Journal of the Operations Research
Society of Japan
Vol. 43, No. 1, March 2000

A PATH-EXCHANGE-TYPE LOCAL SEARCH ALGORITHM FOR VEHICLE ROUTING AND ITS EFFICIENT SEARCH STRATEGY

Narihiro Park
The University of Tokyo

Hiroyuki Okano
IBM Research, Tokyo Research Laboratory

Hiroshi Imai
The University of Tokyo

(Received December 15, 1998; Revised April 7, 1999)

Abstract This paper proposes a fast path-exchange-type local search algorithm for vehicle routing problems (VRPs) that uses a new search strategy to restrict the search to promising neighbors. An efficient search strategy for path exchange that uses edge lengths to give search radii is proposed. Intensive numerical experiments with VRP instances of up to 10,000 vertices were conducted, and the results are compared with those given by the k-Opt algorithms. It is shown that our algorithm is faster than the 3-Opt and yields the quality of the solution comparable with the 4-Opt.

1. Introduction

Routing problems are key elements in the logistics and the manufacturing field. For example, the routing of vehicles starting at a depot and delivering goods to customers is a typical problem encountered in logistics, and has been studied for a long time. A *vehicle routing problem* in which the capacity of identical vehicles is taken into account is called the *capacitated* vehicle routing problem (CVRP), and one in which the service time window for each customer is taken into account is called the VRP with *time windows* (VRPTW). Another example of a routing problem, called the *traveling salesman problem* (TSP), can be found in the drilling of a printed circuit board where the length of a single connected route (tour) is to be minimized. This type of routing problem is also found in logistics.

Since routing problems in general belong to the class of NP-hard combinatorial optimization problems, which means that there is no hope of finding polynomial-time exact algorithms unless $P = NP$, approximation algorithms play an important role in practical applications. In the literature, approximation algorithms for routing problems generally consist of *constructive* and *local search* algorithms. Additional algorithms, called *meta-heuristics*, that guide a search to attractive solutions are also widely used [4, 5, 20, 14, 18, 16, 8]. Among these algorithms, in this paper, we focus on local search algorithm, which is applied repeatedly in many modern algorithms based on meta-heuristics (see [3] and references therein).

Local search algorithm finds neighboring solutions $N(s)$ of a current solution s, and selects a new current solution $t \in N(s)$ which reduces the objective cost; that is, $f(t) < f(s)$. For example, a local search algorithm that applies *2-exchange*, where two edges in a route are removed and another two edges are added to form a connected route, is called the *2-Opt*. When a solution s has no gainful neighbor, s is said to be *2-optimal*, and the time complexity of checking 2-optimality in a naive implementation is $O(n^2)$ for the CVRP, and $O(n^3)$ for the VRPTW, where n is the number of customers (vertices) in a route. Many studies have focused on reducing these orders in two ways: by searching only promising neighbors and

by checking the feasibility of side constraints efficiently. For example, a *search strategy* for 2-exchange to restrict a search to promising neighbors that uses the lengths of removed edges is widely known, and was efficiently implemented for the Euclidean TSP by Bentley [2] (see Section 2). For the VRPTW, Savelsbergh introduced a search strategy with an efficient way of feasibility checking for various neighborhoods including 2-exchange and Or-exchange, where the feasibility checking of a single exchange requires only a constant time [17, 9].

In this paper, we propose a new search strategy for *path exchange*, that is applicable to routing problems, and that works under any side constraints. Path exchange has been regarded as an effective neighborhood for vehicle routing [19], though no rigid description of its efficient search strategy has yet been introduced without assumptions of side constraints; for example, Savelsbergh's search strategy only works under certain side constraints, such as precedence relations or time windows of customers. Note that Kubo [10] showed a search strategy that determines promising neighbors by edge lengths for limited operations.

Our new search strategy gives *search radii* according to which promising neighbors are found only within a certain distance of a current *base vertex*, and enables us to check *l-path-optimality* in $O(\bar{k}l^2n)$, where, by l-path-optimality, we mean that a solution has no gainful path exchange with a maximum path length l, and $\bar{k}$ and n are the average number of promising neighbors found within the search radii and the number of vertices, respectively. Note that, in the case where each of m vehicles visits $O(m^{-1}n)$ vertices, the time complexity reported by Taillard, Badeau, Gendreau, Guertin, and Potvin [19] is $O(m^{-2}n^4)$, and that it is greater than ours even when l is set to $O(m^{-1}n)$.

