Noname manuscript No.
(will be inserted by the editor)

Exploiting sets of independent moves in VRP

Tommaso Bianconcini · David Di Lorenzo · Alessandro Lori · Fabio Schoen · Leonardo Taccari

Received: date / Accepted: date

Abstract Most heuristic methods for VRP and its variants are based on the partial exploration of large neighborhoods, typically by means of single, simple moves applied to the current solution. In this paper we define an extended concept of independent moves and show how even a very standard heuristic method can significantly improve when considering the simultaneous application of carefully chosen sets of moves. We show in particular that, when choosing a set such that the total cost variation is equal to the sum of the variations induced by each single move, the quality of solutions obtained is in general very high. When compared with numerical results obtained by some of the best available heuristics on challenging, large scale, problems, our simple algorithm equipped with the application of optimally chosen independent moves displayed very good quality.

Keywords VRP · tabu search · matheuristic · independent moves

Introduction

Most heuristic approaches for capacitated vehicle routing problems (VRPs) are based on a clever exploration of suitably defined neighborhoods of one

T. Bianconcini
tommaso.bianconcini@fleetmatics.com

D. Di Lorenzo
E-mail: david.dilorenzo@fleetmatics.com

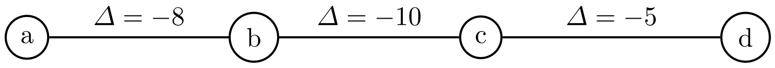
A. Lori
E-mail: alessandro.lori@fleetmatics.com

L. Taccari
E-mail: leonardo.taccari@fleetmatics.com

F. Schoen
DINFO - University of Florence E-mail: fabio.schoen@unifi.it

(or several, in population-based methods) current solution. It is well known that the behavior of most successful heuristics is strongly influenced by the quality of the neighborhood definition and by the capability to perform a good neighborhood exploration. Much research has been devoted to efficient ways to explore very large scale neighborhoods, sometimes with the aim of trying not to waste precious CPU time by re-evaluating the effect of the same move more than once: this has been achieved mainly through the use of some kind of caching mechanism. Although caching is very relevant in practice, as it significantly cuts the total CPU time devoted to neighborhood exploration, in most cases large neighborhoods are explored with the aim of selecting a single "best" move to be applied.

Our approach is based on the idea of being able to select, at each step of an algorithm, a set of moves which can be simultaneously applied to the current solution. The benefits expected by the approach proposed in this paper are at least two: on one hand, in this way we are performing an exploration which profits of advantages similar to those obtained by caching, but without requiring a caching mechanism to be implemented. On the other hand, and in our opinion more relevantly, the simultaneous application of several moves generates a less greedy move acceptance criterion, in which the single most improving move might not be applied, in favor of the application of several, less profitable moves which, together, perturb the solution in a more significant way. In fact, it is easy to observe that the usual strategy of selecting a single most improving move at each iteration of a heuristic method sometimes changes the current solution in such a way that other moves, with comparable impact on the quality of the solution, cannot be anymore applied as a consequence of a change in current routes. The next figure illustrates, through an artificial example, the basic idea.



In this figure the labels over each edge represent the total cost variation in a VRP solution obtained by removing that edge and suitably modifying a different portion of the VRP solution in order to obtain a feasible solution. If a standard local search algorithm is employed, the move to be chosen is to remove edge $\{b, c\}$, as this gives the largest improvement. However, by doing so, in the following stage of the algorithm, edges $\{a, b\}$ and $\{c, d\}$ might no longer be part of the solution. On the contrary, a move which removes simultaneously (if feasible) both edge $\{a, b\}$ and edge $\{c, d\}$ yields a total improvement which is significantly higher. In order to be able to design a method which takes an idea of this kind into a practical method, a careful definition of sets of moves which can be simultaneously applied is required, as well as a set of rules for deducing the overall improvement – in this trivial example we assumed that the improvements due to each single move were additive.

Our idea in this paper is that, through the use of state of the art mathematical programming solvers, the choice of a whole set of moves which globally improve as much as possible the current solution is very beneficial in terms of the quality of the resulting routes. We choose one of the most basic and widely studied models, the Vehicle Routing Problem with capacity constraints, and implement a very standard tabu search algorithm. Our objective is to show that even for the most basic VRP model, studied by a large community with a huge number of existing computational approaches, and using an elementary heuristic method, enlarging the set of moves to be applied at each iteration significantly improves the effectiveness of the method. Indeed, our experimentation on a large test set of medium / large sized instances shows that the idea of simultaneous moves can bring a standard VRP method to the quality of much more refined and complex algorithms.

The main contribution in this paper consists in a definition of the concept of *independent* moves, which is crucial in order to be able to select sets of moves which can be applied simultaneously and which guarantee an overall cost variation which is easily computed. Given the general definition, we formally define a set of standard moves, very common in VRP heuristics, and show the conditions which guarantee that these moves are indeed independent. The selection of the best set of independent moves to be applied to the current solution is then delegated to a Mixed Integer Programming (MIP) algorithm. Extensive computational experiments carried out the 100 instances from the new test set described in [23] confirm that even a simple implementation of a basic local search heuristic method can be significantly improved by means of the simultaneous selection of independent moves, to the point that for most of the instances the resulting method stops within a few percent points from the putative optimum.

The paper is organized as follows. Some relevant papers are briefly discussed in Section 1. Section 2 introduces the notion of independent moves, which forms the basis for the definition of the algorithm in Section 3. Numerical results are analyzed in Section 4, and conclusions are drawn in Section 5. Theoretical properties used in Section 2 are proved in Appendix A.

1 Review of the literature

The literature on VRP is very large and it is out of scope to review it here – for a general reference on the subject the interested reader might consult, e.g., [22]. In this section we would like to mention just those approaches which are more closely related to the one proposed in this paper, namely, heuristic approaches that leverage MIP algorithms as well as those in which exponential-size neighborhoods are efficiently explored by other means.

In recent years several authors have proposed to include MIP models within heuristic algorithms in order to exploit the computational efficiency of modern MIP codes and improve the quality of local search methods. These approaches are usually denoted by the term *matheuristic*. Recent approaches include, e.g., [2,7,15,20,21]. Several of them use MIP as a tool to reconstruct solutions after a *destroy* operation, while others employ techniques like local branching to build and explore MIP-defined neighborhoods. To the best of our knowledge, the most widely used approach based on mixing MIP and heuristics for VRP is the selection of suitable routes by means of the solution of a set-covering or partitioning model. In practice, in any local search based method, a number of routes is generated which might be combined *a posteriori* via a set covering model. This idea, first introduced in [9] and used, among others, in [11,18], is indeed interesting and, while it is not the core of our algorithm, it has also been used in this paper to get some minor improvement at the end of the algorithm, as we will further explain later.

