Efficient Local Search for CVRP, Move Decomposition and Sequential Search

Stefan Irnich* Tore Grünert† Birger Funke†

*Deutsche Post Lehrstuhl für Optimierung von Distributionsnetzwerken, Aachen University
Templergraben 64
52062 Aachen, Germany
sirnich@or.rwth-aachen.de

†GTS Systems and Consulting GmbH
Raiffeisenstr. 10
52134 Herzogenrath, Germany
{gruenert,funke}@gts-systems.de

1 Introduction

Nearly all heuristics and metaheuristics for vehicle routing and scheduling problems (VRSP) rely heavily on the definition of neighborhood solutions and local search (LS). This paper introduces new LS techniques applicable to well-known neighborhoods of the capacitated vehicle routing problem (CVRP). These techniques are *sequential search* based on the gain criterion of Lin and Kernighan [5] and *indirect search techniques*, which are based on the decomposition of moves into so-called partial moves. Indirect search allows to identify the best improving neighbor solution in the node exchange neighborhood, which is of size $\mathcal{O}(n^4)$, with quadratic effort. The superiority of sequential and indirect search over straightforward approaches is not only of theoretical interest, but is essential for solving large-scale CVRP. We observed speedups by factors between 10 and 200.

The extended abstract is organized as follows. In the remainder of this section, we clarify the definition of moves in LS algorithms and their decomposition into partial moves, and formalize the concept of gains of (partial) moves. Section 2 gives a short overview over different neighborhoods for the CVRP. The search acceleration techniques and corresponding neighborhoods are presented in Section 3 and computational results in Section 4.

1.1 Local Search and Move Decomposition

We assume that the reader is familiar with the definition of the CVRP and LS. Given a combinatorial optimization problem $\min_{x \in X} c(x)$, the basis of all LS methods is the use of a set of *elementary moves* that transform a given solution $x \in X$ into a different, so-called,

neighbor solution x'. The set of all solutions that can be reached from the current solution using the set of moves is called the *neighborhood* of the current solution w.r.t. the move set, i.e. $\mathcal{N}(x) \subset X$.

For a precise definition of the term *move*, it is helpful to consider an enclosing superset of *solutions*, $Z \supseteq X$ (i.e. route plans which do not necessarily respect the capacity constraints). The idea is that some of the moves $m \in M$ might transform a feasible solution x into an object $m(x)$, which has a structure similar to a feasible solution, but does not necessarily satisfy all constraints that define feasible solutions. An example is the swap of two customers between two CVRP tours, which might violate a capacity constraint. In general, we denote by M the *set of moves* where a move $m \in M$ is a (possibly partial) map from Z to itself, i.e. $m : Z \to Z$. For a given $x \in Z$, the *extended neighborhood* $\hat{\mathcal{N}}(x) = \{m(x) : m \in M\}$ contains all neighbors of x, either feasible or infeasible. Clearly, the neighborhood $\mathcal{N}(x) \subset X$ is given by $\mathcal{N}(x) = \hat{\mathcal{N}}(x) \cap X$. Every move, $m \in M$, with $m(x) \in X$ is called a *feasible move* w.r.t. x.

In order to analyze different moves, we will decompose them into smaller parts, the so-called *partial moves*. A given decomposition $m = p_l \circ \ldots \circ p_2 \circ p_1$ of a move m into $l \geq 2$ partial moves $p_1, p_2, \ldots, p_l$ means that an $x \in Z$ is first transformed into $p_1(x)$, second $p_1(x)$ is transformed into $p_2(p_1(x))$, and so on. Of course, we have to consider the structures that occur after having applied one or several partial moves. In general, the ith partial move transforms elements of an intermediate structure Y_{i-1} to elements of another intermediate structure Y_i, while for the first and last structure $Y_0 = Y_l = Z$ holds. As a result, $m : Z \to Z$ decomposes into

$$m: \quad Z = Y_0 \xrightarrow{p_1} Y_1 \xrightarrow{p_2} Y_2 \xrightarrow{p_3} \quad \ldots \quad \xrightarrow{p_{l-1}} Y_{l-1} \xrightarrow{p_l} Y_l = Z.$$

We neither claim that the decomposition into partial moves is self-evident nor that it is unique. However, it strongly influences the efficiency of corresponding search algorithms. The paper provides several examples of such decompositions.