For fast implementation of the search strategy, we use the k-d tree [1] and the *neighbor-list* [6] for proximity searching in numerical experiments. The number of promising neighbors is bounded by a constant K if we use a neighbor-list of size K for proximity search. As we found through intensive numerical experiments, $K = 50$ gives a good trade-off in practice between the computation time and the quality of the solution. Our new search strategy then gives the time complexity $O(Kl^2n)$ of checking l-path-optimality on the assumption that the neighbor-list is used.

We also report various observations that may be useful to practitioners, such as the existence of a trade-off between maximum path lengths l and computation times and comparison with TSP heuristics including the 3-Opt, 4-Opt, and Lin-Kernighan algorithms [11].

We define vehicle routing problems (VRPs), the two neighborhood operations, and the local search algorithm that applies these operations in the next two subsections. In Section 2, we review the known search strategy for 2-exchange and a technique for a proximity search. In Section 3, a new search strategy for path exchange is proposed. Section 4 describes an intensive numerical study. Finally, we summarize our paper in Section 5.

1.1. Vehicle routing problems

The local search algorithm that we introduce in this paper is applicable with small modifications to various types of routing problem, such as vehicle routing with multiple depots or with mixed truck sizes. However, we consider only the capacitated vehicle routing problem (CVRP). Note that because our search strategy works under any side constraints, it is easy to incorporate efficient feasibility-checking algorithms and apply them, for example, to the VRPTW.

In the CVRP we are given a set $V = \{v_0, v_1, \ldots, v_{n-1}\}$ of vertices and a cost $d_{ij} \geq 0$ for each vertex pair $\{v_i, v_j\}$, where we assume costs are symmetric; that is, $d_{ij} = d_{ji}$. Vertex v_0 represents a depot at which vehicles of capacity Q are based, and v_i represents customers who require goods in quantities $q_i > 0$. The CVRP involves finding vehicle routes with

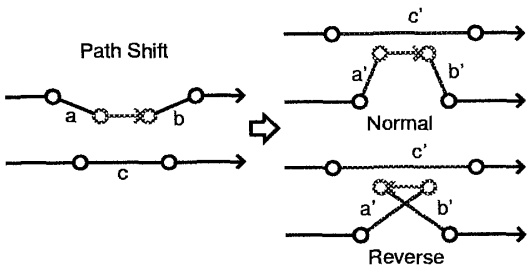

Figure 1: Shift operations

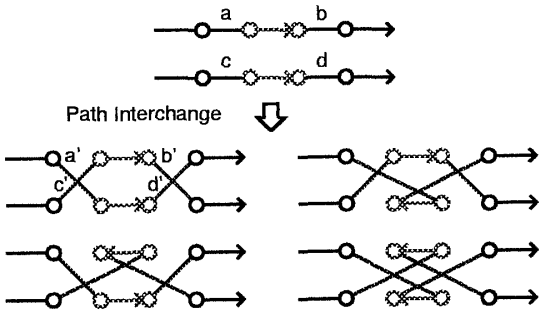

Figure 2: Interchange operations

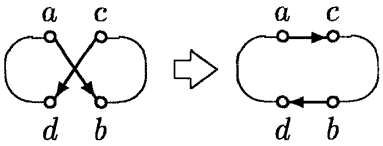

Figure 3: A 2-exchange

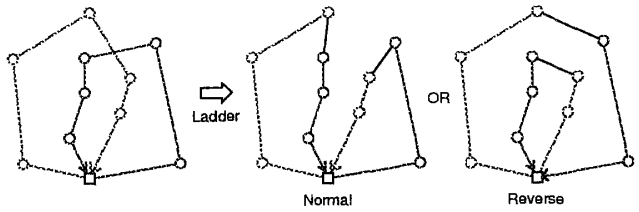

Figure 4: Ladder operations

the minimal total cost, each starting and ending at the depot, so that every customer is visited exactly once and the total quantity of goods carried on any route does not exceed the vehicle capacity Q.

We can view VRPs in the context of the TSP by incorporating a sufficient number of additional *depot vertices* $V' = \{v_n, v_{n+1}, \ldots\}$ as duplicates of v_0, and by constraining the total quantities of goods transported on paths between any pair of depot vertices $v \in V' \cup \{v_0\}$ not to exceed the vehicle capacity Q. By doing so, in Section 4, we can compare our local search algorithm with existing TSP heuristics even in the context of the CVRP. See [12] for a technique to apply a TSP heuristic for vehicle routing.

1.2. Neighborhood operations and the local search algorithm

We now introduce the two neighborhood operations, path exchange and 2-exchange, that we focus on in this paper.

