March 12, 1999
RT0309
Computer Science 16 pages

Research Report

Delivery route optimization and its applications, with examples from logistics in banks

H. Okano

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Abstract

This paper proposes an algorithm for delivery route optimization problems and describes its application to real problems in Japanese banks. For delivery route optimization problems in the real world, an algorithm is required to minimize the number of routes while handling various complicated constraints. Techniques for modeling real problems with an abstract model provided by the algorithm are also required. The algorithm proposed in this paper provides a model and constraints that were designed for describing real problems, such as deliveries from multiple depots, time windows of customers, and reuse of vehicles. A mechanism for minimizing number of routes used by the algorithm, called the *objective tuning method*, guides local search so that selected routes will shrink and disappear. Through case studies of logistics problems in Japanese banks, the algorithm is shown to be capable of modeling and solving real problems.

Keywords

delivery route optimization, vehicle routing problem, objective tuning method, iterated local search

1. Introduction

Delivery route optimization in the real world involving various constraints has recently become feasible as a result of the ready availability of digital road maps at low prices, the development of optimization techniques represented by meta-heuristics, and the increase in computing power. At the same time, companies in various fields, such as manufacturing, sales, and banking, have begun to focus on reducing their logistics costs by applying delivery route optimization. In this paper, an algorithm that aims to handle real problems of delivery route optimization is proposed, and its applications to logistics problems in Japanese banks are introduced.

Delivery route optimization problems, which are also called vehicle routing problems (VRPs), vary widely according to the models and constraints involved. An algorithm developed by the authors currently handles the following abstract model and constraints:

- Different types (capacities) of vehicles start from multiple depots, visit certain points, and return to the original depots, while each vehicle can return to its original depot at any time to load or unload goods. (This is called reuse of vehicles.)
- A specified quantity of goods is delivered to or picked up from each point by a vehicle.
- For each depot, various values are specified: the number of vehicles based at the depot, the working times of the drivers (e.g., from 9:00 to 17:00), the rest times of the drivers (e.g., one hour after 12:00), and so on.
- For each point, various values are specified: the quantity of goods to be delivered or picked

up, the time required for loading or unloading, the time window in which vehicles are allowed to visit, the maximum or minimum size of vehicles that can visit the point (e.g., below four tons), the base depots of the vehicles that can visit the point, and so on.

Note that it is called an abstract model because it may be mapped to concrete models in ways that differ from the above description.

Because delivery route optimization problems belong to the class of NP-hard combinatorial optimization problems, approximation algorithms are generally used for them. Early studies of delivery route optimization discussed approximation algorithms whose objective was to minimize the total running time, and that handled simple constraints such as load capacity and time windows. In the real world, however, it is necessary to consider many constraints, and thus it sometimes turns out that such algorithms that assume simple constraints are not applicable. For example, a well-known lexicographic search for the traveling salesman problem with time windows (TSPTW) by Savelsbergh [1], which can check the feasibility of an edge exchange in $O(1)$ time, is not applicable if penalized (infeasible) solutions are allowed. This is because, when solutions may have penalties, it is necessary to scan each route affected by a neighborhood operation in order to calculate a penalty along the route. Note that, when real problems with various complicated constraints are considered, it is reasonable to allow penalized solutions, because the existence of feasible solutions is not guaranteed, and also because a search space should not be restricted. Moreover, many studies in the literature handle problems whose objectives are to minimize the total running time of vehicles, while the objectives of real problems are to minimize the number of routes (vehicles used). A few rare examples that handle minimization of the number of routes are a tabu-search-based algorithm by Potvin et al. [2] that performs vehicle reduction and time reduction by turns, and tabu- and guided-local-search-based algorithms by de Backer et al. [3] that introduce dummy routes to serve unvisited points and penalize them according to their lengths. Note that algorithms for minimizing the total running time are not sufficiently applicable to real problems, because (transitions to) solutions that minimize the total running time are not necessarily the same as (transitions to) solutions that minimize the number of routes, and they sometimes differ greatly. Therefore, to be applicable to real problems, such an algorithm is required to minimize the number of routes while handling various complicated constraints.

