J Sched (2009) 12: 257–280
DOI 10.1007/s10951-008-0081-9

The integrated production–inventory–distribution–routing problem

Jonathan F. Bard · Narameth Nananukul

Received: 14 May 2007 / Accepted: 4 July 2008 / Published online: 20 August 2008

Abstract The integration of production and distribution decisions presents a challenging problem for manufacturers trying to optimize their supply chain. At the planning level, the immediate goal is to coordinate production, inventory, and delivery to meet customer demand so that the corresponding costs are minimized. Achieving this goal provides the foundations for streamlining the logistics network and for integrating other operational and financial components of the system. In this paper, a model is presented that includes a single production facility, a set of customers with time varying demand, a finite planning horizon, and a fleet of vehicles for making the deliveries. Demand can be satisfied from either inventory held at the customer sites or from daily product distribution. In the most restrictive case, a vehicle routing problem must be solved for each time period. The decision to visit a customer on a particular day could be to restock inventory, meet that day's demand or both. In a less restrictive case, the routing component of the model is replaced with an allocation component only.

A procedure centering on reactive tabu search is developed for solving the full problem. After a solution is found, path relinking is applied to improve the results. A novel feature of the methodology is the use of an allocation model in the form of a mixed integer program to find good feasible solutions that serve as starting points for the tabu search. Lower bounds on the optimum are obtained by solving a modified version of the allocation model. Computational testing on a set of 90 benchmark instances with up to 200 customers and 20 time periods demonstrates the effectiveness of the approach. In all cases, improvements ranging from 10–20% were realized when compared to those obtained from an existing greedy randomized adaptive search procedure (GRASP). This often came at a three- to five-fold increase in runtime, however.

Keywords Tabu search · Production planning · Lot-sizing · Inventory · Vehicle routing problem · Allocation model

J.F. Bard (✉) · N. Nananukul
Graduate Program in Operations Research & Industrial Engineering, The University of Texas, 1 University Station C2200, Austin, TX 78712-0292, USA
e-mail: jbard@mail.utexas.edu

N. Nananukul
e-mail: narameth@mail.utexas.edu

1 Introduction

The role that the customer plays in logistics management is becoming increasingly important as the areas in which businesses compete expand to include quality, on time delivery, warranty and repair services, pricing contracts, and remanufacturing. Many companies now realize that greater value can be offered to their customers by improving the timeliness and consistency of delivery. Achieving a level of integration that will yield new benefits requires that the production and distribution decisions be made daily to balance setup, holding and delivery costs while tightly managing available resources. These decisions have traditionally been made separately, often at the planning stage without regard to scheduling, but their integration can have a significant impact on overall system performance. Chandra and Fisher (1994) have shown, for example, that solving the production scheduling and vehicle routing problems simultaneously can result in total operating cost reductions from 3 to 20%.

A vendor managed inventory replenishment (VMI) system is a good example of the type of integration mentioned above (e.g., see Cetinkaya et al. 2006; Zhao et al. 2007). In the VMI model a vendor observes and controls the inventory levels of its customers, as opposed to conventional approaches where customers monitor their own inventory and decide the time and amount of products to reorder. One of the benefits of VMI is that vendors can achieve a more uniform utilization of transportation resources leading to lower distribution costs. It also offers them the flexibility to choose the most preferred transportation mode. Customers benefit from higher service levels and greater product availability due to the fact that vendors can use existing inventory data at their customer sites to more accurately predict future demand. When a single entity is responsible for both planning and scheduling, efficiencies are realized at all nodes in the system. This has been the underlying motivation for the emphasis on supply chain management that is now widespread in the economy (e.g., see O'Brien and Tang 2006).

In general, the problem of optimally coordinating production and transportation is called the production–inventory–distribution–routing problem (PIDRP) (e.g., see Lei et al. 2006). Addressing these components in a single framework offers a holistic view of the logistics network and provides a good starting point for the full integration of the supply chain. The PIDRP commonly arises in the retail industry where customers or outlets rely on a central supplier or manufacturer to provide them with a given commodity on a regular basis. In the version of the problem addressed here, a manufacturer must develop minimum cost production and distribution schedules for a single product that are sufficient to meet all customer demand over the planning horizon.

The purpose of this paper is to outline the full model of the PIDRP along with a composite solution methodology that gives verifiably high quality results within acceptable runtimes. For planning purposes, this means within one or two hours. The primary components of the methodology are an allocation model for obtaining initial solutions and lower bounds on the optimum and a tabu search metaheuristic (Glover and Laguna 1997) with path relinking (Resende and Ribeiro 2005) for improving the results. The tabu search is distinguished by its neighborhood structure, short- and long-term memory functions, and search strategies.

In the next section, the PIDRP is outlined and a portion of the relevant literature is reviewed with an emphasis on the most recent work. In Sect. 3, a formal definition of the allocation model is given, which takes the form of a mixed-integer program (MIP). This is followed in Sect. 4 by our tabu search algorithm. Our lower bounding model is described in Sect. 5 and several theoretical statements are made to offer some improvement to the results. The computational results are highlighted in Sect. 6 for benchmark instances with up to 200 customers and 20 time periods. The analysis is discussed in Sect. 7 and several ideas are presented for extending the work.

2 Problem statement and literature review

Although the PIDRP can be defined more generally, our focus is on a single facility that must meet the demand of its customers for a single commodity. To ensure timely distribution and to avoid shortages, excess production can be stored at either the plant or at the customer sites up to some limits; however, inventory cannot be transferred between sites and stockouts are not permitted. It is further assumed that demand is known for each day of the planning horizon and that all initial inventories are given. In the model, two lot-sizing decisions must be made. The first concerns the production–inventory tradeoff; the second relates to the daily distribution decisions over the planning horizon.

With regard to the latter, deliveries are made routinely by a fleet of homogeneous vehicles that begin and end their run at the plant. In the most complex scenario investigated, a vehicle routing problem (VRP) must be solved daily to determine, in conjunction with the production decisions, which customers to visit and how much product to deliver to each. Due to either vehicle capacity limits or favorable cost tradeoffs, it may be desirable to visit a customer on a day in which ample stocks exist at his site in order to build up inventory. In fact, this might be the only feasible option if all demand is to be met.

The PIDRP is different than traditional VRPs because it requires multiple customer visits to satisfy demand spread out over an extended period of time. It is most similar to the inventory routing problem, IRP (Abdelmaguid and Dessouky 2006; Bard et al. 1998; Golden et al. 1984, and Dror and Ball 1987) and the periodic routing problem, PRP (Gaudioso and Paletta 1992; Mourgaya and Vanderbeck 2007, and Parthanadee and Logendran 2006). Although there has been much research on these two problems, little of it carries over to the PIDRP, which has only been studied intermittently. The primary reason relates to the formidable complexity of its structure, as defined by a combination of a capacitated lot-sizing problem (Gutiérrez et al. 2007; Pochet and Wolsey 2006) and a capacitated, multiperiod VRP. The full problem has so far proven to be beyond the capability of exact methods. By decoupling of the lot-sizing and routing decisions, though, several researchers have had some success in finding good solutions with heuristics. Chandra and Fisher (1994), for example, first determine a production schedule without regard to the logistics. Next, they developed a distribution schedule for each planning period based on the results obtained from the first-stage model. This approach worked well when there was enough inventory in the system to buffer production from the distribution operations but consequently led to increased holding costs.

The IRP and the PRP are relaxations of the PIDRP, differing in several ways. Neither, for example, takes the production decision and inventory level at the plant into consideration. In addition, the PRP assumes that the delivery patterns defined by delivery frequencies or delivery days are given in advance and tries to select the most suitable pattern for each customer.

Golden et al. (1984) were the first to investigate the interrelated problem of inventory allocation and vehicle routing. The particular application involved an energy-products company that distributed liquid propane to its customers. They developed a heuristic for designing an integrated delivery planning system aimed at comparing the distribution rule used by the company with their approach. Historical data was used to calculate an average consumption rate for each customer, and the latest replenishment data was used to calculate when each customer could be expected to need a resupply. The next replenishment was scheduled based on this information. The proposed heuristic included a customer selection algorithm that decides which customers to visit on each day in a cost-effective way and a VRP component to construct daily routes. Testing was done on instances with up to 3000 customers. The results were compared to those from a simulation experiment and showed an improvement of 8.4% in the number of gallons/hour, a 50% reduction of stockouts, and a 23% reduction in total costs. The basic methodology was extended by Bard et al. (1998) to better account for demand uncertainty and the use of satellite facilities for extending daily tours.

Parthanadee and Logendran (2006) considered a multi-product, multi-depot periodic distribution problem and formulated it as a MIP. In the model, they assumed that the daily demand of each customer was known and that all deliveries could be completed in one day within specified time windows and allowing for multiple vehicle trips. Backordering of products at the depots was allowed. To rationalize deliveries over the planning horizon, a set of predefined patterns was introduced and ranked by the customers. Within a tabu search heuristic, a penalty scheme was used to direct the search away from those patterns that were deemed undesirable.

In their version of the PRP, Mourgaya and Vanderbeck (2007) proposed a column generation-based heuristic to fix dates for customer visits and to assign customers to vehicles. The daily sequencing decisions were left to an operational model. In formulating the problem two objectives were considered: (1) optimizing the compactness of the geographical regions to which customers were assigned, and (2) balancing the workload evenly between vehicles. Using a Dantzig–Wolfe reformulation, they found that the resulting master problem provided substantially stronger lower bounds than the LP relaxation of the original problem and that there were fewer difficulties due to symmetry during branch and bound. The pricing subproblem decomposed into τ clustering problems, one for each time period. Computational tests were performed using 20, 50, and 80 customer data sets that involved comparisons of various implementation options related to initialization, column generation strategies, number of passes in the rounding heuristic, and problem specifications.

Zhao et al. (2007) studied the integration of inventory control and vehicle routing for a distribution system in which a set of retailers with constant rates of demand were resupplied with a single item from a central warehouse. The objective was to determine inventory policies and routing strategies such that the long-run average costs were minimized and all demand was satisfied. In their model, no inventory capacity constraints were imposed on the warehouse or on the retailers. Testing was done on problem instances with 50 and 75 retailers.

There has been a vast amount of research in the areas of production planning and inventory management over the last 40 years, with the field now including all aspects of the supply chain. Anily and Federgruen (1993), for example, analyzed fixed partition policies for the inventory routing problem with constant deterministic demand rates and an unlimited number of vehicles. The routing patterns were determined by using a modified circular clustering scheme. After the customers were grouped, those within a partition were assigned to one or more regions. Demand was calculated by summing the demand of the individual customers assigned to a region taking into account percentage allocations. The routing logic required that all customers in a region be visited as long as there was a need to visit one. A lower bound on the long-run average cost was also determined to provide an understanding of the effectiveness of the routing scheme. For extensions to multiple items and warehouses, see Federgruen and Tzur (1999).

Lei et al. (2006) proposed a two-phase solution approach for the PIDRP with multiple plants and a heterogeneous fleet. Our methodology is very similar to theirs. In phase one, a restricted version of the problem that contained all but the routing constraints was solved. The results provided a production schedule and the number of items to be delivered to each customer in each period. The solutions were shown to always be feasible to the original problem; however, they did not allow for the consolidation of less-than transporter loads (LTL), which could have reduced overall transportation costs. To address this issue, a routing heuristic based on an extended optimal partitioning procedure was used in phase two to consolidate the LTL assignments into more efficient delivery schedules. Testing showed that the approach gave good solutions to instances with up to 50 customer sites over 2 to 4 planning periods.

Boudia et al. (2006, 2007) proposed both a memetic algorithm with population management (MAPA) and a reactive greedy randomized adaptive search procedure (GRASP) with path-relinking (Resende and Ribeiro 2005) to solve the PIDRP. The model included a single plant and a set of customers located on a grid. Holding costs at the customer sites were assumed to be negligible compared to the holding costs at the plant and so were ignored. Like ours, the objective was to minimize the sum of production, holding and transportation costs while ensuring that all demand was satisfied over the planning horizon.

As an enhancement, the authors proposed two strategies to combine path relinking and GRASP. The first strategy was to perform path relinking upon the termination of GRASP; the second was to perform path relinking every time GRASP improved on the incumbent. The computational results showed that on average the first strategy performed better than the second on the 50 and 100 customer instances with 20 time periods but was not as good on the 200 customer instances. The algorithms converged within 100 s on a 2.8 GHz computer in all cases but no lower bounding procedures were provided to gauge the quality of the solutions. The reactive GRASP on average provided more savings than the solutions obtained with the GRASP alone by 0.8% and the solutions obtained with the relinking feature by 0.42%.

3 Model formulation

We are given a single production facility and a set of n customers geographically dispersed on a grid. Each customer $i \in N = \{1, \dots, n\}$ has a fixed nonnegative demand d_{it} in time period t of the planning horizon that must be satisfied, i.e., shortages are not permitted. If production takes place at the facility in period t, then a setup cost f_t is incurred, $t \in T_0 = \{0, 1, \dots, \tau\}$. A limited number of items can be produced in each time period and a limited number can store at a unit cost of h^P. In general, it is natural to equate a period with a day, which we do, but when production is scheduled for more than one shift in a day or when transportation times are measured in weeks, a broader interpretation of a period is appropriate.

In constructing delivery schedules, each customer can be visited at most once per day and each of θ homogeneous vehicles can make at most one trip per day. The latter restriction implies that all routes overlap in time. If c_{ij} is the cost of going from customer i to customer j and x_{ijt} is a binary variable equal to 1 if customer i is the immediate predecessor of customer j on a route in period t (0 otherwise), then the routing costs are given by $\sum_{ijt} c_{ij} x_{ijt}$ in the full model. A limited amount of inventory can be stored at customer i's site at a unit cost of h_i^C, but transshipments between customers are not permitted (cf. Herer et al. 2006).

Moreover, it is assumed that all deliveries take place at the beginning of the day and arrive in time to satisfy demand for at least that day. All production on day t is available for delivery only on the following morning (this is common in food production and distribution, e.g., see Villegas and Smith 2006) and all inventories are measured at the end of the day. Demand on day t can be met from deliveries on day t and from ending inventory on day $t-1$ at the customer site. Initial customer inventory on day 0 simply reduces demand on subsequent days, while initial inventory at the plant must be routed; at the end of the planning horizon all inventory levels are required to be zero.