Our path exchange selects two distinct paths, p and q, in which the numbers of vertices $L(p)$ and $L(q)$ are at most l ($l \in \{1, 5, 10, 15\}$ in numerical experiments in Section 4). In VRPs, two paths are selected from different routes. Either p or q may be empty, in which case the exchange is called (l-*path*) *shift* (Figure 1); otherwise, it is called (l-*path*) *interchange* (Figure 2). Note that shift with $l \leq 3$ was first introduced by Or [13] and general exchange operation, called λ-interchange, in which λ vertices are exchanged between routes was first introduced by Osman [14] and used with $\lambda \leq 2$. Thanks to our efficient search strategy, we can exchange paths longer than two in numerical experiments.

There are two types of 2-exchange: intra-route and inter-route. Intra-route 2-exchange removes two edges in a route and adds another two edges to form a connected route (Figure 3). Inter-route 2-exchange (which we call *ladder* in this paper), removes two edges from

distinct routes and add two edges to form different distinct routes (Figure 4). There are two types of ladder: normal and reverse. A normal ladder preserves the directions of paths (precedences of vertices), while a reverse ladder does not. Potvin, Kervahut, Garcia, and Rousseau called the normal ladder the 2-Opt* [15] and used only this type of ladder operation.

The local search algorithm proposed in this paper applies both path exchange and 2-exchange. Our local search algorithm has two modes – *first improve* and *best improve* – and proceeds in the following manner: an arbitrary *base* vertex v is selected, promising *partner* vertices are selected according to our search strategy, and a neighbor $t \in N(s, v)$ of a current solution s with the least cost for the current base v is selected. In the first-improve mode, the selected neighbor t is *applied* ($s \leftarrow t$) for each base if it is gainful; that is, if $f(t) < f(s)$. In the best-improve mode, all the vertices $v \in V$ are tested as bases, and the neighbor with the least cost is applied if it is gainful.

2. Preliminaries

2.1. A search strategy for 2-exchange

We first review the well-known search strategy for 2-exchange. Consider a 2-exchange operation that removes two edges (a, b) and (c, d) in a route and adds other edges (a, c) and (b, d) to form a connected route (Figure 3). If the operation is gainful – that is, if the total cost of the new route is less than that of the old one –

$$d_{ac} + d_{bd} < d_{ab} + d_{cd} \tag{2.1}$$

must be satisfied. This requires that at least one of the following should be satisfied:

$$d_{ac} < d_{ab}, \tag{2.2}$$

$$d_{bd} < d_{cd}. \tag{2.3}$$

Therefore, supposing that we want to find the second edge (c, d) after removing the first edge (a, b), we have only to search for edges whose adjacent vertex c is within a distance d_{ab} of a. We call a search origin a a *base*, and a corresponding vertex c a *partner*.

In Section 3, we expand this technique for path exchange and propose a new search strategy.

2.2. Proximity search

For fast implementation of the 2-Opt, we need an efficient method of proximity searching. In a metric space, we can use the k-d tree [1], which enable us to efficiently find all vertices within a certain distance of a vertex.

Another possible choice is the *neighbor-list* introduced by Johnson, Bentley, McGeoch, and Rothberg [6]. This is a set of vertex lists in which each list corresponds to a vertex and contains the sorted K nearest-neighbor vertices. Proximity search with the neighbor-list is much faster than that with the k-d tree, since it is done simply by scanning the list. It may be noted that the neighbor-list is also applicable in a non-metric space.

Using the neighbor-list automatically limits the number of vertices to be considered in a search based at any vertex to the size K of the neighbor-list. In other words, we have to give up considering vertices farther than the K-th nearest vertex, which means that we might miss gainful operations involving edges whose adjacent vertices are farther than the K-th nearest vertex. On the other hand, we can perform an iteration of the 2-Opt (a search based at a vertex) within a constant time $O(K)$, and thus the time complexity of checking 2-optimality is $O(Kn)$.

Johnson et al. reported that when $K \geq 20$, the qualities of solutions for the TSP are not much worse than those obtained with $K = (n-1)$ where n is the number of vertices. In Section 4, we use $K = 50$ in our experiments for the CVRP.