1.2 Costs and Gains of (Partial) Moves

Recall that $c(x)$ is the cost of a solution $x \in Z$. We denote the *gain* of move $m \in M$ applied to solution $x \in Z$ by $g(m, x) := c(x) - c(m(x))$. For the CVRP cost is related to the edges chosen in the solution x. We assume that the overall cost is the sum of the costs $c_{ij}(x)$ associated with the edges in the solution, i.e. $c(x) = \sum_{(i,j) \in solution(x)} c_{ij}(x)$. In order to implement efficient pruning rules in LS, it is necessary to allocate a *gain* $g(p_i, x)$ to each of the partial moves p_i, $i = 1, \ldots, l$, depending only on the current solution x, but not on any intermediate solution. Let the move $m \in M$ be decomposed into partial moves $p_l \circ \ldots \circ p_2 \circ p_1$. For the gain functions it is desirable that the gain of a move m is the sum of the gains of its partial moves $p_1, \ldots, p_l$. A decomposition $m = p_l \circ \ldots \circ p_2 \circ p_1$ is called *cost-independent* if

$$g(m, x) = \sum_{i=1}^{l} g(p_i, x) \tag{1}$$

holds. If the equality is not fulfilled in all cases, then sufficient conditions which guarantee (1) can be given. These are called *legitimacy conditions*, cf. [3]. For example, legitimacy conditions might require that only compatible subsets of partial moves occur simultaneously or restrict the ordering of partial moves.

For the case of identical intermediate structures (i.e. $Y_i = Z$, $i = 0, 1, \ldots, l$), the decomposition $m = p_l \circ \ldots \circ p_2 \circ p_1$ is *order-independent* if $m(x) = p_{\pi(l)} \circ \ldots \circ p_{\pi(2)} \circ p_{\pi(1)}(x)$ holds for all solutions $x \in Z$ and all permutations π of $\{1, 2, \ldots, l\}$. It is called *cyclic-independent* if the same holds only for cyclic permutations π. We will show below that cyclic-independent move decompositions are essential in the development of efficient search algorithms.

2 Neighborhood Types

For the description of different neighborhood types, we assume that CVRP solutions x are given by their edges. Then, each solution $x \in Z$ can be transformed into any other solution $x' \in Z$ by deleting and adding a number of edges. Hence, every LS method could, in principle, be regarded as a special variant of an edge exchange. On the other hand, some transformations can be described better by considering nodes. Typical examples are the relocation of one or several nodes from one route to another or from their current positions to different positions within the same route. In the following, we will use the term *edge exchange* when the number of directly involved nodes is larger than the number of directly involved edges, and we use the term *node exchange* when the opposite is true. **Node-exchanges** are Relocation, Exchange, λ-Interchange, Node-Ejection-Chains, and Cyclic-Transfers. **Edge-exchanges** are k-Opt, k-Opt*, Or-Opt and special inter-route-neighborhoods like Cross-Exchange, String-Relocation, String-Exchange. Another well-known category of neighborhoods are the **combinatorial neighborhoods** which are beyond the scope of this paper. For further details see [2].

3 Search Techniques

Given a neighborhood for the CVRP, how can one search efficiently in order to reach a local optimum as quickly as possible? We will distinguish between two generic approaches: *direct search by enumeration* and *indirect search by optimization*. By direct search, we mean approaches that subsequently add and/or delete edges or nodes and evaluate the result of these operations directly. Usually one can consider these approaches as some type of tree search method. Indirect search methods try to map the problem of finding a best improving solution in the neighborhood into some optimization problem, such as a shortest path, assignment or set packing problem. The search for improving neighbor solutions is then equivalent to solving the optimization algorithm exactly or by a heuristic.

3.1 Sequential Search

Sequential search is a direct search method. It is a cost or gain oriented search algorithm that exploits the cost-independence of moves and cyclic independence of neighborhoods. The basic idea of this approach is to consider all relevant partial moves of a cyclic independent neighborhood recursively. The elements in sequential search are *not* selected according to any order specified by the current solution but rather considering *candidate lists*. These candidate lists are sorted in the order of increasing cost of the elements. Sequential search is particularly

attractive if the number of elements in every selection step (i.e. the length of the candidate list) is reduced from $\mathcal{O}(n)$ to some small constant.

In order to search a neighborhood exactly by sequential search, it has to be *cyclic-independent*. The second requirement is that the partial moves have to be *cost-independent*. Many of the above-mentioned node- and edge-exchange neighborhoods have a cyclic-independent decomposition into partial moves.

The attractiveness of sequential search in cyclic neighborhoods is due to the following theorem of Lin and Kernighan, [5]: *If a sequence of numbers* $(g_i)_{i=1}^k$ *has a positive sum* $\sum_{i=1}^k g_i > 0$, *there is a cyclic permutation* π *of these numbers such that* every partial sum *is positive, i.e.* $\sum_{i=1}^p g_{\pi(i)} > 0$ *for all* $1 \leq p \leq k$.