The goal is to construct a production plan and delivery schedule that minimizes the sum of all costs while ensuring that each customer's demand is met over the planning horizon. In so doing, four critical decisions have to be made:

- How many items to manufacture each day.
- When to visit each customer.
- How much to deliver to a customer during each visit.
- Which delivery routes to use.

As part of the solution methodology, we do not attempt to solve the full model but instead investigate a relaxation referred to as the *allocation model*. In particular, the routing term in the objective function is replaced by a distribution component. The following additional notation is used in the developments.

Parameters

Q	capacity of each vehicle
$I_{\max}^P$	maximum inventory that can be held at the production facility
$I_{\max,i}^C$	maximum inventory that can be held by customer i
C	production capacity of the plant
f_{it}^C	fixed cost of making a delivery to customer i on day t
e_{it}^C	variable cost of delivering one item to customer i on day t

Decision variables

p_t	production quantity on day t
z_t	1 if there is production on day t; 0 otherwise
I_t^P	inventory at the production facility at the end of day t
I_{it}^C	inventory at customer i at the end of day t
w_{it}	amount delivered to customer i on day t
z_{it}^C	1 if a delivery is made to customer i on day t; 0 otherwise

Model

$$\phi_{\text{IP}} = \text{Minimize} \quad \sum_{t\in T} f_t z_t + \sum_{t\in T}\sum_{i\in N} f_{it}^C z_{it}^C + \sum_{t\in T}\sum_{i\in N} e_{it}^C w_{it} + \sum_{t\in T_0\setminus\{\tau\}} h^P I_t^P + \sum_{t\in T\setminus\{\tau\}}\sum_{i\in N} h_i^C I_{it}^C \tag{1a}$$

$$\text{subject to}\quad I_t^P = I_{t-1}^P + p_t - \sum_{i\in N} w_{it},$$

$$\forall t \in T_0; \tag{1b}$$

$$\sum_{i\in N} w_{it} \le I_{t-1}^P, \quad \forall t \in T; \tag{1c}$$

$$p_t \le C z_t, \quad \forall t \in T_0 \backslash \{\tau\}; \tag{1d}$$

$$p_0 \ge \sum_{i\in N} (d_{i1} - I_{i0}^C); \tag{1e}$$

$$I_{it}^C = I_{i,t-1}^C + w_{it} - d_{it},$$

$$\forall i \in N, t \in T; \tag{1f}$$

$$w_{it} \le D_{it}^{\max} z_{it}^C, \quad \forall i \in N, t \in T; \tag{1g}$$

$$\sum_{i\in N} w_{it} \le \lfloor 0.8 Q\theta \rfloor, \quad \forall t \in T; \tag{1h}$$

$$0 \le I_t^P \le I_{\max}^P, \qquad 0 \le I_{it}^C \le I_{\max,i}^C,$$

$$\forall i \in N, t \in T \backslash \{\tau\};$$

$$I_\tau^P = I_{i\tau}^C = 0, \quad \forall i \in N; \tag{1i}$$

$$z_t \in \{0,1\}, \qquad z_{it}^C \in \{0,1\}, \qquad p_t \ge 0,$$

$$w_{it} \ge 0, \quad \forall i \in N, t \in T, \tag{1j}$$

$$\text{where } T = T_0 \backslash \{0\} \quad \text{and}$$

$$D_{it}^{\max} = \min\left\{Q, \sum_{l=t}^{\tau} d_{il}\right\}.$$

The objective function minimizes the sum of production setup costs, a surrogate for the routing costs (second and third terms), holding costs at the plant, and holding costs at the customer sites. An explanation of the values used for the coefficients f_{it}^C and e_{it}^C is given in the next section. Because all demand must be met, production costs, which are assumed to be linear and independent of time, can be omitted, as can any initial inventory costs. Constraints (1b) and (1f) are inventory flow balance equations in which it is assumed that the initial inventories I_0^P and I_{i0}^C are given for all customers $i \in N$. Constraint (1d) limits production on day t to the capacity of the plant. A simple way to tighten this constraint is to replace C with $C_t = \min\{C, I_{\max}^P, D_{t+1}^{\max}\}$, where the third term is the demand of all customers for the remainder of the planning horizon. The assumption that items produced on day t are only available for delivery on day $t+1$ implies that $p_t \le I_{\max}^P$ and $p_\tau = 0$. It is possible to strengthen the latter inequality by subtracting from $I_{\max}^P$ the reduction in inventory due to deliveries on day t to get $p_t \le I_{\max}^P - (I_{t-1}^P - \sum_{i\in N} w_{it})$, but this constraint is dominated by (1b). To ensure that demand on day 1 can be met, it is necessary to include (1e) which allows production on day 0. If $I_0^P = I_{i0}^C = 0$, then $p_0 \ge \sum_{i\in N} d_{i1}$ or the problem is infeasible.

As indicated by constraints (1c), the total amount available for delivery on day t is limited by the amount in inventory at the plant on day $t-1$. The specific amount delivered to customer i is limited by the parameter $D_{it}^{\max}$ in (1g), which is the smaller of the vehicle capacity Q or the cumulative demand from day t to the end of planning horizon τ. Constraints (1h) represent an aggregate relaxation of the routing constraints common to capacitated VRPs. They simply restrict the total amount that can be delivered on day t to a fixed percentage of the total transportation capacity, and provide a hedge against the need for split deliveries. Testing showed that using a value of 80% always yielded feasible solutions. To conclude the formulation, variable bounds are specified in (1i) and (1j).

To obtain the full PIDRP model, several modifications to (1a)–(1j) are needed. First, the second and third terms in (1a) should be omitted and replaced with $\sum_{ijt} c_{ij} x_{ijt}$; next, constraint (1h) should be deleted; finally, routing constraints that take into account the load on the vehicles must be added.

Figure 1 depicts an abbreviated network flow diagram of the PIDRP. The top portion of the figure represents the production facility, denoted by C_0, for days 0 through τ. Instead of demand driving the decisions on each day t for the plant, a series of delivery decisions has to be made for each customer i. The amount delivered, denoted by w_{it}, is combined with customer i's inventory $I_{i,t-1}^C$ at the beginning of day t to satisfy demand, d_{it}. The corresponding flow is shown in the bottom portion of the figure. Because the inventory at the end of the planning horizon at both the plant and the customers' sites is required to be zero, there is no horizontal flow exiting node τ in either network.

The size of model (1) is determined largely by constraints (1f) and (1g) and the number of binary variables, z_{it}^C, all of which grow at a rate proportional to $O(n\tau)$. The majority of the other constraints only grow at a rate proportional to $O(\tau)$. Problem instances with $n\tau \le 30$ can be solved with CPLEX 8.1 in less than 10 min; for example, with $n=5$ and $\tau \in \{2,4,6\}$, corresponding solution times, $t_{\text{cplex},\tau}$, were 4, 196, and 494 s. However, for $n=10$ and $\tau=4$, CPLEX did not converge within 2 hours on any instances, and at the point exhibited an optimality gap of over 10%.

Although not specified in (1i) and (1j), our solution methodology requires that the delivery quantities w_{it} be integral. The following proposition shows that the allocation model always returns integer values for not only w_{it}, but for p_t, I_t^P and I_{it}^C as well.

Proposition 1 *When the setup variables z_t and z_{it}^C are fixed at* 0 *or* 1, *there exists and optimal solution to model* (1a)–(1j) *such that w_{it}, p_t, I_t^P, and I_{it}^C are integral for all $i \in N$ and $t \in T_0$.*

Proof We show that when z_t and z_{it}^C are fixed, the remaining components of the model are equivalent to a pure net-

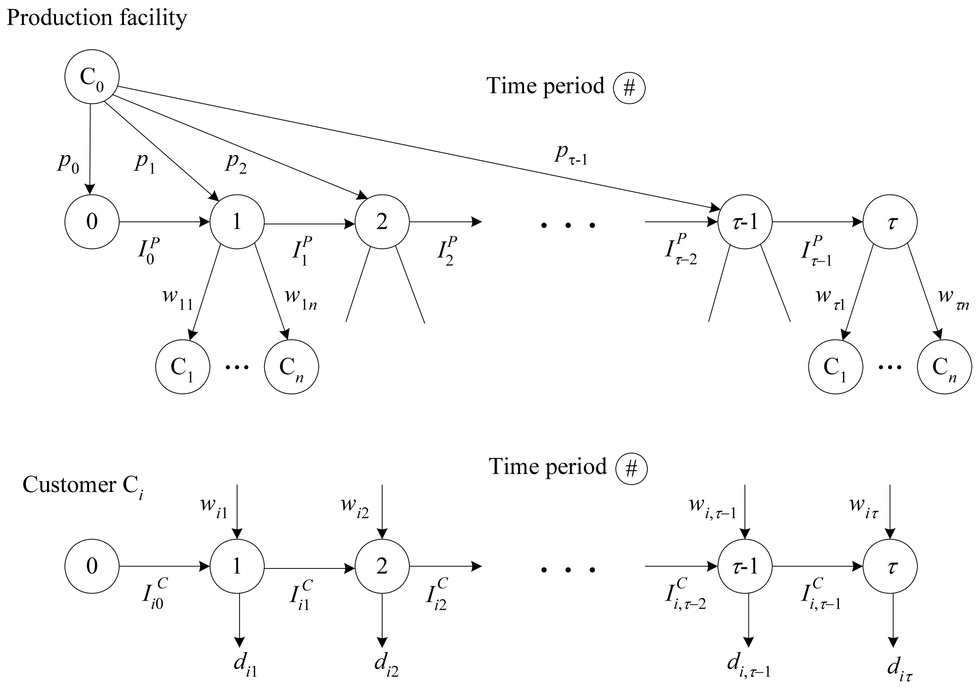

Fig. 1 Network flow representation of production–inventory–distribution problem

work flow problem whose constraint matrix is known to be totally unimodular. The first step is to represent the flows at the plant more accurately by accounting for the fact that production on day t is held in inventory that day and is only available for delivery on day $t+1$. This can be done by creating three inventory nodes at the plant in each period: one for serving customers, a second to bind the flow to customers, and a third to receive the current day's production. This division is shown in Fig. 2, where primes and double primes are used to distinguish the three nodes. At node $1'$, for example, items may be withdrawn and delivered to customers; however, an upper bound of $\lfloor 0.8\theta Q \rfloor$ is placed on the arc from node $1'$ to node $1''$ to ensure adherence to the capacity restrictions (not shown). Items produced on day 1 are channeled to node 1 rather than node $1'$. At the end of the day, whatever inventory remains flows to node $2'$, as indicated by the variable $I_1^{P'}$.

When z_t and z_{it}^C are fixed, the remaining variables are bounded by integer values. Conservation of flow at the nodes with primes is $I_{t-1}^P = I_{t-1}^{P'} - \sum_{i \in N} w_{it}$, which is equivalent to (1c) given the variable bound $I_{t-1}^P \geq 0$ for all t. Conservation of flow at the nodes without primes is essentially (1b) with I_{t-1}^P replaced with its equivalent. The delivery limits in (1h) are enforced by the bounds on the arcs between the nodes with single and double primes. Finally, constraint (1f) represents conservation of flow at the customer sites and is already in pure network form. Thus, we have a bounded pure network flow problem that is guaranteed to have an optimal integral solution. □

4 Solution methodology—tabu search

A two-phase approach is used in the design of our reactive tabu search algorithm for solving the PIDRP. In the first part of phase 1, an initial solution is found by solving the allocation model (1a)–(1j). The results provide customer delivery quantities w_{it} for all $i = 1, \ldots, n$ and $t = 1, \ldots, \tau$. In the second part, these values become the demand for τ independent routing problems. An efficient CVRP subroutine also based on tabu search (Carlton and Barnes 1996) is called to find solutions. It is treated as a "black box" and will not be discussed here. In phase 2, neighborhood search is performed to improve the allocations and routing assignments found in phase 1.

4.1 Initial solution

Absent of the routing component, model (1) represents a pure lot-sizing distribution problem. The two terms $\sum_{it} f_{it}^C z_{it}^C$ and $\sum_{it} e_{it}^C w_{it}$ in (1a) serve as surrogates for the actual routing costs given by $\sum_{ijt} c_{ij} x_{ijt}$. In our implementation, the specification of the coefficients f_{it}^C and e_{it}^C depends on the dimensions of the problem. For instances where $n^2\tau \leq 500$, we set $f_{it}^C = 2c_{i0}$ and $e_{it}^C = 0$ for all i and t; that is, the routing costs on any day t are approximated solely by the cost of a round trip between the depot and customer i. When $n^2\tau > 500$, indicating a fairly large instance, it is not practical to solve (1a)–(1j) with the setup variables z_{it}^C included. In this case, we set $z_{it}^C = f_{it}^C = 0$ and put $e_{it}^C = 2c_{0i}/d_{it}$ for all i and t; that is, the cost of making a delivery to customer i on day t is approximated by the

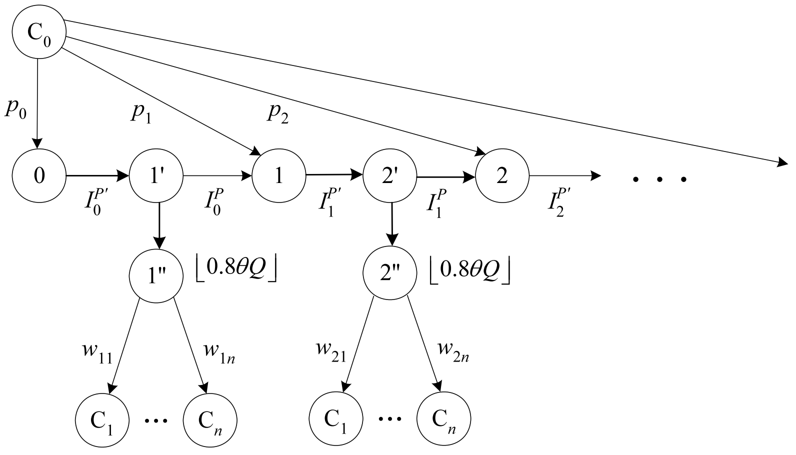

Fig. 2 Network flow with detailed inventory nodes

round trip cost of visiting customer i directly from the depot divided by his demand on day t. Empirically, solutions to the allocation model with these settings were seen to be close to those of the full model when the production and fleet capacities were larger than the total daily demand. This is shown in the computations section of the paper.