3. A New Search Strategy for Path Exchange

3.1. Search strategy for l-path interchange

Consider a path-interchange operation that removes edges a and b in one route and edges c and d in another route, and adds new edges a', b', c', and d' (Figure 2). If the total cost of the new solution is less than that of the old one,

$$a' + b' + c' + d' < a + b + c + d \tag{3.1}$$

must be satisfied, where the names of edges denote their costs. This requires that at least one of the following is satisfied:

$$a' < a, \tag{3.2}$$
$$b' < b, \tag{3.3}$$
$$c' < c, \tag{3.4}$$
$$d' < d. \tag{3.5}$$

Consequently, we can efficiently find path-interchange operations that are gainful. At each iteration of our local search algorithm, a base vertex and its adjacent edge to be removed are fixed, and candidate targets for the operation are searched for in the following manner: partner vertices within a certain distance (search radius) of the base are searched for first, and then neighborhood operations that include the base and the partner are examined. For each pattern of path interchange in Figure 2, because the interchange is symmetric, we only have to search for partners that are within a distance a of the base, where a is the first edge to be removed. A case to note is when the base or partner is a vertex representing the depot. As is mentioned in the first section, we view a set of VRP routes as a TSP tour by adding depot nodes as duplication of the original depot vertices. It follows that each depot vertex has only one pair of adjacent vertices, and can be handled in the same way as other vertices. For example, when the depot is found within a search radius from a base, all the depot nodes, each represent a distinct route, are examined as partners.

Note that the factor of l does not appear in the search radius. Note also that, when we use a neighbor-list of size K, the number of partners is at most K. A detailed algorithm for l-path interchange is described in Section 3.3.

3.2. Search strategy for l-path shift

Consider a path-shift operation that removes edges a and b in a route and an edge c in another route, and adds new edges a', b', and c' (Figure 1). If the total cost of the new solution is less than that of the old one.

$$a' + b' + c' < a + b + c \tag{3.6}$$

must be satisfied, where the names of edges denote their costs. If we suppose that $a' > c$ and $b' > c$, then at least one of the following must be satisfied:

$$a' < a + b - c', \tag{3.7}$$
$$b' < a + b - c'. \tag{3.8}$$

```
// Given base ∈ route1 and partner ∈ route2
i := base;
For count1 := 1 to l {
    i' := (count1 − 1)-th vertex after i;
    j := the next vertex after the partner;
    For count2 := 1 to l {
        j' := (count2 − 1)-th vertex after j;
        If (interchange of paths (i, i') and (j, j') leads to overcapacity on route1)
            Exit loop of count2;  // In order to stretch path (i, i');
        Else
            examine interchange of paths (i, i') and (j, j').
    }
}
```

Figure 5: Detailed algorithm for l-path interchange incorporating the capacity constraint

Consequently, we can search for path-shift operations in two cases: (1) when a base is adjacent to c, and c is the first edge to be removed, (2) when a base is adjacent to a, and a is the first edge to be removed. Path-shift operations in which $a' \leq c$ or $b' \leq c$ holds can be found in case 1 with a search radius c, where a path to shift starts at the partner vertex adjacent to a. Those in which $a' > c$ and $b' > c$ both hold can be found with a search radius $a + b - c'$, where a path to shift starts at the base vertex adjacent to a.

When an algorithm is designed to search for two or more path-shift operations starting at a base as shown in Section 5 for the interchange operation, a search radius should be long enough for finding all the possible patterns. Therefore, a search radius for path-shift operations is set to $\max(c, a+b-c')$, the maximum value for cases 1 and 2, and $a+b-c'$ is further determined to be the largest value for all the choices of a, b, and c'. Note that only one operation may be applied at a search starting at each base even when the algorithm searches for multiple patterns of operations.

3.3. Detailed algorithm for l-path interchange

Figure 5 is a detailed algorithm for l-path interchange in our local search algorithm for the capacitated VRP. It is one of the four patterns in Figure 2; the other three are obtained by modifying it slightly. Figure 6 shows the order in which a path is stretched in the algorithm.

Let m be the minimum possible number of vehicles ($m \simeq \lceil \sum_i q_i/Q \rceil$), and let $\bar{k}$ be the average number of partner vertices found in an iteration. (If we use the neighbor-list, $\bar{k}$ is bounded by the size K of the neighbor-list, which is a constant value.) In the case where the number of vertices visited by each vehicle is $O(\frac{n}{m})$, the time complexity of checking l-path-optimality reported by our search strategy is $O(\bar{k}m^{-2}n^3)$, since we only search for *partner paths* of $O\left(\bar{k}\frac{n}{m}\right)$ from each base, the number of *base paths* for each base is $O\left(\frac{n}{m}\right)$, and the number of bases is n. Note that one by Taillard, et al. [19] is $O(m^{-2}n^4)$.

Moreover, if an upper limit is specified for the path length l, as we do so in our local search algorithm, the number of partner paths from each base becomes $O(\bar{k}l)$ regardless of the number of vertices included in each route, and that the complexity is reduced to $O(\bar{k}l^2n)$.