This paper introduces an algorithm developed by the authors, which is an example of such algorithms for handling real problems, and which handles the above-mentioned model and constraints. The algorithm uses iterated local search as a meta-heuristic framework, and has a

mechanism for reducing the number of routes in escaping from local opts and in local searching. Two escape methods are provided in the algorithm: one is to extract nodes in selected routes and insert them into any nonempty routes, and the other is to simply tune the objective function values of selected routes to augment them (*the objective tuning method*). The objective tuning method also guides a local search so that selected routes will shrink and disappear. The considered optimization objectives are minimization of the number of routes, minimization of the total running time, and minimization of the maximum number of routes, which involves minimization of the maximum value among a number of routes assigned to each depot. When the primary objective is minimization of the number of routes or the maximum number of routes, the secondary objective, minimization of the total running time, is also considered. Because the models and constraints of real problems are not necessarily the same as those provided by algorithms, it is generally necessary to create concrete models for real problems by using abstract models of algorithms. For example, many problems involve deliveries on multiple days, where delivery points need to be grouped into two subsets so that each of them can be served once in two days, while some points need to be served on both days. Although the above-mentioned model does not seem to be applicable to this type of problem it turns out to be applicable if a depot is logically represented by two depots, and solving the problem will minimize the maximum number of routes. For other types of problems, it may necessary to simplify the actual model to handle the abstract models provided by algorithms. Motivations for solving real problems may be classified into two types: tactical, where the aim is to estimate a cost after changing a delivery operation, and operational, where the aim is to make plans of daily operations. Models of the former type are often allowed to be simplified. Precisely speaking, for tactical problems, the person who uses a delivery route optimization tool maybe not a customer who has a real problem but a consultant who wants to analyze the total cost, including the hidden costs of modeling; therefore, factors neglected in simplifying models are compensated for. As a real-world example of delivery operations, this paper introduces the replenishment of cash in autonomous teller machines (ATMs).

In Section 2, general approaches to the delivery route optimization are introduced, while in Section 3, the authors' algorithm is proposed. Section 4 describes case studies of delivery route optimization in Japanese banks, and Section 5 concludes the paper.

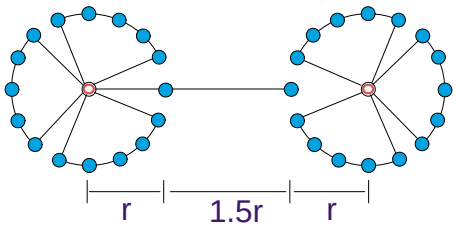


Fig. 1. A simple instance in which the optimal solutions for two optimization objectives, minimization of the number of routes and minimization of the total running time, are different.

2. General approaches to delivery route optimization

Almost every approximation algorithm for delivery route optimization uses local search. Local search is a type of algorithm in which such a solution t that $F(s, t \in N(s)) = Y$ is found repeatedly to update $s = t$ until $F(s, {}^{\forall}t \in N(s)) = N$ is satisfied, where $N(s)$ is a set of neighborhood solutions of a solution s, and $F(s,t) = \{Y, N\}$ is an evaluation function for deciding whether a solution t is more optimal than a solution s; for example, given an objective function $f(x)$, it is defined as $F(s,t) = \{Y,\ N : \ Y \text{ if } f(s) > f(t),\ \ N \text{ if } f(s) \leq f(t)\}$.

Various neighborhood operations for delivery route optimization have been proposed, such as 2-opt, Or-opt, shift, interchange, and cross [6], all of which aim to reduce the running time. No neighborhood operation that is effective for reducing the number of routes has been proposed, and actually it seems impossible to devise such operations. Consequently, the transitions of solutions should be well guided, for example, in a local search or a meta-heuristic. Note that there are cases in which a solution should locally move in a direction that increases the total running time in order to reduce the number of routes. For example, the minimum number of routes is achieved for the instance in Fig. 1 when the total length is longer than the shortest route solution by r. Because, on the other hand, problems of sizes often seen in the real world have many bad local opts, algorithms should include mechanisms for escaping from local opts. The following are existing meta-heuristic frameworks in which local search is coupled with such mechanisms, and by which algorithms for delivery route optimization have been constructed:

- Algorithms that start with a solution and find neighboring solutions one by one
  - Simulated Annealing (SA) [9]
  - Iterated Local Search (ILS) [4,5]
  - Tabu Search (TS) [3,7,8,9]
  - Guided Local Search (GLS) [3,10]
- Algorithms that start with a set of solutions and find neighboring sets of solutions one by

one

– Genetic Algorithm (GA) [11,12,13]

– Rochat-Taillard's algorithm (RT) [14]

The ideas behind them are that a better local opt may exist near a good local opt, and that a better local opt may be constructed by combining good local opts. For more details, see a chapter in a text by Gendreau et al. [15]. According to the literature, they have search powers to a certain extent; however, when they are applied to real problems, good results cannot be expected without incorporating mechanisms to reduce the number of routes, because the aim of a local search used in them as a subroutine is to minimize the running time. The algorithm proposed in the next section offers an example of such a mechanism.