The theorem implies that, for finding an improving move $m = p_k \circ \ldots \circ p_1$ within a given neighborhood which decomposes into cyclic-independent partial moves, we need only consider those moves where $G_i := \sum_{l=1}^i g(p_i, x) > 0$ for all $i = 1, \ldots, k$. The direct implication is that at stage i of the search we need only consider moves with a gain $g(p_i, x) > -G_{i-1}$. Thus, the total gain at stage $i-1$ limits the choice of a partial move at stage i. We refer to this rule as the *gain criterion*. The gain criterion is fundamental for the effectiveness of the Lin-Kernighan algorithm for the TSP, see [5]. The gain criterion is also exploited in effective algorithms for 2- and 3-Opt for the TSP, see, e.g., [1, 4]. It is easy to see that it is also applicable to 2-Opt, Or-Opt, and 2-Opt* moves for the CVRP. The feasibility of these moves can always be checked in constant time $\mathcal{O}(1)$ when additional information like the residual capacity of a route and accumulated load is stored at each node of the current solution. However, we are not aware of any of the 'modern' metaheuristics for VRSPs that exploit this very important speedup technique.

We now give examples of how sequential search can be applied efficiently to node exchanges in the context of the CVRP.

Swap The swap move m_{ij}^{swap} replaces a node i by a node j and vice versa. Consequently, the four edges $(i-1,i),(i,i+1),(j-1,j),(j,j+1)$ are deleted and the four edges $(i-1,j),(j,i+1),(j-1,i),(i,j+1)$ are added to the current solution x. The swap move is composed of two *dependent* relocation partial moves $m_{i,j-1}^{reloc} \circ m_{j,i-1}^{reloc}$. (The relocation move $m_{k,\ell}^{reloc}$ moves node k from its current position and inserts it into the position between ℓ and $\ell+1$.)

There exist several decompositions of m_{ij}^{swap} into two cost-independent partial moves of the same type. One possibility is to define p_{ij} as a partial move which deletes the edges $(i-1,i),(i,i+1)$ and then adds the edges $(j-1,i),(i,j+1)$. Then, $m_{ij}^{swap} = p_{ij} \circ p_{ji} = p_{ji} \circ p_{ij}$ is a cyclic decomposition into cost-independent partial moves. The partial gain of p_{ij} is $g(p_{ij}, x) = c_{i-1,i} + c_{i,i+1} - c_{j-1,i} - c_{i,j+1}$. When looking for improving swap moves by sequential search, the gain criterion tells us that we can restrict our attention to a first partial move p_{ij} with positive gain. We propose to search for those p_{ij} with positive gain by first considering all nodes $i \in V$. The task is then to restrict the search for possible nodes j under the condition that i is known. This can be done with candidate lists of edges incident to node i ordered increasingly by their cost. Let $\alpha := \frac{c_{i-1,i}+c_{i,i+1}}{2}$, which is a fixed constant when node i is chosen. The condition $g(p_{ij}, x) > 0$ is equivalent to $(c_{j-1,i} - \alpha) + (c_{i,j+1} - \alpha) < 0$ which implies $c_{j-1,i} < \alpha$ or $c_{i,j+1} < \alpha$. As a consequence, only ingoing edges $(j-1,i)$ of cost less than α and

outgoing edges $(i, j+1)$ of cost less than α have to be considered. Finally, when j is chosen, the total gain $g(p_{ij}, x) + g(p_{ji}, x)$ can be checked in constant time.

Other neighborhoods The report [2] shows that node-ejection-chains and cyclic transfers are additional examples of cyclic-independent and cost-independent neighborhoods. The same holds for the string exchange neighborhood. Hence, the gain criterion and sequential search are applicable.

3.2 Indirect Search

Next, we present an indirect search method to scan the exchange neighborhood of size $\mathcal{O}(n^4)$ with quadratic effort $\mathcal{O}(n^2)$. In an *exchange move*, one node is moved from a first tour r^1 to the second tour r^2 and a second node is moved vice-versa. Clearly, each exchange move can be represented as the composition of two (partial) relocation moves, i.e. $m^{exchange}_{i_1,j_1,i_2,j_2} = m^{reloc}_{i_1,j_1} \circ m^{reloc}_{i_2,j_2}$. The size of the exchange neighborhood is $\mathcal{O}(n^4)$, since it is the combination of two relocation moves. For the exchange move it is easy to verify that its partial moves $m^{reloc}_{i_1,j_1}$ and $m^{reloc}_{i_2,j_2}$ are cost- and permutation-independent, if $j_1 \notin \{i_2 - 1, i_2\}$ and $j_2 \notin \{i_1 - 1, i_1\}$ holds.