Solving model (1) determines the amount of production in each day t ($\bar{p}_t,\ t = 0, 1, \ldots, \tau$), and the quantity delivered to each customer i on each day t ($\bar{w}_{it},\ t = 0, 1, \ldots, \tau;\ i = 1, \ldots, n$). Given these values, an initial solution to the PIDRP is found by calling the VRP subroutine to construct delivery routes for each day t.

4.2 Neighborhood definition

A neighborhood is a set of points that can be reached from the current solution by performing one or more moves that, in general, take the form of insertions, exchanges or replacements. From any incumbent, a new solution is determined by identifying the best solution within its neighborhood. For the PIDRP, we define the neighborhood as all feasible points that can be reached by two types of moves between periods. The first is called a *swap* and involves a (partial) exchange of delivery quantities between two customers i_1 in period t_1 (with quantity $\bar{w}_{i_1t_1}$) and i_2 in period t_2 (with quantity $\bar{w}_{i_2t_2}$), where t_2 is the first period after t_1 in which $\bar{w}_{i_2t_2} > 0$. For customer i_1, the move considers the maximum portion of $\bar{w}_{i_1t_1}$ that can be reassigned to period t_2 without causing a shortage in period t_1 to be exchanged with the full amount $\bar{w}_{i_2t_2}$. (We show below that it is suboptimal to transfer less than the maximum portion of $\bar{w}_{i_1t_1}$). If customer i_1 was not scheduled for a delivery in period t_2, then he must be inserted into one of the θ routes. In general, a swap produces a change in holding costs and a change in transportations costs in periods t_1 and t_2. If the net effect is negative, then the swap is beneficial. Note once again that in the full model, the transportation costs are the sum of the routing costs c_{ij} and hence do not depend on the delivery quantities.

Proposition 2 *Let $\bar{w}_{i_1t_1}$be the planned delivery quantity to customer i_1 in period t_1 and let $\bar{I}^C_{i_1t_1}$ be the net inventory position after demand is satisfied, i.e., $\bar{I}^C_{i_1t_1} = \bar{I}^C_{i_1,t_1-1} + \bar{w}_{i_1t_1} - d_{i_1t_1}$. For any feasible swap between customers i_1 and i_2 in periods t_1 and t_2, the optimal swap amount is equal to* $\min\{\bar{w}_{i_1t_1}, \bar{I}^C_{i_1t_1}\}$.

Proof Consider any swap amount χ for customer i_1 in period t_1 such that $\chi < \min\{\bar{w}_{i_1t_1}, \bar{I}^C_{i_1t_1}\}$. We will show that there exists a swap amount χ^* such that $\chi < \chi^* \le \min\{\bar{w}_{i_1t_1}, \bar{I}^C_{i_1t_1}\}$ that results in a better *move_value*.

The *move_value* is equal to the sum of the change in holding costs for customers i_1 and i_2, the change in transportation costs in periods t_1 and t_2, and the change in holding cost at the plant. For the case where the swap amount is χ, the *move_value* is $-h^C_{i_1}\chi + \varepsilon^1_{t_1} + \varepsilon^1_{t_2} + h^C_{i_2}\bar{w}_{i_2t_2} + \rho^1$, where $\varepsilon^1_{t_1}$ and $\varepsilon^1_{t_2}$ represent the change in transportation costs in period t_1 and t_2, respectively, and ρ^1 is the change in holding cost at the plant. For the latter, we have $\rho^1 = -h^P(\chi - \bar{w}_{i_2t_2})(\tau^1_{t_2} - \tau^1_{t_1})$, where $\tau^1_{t_1} \le \tau^1_{t_2}$ are, respectively, the periods associated with t_1 and t_2 in which production must be adjusted to ensure that the swap is feasible. (Although the results are stated for a single pair of adjustment periods, $\tau^1_{t_1}$ and $\tau^1_{t_2}$, it is easy to show that they hold when production must be adjusted in more than these two periods.)

Similarly, for the case where the swap amount is χ^*, the *move_value* is equal to $-h^C_{i_1}\chi^* + \varepsilon^2_{t_1} + \varepsilon^2_{t_2} + h^C_{i_2}\bar{w}_{i_2t_2} + \rho^2$. (The symbols have the same meaning as above but a superscript 2 is used instead of 1.) To compare transportation costs, the following two cases must now be considered.

Case 1: When $\chi^* < \bar{w}_{i_1t_1}$ (the situation when $\chi^* = \bar{I}^C_{i_1t_1}$) there is no difference in transportation costs, that is, $\varepsilon^1_{t_1} = \varepsilon^2_{t_1}$ and $\varepsilon^1_{t_2} = \varepsilon^2_{t_2}$, because the customers who are scheduled for a delivery in period t_1 are the same as those scheduled for a delivery in t_2.

Case 2: When $\chi^* = \bar{w}_{i_1 t_1}$ it is not necessary to make a delivery to customer i_1 in period t_1 so $\varepsilon_{t_1}^2 \le \varepsilon_{t_1}^1$ in period t_1 and $\varepsilon_{t_2}^2 = \varepsilon_{t_2}^1$ in period t_2.

Because $\chi^* > \chi$, more product is moved to a later period, which implies a greater reduction in the holding cost at the plant: $\rho^2 < \rho^1$. Finally, by noting that the amount of product moved to an earlier period, $\bar{w}_{i_2 t_2}$, is the same in both cases, we have $-h_{i_1}^C \chi + \varepsilon_{t_1}^1 + \varepsilon_{t_2}^1 + h_{i_2}^C \bar{w}_{i_2 t_2} + \rho^1 > -h_{i_1}^C \chi^* + \varepsilon_{t_1}^2 + \varepsilon_{t_2}^2 + h_{i_2}^C \bar{w}_{i_2 t_2} + \rho^2$. □

The second move is called a *transfer* and examines each customer i one at a time and tries to reassign the delivery quantity $\bar{w}_{it_1}$ scheduled for t_1 to the latest period, call it t_2, preceding t_1 in which a delivery is scheduled for at least one customer i; that is, $t_2 = \max\{t : t < t_1, \exists \bar{w}_{it} > 0$ for some $i \in N\}$. In this case, we have the following result.

Proposition 3 *For any customer i, when considering a transfer move between periods t_1 and t_2, it is suboptimal to reassign less than the full amount $\bar{w}_{it_1}$ to period t_2.*

Proof If only a portion of $\bar{w}_{it_1}$ were transferred, then the holding costs would increase in t_1 for customer i, while the transportation costs would remain the same in both periods so there would be no benefit in considering intermediate cases. □

Whether a swap or a transfer, only moves that result in feasible solutions are permitted, so it is necessary to check for violations of the production constraints and the inventory bounds at the plant and the customer sites. Moreover, a VRP must be solved in periods t_1 and t_2 to see whether feasible routes can be found after the move, and if so, what their (optimal) costs are. The value of a move is determined in part by these costs, which must be calculated for each candidate. To begin, *Feasibility_Check_Algorithm* is called to determine whether the move violates any of the production, storage, or vehicle capacity constraints. If not, then *Move_Value_Algorithm* is called to determine the net benefit (see Nananukul 2008).

Complexity of neighborhood A swap involves an exchange in delivery quantities between pairs of customers in two different periods $t_1 < t_2$, so the number of possible candidates in the worst case is $O(n^2)$ in each period. Calculating the portion of $\bar{w}_{i_1 t_1}$ that can be exchanged without causing a shortage in period t_1 can be done in $O(1)$. The feasibility check is $O(n + \tau)$ and determining the move value by solving the VRP takes $O(n^3)$. For τ periods then, the amount of work associated with a swap is $O((n^3 + n + \tau) \cdot n^2\tau) = O(n^5\tau + n^2\tau^2)$. For transfer moves, deliveries to each of the n customers in period t are considered for reassignment to period $t - 1$. In the worst case, there are $O(n)$ possibilities in each period. Taking the feasibility check and the move value computations into account, the amount of work associated with a transfer for all τ periods is $O((n^3 + n + \tau) \cdot n\tau) = O(n^4\tau + n\tau^2)$. Thus, the neighborhood size is $O(n^5\tau + n^2\tau^2)$.

Example of moves Figure 3 depicts a swap between customers 1 and 3 in periods 2 and 3, respectively. The periods are numbered on the far left and the customers on a specific route are represented by circles. The depot (customer 0) is at the start and end of each route, implying that a single vehicle only is required in each period. The parameter values for the five customers in the example are as follows. The holding cost for customer 1 is $h_1^C = 20$, while the holding costs for customers 2 to 5 are $h_2^C = h_3^C = h_4^C = h_5^C = 10$. Initial inventories at all customer sites in period 2 are $I_{11}^C = I_{21}^C = I_{31}^C = I_{41}^C = I_{51}^C = 0$. It is also assumed that the storage capacity at the customer sites is unlimited and that the vehicle capacity $Q = 60$.

Beginning at the depot, route costs are calculated by summing the individual link costs, c_{ij}, between customers i and j on the route, where $c_{01} = 175$, $c_{04} = 10$, $c_{12} = 25$, $c_{02} = c_{20} = 50$, $c_{23} = 75$, $c_{03} = c_{30} = 25$, $c_{45} = 30$, and $c_{51} = 10$. The route cost in period 2, for example, is $c_{04} + c_{45} + c_{51} + c_{12} + c_{23} + c_{30} = 175$. Demand for customers 1 to 5 in period 2 is 8, 7, 7, 6, and 8, respectively, and in period 3 is 4, 4, 8, 6, and 8. These values are shown in the squares below the circles in Fig. 3 only if a delivery is scheduled for the customer in the period of interest.

Before the swap, customer 1 is scheduled for deliveries of 8 and 4 items in periods 2 and 3, respectively, and customer 3 is scheduled for a delivery of 15 items in period 2. After the swap, the amount to be delivered to customer 1 is increased to 12 in period 2 and reduced to 0 in period 3. Also, the amount to be delivered to customer 3 is decreased to 7 in period 2 in accordance with Proposition 2 and a new delivery of 8 units is scheduled in period 3. Thus, for customer 3, the exchange involves the maximum number of items, 8, that can be moved in period 2 without causing a shortage (customer 3 has a demand of 7 in period 2 and currently, 15 items are scheduled for delivery). For customer 1, all 4 units that were to be delivered in period 3 are now scheduled for period 2. Note that before and after the swap the total number of items scheduled for delivery in either period do not exceed the vehicle capacity. Before the swap, a total of 58 and 12 items are scheduled to be delivered in periods 2 and 3, respectively, while after the swap the delivery quantities are 54 and 16.

The results indicate that a big cost reduction is achieved by eliminating the link from the depot to customer 1 ($c_{01} = 175$) in period 3. This cannot be realized, though, by simply rescheduling the 4 items to be delivered to customer 1 in period 3, to period 2 because the vehicle capacity, 60, would

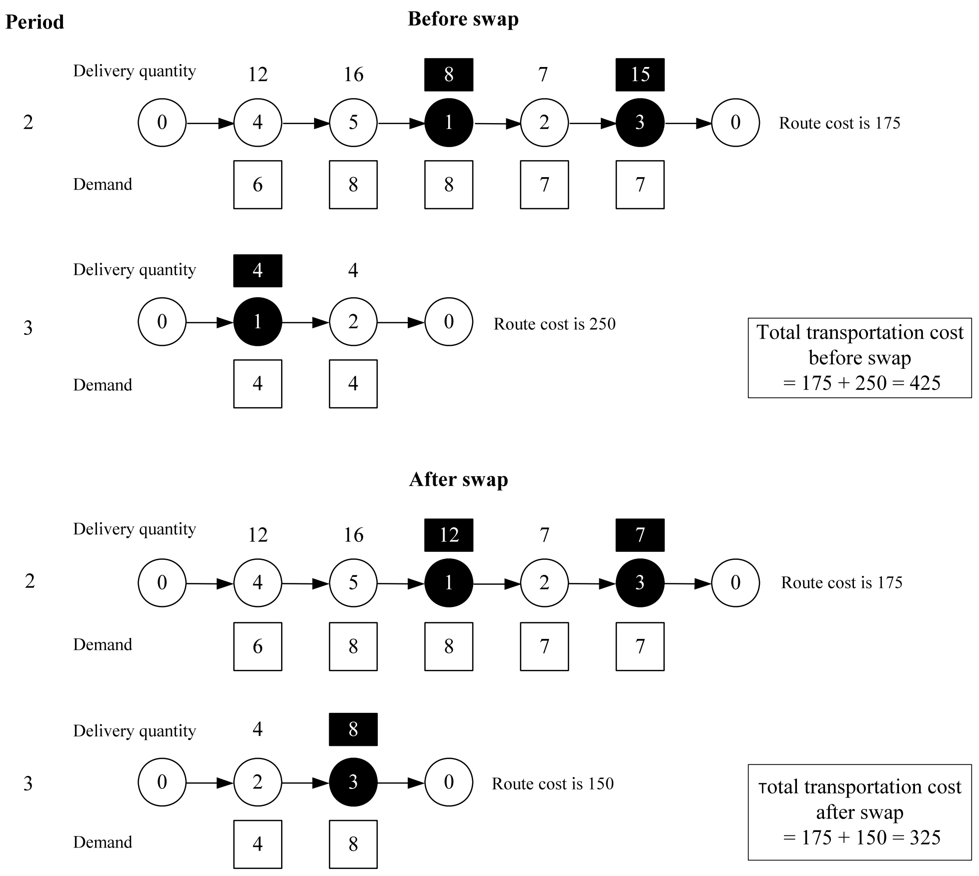

Fig. 3 Example of an exchange or swap move between customers 1 and 3

be exceeded. A swap between periods is required. Before the swap the transportation cost in period 3 is $c_{01} + c_{12} + c_{20} = 250$ and after the swap it is $c_{02} + c_{23} + c_{30} = 150$, where the solution to the VRP for period 3 indicated that it was optimal to insert customer 3 after customer 2. In period 2, there is no change in route so the transportation cost remains the same, as do the holding cots. Calculating the difference between the before and after costs gives the *move_value*, which in this case is -100.