3. An algorithm for delivery route optimization in real problems

Delivery route optimization problems in the real world require an algorithm that can minimize the number of routes while handling complicated constraints. In this section, one such approach, an algorithm that guides a local search to escape from local opts and to reduce the number of routes is proposed.

3.1. Local search

The algorithm by the authors for delivery route optimization problems involves the following objective functions:

$L(s)$ Sum of the total running time and the total stopping time of vehicles

$P(s)$ Penalty (the extent to which constraints are violated)

$R(s)$ Number of routes,

where s is an input solution. When the optimization objective is minimization of the number of routes or the total running time, the number of routes $R(s)$ is defined as follows:

$R(s) = \sum_{i \in D} r_i$,

where r_i is a number of routes assigned to a depot i, and D is a set of depots. When the optimization objective is minimization of the maximum number of routes, $R(s)$ is defined as follows:

$$
\begin{aligned}
R(s) &= r_{max} + \left(\sum \frac{(r_i / r_{max})^2}{r_{sum}} \text{ for all } i \in D,\ r_i \neq r_{max} \right), \\
r_{sum} &= \sum_{i \in D} r_i, \\
r_{max} &= \max_{i \in D} r_i.
\end{aligned}
$$

Note that the closer the numbers of routes assigned to each depot, the greater the second term of $R(s)$.

An evaluation function $F(s,t)$ referenced in local search is defined in accordance with the optimization objective as follows:

Minimization of the running time:

$$F(s,t) = \begin{cases} Y & P(s) \geq P(t) \text{ and} \\ & L(s) + P(s) > L(t) + P(t) \\ N & \text{otherwise} \end{cases}$$

Minimization of the number of routes,
minimization of the maximum number of routes:

$$F(s,t) = \begin{cases} Y & \left[\begin{array}{c} P(s) \geq P(t) \text{ and} \\ L(s) + P(s) > L(t) + P(t) \text{ and} \\ (P(s) > 0 \text{ or } R(s) \geq R(t)) \\ \text{or} \\ P(s) \geq P(t) \text{ and } R(s) > R(t) \end{array} \right] \\ N & \text{otherwise} \end{cases}$$

The local search used in the algorithm applies four neighborhood operations: 2-opt, cross, shift, and interchange. 2-opt is a path reversion within a route. Cross is an interchange of paths between two routes such that one end of each path is a depot. Shift and interchange are relocation of a path from one route to another and interchange of paths between two routes, respectively, where the maximum lengths of paths are limited to a constant.

The object of each operation in local search — that is, the internal data representation of the solution — is a single connected loop of a doubly linked list, where each node in the list has one of the following types a priori:

- A depot (DC: distribution center),
- A point to deliver or pick up goods (visit),
- A point that expresses a return to an original depot (MDC: middle DC).