Approaches for the exploration of exponential-size neighborhoods for combinatorial problems that do not employ MIP algorithms have also been widely studied. In such methods, the large-scale neighborhoods are built in such a way as to allow a polynomial algorithm to explore them. An example is the dynasearch algorithm proposed for a machine scheduling problem and the TSP in [4,16], which is based on combinations of mutually independent basic moves, where the optimal sequence of search steps is selected via dynamic programming.

On the other hand, the literature on methods focused, more specifically, on the selection of generic subsets of simple, standard VRP moves is quite scarce. A notable example in this category is [8]. In this paper a definition of independence among a small set of moves is considered in order to be able to select a bunch of improving moves simultaneously, in the same spirit of what we are proposing here. The idea of the authors was to restrict so much the set of moves which can be applied simultaneously, to be able to cast the optimal selection of moves as a constrained shortest path problem in a directed graph. This way, they can use a label setting algorithm as a fast heuristic. This selection tool is quite limited in the set of candidate moves considered, compared to what we propose here. In fact, they define compounded neighborhoods for the VRP only for two of the six operators we consider (namely, swap and insertion); their definition of independence is also more restrictive, as it forbids pairs of moves that we consider as independent (such as intertwined swaps); moreover, they allow for inter–route selection of independent moves by concatenating different routes into a single one, thus imposing an arbitrary order, and by setting an arbitrary orientation of the orders within each route.

Our aim in this paper is to greatly enlarge the set of moves, both feasible as well as infeasible, among which to choose those to be applied. With respect to [8] thus we expand the types of moves considered introducing a broader definition of independence, we allow the simultaneous choice of moves of different types, we allow the selection of moves in any route, and overall we greatly expand the number of moves considered; of course this enlargement in the type and number of moves to be considered requires the use of a MIP solver, instead of a much simpler shortest path algorithm.

Another recent example is [17] where the idea of choosing a set of independent moves to be simultaneously applied is studied with the aim of proposing a highly parallel, GPU–based, implementation. At each iteration a subset of independent moves is extracted by means of a heuristic procedure and then all these moves are applied simultaneously within an Iterated Local Search algorithm. The framework is similar to the one proposed here, but with a very limited set of move types and it is focused on the TSP context, in which capacity constraints are not considered.

2 Independent moves for VRP problems

The problem considered in this paper is the capacitated VRP with symmetric costs and a unique depot. Although here we are restricting our attention to one of the most basic VRP models, many concepts and properties can be extended to more general cases.

Let $G = (V, E)$ be an undirected, complete, graph where V is the set of nodes, i.e., a set of orders to be satisfied, plus a depot which we consider to be unique. E is the complete set of (undirected) edges, with symmetric costs $c_{ij} \geq 0 \; \forall i, j \in V$. Formally, we identify a (VRP) solution s by the set of its selected edges $E_s \subset E$. We assume that a set of R vehicles is given, each with an associated capacity Q. To each node (order) $v \in V$ a positive load (demand) d_v is associated.

Definition 1 A solution is *well-formed* if all orders (except the depot) have degree 2, and if each cycle (or tour) in the solution contains the depot node.

In the above definition, each cycle corresponds to a route. A well-formed solution is *feasible* if all the routes also satisfy the capacity constraints, i.e., the sum of loads of the orders in each route is not greater then Q. For the standard capacitated VRP problem we are facing, we look for a least cost feasible solution with R vehicles, noting that a vehicle can also be assigned an empty route.

In what follows, let us discard, for the moment, vehicle capacity constraints, to focus on the effect of the moves on the structure of the solution. A *move* m is defined as a set of edges $R_m \subseteq E$ that must be removed from the solution, and a set of edges $I_m \subseteq E$ which must be inserted. We also say that the edges in R_m are *affected* by m. We denote by $m(s)$ the solution obtained applying the move m to s, formally defined as:

$$m(s) := (E_s \setminus R_m) \cup I_m \tag{1}$$

Definition 2 A move m is said to be *legal* for a well-formed solution s if

1. $R_m \subseteq E_s$
2. $(I_m \cap (E_s \setminus R_m)) = \emptyset$
3. $m(s)$ is well-formed.

Definition 3 A set M of legal moves over a solution s is called an *independent set* if

1. $(R_{m_i} \cup I_{m_i}) \cap (R_{m_j} \cup I_{m_j}) = \emptyset \quad \forall m_i, m_j \in M$ (no edge overlap)
2. $M(s) = \left(E_s \setminus \left(\bigcup_{m \in M} R_m\right)\right) \cup \left(\bigcup_{m \in M} I_m\right)$ does not contain subtours

where $M(s)$, with a slight abuse of notation, denotes the solution obtained applying all the moves $m \in M$ to s.

It holds that:

Proposition 1 *If M is a set of independent moves on a well-formed solution s then*

1. $M(s)$ is a well-formed solution
2. $cost(M(s)) = cost(s) + \sum_{m \in M} \Delta(m)$ (cost-additivity)

where $\Delta(m)$ is the cost difference induced by move m, or

$$\Delta(m) = \sum_{e \in I_m} c_e - \sum_{e \in R_m} c_e.$$

Proof In order to verify that the solution is well-formed, consider first a node $v \in V$ different from the depot. Since all moves in M are legal, if a move m cuts an edge incident to v, it will also insert a new edge touching v, and if it cuts two edges, it will insert two distinct edges. Due to the no overlap hypothesis, no edge can be removed or inserted twice by different moves, thus the degree two of the node is preserved. By hypothesis, the new solution does not contain subtours, hence the solution is well formed.