Using the same data and starting with the schedule in the bottom portion of Fig. 3, Fig. 4 gives an example of a single customer transfer move. Before the transfer, customer 2 is scheduled to receive deliveries of 7 and 4 items in periods 2 and 3, respectively, where now $t_1 = 3$ and $t_2 = 2$. The transfer eliminates the delivery in period 3 by moving all 4 items to period 2 in accordance with Proposition 3, and is feasible because the total load on the vehicle only goes up to 58, which is less than the capacity. The move increases the holding cost for customer 2 from 0 of 40, but the transportation cost in period 3 is reduced by 100, giving a *move_value* of -60. More specifically, after the transfer the net change in holding cost is $4 \times 10 = 40$ and is associated with customer 2, the total transportation cost in period 3 is $c_{0,3} + c_{3,0} = 50$, and the net change in transportation cost in period 2 is 0. As a result, the *move_value* is $50 + 40 - 150 = -60$.

4.3 Tabu list and aspiration criterion

Tabu search transcends local optimality by forbidding certain moves on a temporary basis. In general, the process is implemented with a short-term memory structure that proscribes a subset of the moves in a neighborhood. The *tabu list* stores all forbidden moves. Its length, normally called the *tabu tenure*, indicates how many iterations a certain move is forbidden. The *tabu status* indicates whether a move on the tabu list is currently forbidden.

In our implementation, we use a reactive tabu list meaning that the tabu tenure is dynamic. We also specify an aspiration criterion that allows the tabu status of a move to be overridden. Specifically, each entry on the tabu list is a combination of a pair of customers and a pair of time periods represented by the 4-dimensional vector (customer i_1, customer i_2, period t_1, period t_2). Note that for a transfer move where only a single customer is involved, a default value of zero is used for customer i_2. The length of tabu list is kept constant as long as progress is being made, but it is increased when there is no improvement in some fixed number of iterations. Because a move associated with a customer in one

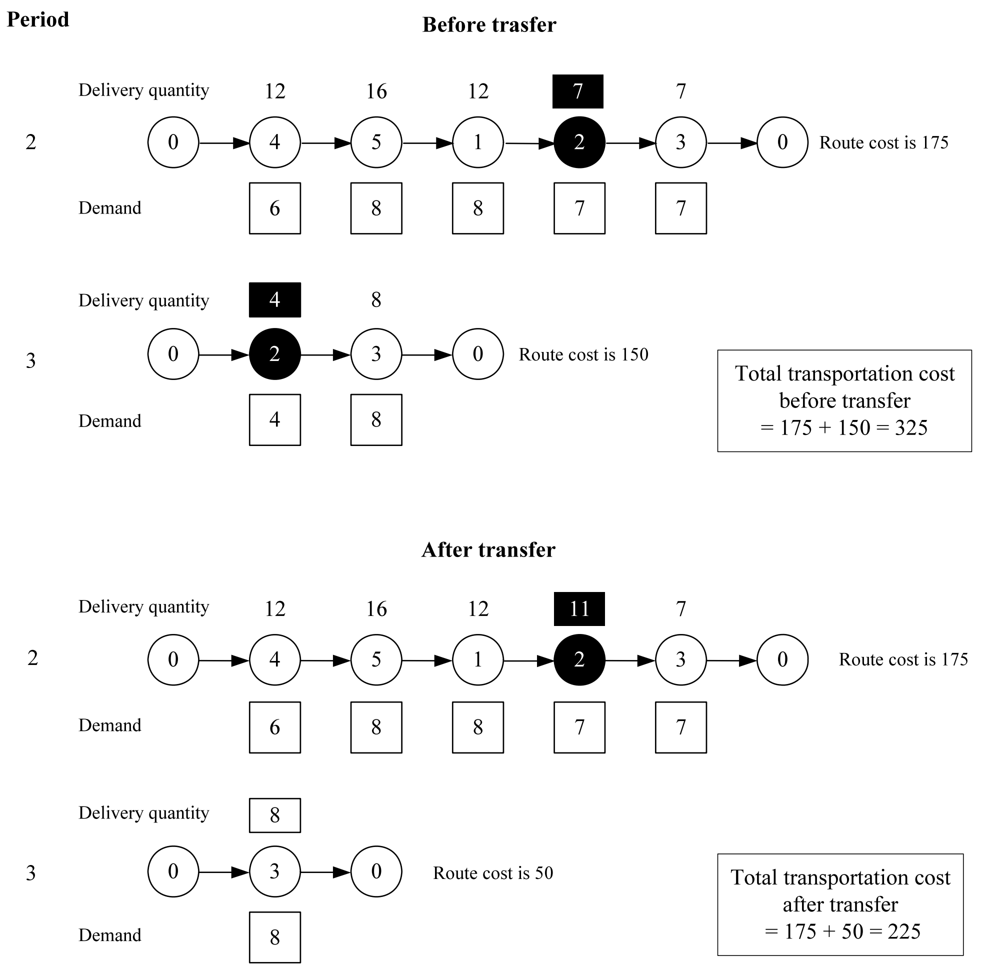

Fig. 4 Example of a transfer move that assigns a delivery to an earlier period

period could create a series of moves in the following periods, thus affecting up to $\tau - 1$ periods, we set the tabu tenure proportional to τ. Its initial value is set to $\tau/2$ and then increased to τ when there is no improvement for 5 iterations. However, when a new incumbent is found, the tabu tenure is set back to $\tau/2$. The process is repeated until the tabu search stopping criteria are met.

Although some implementations allow infeasible moves as an additional means of overcoming local optimality, we do not. Preliminary testing showed that when infeasible moves were considered at each iteration, runtimes became excessive due to the increase in neighborhood size for even small instances with up to 20 customers.

After the current neighborhood is searched and a new incumbent is found, the move that led to the incumbent is added to the tabu list. The tabu status of a move can be overridden, though, when a certain aspiration condition is met. In our case, this condition is associated with a move on the tabu list that gives a better solution than the incumbent.

4.4 Search strategy

Two important strategies for tabu search are intensification and diversification. Intensification focuses on creating solutions that have good attributes (with respect to routing, for example, solutions that include low cost arcs). Diversification is the ability of the algorithm to expand the search by generating solutions that have attributes different from those encountered at previous iterations. These two strategies counterbalance and reinforce each other.

In our algorithm, intensification is implemented by using incentives and diversification by using penalties and short-term memory. The latter is provided by the tabu list. Incentives and penalties are a function of long-term memory and are represented by the $(n \times n \times \tau \times \tau)$-dimensional matrices $\boldsymbol{F^I}$ and $\boldsymbol{F^P}$, respectively. For a move that involves customer i_1 in period t_1 and customer i_2 in period t_2, an element $F^I_{i_1 i_2 t_1 t_2}$ of $\boldsymbol{F^I}$ represents the number of times the set $\{i_1, i_2, t_1, t_2\}$ has been involved in a move that improves the solution. Similarly, an element $F^P_{i_1 i_2 t_1 t_2}$ of $\boldsymbol{F^P}$ represents the number of times the set $\{i_1, i_2, t_1, t_2\}$ has been involved in a nonimproving move. Any move associated with these sets is either rewarded or penalized in accordance to the values of $F^I_{i_1 i_2 t_1 t_2}$ and $F^P_{i_1 i_2 t_1 t_2}$. With respect to the above example, if a candidate swap involves customers i_1 and i_2 in periods t_1 and t_2, then the actual value of the move is $move\_value - \rho^I F^I_{i_1 i_2 t_1 t_2} + \rho^P F^P_{i_1 i_2 t_1 t_2}$ rather than *move_value* alone, where ρ^I and ρ^P represent incentive and penalty multipliers (see next section for more detail).

Implementing exhaustive search for the neighborhood defined in Sect. 4.2 is possible for small size problems ($n^2\tau \le 500$). This was the approach that we took during algorithmic development. For larger instances ($n^2\tau > 500$), including those in the Boudia et al. (2007) data sets, it is not practical to examine all candidate moves. To reduce the computational effort, we randomly select a subset of moves and then adaptively decide according to the progress of the algorithm, which of two rules to follow. The first rule places a 4 to 1 emphasis on transfers over swaps and is used when the most recent tabu iteration resulted in an overall cost reduction. The second rule reverses the emphasis and is used when the most recent tabu iteration did not improve upon the incumbent. More detail is provided in Nananukul (2008).

At this point, all moves on the candidate list are processed and the one with the best *move_value* is selected. The relevant production and inventory levels are updated, the stopping criteria are checked, and if not met, the next tabu iteration is performed.

4.5 Algorithm description

Figure 5 summarizes the major steps of the tabu search algorithm for large instances. In the first step, an initial feasible solution is created with the procedure described in Sect. 4.1. The result is saved as both the *current solution* and the *best solution* found. Then the algorithm iterates until either a prespecified *max_iterations* is reached or there is no improvement in *max_no_improve* iterations. Of all the admissible candidates, the move selected at each iteration is the one that has the smallest value of $move\_value + move\_penalty + move\_incentive$. A move is considered admissible if it is not on the tabu list or its tabu status has been overridden by the aspiration criterion. For the current iteration, once the best move is found it is executed and the tabu data structures are updated.

The parameters and data structures used in the algorithm are as follows:

- *num_iteration*: current iteration number
- *max_iterations*: maximum number of iterations allowed
- *curr_soln*: current solution after executing the best move
- *curr_cost*: cost of current solution
- *best_soln*: incumbent solution
- *best_cost*: cost of incumbent solution
- *move_value*: difference in the cost before and after the swap move
- *best_move_value*: lowest *move_value* among all candidates
- *freq_matrix*: long-term memory function that stores the frequency of each possible move
- *move_penalty*: additional penalty imposed by long-term memory *freq_matrix* $\boldsymbol{F^P}$; when a swap involves customer i in period k and customer j in period l, $move\_penalty = \rho^P F^P_{ijkl}$, ρ^P is set to 5% of the objective function value of the initial feasible solution divided by *max_iterations*. This value gives an approximation of the average amount of penalty per iteration.
- *move_incentive*: additional incentive imposed by long-term memory *freq_matrix* $\boldsymbol{F^I}$; when a swap involves customer i in period k and customer j in period l, $move\_incentive = -\rho^I F^I_{ijkl}$, where ρ^I is set to ρ^P because we penalize a move that results in a non-improved solution the same amount as the incentive of the same move that results in an improved solution.
- *best_move_penalty*: *move_penalty* for the best move of all candidates
- *best_move_incentive*: *move_incentive* for the best move of all candidates
- *no_improve*: number of consecutive iterations during which no better solutions are found
- *max_no_improve*: maximum number of consecutive no improvement iterations allowed
- *tabu_list*: short-term memory function that stores a list of moves that are forbidden
- *tabu_size*: total number of iterations for which a move is held on the *tabu_list*
- *candidate_rule*: candidates that are considered during each iteration in accordance with rules 1 and 2
- *candidate_move*: solution that results when the moves associated with *candidate_rule* are applied
- *Admissible_move*: candidate move that either satisfies the aspiration criterion or is not on the *tabu_list*

When evaluating each candidate move, it may be possible to achieve a further cost reduction by adjusting the production (and inventory) levels which are not necessarily optimal for the updated delivery schedule. Given the updated values of w_{it}, Proposition 1 allows us to solve a linear program to find the optimal production and corresponding inventory levels; however, doing so at each iteration is too costly because of the large number of possible moves, so in the implementation exact solutions are computed once every five iterations. At all other times, we attempt to improve the solution by calling the heuristic *Production_Level_Adjustment_Algorithm*. See Appendix for the details.

Fig. 5 Tabu search algorithm

```
1.  Generate initial feasible solution and save it as curr_sol and best_soln.
2.  Calculate the cost of curr_sol and set best_cost = curr_cost.
3.  Initialize tabu_list and freq_matrix, set no_improve = 0 and set candidate_rule = 1.
4.  Do{
        best_move_value = $\infty$
        for (all candidate_move)
        {
            if(candidate_move is not on tabu_list or if it satisfies aspiration criterion )
            {
                if(Feasibility_Check_Algorithm return <true>)
                {
                    call Move_Value_Algorithm, store result in move_value
                    if(move_value + move_penalty + move_incentive <  best_value +
                        best_move_penalty + best_move_incentive)
                    {
                        best_move_value = move_value
                        best_move_penalty = move_penalty
                        best_move_incentive = move_incentive
                        best_move = current_move
                    }
                }
            }
        }
        execute best_move
        curr_cost = curr_cost + best_move_value
        update tabu_list and freq_matrix and adjust production levels
        if(curr_cost < best_cost)
        {
            best_cost = curr_cost
            best_soln = curr_soln
            no_improve = 0
            set candidate_rule = 1
        }else{
            no_improve = no_improve + 1
            set candidate_rule = 2
        }
    }While(num_iteration < max_iterations or no_improve < max_no_improve)
```

5 Lower bound computations

To gauge the quality of the solutions provided by tabu search, lower bounds on the true optimum are needed. The simplest way to obtain a lower bound is to solve the LP relaxation of the full model. Initial testing on small instances showed gaps of roughly 30% to 260%—not a useful measure. As an alternative, we developed a procedure based on the allocation model that gives much better results.

To obtain a lower bound on the true optimum, and hence on the feasible solutions provided by tabu search, a valid relaxation of the full model must be solved. To formulate such a relaxation, we begin with (1a)–(1j) and make several modifications. First, the cost coefficients f_{it}^C for each customer i are set to the shortest distance to either the depot or any of the other customers, that is, $f_{it}^C = \min\{c_{ij} : j \in N_0\backslash(i)\}$. Next, the cost coefficients associated with w_{it} are set as follows: $e_{it}^C = c_0/Q$, where $c_0 = \min\{c_{0j} : j \in N\}$. Finally, the right-hand side of (1h) is increased to $Q\theta$ in each period t and the corresponding allocation model solved to get ϕ_{LB}.

Proposition 4 *The solution to the modified allocation model* (*lower bounding model*) *provides a valid lower bound on the true optimum to the PIDRP*; *that is*, $\phi_{\mathrm{LB}} \le \phi_{\mathrm{PIDRP}}$.

Proof First note that any solution that is feasible to the full model is feasible to the modified allocation model. This follows because (1h) with its right-hand side set to $Q\theta$ is a valid relaxation of the routing constraints in the full model. We now show that the modified costs $\sum_{it} f_{it}^C z_{it}^C + \sum_{it} e_{it}^C w_{it}$ underestimate the true costs $\sum_{ijt} c_{ij} x_{ijt}$, which in any feasible solution contain one arc cost for each customer that receives a delivery in period t and one arc cost for each vehicle used in period t. By design, the first term underestimates the individual customer arc costs and the second underestimates the vehicle arc costs from the depot. With respect to the vehicle arc costs, note that the number of vehicles required to deliver $\Sigma_{it} w_{it}$ units in period t is at least $\lceil \Sigma_{it} w_{it}/Q \rceil$. Removing the integer requirement and multiplying by c_0 gives the desired result. □

A slightly improved lower bound can be obtained when the holding costs are small and production setup costs are large with respect to the vehicle cost, which is the case here.