Neighborhood operations are decomposed into *flip* and *move* operations, where *flip* is a reversion of a path and *move* is a relocation of a path in the doubly linked list. A route is made to disappear by applying neighborhood operations when a path between a neighboring pair of DCs becomes empty (or MDCs only), and a route is generated when visits are inserted into an empty path between DCs.

```
1:  Initialize a list L_B of pairs of a base and gains so that it is empty.
2:  b := s.                          /* Let b be the current solution s. */
3:  for each node v ∈ s {            /* Select a node in a doubly linked list. */
4:      Consider v as a base so that N(s) will be calculated accordingly.
5:      for each neighborhood solution t ∈ N(s)
6:          if (F(b,t) = Y)          /* If a gainful neighborhood is found, */
7:              b := t.              /* update the current best solution. */
8:      if (b ≠ s)                   /* If a gainful neighborhood has been found, append the base */
9:          L_B := L_B ∪ {v, L(s) − L(b), P(s) − P(b), R(s) − R(b)}. /* v and gains to L_B. */
10: }
11: Sort L_B in decreasing order of gains, while F is referenced to compare gains.
12: Initialize a list Q_LS of non-searched vertices so that it is empty.
13: Initialize a list Q'_LS of updated routes so that it is empty.
14: for each node v ∈ L_B {          /* Select a node in L_B. */
15:     Consider v as a base so that N(s) will be calculated accordingly.
16:     b := s.                      /* Let b be the current solution s. */
17:     for each neighborhood solution t ∈ N(s)
18:         if (F(b,t) = Y)          /* If a gainful neighborhood is found, */
19:             b := t.              /* update the current best solution. */
20:     if (b ≠ s) {                 /* If a gainful neighborhood has been found, */
21:         Insert the end nodes of updated edges into Q_LS.
22:         Insert the first nodes (DCs) of updated routes into Q'_LS.
23:     }
24: }
25: while (|Q_LS| ≠ 0) {
26:     Remove a random node v from Q_LS.
27:     Consider v as a base so that N(s) will be calculated accordingly.
28:     b := s.                      /* Let b be the current solution s. */
29:     for each neighborhood solution t ∈ N(s)
30:         if (F(b,t) = Y)          /* If a gainful neighborhood is found, */
31:             b := t.              /* update the current best solution. */
32:     if (b ≠ s) {                 /* If a gainful neighborhood has been found, */
33:         Insert the end nodes of updated edges into Q_LS.
34:         Insert the first nodes (DCs) of updated routes into Q'_LS.
35:     }
36: }
```

Fig. 2. A local search with the simple-best-improve strategy

In the searching part of neighborhood operations, a *base* node (a search origin) is first selected, *partner* nodes are then selected, one by one, to identify paths to reverse or paths and places to relocate or interchange, and the gain (reduction in cost) of each operation is examined. Once a base has been selected, the radius from the base is determined within which promising partners that provide neighborhood operations to reduce the running time can be found. The proximity search with a search radius, when the metric is L_2 or L_∞, is generally performed by a k-d tree [16]; however, because the metric in the problem under consideration is determined according to the shortest paths on a digital road map, the algorithm uses a set of sorted lists for all the nodes, called a *neighbor-list*, in which nodes are sorted in order of distance.

Figure 2 shows the procedure of local search using a search strategy devised by the authors, called *simple-best-improve*. In this strategy, the first transition is the same as in the steepest decent method, and the following transitions are performed by the first-improve according to the information on gradients obtained during the first transition. The local search with the simple-best-improve strategy can find more optimal solutions than that with the first-improve strategy.

3.2. Escape methods

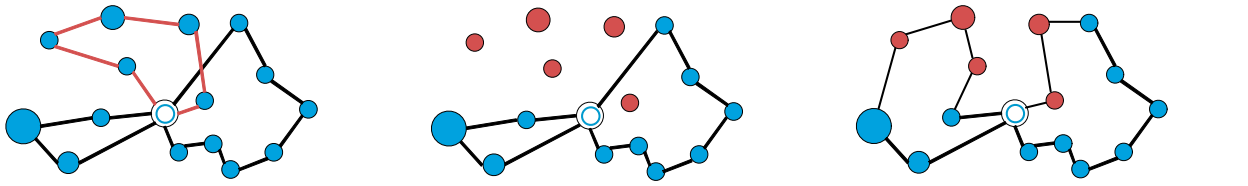
Escape methods are for changing a local opt so that the next application of the local search may output a different (preferably a more optimal) solution. The authors devised the following two escape methods (Figs. 3 and 4):

Non-increasing route extraction method: Extract one or more routes, and insert each extracted node (visit) into the best position in any non-empty routes.

Objective tuning method: Select one or more routes, set an objective tuning flag to the first node (DC) of each route so that their objective functions will be augmented in the local search.

When the optimization objective is minimization of the number of routes and the current best solution is feasible ($P = 0$), tuning flags are also set to the first nodes (DC) of empty routes. When the objective is minimization of the maximum number of routes and the target solution is feasible, tuning flags are set in the same way for routes based at depots whose number of routes is the smallest. If the tuning flag of a route is set upon evaluation of objective functions, its running time L is added to the penalty P, and the penalty is further multiplied by a constant.

According to experiments using artificial test instances, the objective tuning method performs better than other methods [4,5]. The non-increasing route extraction method, when the current



A local opt. A route is extracted. Each extracted node is inserted into any nonempty route.