Concerning cost-additivity, the cost of a solution is given by the sum of its edge costs:

$$\begin{aligned} cost(M(s))) &= \sum_{e \in \left(E_s \setminus \left(\bigcup_{m \in M} R_m\right)\right) \cup \left(\bigcup_{m \in M} I_m\right)} c_e \\ &= \sum_{e \in E_s} c_e - \sum_{e \in \bigcup_{m \in M} R_m} c_e + \sum_{e \in \bigcup_{m \in M} I_m} c_e \qquad (2) \\ &= \sum_{e \in E_s} c_e - \sum_{m \in M} \sum_{e \in R_m} c_e + \sum_{m \in M} \sum_{e \in \cup I_m} c_e \qquad (3) \\ &= cost(s) + \sum_{m \in M} \Delta(m) \end{aligned}$$

where (2) follows from the fact that m_i and m_j are legal moves for s, and (3) follows from the hypothesis of no edge overlap between the moves in M. $\square$

Proposition 1 tells us that it is possible to combine moves that involve disjoint sets of edges, as long as they do not create subtours. Also note that, in a practical setting, according to our definition of independent moves it does not matter in which order the moves of an independent set are applied.

Although the concept of independent moves is general enough, its relevance in a specific algorithm depends on the moves the algorithm is allowed to perform. In the following section we will describe the algorithm we used to prove the usefulness of applying simultaneous moves, and a set of well known and widely used moves will be defined. In Appendix A we will show under which conditions a set of moves is independent in the sense of Definition 3, so that it can be applied in order to generate a well-formed solution.

3 The proposed algorithm

In this paper we start from a standard method based on neighborhood exploration, tabu search and destroy-and-repair. The basic algorithm consists in applying a set of standard moves to the incumbent solution, within a prescribed set of possible move types; according to the tabu scheme, the best move(s) in the neighborhood are applied to the current solution, even if this might imply a worsening of the objective function. These moves and their inverse ones are labeled as tabu, and their application is forbidden for the next $\lfloor\sqrt{n}\rfloor + 1$ iterations, where $n = |V| - 1$ is the number of orders. When, for a certain number of iterations, no improvement is observed, a set of destroy and repair moves are applied in order to let the algorithm escape a local minimum, as it is quite standard in many recent VRP approaches (see, e.g., [3,6,12,13]). A high level description of the overall approach is reported in the following scheme:

```
Input: An initial solution $xInitial$
$xLocalBest \leftarrow xInitial$;
$xIncumbent \leftarrow xLocalBest$;
$xBest \leftarrow xLocalBest$;
$c \leftarrow \infty$;
while $\neg terminate()$ do
    while $\neg innerTerminate()$ do
        $\mathcal{M} \leftarrow getIndependentSet(xIncumbent)$;
        $xCandidate \leftarrow apply\,(\mathcal{M}, xIncumbent)$;
        if $cost(xCandidate) < c$ then
            $xLocalBest \leftarrow xCandidate$;
            $c \leftarrow cost(xCandidate)$;
            if $c < cost(xBest)$ then
                $xBest \leftarrow xCandidate$;
            end
        end
        $xIncumbent \leftarrow xCandidate$;
    end
    $xIncumbent \leftarrow destroyAndRepair(xLocalBest)$;
end
return $xBest$;
```

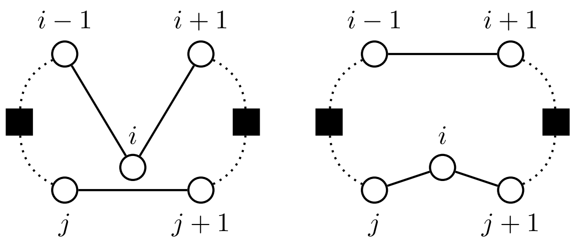
Algorithm 1: Tabu Search with Independent Set (TSIS).

The core of the TSIS (Tabu Search with Independent Sets) algorithm is in the *getIndependentSet* operator. In standard tabu search methods this is replaced by the selection of the single most improving (or least worsening) non-tabu move within a specific neighborhood of the current solution. In our approach this is extended, as we will see, by means of the selection of a set of moves suggested by the solution of a MIP model. The candidate solution is hence obtained by applying to the incumbent solution all the moves contained in $\mathcal{M}$. Another relevant feature of the method is the definition of the *destroyAndRepair* method, which introduces some diversification, as typically done in many heuristic approaches. In the following subsections we will list the elementary moves included in the neighborhood exploration and the elementary destroy and repair operators employed in our experiments.

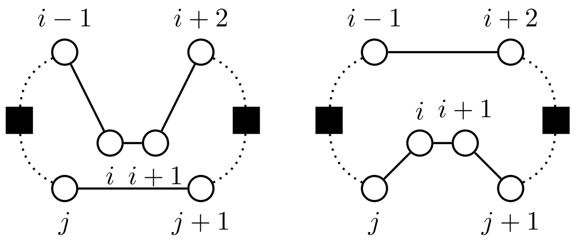
3.1 Neighborhood exploration

Several standard and low–complexity operators are defined in order to build a rich set of moves among which we aim to chose a subset of promising ones. These moves are elementary enough to allow for a complete enumeration of all of their possible application to the current solution.

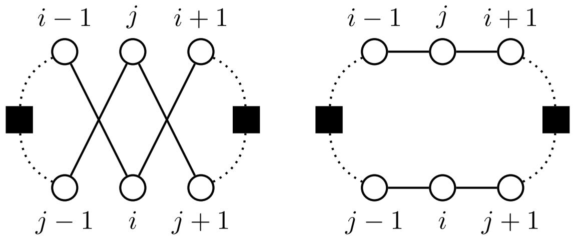
– **Relocate Operator**: an order i is moved from its original route onto a different one in a specific position;



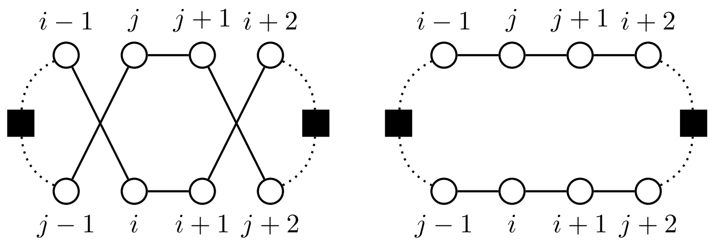
– **Relocate Pair Operator**: a pair of adjacent orders i, $i+1$ is moved from its original route onto a different one. These orders are kept adjacent in the destination route;



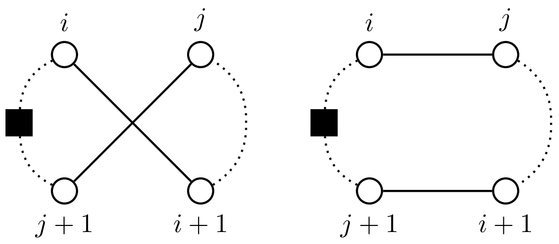
– **Exchange Operator**: two orders i and j (either from the same or from different routes) are swapped;



