillation operates in the tabu search procedure by orienting moves in and out of the feasible region. A penalty is imposed on solutions that are infeasible, i.e. those which violate some constraints. For each $i \in R$ and $j \in J$, let $x_{ij,n+1}$ be amount of unfulfilled demand of retailer i for item j, i.e.,

$$x_{ij,n+1} = \sum_t D_{ijt} - \sum_t x_{ijt}.$$

Let $f(x)$ be a function that takes value x if $x > 0$, or 0 otherwise. Denote the amount of items exceeding the capacity of retailer i on day t by

$$v_{it} = f(\sum_j x_{ijt} - Q_i);$$

the amount exceeding the capacity of the warehouse on day t by

$$v_t = f(\sum_j z_{jt} - Q_W);$$

and the amount exceeding the capacity of a vehicle serving route r on day t by

$$v_r = f(\sum_{i,t} (h_{irt} \cdot \sum_j x_{ijt}) - Q_V).$$

We impose an infeasible solution with a penalty cost directly proportional to the degree of constraint violation:

$$P_1 \sum_i \sum_j x_{ij,n+1} + P_2 (\sum_i \sum_t v_{it} + \sum_t v_t + \sum_r v_r),$$

where $P_1 < P_2$ are pre-defined parameters. For the intensification phase, they are set high values to reduce the chance of reaching an infeasible solution, while for the diversification phase, they are set lower values. Experimentally, our tabu search scheme is effective when $P_1 = P_2/2$.

Problem	Initial	Final	Iterations	Time(sec)
R101	48929	46022	3	19
R102	47852	44412	3	20
R103	47915	43638	4	40
R104	45692	43184	3	35
R105	46646	43450	3	23
R106	44545	42651	3	35
R107	43050	42193	3	30
R108	41495	40403	2	14
R109	46376	44322	3	24
R110	48283	45283	3	25
R111	47564	44960	3	34
R112	46613	44408	3	38
C101	105608	90667	2	15
C102	146067	104105	3	34
C103	131945	106930	4	44
C104	153228	111894	3	30
C105	120002	106073	2	17
C106	128149	112500	3	34
C107	126839	105747	5	55
C108	128345	102260	5	59
C109	142795	108688	4	38
RC101	110842	106259	4	39
RC102	122789	110580	2	13
RC103	132193	110467	6	70
RC104	119634	102381	4	45
RC105	127889	117552	5	44
RC106	106396	99141	4	32
RC107	117967	109035	2	17
RC108	108388	105446	4	43

Table 1: Experimental Results (Based on Extended Solomon's Benchmarks)

Experimental Results

In this section, we report some preliminary experimental results. To our knowledge, no benchmark test data available in the literature matches our problem exactly. Hence, our experimental results are based on the test data we generated as follows.

Since IRPTW contains VRPTW as a sub-problem, we adopt the well-known Solomon benchmark problems listed in Solomon (1987) to generate the locations and time-windows of the retailers and warehouse (depot). The demands of retailers are randomly generated in the range [0,30]. The capacities of the vehicles, retailers and warehouse are set as 200, 300 and 10000 respectively. In this paper, we consider 3 types of items, 3 suppliers, 1 warehouse and 50 retailers over a 5-day planning horizon. In terms of cost parameters, the inventory cost at the warehouse (C_j), inventory cost of retailer (C_{ij}) and backlogging cost (B_{ij}) are set to be 1, 2 and 4 per unit item per day respectively. The transportation cost of each route is 5 times its total distance plus a fixed vehicle usage cost of 50.

DP is implemented based on the Tabu Search described above, and VRPTW is based on an efficient engine developed inhouse. Note that our concern is not the absolute quality of the solution, but rather, the improvement that can be derived when the two engines collaborate (versus the conventional sequential pipeline approach). Under our proposed collaborative framework, any improvement made to the DP algorithm or VRPTW algorithm will increase the overall solution quality of IRPTW.

We run our program on a Pentium 300 PC and the results are given as follows. In this table, we plot the different test instances against: (1) the initial objective value, (2) final objective value, (3) total number of iterations taken, and (4) CPU run time.

¿From Table 1, we observe that the average number of iterations to convergence is 3.38, and the average percentage improvement in the objective value of the final solution over the initial solution is 10.5%.

References

M. W. Carter, J. M. Farvolden, G. Laporte, J. Xu, Solving an Integrated Logistics Problem Arising in Grocery Distribution, *INFOR*, **34:4** (1996), 290–306.

L. M. Chan, A. Fedegruen and D. Simchi-Levi, Probabilistic Analysis and Practical Algorithms for Inventory-Routing Models, *Operations Research*, **46:1** (1998) 96–106.

Y. Caseau and T. Kokeny, An Inventory Management Problem, *Constraints*, **3**, (1998), 363–373.

J. Desrosiers, Y. Dumas, M. M. Solomon and F. Soumis, Time Constrainted Routing and Scheduling, in: M. O. Ball *et al.* eds., *Handbooks in Operations Research and Management Science* Vol **8**: Network Routing, (North-Holland, 1995), 35–139.

M. Florian, J. K. Lenstra, and A. H. G. Rinnooy Kan, Deterministic Production Planning: Algorithm and Complexity, *Management Sc.*, **26:7** (1980), 669–679.

A. Fedegruen and P. Zipkin, An Efficient Algorithm for Computing Optimal (s,S) Policies, *Operations Research*, **22** (1984), 1268–1285.

S. C. Graves, A. H. G. Rinnooy Kan and P. H. Zipkin eds., *Handbook in Operations Research and Management Science* Vol **4**: Logistics of Production and Inventory, (North-Holland, 1993).

M. R. Garey and D. S. Johnson, *Computers and Intractibility,* (Freeman and Company, 1979).

F. Glover and M. Laguna, *Tabu Search*, (Kluwer Academic Publishers, 1997).

J. K. Ho and E. Loute, Computational Experience with Advanced Decomposition of Decomposition Algorithms, *Math Programming*, **27:3**, (1983), 283–290.

M. W. P. Savelsbergh, Local Search for Routing Problems with Time Windows, *Annals of Operations Research*, **4**, (1986), 285–305.

M. M. Solomon, Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints, *Operations Research*, **35**, 1987, 254–265.