Lemma 1 *Assume that the minimum production setup cost $f^P \equiv \min\{f_t : t = 0, 1, \dots, \tau - 1\}$ is much larger than the cost c_0 of using a vehicle in any period, and that the customer holding costs are negligibly small, i.e., $h_i^C \approx 0, \forall i \in N$. Let $R(t) = \lceil \sum_i w_{it}/Q \rceil - \sum_i w_{it}/Q$ and $r(t) = \sum_i w_{it}/Q - \lfloor \sum_i w_{it}/Q \rfloor$ be the fraction of a vehicle corresponding to the difference between the rounded up and rounded down number of vehicles in a solution in period t, respectively. Under the stated conditions, if*

$$c_0 r(t) + (t' - t)h^P r(t)Q - (t'_P - t_P)h^P r(t)Q - c_0 - c_0 R(t') \geq 0, \quad \forall t' > t, \tag{2}$$

$$c_0 r(t) - (t - t')h^P r(t)Q + (t_P - t'_P)h^P r(t)Q - c_0 - c_0 R(t') \geq 0, \quad \forall t' < t, \tag{3}$$

then the lower bound ϕ_{LB} can be increased by rounding up the number of vehicles used in period t and multiplying the corresponding fraction $R(t)$ by c_0. Summing over all periods gives the following adjusted lower bound

$$\phi_{\text{LB}} \leftarrow \phi_{\text{LB}} + \sum_{t=1}^{\tau} c_0 \left(\left\lceil \sum_i w_{it}/Q \right\rceil - \sum_i w_{it}/Q \right). \tag{4}$$

Proof The assumption that $f^P \gg c_0$ implies that it is never advantageous to set up for production in order to avoid using an additional vehicle in any period. Therefore, the result follows if this is the only feasible option for eliminating a vehicle in some period t.

Rounding up number of vehicles used in each period will not change the optimality of the solution if the increase in cost, $c_0 \cdot R(t)$, in period t does not exceed the cost of rescheduling the delivery quantity $Q \cdot r(t)$ to another period t'.

Case 1: $t' > t$

Given ϕ_{LB}, the objective function value when rounding up the number of vehicles used without rescheduling in both periods t and t' is

$$\text{Cost}_1 = c_0 R(t) + c_0 R(t') + \phi_{\text{LB}}. \tag{5.1}$$

Now, let t_P and t'_P be the periods where production of $Q \cdot r(t)$ takes place before and after rescheduling, respectively. The objective function value after rescheduling is bounded below by

$$\begin{aligned}\text{Cost}_2 = \phi_{\text{LB}} &- c_0 r(t) + (t' - t)h^P r(t)Q \\ &- (t'_P - t_P)h^P r(t)Q \\ &- (t' - t)\max\{h_i^C, \ i \in N\} r(t)Q \\ &+ c_0 \left(\left(\sum_i w_{it'} + r(t)Q \right)/Q - \sum_i w_{it'}/Q \right).\end{aligned} \tag{5.2}$$

Thus, if $\text{Cost}_2 \geq \text{Cost}_1$ for all $t' > t$, then rounding up the number of vehicles used in period t does not change the optimality of the solution.

Case 2: $t' < t$

Using the same logic, the objective function value when rounding up the number of vehicles used after rescheduling is bounded below by

$$\begin{aligned}\text{Cost}_3 = \phi_{\text{LB}} &- c_0 r(t) - (t - t')h^P r(t)Q \\ &+ (t_P - t'_P)h^P r(t)Q \\ &+ (t - t')\min\{h_i^C, \ i \in N\} r(t)Q \\ &+ c_0 \left(\left(\sum_i w_{it'} + r(t)Q \right)/Q - \sum_i w_{it'}/Q \right).\end{aligned} \tag{5.3}$$

The implication of (5.1) and (5.3) is that rounding up the number of vehicles used in period t does not change the optimality of the solution if $\text{Cost}_3 \geq \text{Cost}_1$ for all $t' < t$.

Now, when $h_i^C \approx 0$, $\forall i \in N$, (5.1), (5.2), and (5.3) give

Case 1: $t' > t$

$$c_0 r(t) + (t' - t)h^P r(t)Q - (t'_P - t_P)h^P r(t)Q - c_0 - c_0 R(t') \geq 0.$$

Case 2: $t' < t$

$$c_0 r(t) - (t - t')h^P r(t)Q + (t_P - t'_P)h^P r(t)Q - c_0 - c_0 R(t') \geq 0.$$ □

A second and third improvement in the lower bound can be obtained by taking into account the fact that exactly $\lceil \Sigma_{it} w_{it}/Q \rceil$ vehicles must return to the depot in any solution to the full model. A fourth improvement can be obtained by recognizing that some minimum number of vehicles must be used in each period in order to meet demand. The proofs of Lemmas 2 and 3 similar to the proof of Lemma 1 and can be found in Nananukul (2008).

Lemma 2 *Let $y_t = \lceil \sum_{it} w_{it}/Q \rceil$ be the minimum number of vehicles used in period t and let $c_i = \min\{c_{ij} : j \in N\}$ be the least cost transition for customer i to another customer. Also, let $N^*(t) = \{i : z_{it}^C = 1,\ f_{it}^C = c_{i0}\}$ be the set of customers who receive a delivery in period t and whose corresponding cost is c_{i0}, and let $\Delta c_i = c_i - c_{i0}$ be the opportunity cost of not using the minimum cost arc in a solution. Now, order the $n^*(t) = |N^*(t)|$ customers such that $\Delta c_{i_1} \le \Delta c_{i_2} \le \cdots \le \Delta c_{i_{n^*(t)}}$ and let $l(t) = n^*(t) - y_t$ be the excess number of customers whose delivery cost is c_{i0}. Assume that the minimum production setup cost $f^P = \min\{f_t : t = 0, 1, \ldots, \tau - 1\}$ is much larger than the cost c_0 of using a vehicle in any period. If*

(i) $\Delta c_i \le (t'-t)w_{it}h^P - (t'_P - t_P)w_{it}h^P - (t'-t)w_{it}h_i^C - f_{it}^C, \forall t' > t$, *and*
(ii) $\Delta c_i \le -(t-t')w_{it}h^P + (t_P - t'_P)w_{it}h^P + (t-t')w_{it}h_i^C - f_{it}^C, \forall t' < t$,

then the lower bound ϕ_{LB} can be adjusted as follows

$$\phi_{\text{LB}} \leftarrow \phi_{\text{LB}} + \sum_{t=1}^{\tau} \sum_{k=1}^{l(t)} \Delta c_{i_k}. \tag{6}$$

When the holding cost at customer i is negligibly small, that is, $h_i^C \approx 0$ for all $i \in N$ (the case for our data), then conditions (i) and (ii) in Lemma 2 become

(i) $\Delta c_i \le (t'-t)w_{it}h^P - (t'_P - t_P)w_{it}h^P - f_{it}^C, \forall t' > t$,
(ii) $\Delta c_i \le -(t-t')w_{it}h^P + (t_P - t'_P)w_{it}h^P - f_{it}^C, \forall t' < t$

Lemma 3 *Expanding on the notation introduced in Lemma 2, let $L^*(t) = \{i : z_{it}^C = 1, f_{it}^C \ne c_{i0}\}$ be the set of customers who receive a delivery in period t and whose corresponding cost is not c_{i0}. For $i \in L^*(t)$, let $\Delta c_i = c_{i0} - c_i$ be the increase in cost that would result if customer i was assigned the arc connected to the depot in a solution instead of c_i. Assume that $y_t > n^*(t)$ and let $l(t) = y_t - n^*(t)$ be the number of customers whose delivery cost should be set to c_{i0}. Now identify the $l(t)$ smallest values of Δc_i for $i \in L^*(t)$; that is, $\Delta c_{i_1} \le \Delta c_{i_2} \le \cdots \le \Delta c_{i_{l(t)}}$, where $i_1, \ldots, i_{l(t)} \in L^*(t)$. Assuming that the minimum production setup cost $f^P = \min\{f_t : t = 0, 1, \ldots, \tau - 1\}$ is much larger than the cost c_0 of using a vehicle in any period, if*

(i) $\Delta c_i \le (t'-t)w_{it}h^P - (t'_P - t_P)w_{it}h^P - (t'-t)w_{it}h_i^C - f_{it}^C, \forall t' > t$, *and*
(ii) $\Delta c_i \le -(t-t')w_{it}h^P + (t_P - t'_P)w_{it}h^P + (t-t')w_{it}h_i^C - f_{it}^C, \forall t' < t$,

then the lower bound ϕ_{LB} can be adjusted as follows

$$\phi_{\text{LB}} \leftarrow \phi_{\text{LB}} + \sum_{t=1}^{\tau} \sum_{k=1}^{l(t)} \Delta c_{i_k}. \tag{7}$$

Lemma 4 *Define $v(t)$ as the minimum number of vehicles needed in period t. Let $i^* = \arg\min\{c_{i0}, i \in N\}$ and let $\Delta c_{it} = c_{0i} - c_0$ be the incremental cost for customer $i \in N \backslash \{i^*\}$ in period t for not using the minimum cost arc (with cost c_0) from the depot in a solution. For period t, identify the $v(t) - 1$ smallest values of Δc_{it} and the corresponding customers. The following adjusted lower bound is valid for the modified allocation model*

$$\phi_{\text{LB}} \leftarrow \phi_{\text{LB}} + \sum_{t=1}^{\tau} \sum_{k=1}^{v(t)-1} \Delta c_{i_k t}. \tag{8}$$

Proof The number of outbound arcs from the depot to customers in each period must be at least $v(t)$ in any feasible solution to the full model. Because each customer can be visited only once in each period, this implies that the outbound arcs from the depot cannot be the same for all vehicles. By selecting the $v(t) - 1$ smallest incremental costs associated with the arcs leaving the depot in period t, we increase ϕ_{LB} by $\sum_{k=1}^{v(t)-1} c_{i_k t}$ while simultaneously reducing it by $c_0(v(t) - 1)$. Summing over the planning horizon gives a valid lower bound on the total vehicle costs because we are only adjusting the cost for one less than the minimum number of vehicles required in each period. □

The bound improvement in (8) can be calculated in a preprocessing step. The other improvements given in (4), (6), and (7) can be obtained directly by solving the allocation model after several additional modifications are made. For (4), it would be necessary to replace $\Sigma_{it} w_{it}/Q$ in the objective function with a nonnegative integer variable y_t ($t \in T$), and set the right-hand side of (1h) to Qy_t. For (6), it would be necessary to introduce $n \times \tau$ binary variables ξ_{it} ($i \in N, t \in T$), to identify whether cost coefficient c_{i0} or c_i is to be used for customer i in period t, and $n \times \tau$ constraints of the form $x_i + \xi_{it} \le 1$ to ensure that at most one coefficient is selected. A similar modification would be required for (7). Testing has shown that these additions can measurably extend computation times so we have relied on Lemmas 1–4 to improve the initial bound.

6 Computational results

All computations were performed on a 2.53 GHz processor with 512 MB of RAM. The optimization models and the tabu search code were implemented in Java Netbean 4.1 and linked to the CPLEX 8.1 libraries. CPU times were obtained through both CPLEX and the time function in Java. For testing purposes, we used the three data sets provided by Boudia et al. (2007) containing 30 instances of 50, 100, and 200 customer problems, all with a 20-period planning horizon and holding costs $h^P = 1$, $h_i^C = 0$ for all $i \in N$. These instances were randomly generated on a 100×100 Euclidean grid. For each customer i, demand was uniformly distributed between 0 and the storage capacity $I^C_{\max,i}$, and for each period t was positive with a probability that varied from 0.5 to 1. The value of $I^C_{\max,i}$ depended on i and the number of customers in the specific data set. The vehicle capacities were $Q_{50} = 8000$, $Q_{100} = 8000$, $Q_{200} = 12{,}000$, and the number of vehicles were $\theta_{50} = 5$, $\theta_{100} = 9$, $\theta_{200} = 13$. The first set of experiments was designed to gauge CPLEX's performance on the full problem and did not make use of the Boudia et al. data sets.

6.1 Preliminary testing

To determine the limits of CPLEX to solve the PIDRP directly, we randomly generated 10 instances on a 100×100 grid. The first step was to fix the number of customers $n \in \{5, 10, 15, 20\}$ and the number of time periods $\tau \in \{2, 4, 6, 8\}$ by selecting a range of values from these sets. The remaining parameter values and the output can be found in Nananukul (2008).

To summarize, the results indicated that when either the number of customers or the number of periods is increased, the runtime increased dramatically in almost all cases. This was to be expected because the PIDRP requires the solution of a CVRP in each period. When the number of customers was increased, the number of binary variables and constraints in each CVRP increased by $O(n^2)$ in the worst case, implying a significant increase in both the number of binary variables and the number of constraints in the PIDRP model. In all instances, the LP solution was obtained in less than a second but CPLEX's MIP solver required anywhere from 4 to > 7200 s, the runtime limit imposed. The instances that could not be solved by CPLEX were the ones that had $n\tau \geq 40$.

To test the performance of the VRP subroutine (Carlton and Barnes 1996) we created eight single-period problem instances in which the numbers of customers ranged from 5 to 150. The same generating procedure used for the PIDRP instances were used here but the inputs were limited to customer locations, customer demand, vehicle capacity, and number of vehicles.

The results indicated that the VRP subroutine gives high quality solutions, denoted by ϕ_{VRP}, within a much smaller amount of time compared to CPLEX, and is especially effective for the larger instances with 30 or more customers. For the small size problems ($n = 5, 10$), the VRP subroutine provided the same solution as CPLEX within about the same runtime. The gap between ϕ_{VRP} and ϕ_{cplex} was less than or equal to zero in all but one case. Also, the VRP runtimes were substantially less than those of CPLEX, with the difference increasing dramatically as the number of customers increased.