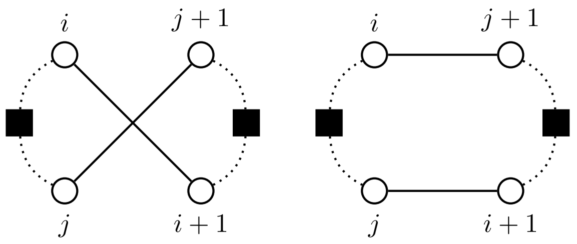
– **Exchange Pair Operator**: two edges, either from different routes or from the same one, are swapped;



– **2-Opt Operator**: 2 edges $\{i, i+1\}$ and $\{j, j+1\}$ from the same route are replaced by the edges $\{i, j\}$ and $\{i+1, j+1\}$;



– **2-Opt* Operator**: given two edges $\{i, i+1\}$ and $\{j, j+1\}$ belonging to two different routes, orders $i+1$ and $j+1$ are swapped.



3.2 Destroy & Repair

After a certain number of non-improving iterations have been observed, the current solution is perturbed by means of one of the following destroy operators (see, e.g., [12]):

– **Random Removal**: p orders are randomly chosen and removed from their respective routes;
– **Cluster Removal**: an order i is randomly chosen. i and the $p-1$ orders closest to i are removed from their respective routes;
– **Pair Removal**: $p/2$ orders are randomly chosen. For each of these orders, the closest order not already selected is also chosen. Then all the selected orders are removed from their respective routes;

– **Smart Removal**: inspired by [19]. An order i is randomly chosen. Then the $p - \ell - 1$ order closest to i and the ℓ orders with the closest position to i in the route are selected. The order i and all the selected orders are removed from their respective routes.

The destroy operator is selected randomly at each cycle. To repair the solution, we apply a best insertion heuristic. However, each order cannot be inserted back in the same route it belonged to before the destroy phase.

3.3 MIP formulation for the selection of the best independent set of moves

Given a set $M = \{m_i\}_{i \in \mathcal{I}}$ of legal and non tabu moves over the incumbent solution, let $\mathcal{C} \subset M \times M$ be the set of conflicting move pairs, i.e., all the pairs that violate independence in Definition 3. Formally, the set $\mathcal{C}$ is defined as $(m_i, m_j) \in M \times M$ such that either:

$$(R_{m_i} \cup I_{m_i}) \cap (R_{m_j} \cup I_{m_j}) \neq \emptyset, \text{ or}$$
$$\big(E_s \setminus (R_{m_i} \cup R_{m_j})\big) \cup (I_{m_i} \cup I_{m_j}) \text{ contains subtours.}$$

In the Appendix we report sufficient conditions to evaluate the set $\mathcal{C}$ for the moves defined in Section 3.1.

The neighborhood that we consider in our approach is defined as all the solutions that can be obtained by applying a subset of independent moves (of the type described in the previous section). It is exponential in size, as it is a superset of the one considered in [8].

As we mentioned in the introduction, considering only subsets of independent moves with some further restrictions may allow for the use of polynomial-time algorithms. In our approach, we aim at exploring the whole neighborhood, thus we resort to a MIP algorithm. An optimal set of independent moves can be extracted from M by solving the following mathematical program:

$$\min \quad \sum_{i \in \mathcal{I}} \Delta(m_i)\delta_i \tag{4}$$
$$\text{s.t.} \quad \delta_i + \delta_j \leq 1 \qquad \forall\, i, j : (m_i, m_j) \in \mathcal{C} \tag{5}$$
$$\mathcal{L}_r + \sum_{i \in \mathcal{I}} \lambda(r, m_i)\delta_i \leq Q \qquad \forall \text{ route } r \tag{6}$$
$$\sum_{i \in \mathcal{I}} \delta_i \geq 1 \tag{7}$$
$$\delta_i \in \{0, 1\} \qquad \forall\, i \in \mathcal{I} \tag{8}$$

where the binary variables δ_i are 1 if an only if the move $m_i \in M$ is in the subset of selected moves, $\mathcal{L}_r$ is the load on route r, $\lambda(r, m_i)$ is the variation of the load on route r after applying the move m_i and Q is the capacity of each vehicle.

This is a standard weighted independent set formulation with additional side constraints. Constraints (5) imply that only one move for each pair of conflicting moves can be selected. Constraints (6) are capacity constraints, ensuring that the total capacity of each route is not exceeded. Note that the routing solution is feasible after all the selected moves have been applied. This means that we are also considering single moves that would be infeasible on their own, but, combined with other ones, allow for the overall feasibility to be recovered. This is also instrumental in avoiding local minima, as we will show later. Constraints (7) ensure that at least one move is applied, even if non-improving.

In exact branch-and-cut approaches for independent set problems [14], it is customary to separate valid inequalities that strengthen the formulation. It is out of the scope of this work to solve Formulation (4)–(8) to optimality – indeed, using the MIP solver as a piece of a heuristic framework, we are satisfied with solutions which are sufficiently good. However, we can exploit the structure of our problem to easily enumerate a number of cliques in the conflict graphs. In particular, it is easy to observe that each edge $e \in E$ corresponds, in the conflict graph, to a clique including all the moves that affect that edge, i.e., such that $e \in R_m \cup I_m$. Then, we can add the inequalities:

$$\sum_{i \in \mathcal{I}:\ e \in R_{m_i} \cup I_{m_i}} \delta_i \leq 1 \qquad \forall e \in E_s \tag{9}$$

and remove the subset of Constraints (5) that are dominated by these stronger cuts. and similarly for any set A of arcs,

In the context of a tabu search algorithm, the MIP model can be integrated by treating Constraint (6) as a soft constraint:

$$\mathcal{L}_r + \sum_{i \in \mathcal{I}} \lambda(r, m_i)\delta_i \leq Q + y_r \tag{10}$$

where $y_r \geq 0$ is the excess variable for route r, that is penalized in the objective function with a penalty term $\mu_r y_r$, using the adaptive weights from the tabu search, as we will show in Section 4.