Fig. 3. Non-increasing route extraction method

A route is selected. Length of the selected route is augmented in local search.

Fig. 4. Objective tuning method

solution is infeasible, can efficiently reduce its penalty. The objective tuning method does not change the solution, while it guides the local search so that the selected routes will shrink and disappear. Because it also guides the local search to prevent the use of empty routes, visits removed from shrinking or disappearing routes are inserted into one of non-empty routes if possible.

3.3. The improved iterated local search

The *improved iterated local search* (IILS) is a variant of iterated local search (ILS), to which a terminating condition is added. Figure 5 shows the procedure of the proposed algorithm, which is based on the IILS and equipped with local search with the simple-best-improve strategy, the non-increasing route extraction method, and the objective tuning method. Step 20 in the procedure may be replaced by the non-increasing route extraction method, whereby nodes in a route selected from Q_R are extracted. The list of routes Q_R contains candidate routes for tuning of objectives (or for extraction). When the best solution is updated, routes updated in the last local search are inserted into the list. The list of routes Q'_{LS} contains the first nodes

1: Set an initial solution to the current solution s, and the best solution b.
2: Initialize a list of non-searched routes Q_R so that it contains the first node (DC) for all the routes.
3: $t := LS(s)$. /* Apply the local search to the current solution s. */
4: **if** $(P(t) > 0)$ { /* If the obtained local opt t is infeasible, */
5: $m_i := 0$ for all $i \in D$. /* clear all the objective tuning flags m_i, and */
6: $t := LS(s)$. /* retry the local search. */
7: }
8: **if** $(F(b,t) = Y)$ { /* If the obtained local opt t is more optimal than the best solution b, */
9: $b := t$, $s := t$. /* update the best and current solutions, and */
10: $Q_R := Q_R \cup Q'_{LS}$. /* append the first node (DC) of updated routes into the list. */
11: } **else** { /* If the best solution has not been updated by the local search, */
12: $s := b$. /* revive the best solution. */
13: }
14: $m_i := 0$ for all $i \in D$. /* Clear all the objective tuning flags. */
15: Extract nodes that violate any constraints, and insert each into the best position.
16: **if** $(P(s) > 0)$ { /* If the solution after performing Step 15 still violates any constraints, */
17: Apply the non-increasing route extract method to s.
18: } **else** {
19: **if** $(|Q_R| = 0)$ Terminate the process.
20: $m_i := 1$ for $i \in Q_R$. /* Set an objective tuning flag to a random route i in Q_R. */
21: $Q_R := Q_R - \{i\}$. /* Remove the selected route from Q_R. */
22: $m_i := 1$ for all $i \in D$, $|r_i| = 0$. /* Set objective tuning flags to empty routes. */
23: }
24: Go to 3

Fig. 5. Procedure of the improved iterated local search.

(DCs) of routes updated by neighborhood operations, and is maintained in the local search.

The IILS equipped with the objective tuning method performs as if it were changing the configuration space landscape repeatedly, so that a local opt (a valley) becomes a penalized solution (a high mountain), and finds another local opt. It differs from other algorithms that search within a feasible space, such as tabu search [3,7,8,9]. The idea of penalizing features in a solution is also seen in guided local search (GLS) [3,10,17,18]; the difference is that, in GLS, static features (arcs) of a solution are penalized, so that the next local opt will not contain the selected features, while the proposed objective tuning method penalizes dynamic features, such as selected routes and empty routes in delivery route optimization problems, so that the next local search will remove or not generate the selected dynamic features. Note that the objective

tuning method for delivery route optimization problems guides the local search to reduce the number of routes, while GLS only guides it to escape from local opts.

4. Case studies of the delivery route optimization

This section introduces applications of the delivery route optimization to real delivery operations in Japanese banks. All the introduced examples are from real banks whose internal logistics problems were analyzed by the authors. For reasons of confidentiality, the names of the banks are not revealed, and points are not plotted on a road map. Detailed descriptions of delivery operations are also omitted for reasons of confidentiality and space limitation, and only divided or simplified problems are described.

The algorithm used for optimization is the improved iterated local search equipped with local search with the simple-best-improve strategy, the non-increasing route extraction method, and the objective tuning method. Each optimization was performed until the termination condition (Step 19 in Fig. 5) was met. The execution times for each problem ranged from a few minutes to about 30 minutes on a PC with 116-MHz CPU, depending on the problem size.