We also compared the phase 1 solutions obtained from the allocation model (1) with route optimization in each period with the solutions from CPLEX. It was seen that phase 1 reliably provided high quality solutions in a negligible amount of time compared to CPLEX. In several instances the optimal solution was found by the allocation model and in the other cases it was at most 6% off; however, for some instances, it provided better results than CPLEX. Also, the phase 1 runtimes were no more than a few seconds, not increasing much with either n or τ.

6.2 Small instances

Table 1 summarizes the results for the 50-customer, 20-time period instances. Columns 2 and 3 give the best GRASP solutions (ϕ_{grasp}) from Boudia et al. and the corresponding runtimes (t_{grasp}) for 500 iterations. Their computations were performed on a 2.8 GHz computer with 512 MB of RAM running Windows XP. As a word of caution, it is always problematic to make comparisons across different platforms and different algorithms. There is also some arbitrariness in establishing the termination criteria for metaheuristics like GRASP or tabu search because it is rarely evident when additional iterations will not lead to improvement. Therefore, all runtimes and other performance measures presented below should be interpreted with this in mind.

Columns 4, 5, and 6 are phase 1 solutions (ϕ_{Ph1}), runtimes (t_{Ph1}), and percent deviation from the best GRASP solutions, respectively. Column 7 gives the time to solve the allocation model with CPLEX. An optimality gap of 0% was used in all cases. Columns 8, 9, and 10 report the phase 2 results. In those computations, tabu search was allowed to run for a maximum of 50 iterations but was terminated earlier when no improving solution was found in 5 consecutive iterations. The 50-iteration limit was reached in only two instances. Column 9 gives the iteration on which the best phase 2 solution was found.

From Table 1, we see that the phase 1 solutions are 10.7% better, on average, than the best known solutions, and were obtained with significantly smaller runtimes. In fact, t_{Ph1} was 81% less than t_{grasp}, on average, whereas ϕ_{Ph1} was better than ϕ_{grasp} in all cases. The gap between ϕ_{Ph1} and ϕ_{grasp} ranged from 4 to 17%. As to be expected, ϕ_{Ph2} was smaller than ϕ_{Ph1} in all cases as well, providing an average improvement of 5.8%, as reported in the last column. For the GRASP, runtimes averaged 97.7 s compared to 18.4 s for phase 1 and 433.4 s for phase 2. About 75% of t_{Ph2} resulted from calling CPLEX's LP solver to determine the optimal updated production quantities for each candidate move. If this option is omitted and only the *Production_Level_Adjustment_Algorithm* is used for this purpose, the GRASP is about 1% faster than our two-phase approach. The 5.8% improvement between phase 1 and phase 2, however, drops to approximately 4%.

Table 1 Comparison of solutions for problem instances with 50 customers and 20 periods

Prob. no.	(ϕ_{grasp}) Best GRASP solution	(t_{grasp}) Runtime (s)	(ϕ_{Ph1}) Phase 1 solution	(t_{Ph1}) Runtime (s)	$[(\phi_{\text{Ph1}}-\phi_{\text{grasp}})/\phi_{\text{grasp}}]$ 100%	(t_{ALLOC}) Alloc time (s)	(ϕ_{Ph2}) Phase 2 solution	Iteration ϕ_{Ph2} found	(t_{Ph2}) Runtime (s)	$[(\phi_{\text{Ph2}}-\phi_{\text{Ph1}})/\phi_{\text{Ph1}}]$100%
1	440,505	102.36	418,570	9.18	−5	5.6	399,125	25	226.45	−5
2	448,695	138.17	391,366	16.65	−13	13.0	373,581	29	385.55	−5
3	419,730	95.35	385,897	20.18	−8	16.25	353,058	22	234.16	−9
4	456,398	68.42	401,851	18.17	−12	17.0	361,309	19	216.41	−10
5	434,466	99.37	396,977	20.29	−8	13.89	365,035	16	725.57	−8
6	452,564	98.07	382,417	16.68	−15	14.6	368,082	50	467.14	−4
7	436,812	98.90	388,935	20.21	−11	14.97	369,963	30	404.73	−5
8	420,935	87.32	383,705	18.07	−9	15.91	370,822	30	329.34	−3
9	434,789	142.54	391,442	13.85	−10	15.25	379,401	26	392.90	−3
10	436,221	158.40	388,957	15.30	−11	15.37	370,805	25	408.05	−5
11	433,890	81.77	384,722	20.84	−11	17.83	357,107	30	316.56	−7
12	452,705	85.83	382,746	19.90	−15	16.78	355,199	45	585.90	−7
13	440,771	99.85	381,645	17.74	−13	14.58	366,547	20	340.97	−4
14	419,412	84.45	399,040	20.14	−5	17.23	364,115	35	335.42	−9
15	453,875	86.67	403,862	18.41	−11	14.43	367,659	30	555.09	−9
16	457,310	91.70	377,530	19.16	−17	15.68	360,534	25	493.80	−5
17	455,663	91.08	405,292	16.84	−11	13.33	398,442	20	153.88	−2
18	441,685	88.53	404,254	19.25	−8	16.0	368,600	39	574.6	−9
19	418,896	104.27	394,187	17.52	−6	14.07	377,073	32	488.67	−4
20	452,183	94.12	403,547	16.37	−11	13.43	372,141	39	635.94	−8
21	409,677	78.27	393,013	16.75	−4	13.61	374,743	16	160.18	−5
22	429,116	108.26	380,357	20.61	−11	17.61	347,449	50	1031.96	−9
23	443,184	106.99	387,351	16.52	−13	12.76	362,619	31	794.67	−6
24	426,113	101.40	388,221	20.34	−9	16.37	375,022	23	232.29	−3
25	462,245	86.99	386,524	19.45	−16	16.01	374,926	19	273.34	−3
26	442,029	82.15	397,620	20.29	−10	16.65	366,733	36	811.39	−8
27	444,695	85.69	385,085	20.30	−13	16.70	375,261	12	156.74	−3
28	449,894	187.46	388,354	22.84	−14	19.31	373,155	26	230.68	−4
29	461,555	93.93	400,043	19.09	−13	15.62	379,320	25	543.83	−5
30	434,006	93.73	397,217	20.00	−8	16.55	369,223	33	495.71	−7

The final steps of our methodology involved path relinking and solving the modified version of the allocation model (1) to obtain a lower bound ϕ_{LB}. The results for the 30-customer instances are reported in Table 2. Path relinking is a procedure used to explore the opportunity to improve the solution obtained from any methodology that provides multiple candidate solutions. It is based on the fact that paths between solutions give rise to neighborhoods that contain new solutions with attributes similar to those of the endpoints. Our algorithm explores paths between all pairs of *elite* solutions that were uncovered during tabu search. An elite solution is defined as the first improved solution following any unimproved solution.

The second column of Table 2 lists the number of elite solutions found, which averaged 4.4. Columns 3–5 give the path relinking solution, the corresponding runtime, and the gap with the phase 2 solution, respectively. In most cases, the latter is either negative or negligibly small calling into question the effort required to apply this procedure. The lower bound, ϕ_{LB}, obtained from the modified version of the allocation model is reported in column 6. The adjusted lower bound, ϕ_{ALB}, is presented in column 7 and was obtained by applying (4), (6)–(8). On average, it showed an improvement over the lower bound, ϕ_{LB}, of 0.5% with a standard deviation of 1.23. The gap between the best solution and the adjusted lower bound averaged 12.8% and is

Table 2 Path relinking and lower bound results for instances with 50 customers and 20 periods

Prob. no.	No. elites solns	(ϕ_{PR}) Solution from path relinking	(t_{PR}) Time (s)	$[(\phi_{\mathrm{PR}} - \phi_{\mathrm{Ph2}})/\phi_{\mathrm{Ph2}}]$ 100%	(ϕ_{LB}) from LB model[a]	(ϕ_{ALB}) Adjusted LB	$[(\phi_{\mathrm{Best}} - \phi_{\mathrm{ALB}})/\phi_{\mathrm{ALB}}]$100%	(ϕ_{LP}) LP soln	(t_{LP}) LP time (s)	$[(\phi_{\mathrm{ALB}} - \phi_{\mathrm{LP}})/\phi_{\mathrm{LP}}]$100%
1	8	398,795	180	−0.08	324,463	326,630	22	218,243	4.72	50
2	2	373,374	71	−0.06	325,469	326,324	14	219,206	4.69	49
3	6	353,058	460	0	326,365	329,607	7	217,697	5.56	51
4	6	361,176	300	−0.04	328,090	328,467	10	220,366	5.84	49
5	5	364,819	290	−0.06	324,451	325,044	12	220,178	3.8	48
6	3	368,082	222	0	329,875[b]	332,449	11	221,852	5.3	50
7	9	369,963	720	0	326,712[b]	328,176	13	218,911	4.08	50
8	1	370,822	*	0	324,958	325,893	14	216,969	6.92	50
9	2	379,379	36	−0.01	325,737	326,277	16	218,951	7.38	49
10	5	370,655	260	−0.04	325,846[b]	326,483	14	217,392	4.92	50
11	7	354,025	540	−0.86	329,208	330,539	7	217,596	10.45	52
12	5	354,981	180	−0.06	320,151[b]	326,316	9	215,297	6.33	52
13	3	365,432	208	−0.30	326,512	327,017	12	217,946	6.38	50
14	3	363,404	184	−0.20	321,141	323,498	12	216,514	4.74	49
15	4	367,659	56	0	325,857	326,339	13	219,531	5.69	49
16	1	360,534	*	0	326,324	328,384	10	216,828	5.33	51
17	5	398,442	120	0	328,513	329,996	21	221,842	5.94	49
18	5	368,533	189	−0.02	323,263	324,394	14	219,581	4.08	48
19	6	377,255	480	0.05	326,077	327,712	15	216,825	5.66	51
20	9	372,361	670	0.06	325,181	326,112	14	218,650	4.31	49
21	2	375,228	30	0.13	326,764	327,367	14	218,483	9.36	50
22	7	347,329	810	−0.03	322,818	324,015	7	214,260	6.61	51
23	4	362,619	220	0	326,345[b]	327,824	11	219,942	5.44	49
24	2	375,609	100	0.16	320,055[b]	321,520	17	214,406	4.98	50
25	3	374,682	163	−0.07	330,338[b]	331,818	13	220,062	5.66	51
26	4	366,167	129	−0.15	331,722	334,062	10	220,034	5.16	52
27	1	375,261	*	0	321,615	322,816	16	216,087	5.14	49
28	3	373,464	120	0.08	323,233	324,047	15	215,604	4.12	50
29	5	379,320	255	0	334,784[b]	336,159	13	223,175	5.11	51
30	6	370,012	420	0.21	331,837	335,212	10	218,849	6.33	53

*Only one elite solution

[a]Stop at 500 s

[b]Lower bound model includes integer variables $y(t)$ (Lemma 1 not satisfied)

shown in column 8. With respect to size, the lower bounding MIP contained 2083 constraints and 3183 variables of which 1021 were binary. In all cases, a 500 s limit was placed on CPLEX for these runs, which terminated with an optimality gap that averaged 2.2%.

The last three columns of Table 2 give the LP lower bound (ϕ_{LP}) for the full model (with the routing constraints), the corresponding solution times, and the gaps between the LP solution and the adjusted lower bound (ϕ_{ALB}) obtained from the modified allocation problem in 500 s. The size of this gap, which was about 50%, implies that the solution to the LP relaxation by itself is not very useful. As discussed by Laporte (1992) and others, poor performance like this is due primarily to the weak relaxation associated with the subtour elimination constraints of the form $y_{jt} \le y_{it} - w_{it} + D_t^{\max}(1 - x_{ijt}), \forall i \in N, j \in N_0, t \in T$, where y_{it} is the load on the vehicle just after departing from customer i in period t and $D_t^{\max} = \min\{Q, \sum_{i\in N}\sum_{l=t}^{\tau} d_{il}\}$ is an upper bound on the load on any vehicle in period t. These constraints require the load on a vehicle to be monotoni-

Table 3 Comparison of solutions for problem instances with 100 customers and 20 periods

Prob. no.	(ϕ_{grasp}) Best GRASP solution	(t_{grasp}) Runtime (s)	(ϕ_{Ph1}) Phase 1 solution	(t_{Ph1}) Runtime (s)	$[(\phi_{\text{Ph1}}-\phi_{\text{grasp}})/\phi_{\text{grasp}}]100\%$	(t_{ALLOC}) Alloc time (s)	(ϕ_{Ph2}) Phase 2 solution	Iteration ϕ_{Ph2} found	(t_{Ph2}) Runtime (s)	$[(\phi_{\text{Ph2}}-\phi_{\text{Ph1}})/\phi_{\text{Ph1}}]100\%$
1	790,972	413.42	737,241	118.44	−7	96.50	711,671	21	1079	−3
2	782,906	506.19	734,043	143.27	−6	124.6	698,512	40	1035	−5
3	787,830	346.76	721,767	117.25	−8	95.43	683,270	40	1299	−5
4	779,847	486.51	741,904	119.51	−5	95.31	718,252	21	672	−3
5	796,176	439.06	748,370	118.33	−6	94.87	731,260	14	335	−2
6	793,216	349.02	758,969	141.54	−4	112.68	744,927	15	1143	−2
7	781,317	298.64	729,426	126.22	−7	105.39	695,728	36	1250	−5
8	780,884	499.19	729,542	149.80	−7	119.44	706,058	30	1008	−3
9	784,646	442.30	742,733	130.16	−5	108.22	705,035	39	1125	−5
10	790,156	378.54	727,431	143.40	−8	121.22	696,521	31	985	−4
11	787,596	393.90	734,881	126.43	−7	100.71	711,895	37	835	−3
12	785,170	369.25	730,530	134.65	−7	113.85	703,162	32	1129	−4
13	777,705	505.97	742,061	136.13	−5	111.88	721,066	22	599	−3
14	789,802	467.69	730,899	116.89	−7	95.30	698,548	35	1065	−4
15	790,132	527.02	732,911	157.43	−7	136.64	711,506	23	1139	−3
16	797,322	418.19	753,956	115.16	−5	91.20	714,873	37	1226	−5
17	799,843	520.11	729,914	125.23	−9	106.11	702,314	33	1218	−4
18	787,371	419.09	752,665	152.07	−4	126.81	720,238	32	720	−4
19	806,592	353.69	773,547	113.63	−4	85.87	748,734	36	1349	−3
20	809,340	403.09	748,000	123.80	−8	101.75	729,099	20	1131	−3
21	788,736	477.35	754,214	127.12	−4	102.28	738,746	14	544	−2
22	804,538	412.97	735,752	144.92	−9	123.60	702,849	36	998	−4
23	781,558	429.12	741,379	126.09	−5	101.82	712,717	23	1037	−4
24	798,428	416.88	758,063	142.42	−5	113.80	727,741	37	1380	−4
25	796,591	368.14	745,669	125.21	−6	102.85	725,869	30	1240	−3
26	791,514	403.69	724,380	124.91	−8	104.94	700,719	37	585	−3
27	773,662	495.51	709,640	118.37	−8	98.77	686,382	36	454	−3
28	780,492	416.89	724,556	143.13	−7	117.52	700,980	30	1190	−3
29	799,417	457.51	754,116	135.21	−6	110.31	725,030	32	993	−4
30	785,906	398.11	734,725	154.12	−7	134.15	698,942	32	1300	−5

cally decreasing during the delivery sequence and are rarely if ever binding in the LP solution. The corresponding MIP for the full model contained 54,063 constraints and 54,183 variables of which 51,021 were binary.