We consider the two routes r^1 and r^2 of length $\mathcal{O}(n)$ and w.l.o.g. assume that $i_1, j_2 \in r^1$ and $j_1, i_2 \in r^2$ holds. The following property is very helpful for detecting a best exchange move composed of two *independent* relocation moves: Let $m^{exchange}_{i_1^*,j_1^*,i_2^*,j_2^*}$ be an improving exchange move with *maximum gain* $g(m^{exchange}_{i_1^*,j_1^*,i_2^*,j_2^*}, x)$. Then, node j_1^* is among the three best insertion positions j of all relocation moves $m^{reloc}_{i_1^*,j}$ (taking the maximum gain of all possible nodes j). Similarly, node j_2^* is among the three best insertion positions of all relocation moves which move node i_2^* into the first route. Hence (informally), the best move exchanging two given nodes i_1^* and i_2^* can be found among 9 combinations.

Formally, for each node $i \in N$, the three best insertion position w.r.t. the relocation of i are denoted by $j(i), j'(i), j''(i)$. Then, $m^{reloc}_{i,j(i)}, m^{reloc}_{i,j'(i)}, m^{reloc}_{i,j''(i)}$ are three relocation moves with maximum gain. In order to find the best exchange move $m^{exchange}_{i_1^*,j_1^*,i_2^*,j_2^*}$, define the set of pairs

$$P = \{(i, j(i)), (i, j'(j)), (i, j''(i)) : i \in N\}.$$

The construction of P requires $\mathcal{O}(n^2)$ effort, since all relocation moves have to be scanned while its size is $|P| = 3|N| = \mathcal{O}(n)$. The best exchange move which decomposes into two independent relocation moves, defined by the indices $i_1^*, j_1^*, i_2^*, j_2^*$, can now be computed by considering all combinations $(i_1, j_1) \in P$ and $(i_2, j_2) \in P$ for which $m^{exchange}_{i_1,j_1,i_2,j_2}$ is a feasible move w.r.t. the CVRP capacity constraints and cost-independence. The feasible combination $(i_1, j_1), (i_2, j_2)$ with maximum gain $g(m^{reloc}_{i_1,j_1}, x) + g(m^{reloc}_{i_2,j_2}, x)$ yields $(i_1^*, j_1^*, i_2^*, j_2^*) = (i_1, j_1, i_2, j_2)$. The capacity check can be performed in constant time by comparing demand and residual capacity of i_1 and i_2. Cost-independence means $j_1 \neq i_2 - 1, i_2$ and $j_2 \neq i_1 - 1, i_1$ and is trivial to check.

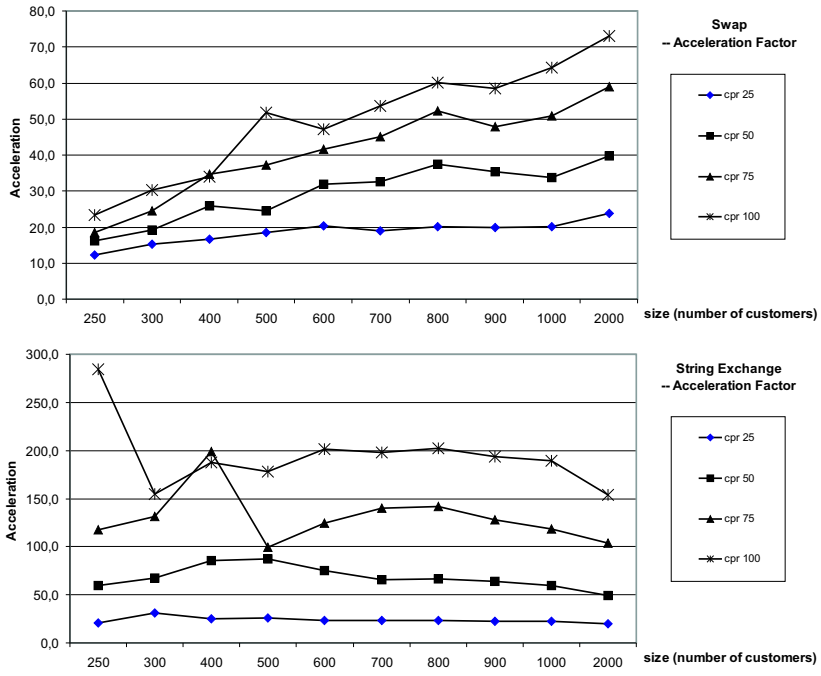
Obviously, the number of combinations to consider is $9n^2$ and, therefore, an $\mathcal{O}(n^2)$-algorithm has been found. With similar techniques it is also possible to scan all remaining *cost-dependent* exchange moves in quadratic time.