For routing problems with side constraints, several studies have provided efficient search strategies for local search, which avoid searching unpromising neighbors by using side constraints. Our search strategy, which uses distance information, can also cooperate with

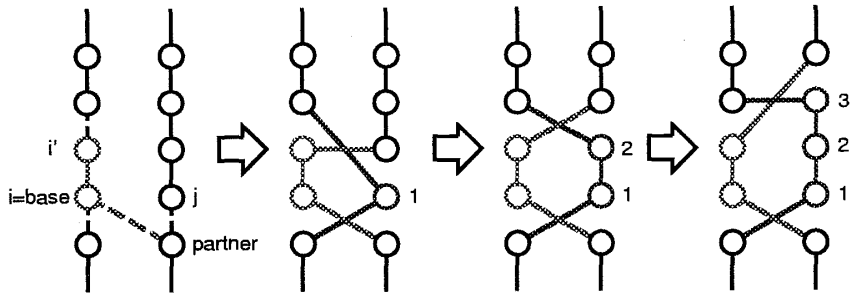

Figure 6: Stretching a partner path

those strategies and further decrease the computation time. Indeed, the algorithm in Figure 5 shows how to utilize capacity constraints. By avoiding estimation of overcapacity interchanges, the number of interchanges whose cost need to be examined at each iteration (each search at a base) is likely to decrease in practice.

4. Numerical Experiments

In this section, we demonstrate through intensive numerical experiments the performance of our local search algorithm using path exchange and 2-exchange, whose efficient search strategy for VRPs was described in the previous two sections. We show the difference in performance of our algorithm for various setups, such as with various maximum path lengths l and with either best-improve or first-improve mode. We compare our algorithm with typical heuristics for the TSP.

As explained in Section 1, we handle VRPs in the context of the TSP; that is, we represent routes as a single connected route and consider additional constraints that reflect the original problems. For the CVRP, we constrain the total quantities of goods transported on paths between any pairs of depot vertices to be within a given vehicle capacity.

We experiment by using random instances generated in a unit square with a size of 100, 316, 1000, 3162, or 10000, and by using benchmark instances for the CVRP. The results of the experiments with random instances for each size are shown as averages over ten runs. In random instances, demand for all vertices is set to 1, and the vehicle capacity Q in random instances is set to $\sqrt{n}$, where n is the number of vertices. This implies that the number of routes is $O(\sqrt{n})$.

We use the nearest-neighbor method for constructing initial solutions, where the nearest unvisited vertices are visited one by one unless the total quantity of goods transported in a route exceeds the vehicle capacity Q times 0.9. (The reason for using a factor of 0.9 is to make it easy to apply shift operations.)

We measure computation times as the CPU times in seconds on an IBM RS/6000, whose CPU is a PowerPC 112 MHz, and the qualities of solutions for random instances as the percentages of the expected Held-Karp lower bound, the expected lower bound of the TSP for random instances in a unit square, which is calculated as follows [7]:

$$\text{Expected Held-Karp lower bound} = \sqrt{N} \times \left(0.70805 + \frac{0.52229}{N^{0.5}} + \frac{1.31572}{N} + \frac{3.07474}{N^{1.5}}\right). \tag{4.1}$$

Note that we measure the quality of solutions by the TSP's lower bound for the CVRP for convenience, since there is no well-established and simple way to compute good lower

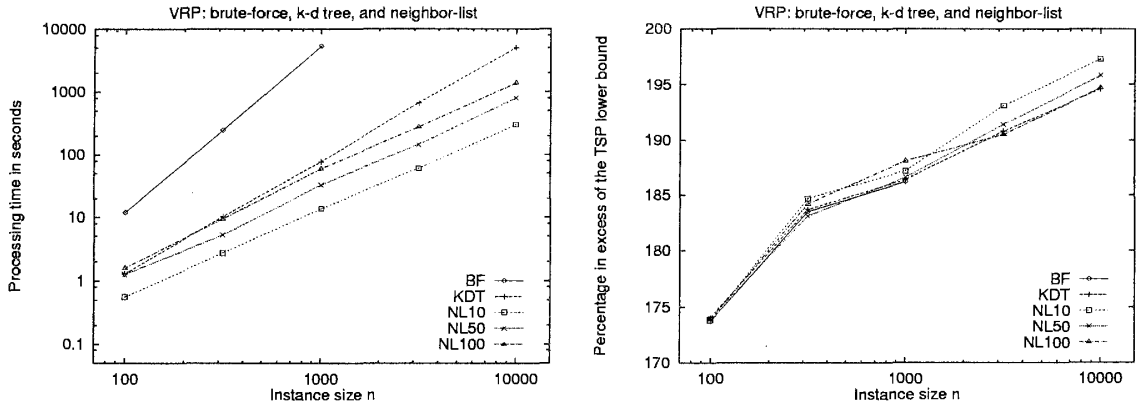

Figure 7: Performance of the local search algorithm with different proximity search methods