An issue that may arise in practice is that, when TSIS reaches a local minimum, and no improving move is available in the neighborhood, the MIP will simply select the less worsening move and discard all the others. This is consistent with the standard tabu search behavior. However, in our case this means wasting a lot of computational power, because of the MIP, just to select one single move. Taking this into account, it seems reasonable to replace the bound in Equation (7) with a larger one:

$$\sum_{i \in I} \delta_i \geq \max\{\lfloor\sqrt{R}\rfloor, 2\}$$

where R is the number of vehicles. This strategy does however have some drawbacks. As an example, consider the case of being only 1 move away from the optimal solution. If the MIP is forced to choose at least 2 solutions, then it will probably choose the “optimal” one, and a second one on an unrelated route. Tabu lists will then prevent such second move to be undone in the next iterations. However, in this case we can see that all the routes comprising the optimal solution have been found by TSIS – they just were not found simultaneously. In such case, solving a set–covering problem at the end of the algorithm would allow us to recover the optimal solution.

3.4 Set-covering refinement phase

Following what we mentioned in the previous section, we can add to our method a final refinement phase with the aim to recover and combine good routes that have been generated during the local search phase.

Let $\mathcal{R}$ be the set of all the feasible routes found by TSIS during all the iterations. Consider a partition of $\mathcal{R}$ into P subsets $\mathcal{R}_i,\ i \in \{1 \dots P\}$ such that two routes r_j and r_k belong to the same subset if and only if the set of nodes touched by the edges of r_j and by the edges r_k are the same. We define such set as $\mathcal{N}_i$, excluding the depot. For every subset $\mathcal{R}_i$, we define

$$c_{\mathcal{R}_i} = \min_{r \in \mathcal{R}_i} cost(r)$$

and

$$s_{\mathcal{R}_i} \in \arg\min_{r \in \mathcal{R}_i} cost(r)$$

where $cost(r)$ is the cost of route r. In case there are more than one lowest cost solution within $\mathcal{R}_i$, $s_{\mathcal{R}_i}$ can be any one of them. We can then formulate the following set–covering problem, introducing binary variables $\delta_i,\ i \in \{1 \dots P\}$, which denote whether the route with the lowest cost of $\mathcal{R}_i$ is included in the solution:

$$\min \quad \sum_{i \in \{1 \dots P\}} c_{\mathcal{R}_i} \delta_i \tag{11}$$

$$\text{s.t.} \quad \sum_{i : u \in \mathcal{N}_i} \delta_i \geq 1 \qquad \forall u \in V : u \text{ is not the depot} \tag{12}$$

$$\delta_i \in \{0, 1\}. \tag{13}$$

Consider the VRP solution s obtained by

$$s = \bigcup_{i : \delta_i = 1} s_{\mathcal{R}_i}.$$

It is easy to see that this represents a feasible solution which covers all the nodes in the original VRP problem, and whose cost is either equal to the cost of best solution found by the tabu search, or lower.

Set covering approaches for the selection of good routes among sets generated by heuristic algorithms have been proposed, e.g., by [1], who also propose a fast method to solve the set covering model, by [5] who used a set covering approach within a local search method based on ant-colony, or by [15], just to cite a few recent papers which are all based on a similar idea.

On a practical note, we implemented two simple tricks that lead to a measurable improvement in the MIP solver speed. Since we know in advance how many routes the solution can be composed of, we can greatly reduce the solution space with the constraint:

$$\sum_{i\in\{1\ldots P\}} \delta_i = R,$$

where R is the number of vehicles. Moreover, the best solution found so far by TSIS is used as a starting point (or "warm start") when we invoke the MIP solver.

4 Numerical results

4.1 Implementation details

The initial solution is computed through a greedy best insertion heuristic. The destroy & repair phase starts after 50 consecutive iterations without improvements on the local solution. The parameters p and ℓ have been set up to:

$$\begin{cases} p &= \lfloor\sqrt{n}\rfloor + 1 \\ \ell &= \lfloor 0.25 \cdot p \rfloor \end{cases},$$

where n is the number of orders. As commonly done in the literature, we did not enforce the capacity as hard constraints, but we rather used soft constraints with linear penalties. The penalty weights μ_r, initialized to 100 for every route r, are updated using the following rules:

- if the whole solution is feasible, then divide the penalties for every route by 1.1;
- if the solution is infeasible, then multiply the penalties by 2, but only for infeasible routes. Leave the other penalties unchanged;
- in any case, restrict the penalties to the range [0.1, 1000].

The same penalty weight of the tabu search are used in the MIP for the selection of independent moves, as mentioned in Section 3.3. In order to achieve better performance, we decided to put a limit to the number of moves to be considered within the MIP. Only the 1000 non-tabu moves with the best objective function improvement are selected at every iteration. Moreover, we set a time limit of 60 seconds (although it is never reached in our experiments) and a relative MIP gap stopping criterion of 10%. In Section 4.2, we report an experiment on the sensitivity of the algorithm to the choice of these parameters.

At the end of TSIS, the set covering model described in Section 3.4 can be used to further improve the best solution. In what follows, we refer to the approach that includes this final refinement phase as TSIS-SC.

We ran the numerical experiments on a Desktop PC Intel(R) Xeon(R) CPU E5-2430 v2 @2.50GHz, with Ubuntu 16.04.1. The algorithm was implemented in C++11, while, as a MIP solver, we used Gurobi 6.5 [10]. In order for the comparison with other single-thread tabu search methods to be fair, we restricted it to only use 1 thread.

4.2 Results

In our computation experiments, we set out to assess the improvement that can be obtained through the use of a clever selection of sets of simultaneous moves, as opposed to a greedy choice. We first performed experiments on a smaller dataset from [8], which contains instances that are widely used in the VRP literature, and than we performed more extensive runs on the dataset recently introduced in [23]. Our aim is to show that an elementary method, equipped with our independent set component, can deliver solutions whose quality is very close to those obtained with much more sophisticated approaches.

Experiments with Ergun et al.'s (2006) dataset

Our first experiments have been performed with the dataset of [8]. The aim was to check whether the much expanded set of candidate moves among which we chose is capable of producing an improvement which justifies the extra computational effort due to the choice of using a MIP solver. The instances we used were those with a single capacity constraint – of course, it would be very easy to extend our model to deal also with an additional, similar, constraint, as in some of the test cases in [8]. We run, as in [8], 5 independent tests on the dataset, with different random seeds, and in Table 1 we report some statistics on the quality of the solution found by our method in 2h of CPU time for the first phase, and a set-covering refinement phase of 10 minutes.