4.1. Delivery operations in banks

Delivery operations in banks are classified as follows:

- Cash delivery
- Internal mail delivery (transportation of drafts, documents, etc.)

Cash delivery includes transportation of cash between branch offices and the head office in order to maintain the amount of cash at each office at a certain level, and also includes transportation of cash in order to replenish (or extract) the amount in ATMs installed at unmanned booths. Although it depends on the banks whether the two types of delivery operations are separated or consolidated, they are generally separated into two or more operations because of the difference in load shapes and required security levels. For example, the operation introduced in the next subsection for replenishing cash in ATMs tends to be treated as a separate operation, because the loads are cartridges to be packed into ATMs, which are quite different from other loads handled in banks, and the required security level is high. This paper introduces three case studies of ATM cash replenishment operations in Japanese banks.

4.2. Operation for replenishing cash in ATMs

Case 1

Problem: In a bank, cash is currently replenished in ATMs installed at about 200 locations in a prefecture from 20 centers. The time allocated for exchanging a cartridge in an ATM is 30 minutes. The working time of cash transportation trucks is 4 hours in the morning. For each ATM, cash needs to be replenished once in a week of 5 working days. Now, is it cost-effective to consolidate the centers to two locations (places are given), reduce the cartridge exchange time to 20 minutes, and expand the working time of cash transportation trucks to 9 hours? (In this case, drivers should take a one-hour rest around noon.)

Model: Because this problem does not involve time windows or specifications for allowable service depots, and does not require irregular schedules for days in a week, it can be modeled straight-forwardly by the abstract model of the algorithm. A problem that includes all the points may be first solved, the resulting routes may be divided into five groups for each center, and each group may be then considered to be a plan for different days. Because the number of trucks currently based at each center is one, the maximum number of routes assigned to each center is constrained to five in the model of the current problem. In the model of the assumed problem, the numbers of routes are not limited.

Results: By performing optimization with the current model, a solution of 48 routes was obtained (Fig. 6). This number is almost the same as the actual number. The number of routes (for five days) assigned to each center is less than five, which means that the total required number of trucks is 20, the same as the actual number. By performing optimization with the assumed model, a solution of 18 routes (Fig. 7) was obtained, in which the numbers of routes assigned to the two centers were 11 and 7, respectively. Next, the assumed model was modified to constrain the maximum number of routes to 10, and a solution was obtained in which the numbers of routes were 9 for both centers. By considering the solution as plans for 5 days, the total number of trucks required turned out to be 4.

In this bank, centers are also normal offices, and thus the maintenance cost of centers will not change after consolidation of centers; however, it turned out that the number of trucks will become one fifth, and therefore, the transportation cost will become about one fifth.

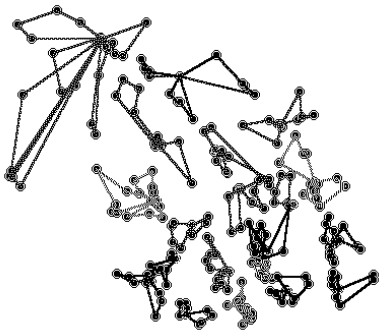


Fig. 6. Result 1 of Case 1

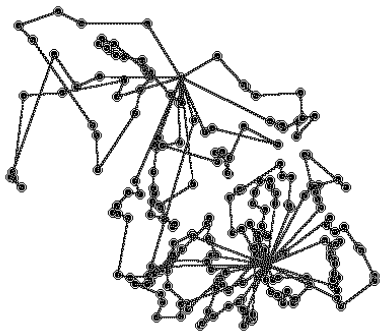


Fig. 7. Result 2 of Case 1

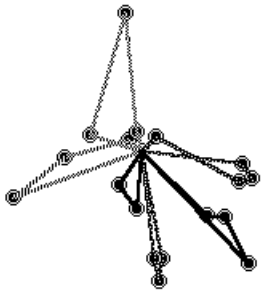


Fig. 8. A result of Case 2

Case 2

Problem: In a bank, cash is replenished in ATMs installed at about 20 locations from one center. The working time of the cash transportation trucks is 4 hours in the morning. The time allocated for exchanging a cartridge in an ATM is 20 minutes. Now, is it cost-effective to expand the working time of cash transportation trucks to 7 hours? (In this case, drivers should take a one-hour rest at the center around noon.)