6.3 Medium and large instances

The results for the 100-customer instances with 20 time periods each are presented in Tables 3 and 4. The column headings are identical to those of Tables 1 and 2, respectively. On average, the GRASP took 42.1 s while tabu search took a total of 1133.8 s or 174% longer. The gap between the GRASP solution and the phase 1 solution obtained from the allocation model after running the VRP subroutine is identified in column 5 and represents roughly a 6.4% improvement. An additional 3.6% is realized in phase 2, giving a total improvement of 10%.

As seen in Table 4, path relinking takes about 465 s and in only for problem no. 2 was an improvement realized. On average, the path relinking solutions were 1.2% worse than the phase 2 solutions. The modified allocation model had 4083 constraints and 6283 variables of which 2021 were binary. A total of 850 s was allotted for the computations. At termination, CPLEX exhibited an optimality gap of roughly 3.5%. The gap between the best feasible solution and the adjusted lower bound is shown in column 8 and averaged 22.9%. The data in the last three columns are as expected.

Table 4 Path relinking and lower bound results for instances with 100 customers and 20 periods

Prob. no.	No. elites solns	(ϕ_{PR}) Solution from path relinking	(t_{PR}) Time (s)	$[(\phi_{\mathrm{PR}}-\phi_{\mathrm{Ph2}})/\phi_{\mathrm{Ph2}}]100\%$	(ϕ_{LB}) from LB model[a]	(ϕ_{ALB}) Adjusted LB	$[(\phi_{\mathrm{Best}}-\phi_{\mathrm{ALB}})/\phi_{\mathrm{ALB}}]100\%$	(ϕ_{LP}) LP soln	(t_{LP}) LP time (s)	$[(\phi_{\mathrm{ALB}}-\phi_{\mathrm{LP}})/\phi_{\mathrm{LP}}]100\%$
1	1	711,671	*	0.00	577,137[b]	582,090	22	281,684	24.18	107
2	4	694,694	240	−0.55	581,028[b]	584,847	19	284,882	22.02	105
3	2	693,600	80	1.51	578,440	581,317	18	283,960	36.05	105
4	1	718,252	*	0.00	571,459[b]	574,748	25	283,138	28.75	103
5	1	731,260	*	0.00	579,452	583,117	25	287,534	21.46	103
6	1	744,927	*	0.00	581,760	586,735	27	286,161	43.87	105
7	3	708,958	540	1.90	571,178	573,184	21	283,575	27.28	102
8	3	713,267	720	1.02	561,926[b]	564,981	25	278,847	34.79	103
9	2	713,779	360	1.24	580,835	583,295	21	286,051	24.54	104
10	2	710,427	270	2.00	573,489[b]	577,928	21	284,019	32.82	103
11	3	729,324	700	2.45	574,951[b]	580,831	23	284,048	20.95	104
12	6	712,832	550	1.38	577,810	581,643	21	285,148	20.62	104
13	5	728,004	650	0.96	568,636	572,655	26	281,696	23.24	103
14	2	705,304	500	0.97	578,958[b]	586,741	19	282,791	21.75	107
15	3	720,050	560	1.20	577,760	580,764	23	286,629	22.73	103
16	3	719,974	627	0.71	576,178	581,498	23	283,683	30.38	105
17	2	725,757	160	3.34	587,829[b]	591,956	19	289,069	11.40	105
18	2	731,189	140	1.52	577,616[b]	580,835	24	285,723	22.17	103
19	3	759,505	780	1.44	573,342	576,318	30	283,849	23.52	103
20	3	737,988	610	1.22	589,667[b]	593,649	23	290,279	33.05	105
21	1	738,746	*	0.00	584,901	590,280	25	289,479	94.78	104
22	2	719,634	400	2.39	574,965	577,327	22	285,958	25.14	102
23	1	712,717	*	0.00	567,679	569,334	25	281,909	23.82	102
24	3	738,178	720	1.43	574,446	577,506	26	285,820	20.57	102
25	6	739,029	590	1.81	573,169	576,299	26	284,631	21.70	102
26	3	707,568	520	0.98	567,682[b]	571,351	23	280,753	23.37	104
27	2	695,961	200	1.40	569,319	572,684	20	283,016	25.37	102
28	2	710,866	400	1.41	564,894	569,282	23	280,900	41.95	103
29	2	740,172	520	2.09	585,442	588,796	23	288,490	63.83	104
30	2	712,385	320	1.92	577,862[b]	582,331	20	286,316	47.36	103

*Only one elite solution

[a]Stop at 850 s

[b]Lower bound model includes integer variables $y(t)$ (Lemma 1 not satisfied)

The LP relaxation of the full model solves quickly with CPLEX but the gap between the solution, ϕ_{LP}, and the adjusted lower bound, ϕ_{ALB}, averaged 104%.

The results for the 200-customer instance with 20 time periods each exhibited the same patterns as the 100-customer instances.

Table 5 presents the comparisons with GRASP, and Table 6 presents the path relinking and allocation model results. The average improvement of tabu search over GRASP was 3% but average runtimes increased from 1903 to 2879 s, or 51%. However, as the number of customers increases so do the objective function values, so small percentage reductions in cost often translate into be large reductions in absolute terms.

From Table 6, we see that path relinking solutions are on average 0.5% worse than the phase 2 solutions so it is doubtful whether the procedure is worthwhile, especially with runtimes averaging 2230 s. To compute the lower bound from the modified allocation model, CPLEX was allowed 1450 s. At the time, the average optimality gap was 3.6%,

Table 5 Comparison of solutions for problem instances with 200 customers and 20 periods

Prob. no.	(ϕ_{grasp}) Best GRASP solution	(t_{grasp}) Runtime (s)	(ϕ_{Ph1}) Phase 1 solution	(t_{Ph1}) Runtime (s)	$[(\phi_{\text{Ph1}}-\phi_{\text{grasp}})/\phi_{\text{grasp}}]100\%$	(t_{ALLOC}) Alloc time (s)	(ϕ_{Ph2}) Phase 2 solution	Iteration ϕ_{Ph2} found	(t_{Ph2}) Runtime (s)	$[(\phi_{\text{Ph2}}-\phi_{\text{Ph1}})/\phi_{\text{Ph1}}]100\%$
1	1,075,528	2078.39	1,051,861	280	−2	179	1,030,684	20	2965	−2
2	1,070,340	1785.75	1,038,017	320	−3	233	1,024,558	11	1737	−1
3	1,070,505	1688.52	1,036,408	265	−3	131	1,036,282	4	1528	0
4	1,068,959	2194.75	1,067,202	289	0	174	1,057,654	14	2190	−1
5	1,060,220	1678.19	1,046,100	266	−1	189	1,031,422	18	2573	−1
6	1,065,700	1948.55	1,052,150	360	−1	204	1,033,233	19	1016	−2
7	1,091,538	1796.72	1,061,457	393	−3	206	1,043,536	35	1942	−2
8	1,060,164	2237.27	1,072,610	349	1	222	1,066,068	10	2978	−1
9	1,055,447	1554.42	1,047,509	321	−1	190	1,036,179	16	2454	−1
10	1,069,590	2090.38	1,057,926	377	−1	214	1,038,559	38	1855	−2
11	1,069,280	2464.75	1,054,845	388	−1	197	1,037,705	21	3428	−2
12	1,057,631	1909.58	1,052,501	397	−1	221	1,040,220	29	2288	−1
13	1,074,180	2087.02	1,075,537	483	0	233	1,063,024	13	2719	−1
14	1,076,460	2264.77	1,054,986	397	−2	216	1,041,786	21	3544	−1
15	1,065,340	1776.81	1,045,066	376	−2	155	1,029,908	25	2554	−2
16	1,067,550	2022.39	1,052,812	410	−1	231	1,033,656	27	3200	−2
17	1,067,007	1826.45	1,053,600	382	−1	187	1,027,433	31	2800	−3
18	1,095,350	1716.27	1,089,858	377	−1	172	1,063,306	29	4740	−2
19	1,063,445	1920.98	1,079,858	372	2	166	1,065,705	23	2797	−1
20	1,049,854	1910.66	1,057,882	360	1	148	1,034,195	37	2600	−2
21	1,055,436	2506.95	1,065,159	477	1	205	1,044,771	32	3900	−2
22	1,066,185	1781.39	1,059,796	369	−1	152	1,045,790	13	2822	−1
23	1,073,265	2201.73	1,033,344	459	−4	214	1,027,042	15	1797	−1
24	1,063,585	1999.14	1,078,993	381	1	151	1,051,610	37	3510	−3
25	1,054,230	1902.87	1,055,464	410	0	202	1,027,772	36	3000	−3
26	1,057,443	1685.31	1,059,502	335	0	129	1,044,315	26	2720	−1
27	1,076,798	1484	1,060,084	409	−2	213	1,047,267	12	2498	−1
28	1,054,225	1508.17	1,052,961	417	0	184	1,042,891	16	4200	−1
29	1,088,853	1463.53	1,040,105	409	−5	239	1,030,156	24	1947	−1
30	1,051,195	1613.92	1,051,980	325	0	155	1,035,703	21	2351	−2

indicating the increased difficulty in solving the corresponding IP. The lower bound lemmas improved the results by 0.61% on average (not shown in table). The gap between the best solution found and the adjusted lower bound was approximately 20.5%, slightly better than for the 100-customer problems. For problem instances of this magnitude, these gaps are within an acceptable range.

7 Discussion and future directions

Good solutions to the PIDRP can yield substantial benefits throughout the supply chain as manufacturers and customers become more integrated. A decade ago, the primary challenges facing manufacturers were establishing online communications and delivery channels. Keeping too much inventory was discouraged and warehouses equaled waste. Today, there is a shift in attitude that allows for increased inventory levels where necessary. Lean inventory is a luxury many companies can no longer afford because timely deliveries depend on a far wider range of factors than can be predicted or controlled. If your supply chain is global, those factors include international transportation providers, as well as infrastructure with varying degrees of quality and import/export regulations. If your supply chain is domestic, you are still affected because certain transportation lanes are

Table 6 Path relinking and lower bound results for instances with 200 customers and 20 periods

Prob. no.	No. elites solns	(ϕ_{PR}) Solution from path relinking	(t_{PR}) Time (s)	$[(\phi_{PR}-\phi_{Ph2})/\phi_{Ph2}]100\%$	(ϕ_{LB}) from LB model[a]	(ϕ_{ALB}) Adjusted LB	$[(\phi_{Best}-\phi_{ALB})/\phi_{ALB}]100\%$	(ϕ_{LP}) LP soln	(t_{LP}) LP time (s)	$[(\phi_{ALB}-\phi_{LP})/\phi_{LP}]100\%$
1	1	1,030,684	*	0	853,521	859,151	20	442,266	120.61	94
2	2	1,010,158	1250	−1.41	851,442[b]	856,489	18	440,625	121.81	94
3	2	1,016,681	1200	−1.89	843,605	847,677	20	438,910	123.88	93
4	4	1,042,854	2141	−1.40	852,993	858,620	21	440,440	132.46	95
5	3	1,023,680	2423	−0.75	853,415	858,986	19	440,789	78.34	95
6	2	1,025,262	2205	−0.77	859,458	863,514	19	443,041	75	95
7	3	1,038,746	2610	−0.46	855,652	859,755	21	442,400	111.27	94
8	1	1,066,068	*	0	856,121[b]	862,333	24	442,292	78.40	95
9	2	1,018,420	1250	−1.71	838,995	845,013	21	434,188	83.01	95
10	8	1,035,240	2625	−0.32	851,118	854,605	21	441,476	76.88	94
11	2	1,037,767	2730	0.01	861,136	866,478	20	444,524	104.01	95
12	5	1,035,350	2242	−0.47	856,766	861,683	20	443,012	105.69	95
13	1	1,063,024	*	0	851,622	855,760	24	442,760	78.43	93
14	2	1,024,491	1920	−1.66	856,951	860,535	19	443,699	103.96	94
15	6	1,026,787	2408	−0.30	848,998	855,899	20	440,104	75.98	94
16	2	1,043,917	1957	0.99	855,689	859,910	20	442,252	101.19	94
17	5	1,022,250	2018	−0.50	853,635	859,039	19	440,699	89.85	95
18	5	1,065,250	2242	0.18	854,938[b]	860,527	24	441,318	90.58	95
19	1	1,065,705	*	0	859,526	867,188	23	442,439	82.06	96
20	4	1,027,134	2425	−0.68	847,312	852,918	20	437,861	73.92	95
21	3	1,049,028	3003	0.41	849,209	853,126	22	440,479	80.13	94
22	1	1,045,790	*	0	848,907	854,260	22	439,190	67.17	95
23	4	1,034,198	2403	0.70	855,591	862,561	19	441,020	73.51	96
24	6	1,045,014	2908	−0.63	854,029	858,675	22	439,128	94.47	96
25	3	1,024,239	2601	−0.34	859,551	864,960	18	442,202	102.96	96
26	4	1,043,128	1673	−0.11	848,613[b]	855,543	22	438,277	93.42	95
27	2	1,030,753	2037	−1.58	863,295[b]	869,693	19	445,644	79.68	95
28	3	1,032,478	2909	−1.00	854,208	859,167	20	440,210	93.27	95
29	5	1,019,371	2179	−1.05	862,445	866,925	18	444,642	99.06	95
30	6	1,027,915	2400	−0.75	859,578	865,404	19	441,228	88.31	96

*Only one elite solution

[a]Stop at 1450 s

[b]Lower bound model includes integer variables $y(t)$ (Lemma 1 not satisfied)

more crowded than ever, and the competition for transportation services has become even more intense.