The comparison between our results and those published in [8] show a clear advantage of our approach for what concerns the solution quality. In all the instances, TSIS-SC is able to beat the best solution found by the algorithm in [8]. Moreover, in 16 out of 19 instances even the average value over the 5 runs of TSIS-SC is at least as good as that of the best solution obtained by Ergun et al.

We also used the dataset of [8] to validate the choice of some parameters and to check the sensitivity of the results with respect to the choice. We performed a very limited set of experiments changing, one at a time, two parameters: the maximum number of non-tabu moves with the best objective function improvement selected for the MIP formulation (default 1000), and the relative gap used for stopping the solution of the MIP model (default: 10%). Both parameters control the trade-off between the accuracy of the move selection phase, and the time devoted to it. Figures 1(left) and 1(right) report a summary of results showing that from one side our choice of parameters was quite good and, on the other side, that the sensitivity of the algorithm to this

Problem	TSIS-SC best	TSIS-SC avg	Ergun et al. best	% gap TSIS-SC vs Ergun et al.
E50-05	524.61	524.61	524.61	0.00
E75-10	835.26	835.26	835.43	-0.02
E100-08	826.14	827.79	826.14	0.00
E100-10	819.56	819.56	819.56	0.00
E120-07	1042.11	1042.11	1042.11	0.00
E150-12	1028.42	1028.69	1033.01	-0.44
E199-17	1291.71	1296.97	1303.21	-0.88
E240-22	707.80	708.06	709.66	-0.26
E252-27	859.47	861.11	869.26	-1.13
E255-14	584.52	585.36	586.44	-0.33
E300-28	997.02	999.16	1010.70	-1.35
E320-30	1082.63	1085.99	1092.29	-0.88
E323-16	743.10	745.17	745.26	-0.29
E360-33	1372.58	1375.78	1385.84	-0.96
E396-34	1347.50	1350.28	1357.97	-0.77
E399-18	920.01	924.28	922.09	-0.23
E420-41	1826.27	1830.81	1854.54	-1.52
E480-38	1626.03	1635.92	1642.92	-1.03
E483-19	1114.05	1122.34	1121.15	-0.63

Table 1 A numerical comparison between the proposed algorithm and the results in [8]. Columns report: the best result (out of 5 independent runs) obtained by TSIS-SC, the average TSIS-SC cost, the cost reported by Ergun et al. (best out of 5 runs) and the percentage gap between TSIS-SC and Ergun best costs.

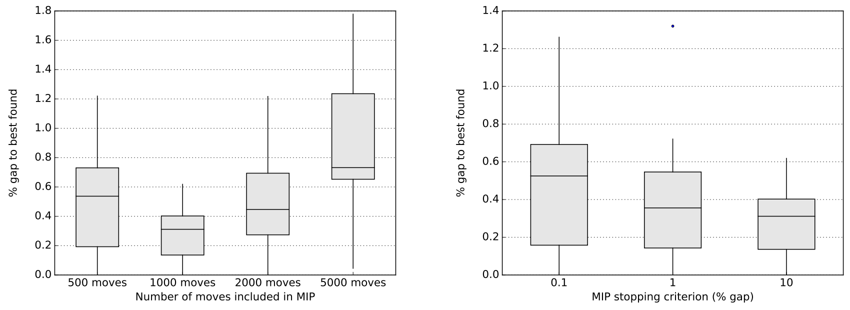


Fig. 1 Sensitivity analysis on Ergun et al.'s dataset: percentage gap between the actual and the overall best observed cost, as a function of the number of candidate moves in each MIP model (left), and the relative gap used as a stopping criterion in the MIP algorithm (right).

choice is not too high.

Concerning the number of candidate moves at each iteration, the number of constraints in the MIP model grows quadratically with their cardinality, so that it becomes challenging to build and solve the problem in a short time if a larger number of them is considered. On the other hand, reducing the search space too much yields a worse overall performance. Figure 1 (right) suggests that reducing the relative gap stopping criterion does not significantly improve the quality of the combined moves: indeed, good solutions are found pretty quickly by the MIP solver, and most of the time would be spent certifying its optimality, with only marginal improvements of the primal bound.

Experiments with the "X" dataset by Uchoa et al. (2017)

We performed our more extensive computational experiments using the dataset recently introduced in [23]. The dataset is comprised of 100 realistic VRP instances, with a number of nodes ranging from 100 to 1000 and a number of vehicles ranging from 10 to 207, generated with a great variety of parameters such as demand distribution, depot positioning, and average route size. Using the same notation used in the dataset, in the following tables a problem with i nodes (including the depot) and j vehicles will be denoted as X-ni-kj. All the gaps in the following results have been computed with respect to the best solution reported in [23].

In order to further validate the correctness of our approach, and to assess the impact of the main algorithmic components, we started by comparing the basic tabu search algorithm (without independent sets of moves) with TSIS. Both algorithms share the same code base, with the algorithmic details described in Section 3, except for the move selection policy. We have put a 2 hours time limit on both algorithms. The results are summarized in Figure 2, with and without the final set-covering refinement phase on the routes generated by the local search. We can see that the TSIS approach gives a substantial improvement with respect to the plain tabu search (TS): using the same amount of computational resources, the solution gap medians and quartiles are roughly reduced by a factor of 3. Additionally, the pure tabu search was unable to find a feasible solution on three problems within the dataset (X-n586-k159, X-n819-k171, and X-n916-k207). Those problems have a number of vehicles which is quite high with respect to the rest of the dataset, and are tightly constrained with respect to capacity. On those problems, the basic tabu search seems unable to solve the underlying bin packing problem defined by the capacity constraints. In our computational experiments, TSIS did not suffer from this limitation.

It is interesting to note that the number of iterations performed by the tabu search in the same amount of time varies from 90 times (on smaller instances) to 5 times (on larger instances) the ones made by TSIS. However, TSIS consistently achieves a much better solution quality, suggesting that it is worth to devote some extra CPU time at every iteration to make sure that a good set of moves is selected.

Focusing on the impact of the different components, it is clear that, although the set-covering is beneficial for both methods, the main improvement is given by the use of the MIP-based move selection component.