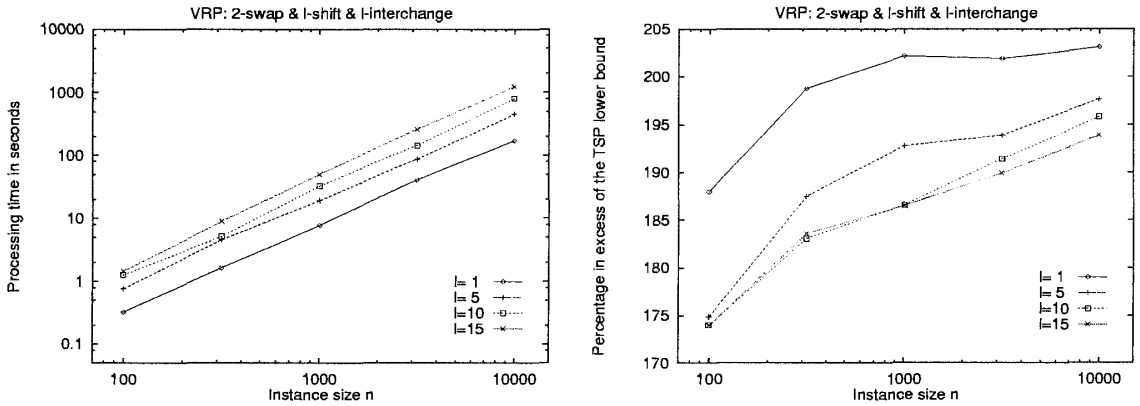

Figure 8: Performance of the local search algorithm with different settings of l

bounds for the CVRP.

4.1. Performance of the fast implementation

Figure 7 shows the computation times and solution qualities of local search algorithms that apply 10-path exchange and 2-exchange. In Figure 7, the BF is a naive implementation, not using the proposed search strategy, that examines all the possible 10-path exchanges. The KDT and the NLs use our search strategy and examine only promising candidates. The KDT uses a k-d tree and the NLs use neighbor-lists of sizes 10, 50, and 100 for proximity search. The computation times shown in the figure do not include the time taken to obtain initial solutions, k-d trees, and neighbor-lists.

Figure 7 shows that our fast algorithms, the KDT and the NLs, are much faster than the BF, which means that the speed of l-path exchange is drastically improved by our efficient search strategy. The figure also shows that the performance of the algorithms depends on the size of neighbor-lists. As the size K increases, the computation time obviously increases and the solution quality is likely to be better, while increasing the size from 50 to 100 makes less difference than increasing it from 10 to 50. We adopt $K = 50$ as a well-balanced size of neighbor-lists for the CVRP, and use this value in the following experiments.

4.2. Trade-off between speed and solution quality

Figures 8 and 9 show the computation times and solution qualities of the local search algorithm that applies l-path exchange where $l \in \{1, 5, 10, 15\}$. In Figure 9, the ladder