Model: The rest time at the center need not be a fixed time interval, and it should be possible to move it forward or backward to obtain a better solution. Therefore, the rest time at the center is treated as returning to the center (reuse of vehicles). To constrain trucks to return to the center around noon, capacities of trucks are set to be lower than the actual values.

Results: By performing optimization with the current model, a solution of 6 trucks was obtained, while by performing optimization with the assumed model, a solution of 3 trucks was obtained (Fig. 8).

The obtained result is the same as one in which half of the routes obtained by the current model are assigned to the afternoon. In this case, it turned out that the number of trucks required can be reduced to half by expanding their working time.

Case 3

Problem: In a bank, the locations at which ATMs are installed are divided into two groups, and cash replenishment operations for these groups are performed by two transportation companies A and B, respectively. The two groups of locations are not separated into areas (spatially). There is only one distribution center. Among the 112 locations assigned to company A, 52 locations need to be replenished once a week, and 60 locations need to be replenished twice a week. Among the 72 locations assigned to company B, 51 locations need be replenished once

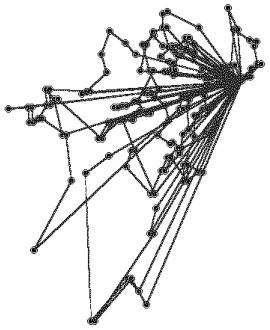


Fig. 9. Result 1 of Case 3

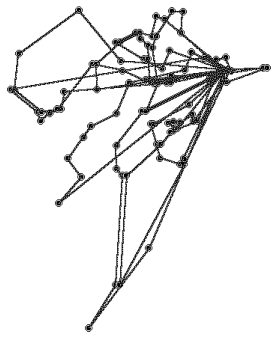


Fig. 10. Result 2 of Case 3

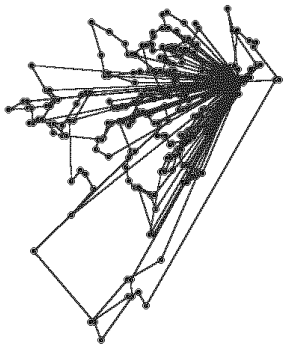


Fig. 11. Result 3 of Case 3

a week, and 21 locations need be replenished twice a week. The replenishment operation is executed on four days in a week. (The other working day is reserved for irregular operations.) The time allocated for exchanging ATM cartridges is precisely specified for each location. Drivers must take a one-hour rest around noon. Now, is it cost-effective to consolidate the two transportation companies into one?

Model: Although there is only one distribution center, represent it as two logical centers (centers 1 and 2). In the input data, each location requiring two replenishments a week is represented as two points, and centers 1 and 2 are specified for each point as allowable base centers, respectively. Each location requiring one replenishment a week is represented normally as a single point, and may be served from either one of the two centers. The objective of optimization is minimization of the maximum number of routes so that the routes assigned to each of centers 1 and 2 may be divided into two groups, and each group may be considered as a plan for a different day.

Results: By performing optimization with the current model, results were obtained that the numbers of routes assigned to centers 1 and 2 were both 10 for locations covered by company A (Fig. 9), and 6 and 5, respectively, for locations covered by company B (Fig. 10). These results mean that the required numbers of trucks are 5 and 3, respectively, which are almost the same as the actual numbers. When the optimization was performed with the assumed model, 14 routes were assigned to both centers (Fig. 11), which means that the total required number of trucks is 7.

The results showed that the required number of trucks can be reduced by one if the two companies commissioned to carry out delivery operations are consolidated into one.

5. Conclusion

In this paper, an algorithm for delivery route optimization was proposed, and its applications to cash transportation operations in Japanese banks were introduced.

The proposed algorithm includes newly devised methods and techniques, such as the simple-best-improve strategy in local searching, two escape methods — the non-increasing route extraction and the objective tuning methods — and the improved iterated local search. Among them, the simple-best-improve strategy in local searching and the improved iterated local search are not specific to delivery route optimization, and can be applied to other types of optimization problems. The objective tuning method can also be applied to other types of optimization problems if the solution representations are carefully designed. The proposed algorithm was used for real problems, and showed adequate level of performance.