Finally, we performed more intensive runs with TSIS-SC only, setting the time limit at 16 hours for every problem, with 2 hours of refinement phase. Results are displayed in Table 2. With "Best TSIS-SC" and "average TSIS-SC" we denote the best and average cost of the solution found by TSIS-SC in

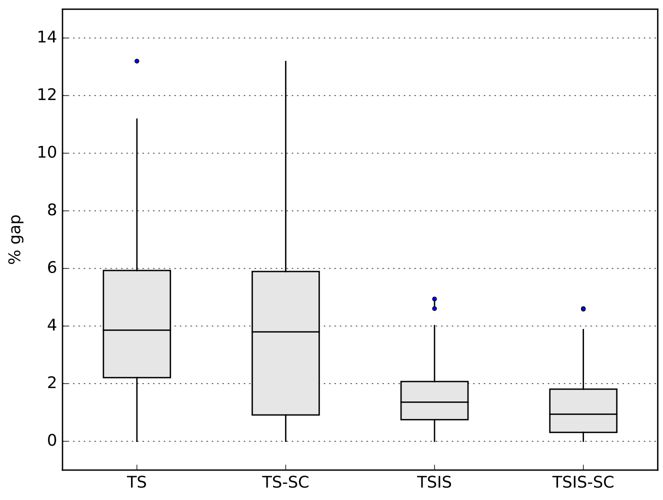


Fig. 2 Boxplot showing the percentage gap to the best solution found by UHGS in [23] after 2 CPU hours of TS and TSIS, with and without set-covering phase (15 minutes). The use of independent sets of moves gives a significant performance boost. The SC phase is especially useful for TSIS due to the reasons mentioned at the end of Section 3.3, but its effect is marginal compared to the difference obtained by the move selection policy.

5 runs; the columns "%gap (best)" report the percentage gap between the best result found by TSIS-SC and the best found in 50 runs of ILS-SP and UHGS respectively. Analogously, the following two columns report the comparison between average results by our method (over 5 runs) and the averages over 50 runs of ILS-SP and of UHGS. We can see that, for problems with up to 200 nodes, TSIS-SC is able to find the optimal solution most of the times, and stays well within the 1% optimality gap when it does not. The optimality gap slightly increases for problems with a high number of nodes, suggesting the need for either more computational resources, or some refinement in the solution strategy.

Name	Best	average	% gap (best)		% gap (average)		Proven
	TSIS-SC	TSIS-SC	ILS-SP	UHGS	ILS-SP	UHGS	Optimal
X-n101-k25	27591	27591.00	0.00	0.00	0.00	0.00	yes
X-n106-k14	26373	26402.80	0.04	-0.02	0.10	0.08	yes
X-n110-k13	14971	14971.00	0.00	0.00	0.00	0.00	yes
X-n115-k10	12747	12747.00	0.00	0.00	0.00	0.00	yes
X-n120-k6	13332	13334.40	0.00	0.00	-0.02	0.02	yes
X-n125-k30	55539	55539.00	0.00	0.00	-0.24	-0.01	yes
X-n129-k18	28940	28956.00	-0.03	0.00	-0.14	0.03	yes
X-n134-k13	10916	10931.20	0.00	0.00	-0.15	-0.03	yes
X-n139-k10	13590	13590.00	0.00	0.00	-0.10	0.00	yes
X-n143-k7	15726	15729.60	0.00	0.17	-0.10	0.19	yes
X-n148-k46	43448	43448.00	0.00	0.00	-0.01	0.00	yes
X-n153-k22	21225	21245.80	-0.54	0.02	-0.72	0.09	yes
X-n157-k13	16876	16877.20	0.00	0.00	0.01	0.01	yes
X-n162-k11	14138	14141.00	0.00	0.00	-0.13	0.00	yes
X-n167-k10	20557	20626.00	-0.02	0.00	0.08	0.31	yes
X-n172-k51	45607	45607.00	0.00	0.00	-0.02	0.00	yes
X-n176-k26	47832	47857.80	-0.64	0.04	-0.81	-0.21	yes
X-n181-k23	25570	25591.20	0.00	0.00	0.08	0.00	yes
X-n186-k15	24152	24173.00	0.03	0.03	-0.05	0.11	yes