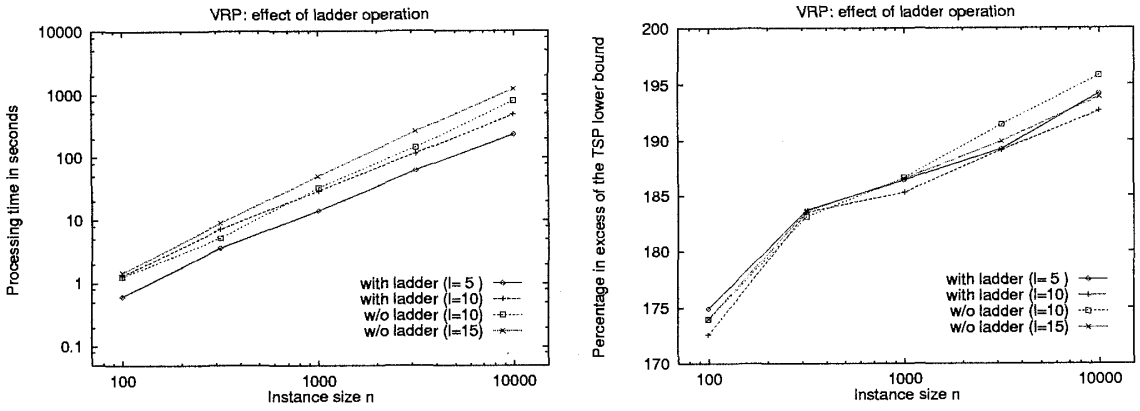

Figure 9: Effect of ladder and l-path exchange

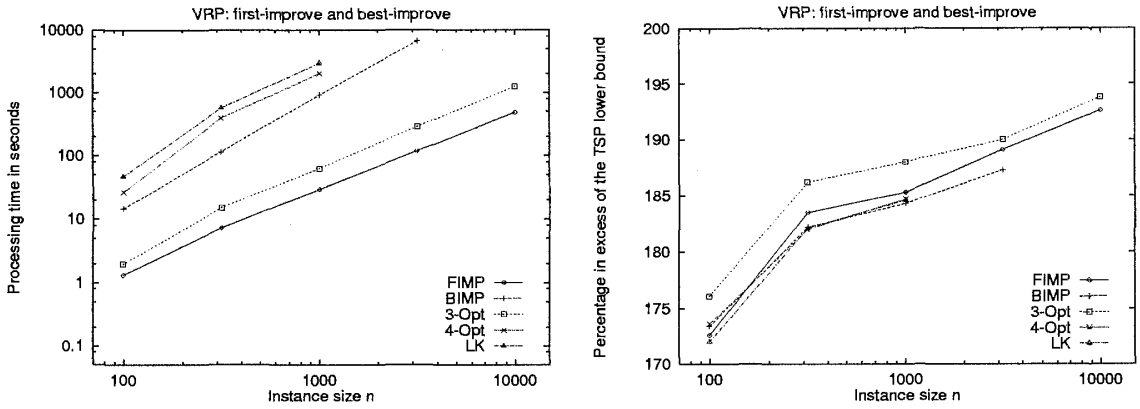

Figure 10: Comparison of various local search algorithms

neighborhood is used in addition to l-path exchange.

A remarkable point is the excellent performance of the ladder neighborhood; that is, we can improve the performance of the l-path-exchange local search by adding the ladder neighborhood. Even the speed is improved, because using ladder is likely to make it faster to reach locally optimal solutions by further improving a solution at each iteration of the local search. Note also that using ladder decreases the effect of l on solution qualities, which means ladder has similar effect to that of expanding the maximum length l of paths in path exchange by about five (Figure 9).

As for the appropriate value of l, we found that $l \geq 10$ is required.

4.3. Comparison with other local search algorithms

Figure 10 shows the computation times and solution qualities of the 10-path exchange algorithm (with the ladder neighborhood for the VRP) in the first-improve (FIMP) and best-improve (BIMP) modes, and of the 3-Opt, 4-Opt, and Lin-Kernighan (LK) algorithms [11]. The implementations of the 3-Opt, the 4-Opt, and the LK are the same program code, where the maximum number of edges exchanged by one operation is limited to three and four in the 3-Opt and the 4-Opt, respectively, and is not limited in the LK. All of the algorithms were implemented by using neighbor-lists. In the k-Opt algorithms, the capacity constraint of each intermediate solution is checked whenever the solution satisfies the tour constraint that the solution should form a connected route.

Table 1: Applications to benchmark instances
(upper: percentages in excess of best-known values, lower: processing times in seconds)

	c50	c75	c100	c120	c150	c199	tai385	Average
FIMP	9.3	3.7	3.8	14.8	5.0	4.7	5.7	6.7%
	0.2	0.6	1.0	1.7	1.6	3.9	11.6	2.9 sec.
BIMP	5.5	5.7	1.6	0.8	5.7	5.3	3.9	4.1%
	1.8	6.7	12.8	15.2	30.4	59.3	375	71.7 sec.
3-Opt	10.5	5.2	5.0	15.0	9.7	8.7	4.9	8.4%
	0.3	1.6	1.0	4.9	3.6	7.5	101	17 sec.
4-Opt	4.8	3.3	2.1	4.3	6.6	5.8	5.6	4.6%
	1.8	15.9	7.3	96.6	36.5	156	3575	556 sec.
LK	6.7	3.3	2.4	1.3	3.5	5.3	4.1	3.8%
	3.0	30.6	15.0	133	82.0	277	6185	961 sec.

The figure shows that path exchange, especially in the first-improve mode, is the best-balanced local search for the CVRP. The quality of a solution obtained by the first-improve path-exchange local search is comparable with that obtained by the 4-Opt, and that obtained by the best-improve local search is comparable with that obtained by the LK, while path-exchange local search is much faster than the k-Opt algorithms.