As examples of how real problems can be modeled by using the model and constraints of the algorithm, cash replenishment operations in Japanese banks were introduced, and three real instances were analyzed in detail. Although one of them is not compatible with the model provided by the algorithm, it can be logically modeled by introducing virtual depots and so on. This type of technique is sometimes necessary in modeling real problems. Note, however, that this type of modeling is not desirable, and it is better to solve the problem by enhancing the modeling power of the algorithm or making it more flexible.

For all three case studies, delivery route optimization resulted in a further reduction of current logistics costs by changing delivery operations. This implies that delivery route optimization could drastically improve delivery planning in various fields where it is currently performed by human experts on the basis of experience and inspiration. For example, the speed of the delivery route optimization, which is much faster than that of human experts, enables a delivery plan for a day to be generated in the morning on that day; consequently, inflexible and costly fixed-route delivery could become obsolete. If a demand forecast is provided with the delivery route optimization, it could be used in mid-term planning of logistics operation commissions. There are many other applications in various fields.

In future studies, the authors plan to enhance the modeling power of the algorithm, and to improve its search power. They also plan to evaluate the algorithm in comparison with other approaches by using benchmark instances, and to apply the proposed methods to other types of optimization problems.

References

[1] M.W.P. Savelsbergh, "An efficient implementation of local search algorithms for constrained routing problems," European Journal of Operations Research, 47, pp. 75-85, 1990.

[2] J.-Y. Potvin, T. Kervahut, B.-L. Garcia, and J.-M. Rousseau, "The vehicle routing problem with time windows; part I: tabu search," INFORMS Journal on Computing, 8, pp. 158-164, 1996.

[3] B. de Backer, V. Furnon, P. Kilby, P. Prosser, and P. Shaw, "Solving vehicle routing problems using constraint programming and metaheuristics," Journal of Heuristics, 1997.

[4] H. Okano, "A new escape method in an iterated local search for the vehicle routing problem," Proc. ORSJ fall annual meeting, 2-C-1, pp. 138-139, 1998 (in Japanese).

[5] H. Okano, "A heuristic algorithm of the vehicle routing problem for minimizing number of routes," Proc. IPSJ SIGMPS, 98-MPS-22, pp. 37-42, 1998 (in Japanese).

[6] G.A.P. Kindervater and M.W.P. Savelsbergh, "Vehicle routing: handling edge exchanges," *in* E.H.L. Aarts and J.K. Lenstra (eds.), *Local Search in Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, pp. 337-360, 1997.

[7] F. Glover, "Tabu Search, part I," ORSA Journal on Computing, 1, pp. 190-206, 1989.

[8] F. Glover, "Tabu Search, part II," ORSA Journal on Computing, 2, pp. 4-32, 1990.

[9] I.H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," Annals of Operations Research, 41, pp. 421-451, 1993.

[10] P. Kilby, P. Prosser, and P. Shaw, "Guided local search for the vehicle routing problem," Proc. 2nd Int. Conf. of Metaheuristics, 1997.

[11] S.R. Thangiah, K.E. Nygard, and P.L. Juell, "GIDEON: a genetic algorithm system for vehicle routing with time windows," Seventh IEEE Conf. Artificial Intelligence Applications, IEEE Computer Society Press, Los Alamitos, CA, pp. 332-328, 1991.

[12] S.R. Thangiah and A.V. Gubbi, "Effect of genetic sectoring on vehicle routing problems with time windows," Proc. IEEE Int. Conf. Developing and Managing Intelligent System Projects, IEEE, New York, pp. 146-153, 1993.

[13] J.-Y. Potvin and S. Bengio, "The vehicle routing problem with time windows; part II: genetic search," INFORMS Journal on Computing, 8, pp. 165-172, 1996.

[14] Y. Rochat and E. Taillard, "Probabilistic diversification and intensification in local search for vehicle routing," Journal of Heuristics, 1, pp. 147-167, 1995.

[15] M. Gendreau, G. Laporte, and J.-Y. Potvin, "Vehicle routing: modern heuristics," *in* E.H.L. Aarts and J.K. Lenstra (eds.), *Local Search in Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, pp. 311-336, 1997.

[16] J.L. Bentley, "Multidimensional binary search trees used for associative searching," Communications of ACM, 18, pp. 509-517, 1975.

[17] C. Voudouris and E. Tsang, "Guided local search," Technical Report CSM-247, Department of Computer Science, University of Essex, 1995.

[18] C. Voudouris, "Guided local search for combinatorial problems," PhD thesis, University of Essex, Colchester, UK., 1997.