In this paper, we have provided an efficient reactive tabu search algorithm for finding high quality solutions to the PIDRP as measured by the optimality gap for small instances and by the solution to our lower bounding model otherwise. Although we included a path relinking feature and several theoretical ways of improving the lower bound, none was very effective. If there is room for improvement in the methodology, the place to begin is with the allocation model (1) and its modified version that was used to obtain lower bounds. Adapting the cut generation procedures of Pochet and Wolsey (2006) for the capacitated lot-sizing problem offers several opportunities for tightening the LP relaxation of (1a)–(1j). A second option that is now underway is to apply branch and price to the full PIDRP using a time period decomposition. Because a VRP must be solved for each period, there are some inherent limitations to this approach. A third option is to cluster the customers first and then solve the PIDP and routing components separately.

Each of these ideas offers computational challenges that the research community has yet to address.

Appendix: Adjustment of production levels

As mentioned in Sect. 4.5, it may be possible to improve the tabu search results when evaluating candidate moves by adjusting the production levels. In the implementation, a linear program is solved every 5 iterations to find the optimal values of $\bar{p}_t$. In the remaining cases, we adjust production levels by calling *Production_Level_Adjustment_Algorithm*. The inputs to this algorithm are the production levels $\bar{p}_t$ associated with the most recent tabu search solution and the output generated by *Find_Adjustment_Period_Algorithm*, which is called when checking the feasibility of a solution.

For a move that involves customer i_1 in period t_1 and customer i_2 in period $t_2, t_1 < t_2$, let $\bar{\eta}_{i_1 t_1}$be the number of items that were originally scheduled to be delivered to customer i_1 in period t_1 that have been rescheduled for delivery in period t_2, and let $\bar{w}_{i_2 t_2}$be the number of items to be delivered to customer i_2 in period t_1 after the move. The logic included in the primary algorithms called in the adjustment of production levels is given below. The first step is to check feasibility of a candidate move. Only the inputs and outputs are stated for this algorithm; more detail can be found in Nananukul (2008).

Feasibility_Check_Algorithm, complexity $O(\tau + n)$.

Input: Tabu search solution $\bar{p}_t$ at current iteration, periods t_1 and $t_2 (t_1 < t_2)$, customer i_1 and i_2, demand of customer i_1 for all $t \le t_1$, demand of customer i_2 from all $t \le t_2$, delivery amounts $\bar{w}_{i_1 t}, t \le t_1$ and $\bar{w}_{i_2 t}, t \le t_2$, and swap amounts $\bar{\eta}_{i_1 t_1}$ and $\bar{w}_{i_2 t_2}$.

Output: Output flag: ⟨true⟩ = feasible or ⟨false⟩ = not feasible
If output flag = ⟨true⟩, then return number of decrease adjustment periods, decrease adjustment periods, decrease adjustment amounts, number of increase adjustment periods, increase adjustment periods, and increase adjustment amounts
\\ comment: The return values are the output from *Find_Adjustment_Period_Algorithm*.

Find_Adjustment_Period_Algorithm, complexity $O(\tau)$.

Input: Period t^*, search type (1 = search for period(s) $t < t^*$ such that production level needs to be decreased, 2 = search for period(s) $t < t^*$ such that the production level needs to be increased), adjustment quantity AQ, and tabu search solution output ($\bar{p}_t$) at current tabu search iteration
\\ comment: $t^* = t_1$ or $t_2 : AQ = \bar{\eta}_{i_1 t_1} - \bar{w}_{i_2 t_2}$. Only positive value of adjustment quantity AQ is used in the algorithm.

Output: Feasible flag (if feasible period(s) is found, then return "1"; otherwise return "−1")
If search type = 1, then return number of decrease adjustment periods, decrease adjustment periods, and decrease adjustment amounts; otherwise return number of increase adjustment periods, increase adjustment periods, and increase adjustment amounts.

```
NumAdjPeriods = 1
For $t = t^* - 1, \ldots, 0$
{
  If ($\bar{p}_t > 0$)
  \\ comment: Only period(s) $t$ with $\bar{p}_t > 0$ are considered in the adjustment.
    If (Search type = 1)
      $TAQ = AQ - \bar{p}_t$
    else
      $TAQ = AQ - (I^P_{\max} - \bar{p}_t)$\\ $TAQ$ stores remaining adjustment quantity after considering adjustment in period $t$.
    end if
    If ($TAQ \le 0$)\\ In this case it means period $t$ is the last period that needs to be considered.
      AdjPeriods[NumAdjPeriods] = $t$
      AdjAmount[NumAdjPeriods] = $AQ$
      NumAdjPeriods = NumAdjPeriods + 1
      Set feasible flag = 1
      If (search type = 1)
        Store results in NumDecAdjPeriods,
        DecAdjPeriods[$n$];
        $n = 1, \ldots,$ NumDecAdjPeriods,
        DecAdjAmount[$n$];
        $n = 1, \ldots,$ NumDecAdjPeriods
    else
        Store results in NumIncAdjPeriods,
        IncAdjPeriods[$n$];
        $n = 1, \ldots,$ NumIncAdjPeriods, IncAdjAmount[$n$];
        $n = 1, \ldots,$ NumIncAdjPeriods
    end if
    return
    else
    AdjPeriods[NumAdjPeriods] = $t$
    If (Search type = 1)
        AdjAmount[NumAdjPeriods] = $\bar{p}_t$
    else
        AdjAmount[NumAdjPeriods] = $I^P_{\max} - \bar{p}_t$
    end if
    NumAdjPeriods = NumAdjPeriods + 1
  end if
  end if
```

```
    $AQ = TAQ$
}
If ($TAQ > 0$)\\ In this case it means that the adjustment is not feasible.
    Set feasible flag $= -1$ and return
end if
```

After the feasibility check is done, the *move_value* for each feasible candidate is calculated by calling *Move_Value_Algorithm*. Because the logic is straightforward, we only give the inputs and outputs.

Move_Value_Algorithm, complexity $O(n^3)$

Input: Periods t_1 and $t_2 (t_1 < t_2)$, costs of current VRP solutions in period t_1 and t_2 ($C_{t_1}^{\mathrm{VRP}}$, $C_{t_2}^{\mathrm{VRP}}$), customer i_1 and i_2, and swap amount $\bar{\eta}_{i_1 t_1}$ (i.e., number of items rescheduled from period t_1 to period t_2 for customer i_1), number of items that were to be delivered to customer i_2 in period t_2 but have been rescheduled for delivery in period t_1 ($\bar{w}_{i_2 t_2}$), adjustment quantity AQ, number of decrease adjustment periods, decrease adjustment periods, decrease adjustment amounts, number of increase adjustment periods, increase adjustment periods, and increase adjustment amounts.
Output: *move_value*

Once all candidate moves are evaluated, the one with the best *move_value* is selected. Finally, the *Production_Level_Adjustment_Algorithm* is called.

Production_Level_Adjustment_Algorithm

Input: Tabu search solution output ($\bar{p}_t$) at current tabu search iteration, Number of decrease adjustment periods, decrease adjustment periods, decrease adjustment amounts, number of increase adjustment periods, increase adjustment periods, and increase adjustment amounts.
Output: Updated values of $\bar{p}_t$.

```
Step 1. Update production levels $\bar{p}_t$ in each period $t$ where the new (tabu search) solution indicates a decrease in production level.
        For $n = 1, \ldots,$ NumDecAdjPeriods
            $\bar{p}$DecAdjPeriods[$n$] $\leftarrow$ $\bar{p}$DecAdjPeriods[$n$]
                                        $-$ DecAdjAmount[$n$]
Step 2. Update production levels $\bar{p}_t$ in each period $t$ where the new solution indicates an increase in production level.
        For $n = 1, \ldots,$ NumIncAdjPeriods
            $\bar{p}$IncAdjPeriods[$n$] $\leftarrow$ $\bar{p}$IncAdjPeriods[$n$]
                                        $-$ IncAdjAmount[$n$]
```

Example (Continued) Considering the example in Fig. 3, assume that the current solution from the allocation model gives $\bar{p}_1 = 66$, $\bar{p}_2 = 0$, and $\bar{p}_3 = 0$. Base on a swap move in Fig. 3, $i_1 = 3, i_2 = 1, t_1 = 2$, and $t_2 = 3$, respectively. Following Step 1 of the above algorithm, NumDecAdjPeriods = 1, DecAdjPeriods[0] = 1, DecAdjAmount[0] = 4, NumIncAdjPeriods = 1, IncAdjPeriods[0] = 1, IncAdjAmount[0] = 4. Continuing, $\bar{p}_1$ is updated to $66 - 4 = 62$. In Step 2, $\bar{p}_1$ is updated to $62 + 4 = 66$. As a result the swap move does not change the production level in period 1.

For the example in Fig. 4, assume that the current solution from the allocation model gives $\bar{p}_1 = 64$, $\bar{p}_2 = 2$, and $\bar{p}_3 = 0$. Based on the transfer move in Fig. 4, $i_1 = 0, i_2 = 2, t_1 = 2$, and $t_2 = 3$, respectively. Following Step 1 of *Feasibility_Check_Algorithm*, *NumDecAdjPeriods* = 2, DecAdjPeriods[0] = 2, DecAdjAmount[0] = 2, DecAdjPeriods[1] = 1, DecAdjAmount[1] = 2, NumIncAdjPeriods = 1, IncAdjPeriods[0] = 1, IncAdjAmount[0] = 4. Following Step 1 of *Production_Level_Adjustment_Algorithm*, $\bar{p}_2$ is updated to $2 - 2 = 0$. In Step 2, $\bar{p}_1$ is updated to $64 - 2 + 4 = 66$.

References

Abdelmaguid, T. F., & Dessouky, M. M. (2006). A genetic algorithm approach to the integrated inventory–distribution problem. *International Journal of Production Research*, *44*(21), 4445–4464.

Anily, S., & Federgruen, A. (1993). Two-echelon distribution systems with vehicle routing costs and central inventories. *Operations Research*, *41*(1), 37–47.

Bard, J. F., Huang, L., Jaillet, P., & Dror, M. (1998). A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Science*, *32*(2), 189–203.

Boudia, M., Louly, M. A. O., & Prins, C. (2006). A memetic algorithm with population management for a production–distribution problem. In A. Doglui, G. Morel, C.E. Pereira (Eds.), *12th IFAC symposium on information control problems in manufacturing* (Vol. 3, pp. 541–546). Saint-Etienne, France, 17–19 May.

Boudia, M., Louly, M. A. O., & Prins, C. (2007). A reactive grasp and path relinking for a combined production–distribution problem. *Computers & Operations Research*, *34*(11), 3402–3419.

Carlton, B. W., & Barnes, J. W. (1996). Solving the traveling salesman problem with time windows using tabu search. *IIE Transactions on Scheduling & Logistics*, *28*(8), 617–629.

Cetinkaya, S., Mutlu, F., & Lee, C.-Y. (2006). A comparison of outbound dispatch policies for integrated inventory and transportation decisions. *European Journal of Operational Research*, *171*(3), 1094–1112.

Chandra, P., & Fisher, M. L. (1994). Coordination of production and distribution planning. *European Journal of Operational Research*, *72*(3), 503–517.

Dror, M., & Ball, M. (1987). Inventory/routing: reduction from an annual to a short period problem. *Naval Research Logistics Quarterly*, *34*(4), 891–905.

Federgruen, A., & Tzur, M. (1999). Time-partitioning heuristics: application to one warehouse, multiitem, multiretailer lot-sizing problems. *Naval Research Logistics*, *46*(5), 463–486.

Gaudioso, M., & Paletta, G. (1992). A heuristic for the periodic vehicle routing problem. *Transportation Science*, *26*(2), 86–92.

Glover, F., & Laguna, M. (1997). *Tabu search*. Norwell: Kluwer.

Golden, B., Assad, A., & Dahl, R. (1984). Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale Systems*, *7*(2-3), 181–190.

Gutiérrez, J., Sedeño-Noda, A., Colebrook, M., & Sicilia, J. (2007). A polynomial algorithm for the production/ordering planning problem with limited storage. *Computers & Operations Research*, *34*(4), 934–937.

Herer, Y. T., Tzur, M., & Yucesan, E. (2006). The multilocation transshipment problem. *IIE Transactions on Scheduling & Logistics*, *38*, 185–200.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*(3), 345–358.

Lei, L., Liu, S., Ruszczynski, A., & Park, S. (2006). On the integrated production, inventory, and distribution routing problem. *IIE Transactions on Scheduling & Logistics*, *38*(11), 955–970.

Mourgaya, M., & Vanderbeck, F. (2007). Column generation based heuristic for tactical planning in multi-period vehicle routing. *European Journal of Operational Research*, *183*(3), 1028–1041.

Nananukul, N. (2008). Lot-sizing and inventory routing for a production–distribution supply chain. PhD dissertation, Graduate Program in Operations Research & Industrial Engineering, The University of Texas, Austin.

O'Brien, C., & Tang, O. (Eds.) (2006). Integrated enterprise and supply chain management. *International Journal of Production Economics*, *101* (special issue).

Parthanadee, P., & Logendran, R. (2006). Periodic product distribution from multi-depots under limited supplies. *IIE Transactions on Scheduling & Logistics*, *38*(11), 1009–1026.

Pochet, Y., & Wolsey, L. A. (2006). *Production planning by mixed integer programming*. New York: Springer.

Resende, M. G. C., & Ribeiro, C. C. (2005). GRASP with path-relinking: recent advances and applications. In T. Ibaraki, K. Nonobe, & M. Yagiura (Eds.), *Metaheuristics: progress as real problem solvers* (pp. 29–63). New York: Springer.

Villegas, F. A., & Smith, N. R. (2006). Supply chain dynamics: analysis of inventory vs. order oscillations trade-off. *International Journal of Production Research*, *44*(6), 1037–1054.

Zhao, Q.-H., Wang, S.-Y., & Lai, K. K. (2007). A partition approach to the inventory/routing problem. *European Journal of Operational Research*, *177*(2), 786–802.