For the CVRP, the capacity constraint severely affects the k-Opt algorithms such as the LK. It limits their neighborhood to a considerable extent, and requires them to spend a lot of time on feasibility checking. Therefore, a combination of path-exchange and 2-exchange is much preferable. Moreover, its fast speed is very attractive, since modern meta-heuristics require many iterations of local searching. By using the fast local search algorithm for path exchange and 2-exchange, meta-heuristics can be iterated many more times and may yield better solutions.

Table 1 shows the computation times and solution qualities of our local search algorithm in the first-improve (FIMP) and best-improve (BIMP) modes, and with the 3-Opt, 4-Opt, and Lin-Kernighan algorithms, when it was applied to six of Christofides instances and one of Taillard. This result also supports the assertion that path-exchange local search is more suitable for the CVRP than the k-Opt algorithms.

5. Conclusion

We have focused on two types of neighborhood operations for vehicle routing problems – path exchange and 2-exchange – and proposed an efficient search strategy for path exchange. Our new search strategy for l-path exchange enables us to check whether no l-path exchange can be applied (l-path-optimality) in $O(\bar{k}l^2n)$ for vehicle routing problems, where $\bar{k}$ is the average number of candidate vertices in each iteration, l is the maximum length of paths, and n is the number of vertices.

We introduced a local search algorithm that applies these neighborhood operations in either the first-improve or best-improve modes. Through intensive numerical study, we showed that the speed of the local search algorithm is drastically improved by our search strategy. We also revealed that use of an inter-route 2-exchange, or ladder, has similar effect to that of expanding the maximum length l of paths in path exchange by about five. Finally, we showed that the quality of the solution given by the local search algorithm in the first-improve mode, which we consider the best-balanced local search for the CVRP, is

comparable with that of the solution given by the 4-Opt.

Acknowledgment
The authors would like to thank anonymous referees for their valuable comments on the original version which improved this paper very much.

References

[1] J.L. Bentley: Multidimensional binary search trees used for associative searching. *Communications of ACM*, **18** (1975) 509-517.

[2] J.L. Bentley: Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, **4** (1992) 387-411.

[3] 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, (1997) 311-336.

[4] F. Glover: Tabu search, part I. *ORSA Journal on Computing*, **1** (1989) 190-206.

[5] F. Glover: Tabu search, part II. *ORSA Journal on Computing*, **2** (1990) 4-32.

[6] D.S. Johnson, J.L. Bentley, L.A. McGeoch, and E.E. Rothberg: Near-optimal solutions to very large traveling salesman problems. in preparation (1996).

[7] D.S. Johnson, L.A. McGeoch, and E.E. Rothberg: Asymptotic experimental analysis for the Held-Karp traveling salesman bound. *Proc. 7th ACM SIAM Symp. on Discrete Algorithms*, (1996).

[8] P. Kilby, P. Prosser, and P. Shaw: Guided local search for the vehicle routing problem. *Proc. 2nd Int. Conf. of Metaheuristics*, (1997).

[9] 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, (1997) 337-360.

[10] M. Kubo: Development of decision support systems for logistics network design and vehicle routing problems. *Proceedings of the Ninth RAMP Symposium*, (1997) 28-42.

[11] S. Lin and B.W. Kernighan: An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, **21** (1973) 498-516.

[12] H. Okano: Extension of a traveling salesman heuristic for vehicle routing. *IBM Research Report*, RT0225 (1997).

[13] I. Or: Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking. *Ph.D. thesis, Department of Industrial Engineering and Management Science* (Northwestern University, Evanston, IL, 1976).

[14] I.H. Osman: Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, **41** (1993) 421-451.

[15] J.Y. Potvin, T. Kervahut, B.-L. Garcia, and J.M. Rousseau: A tabu search heuristic for the vehicle routing problem with time windows; part I: tabu search. *INFORMS J. on Computing*, **8** (1996) 158-164.

[16] Y. Rochat and E. Taillard: Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, **1** (1995) 147-167.

[17] M.W.P. Savelsbergh: An efficient implementation of local search algorithms for constrained routing problems. *European J. of Operational Research*, **47** (1990) 75-85.

[18] E.D. Taillard: Parallel iterative search methods for vehicle routing problems. *Networks*, **23** (1993) 661-673.

[19] E.D. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.Y. Potvin: A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, **31** (1997) 170-186.

[20] 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, Los Alamitos, CA, (1991) 332-328.

[21] 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, (1993) 146-153.

Hiroyuki Okano
IBM Research, Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato
Kanagawa-ken 242-8502, JAPAN
E-mail: okanoh@jp.ibm.com