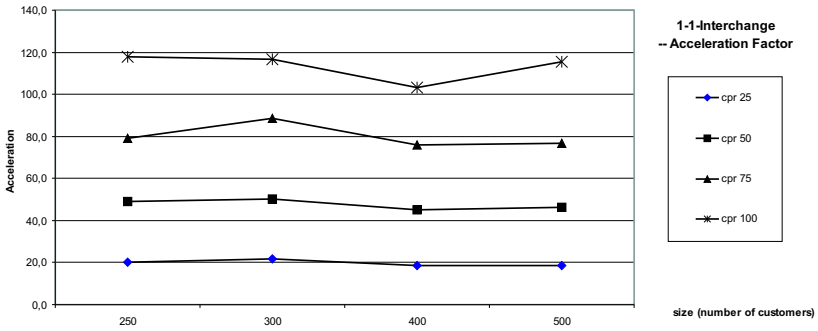
4 Computational Results

In a computational test we compared straightforward implementations and the proposed sequential and indirect search techniques for three neighborhoods: (A) swap as described in Section 3.1, (B) string exchange to which sequential search is applicable, and (C) 1-1-interchange as described in Section 3.2.

Large-scale CVRP instances with sizes from 250 to 2000 customers and demand/capacity ratio (with an average of 25, 50, 75, and 100 customers per route) were randomly generated. For each instance, 18 initial solutions are computed using a parameterized savings heuristic. The three neighborhoods mentioned above and, additionally, the relocate, 2-opt and 2-opt* neighborhoods were used to search for the best improving neighbor. In each iteration, the search is performed with the straightforward implementation as well as our proposed search technique for the same solution (gains are compared to assure that both implementations were coded correctly). LS terminates when a local-optimal solution w.r.t. all of these six neighborhoods is found.

The three figures depicted below give the acceleration factor for swap, string exchange, and 1-1-interchange. For the swap neighborhood, sequential search results in a speedup by factors between 10 and 75, where the factor increases for CVRP instances with increasing size and increasing customers-per-route (cpr) ratio. Sequential search for the string-exchange neighborhood gives speedups by factors between 20 and 200. One observes that the speedup increases with the number of customers, but remains stable for different cpr ratios. For 1-1-interchange, the straightforward $\mathcal{O}(n^3)$ implementation was computationally very costly and, therefore, it was only applied to instances of size ≤ 500 (we observed that indirect search for finding the best 1-1-interchange move has small computation times even for CVRP with more than 5000 customers). Speedups are between 20 and 120, only depending on the cpr ratio.





5 Conclusion

The paper has presented two concepts to accelerate LS for the CVRP. First, sequential search based on Lin & Kernighan's gain criterion is not only applicable to edge-exchange neighborhoods, but also to several classes of node exchange neighborhoods. Second, clever indirect search techniques allow to reduce the search effort by at least one order of magnitude compared to the size of the neighborhood. An example presented is the exchange neighborhood of size $\mathcal{O}(n^4)$ for which the best improving move can be found with quadratic effort.

Computational test have shown that the proposed techniques are extremely useful for tackling large-scale CVRP. We compared sequential search and indirect search with straightforward implementations for swap, string exchange and 1-1-interchange neighborhoods. Speedups by factors between 10 and 200 were observed for CVRP with 250 to 2000 customers. Successful metaheuristics like tabu-search or variable neighborhood search can profit directly from these improvements.

References

[1] J.L. Bentley. Fast algorithms for geometric traveling salesman problems. *Operations Research Society of America*, 4(4):387–411, 1992.

[2] B. Funke, T. Grünert, and S. Irnich. Local search for vehicle routing and scheduling problems: Review and conceptual integration. Technical report, Deutsche Post Lehrstuhl für Optimierung von Distributionsnetzwerken, Aachen University, Templergraben 64, 52056 Aachen, June 2003.

[3] F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. Technical report, US West Chair in Systems Science, University of Colorado, Boulder, School of Business, Campus Box 419, Boulder, CO, 80309, 1992.

[4] D.S. Johnson and L.A. McGeoch. The traveling salesman problem: A case study in local optimization. In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, chapter 8, pages 215–310. Wiley, Chichester, 1997.

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