Name	Best	average	% gap (best)		% gap (average)		Proven
	TSIS-SC	TSIS-SC	ILS-SP	UHGS	ILS-SP	UHGS	Optimal
X-n190-k8	16992	17011.40	-0.54	0.07	-0.77	0.14	yes
X-n195-k51	44225	44245.80	0.00	0.00	0.03	0.00	yes
X-n200-k36	58660	58694.80	0.06	0.14	0.00	0.12	yes
X-n204-k19	19585	19660.60	0.08	0.10	0.18	0.46	yes
X-n209-k16	30692	30749.80	0.08	0.12	-0.05	0.23	yes
X-n214-k11	11052	11082.40	0.61	1.81	-0.40	1.88	yes
X-n219-k73	117595	117597.20	0.00	0.00	0.00	-0.01	yes
X-n223-k34	40480	40554.20	0.02	0.11	0.05	0.14	yes
X-n228-k23	25804	25847.40	0.24	0.24	0.20	0.26	yes
X-n233-k16	19387	19405.40	0.63	0.82	0.36	0.61	yes
X-n237-k14	27089	27164.40	0.17	0.17	0.32	0.36	yes
X-n242-k48	82820	82965.80	0.06	0.02	0.11	0.02	no
X-n247-k50	37278	37282.60	-0.03	0.01	-0.60	0.00	yes
X-n251-k28	38825	38898.20	0.25	0.33	0.15	0.26	no
X-n256-k16	18889	18910.60	0.05	0.05	0.14	0.16	no
X-n261-k13	26775	26834.20	0.26	0.82	-0.13	0.77	no
X-n266-k58	75478	75666.40	0.00	-0.05	0.14	-0.12	yes
X-n270-k35	35351	35407.20	0.08	0.14	0.12	0.11	no
X-n275-k28	21245	21283.00	0.00	0.00	0.13	0.01	yes
X-n280-k17	33651	33715.40	0.08	0.44	-0.16	0.33	no
X-n284-k15	20509	20562.00	1.05	1.39	0.56	1.36	no
X-n289-k60	95736	95864.40	0.44	0.52	0.43	0.41	no
X-n294-k50	47320	47437.20	0.28	0.32	0.39	0.38	no
X-n298-k31	34241	34301.80	0.01	0.03	-0.16	0.03	yes
X-n303-k21	21889	21946.20	0.35	0.65	0.23	0.44	no
X-n308-k13	26037	26068.60	0.53	0.69	-0.12	0.67	no
X-n313-k71	94308	94588.80	0.12	0.23	0.31	0.34	no
X-n317-k53	78359	78387.00	0.01	0.01	0.04	0.00	yes
X-n322-k28	29968	30031.00	0.30	0.33	0.13	0.25	no
X-n327-k20	27713	27763.60	0.41	0.54	-0.18	0.49	no
X-n331-k15	31128	31174.40	0.07	0.08	-0.20	0.05	no
X-n336-k84	139235	139772.60	0.03	0.02	0.22	0.17	no
X-n344-k43	42265	42395.80	0.28	0.39	0.26	0.44	no
X-n351-k40	26184	26334.00	0.63	0.92	0.70	1.23	no
X-n359-k29	52161	52197.20	0.88	1.27	0.23	0.92	no
X-n367-k17	22821	22968.40	-0.35	0.03	-0.15	0.57	no
X-n376-k94	147713	147729.80	0.00	0.00	0.01	-0.01	yes
X-n384-k52	66568	67015.80	0.68	0.74	0.97	1.13	no
X-n393-k38	38510	38587.00	0.55	0.63	0.34	0.55	no
X-n401-k29	66707	66861.40	0.38	0.70	0.22	0.75	no
X-n411-k19	19829	19931.80	0.19	0.56	-0.12	0.95	no
X-n420-k130	107798	107892.60	0.00	0.00	0.05	-0.03	yes
X-n429-k61	66117	66287.40	0.84	0.94	0.82	0.97	no
X-n439-k37	36452	36571.60	0.16	0.16	0.36	0.33	no
X-n449-k29	56514	56605.20	1.35	2.05	0.71	1.89	no
X-n459-k26	24495	24559.60	1.18	1.30	0.40	1.18	no
X-n469-k138	222139	222474.60	0.10	0.03	0.13	-0.06	no
X-n480-k70	90316	90572.20	0.69	0.87	0.78	0.90	no
X-n491-k59	67801	68031.00	1.25	1.75	1.20	1.69	no
X-n502-k39	69484	69553.80	0.29	0.33	0.30	0.32	no
X-n513-k21	24394	24450.80	0.25	0.80	0.07	0.63	no
X-n524-k153	154611	154639.40	-0.06	-0.11	-0.24	-0.22	no
X-n536-k96	95880	96103.60	0.37	0.80	0.42	0.81	no
X-n548-k50	87331	87498.00	0.72	0.59	0.72	0.57	no
X-n561-k42	43346	43460.00	0.92	1.38	0.76	1.38	no
X-n573-k30	51184	51222.60	0.18	0.80	0.10	0.60	no
X-n586-k159	190993	191497.00	0.20	0.24	0.30	0.35	no
X-n599-k92	110342	110773.60	1.18	1.41	1.27	1.57	no
X-n613-k62	60624	61051.40	0.66	1.42	1.00	1.82	no
X-n627-k43	63286	63370.80	0.80	1.48	0.74	1.35	no
X-n641-k35	65330	65631.20	1.35	2.34	1.59	2.24	no
X-n655-k131	106816	106908.00	0.03	-0.01	0.12	0.01	yes
X-n670-k130	147192	147503.80	0.10	0.33	-0.12	0.19	no
X-n685-k75	69985	70210.40	1.95	2.28	1.77	2.27	no
X-n701-k44	83887	84165.40	1.21	1.94	1.35	2.03	no
X-n716-k35	44611	44871.00	1.34	2.50	1.58	2.82	no
X-n733-k159	137085	137512.40	0.18	0.53	0.34	0.68	no
X-n749-k98	79263	79525.00	1.68	1.99	1.60	2.13	no
X-n766-k71	117682	118134.60	1.94	2.62	2.07	2.59	no
X-n783-k48	74720	74963.80	1.73	2.66	1.68	2.68	no
X-n801-k40	74314	74530.80	0.66	0.99	0.71	1.08	no
X-n819-k171	160751	160885.00	1.00	1.35	0.92	1.25	no
X-n837-k142	197341	197726.40	1.30	1.58	1.38	1.67	no
X-n856-k95	89853	90069.40	0.89	0.82	0.89	0.93	no
X-n876-k59	101319	101524.60	1.14	1.61	1.10	1.64	no
X-n895-k37	55796	56129.20	1.98	3.00	2.13	3.10	no
X-n916-k207	332942	333353.00	0.70	0.94	0.73	0.96	no
X-n936-k151	134897	135788.80	0.98	1.32	0.94	1.70	no
X-n957-k87	86912	87102.60	1.42	1.45	1.36	1.49	no
X-n979-k58	121002	121775.20	0.84	1.52	1.27	1.90	no
X-n1001-k43	74059	74589.40	0.38	1.81	0.82	2.24	no
		Min	-0.64	-0.11	-0.81	-0.22	
		Max	1.98	3.00	2.13	3.10	
		Avg.	0.42	0.65	0.36	0.69	
		Median	0.19	0.33	0.15	0.37	

Name	Best TSIS-SC	average TSIS-SC	% gap (best) ILS-SP	UHGS	% gap (average) ILS-SP	UHGS	Proven Optimal

Table 2: Results of the best out of 5 independent runs of TSIS vs 50 runs of ILS-SP (Subramanian) and 50 runs of UHGS (Vidal) as reported in [23]. The first two columns represent the best and the average result obtained by our method; the following two columns report the gap between our best solution and the best reported for ILS-SP and UHGS. Similarly, the two following columns report the average gaps between our average result and the average results obtained by ILS-SP and UHGS. The last column indicates whether the best result here reported has been proven to be optimal according to [23].

We can observe that TSIS-SC displays a very good performance when compared with state of the art, refined methods; the gap with respect to both ILS-SP and UHGS are consistently very low, despite the fact that our method was executed only 5 times, while both benchmark algorithms were run 50 times. In more than a few cases our method found an improved solution and the median gap between our algorithm and UHGS is well below 0.5%, thus confirming the validity of the approach.

5 Conclusions

Our aim while performing this research effort was to check whether the application of carefully chosen sets of independent moves is beneficial for a standard algorithm for VRP. Our intention was to show that, by suitably defining the concept of independence and by exploiting the power of modern MIP solvers, significant advantages can be expected even for the most studied VRP variant. We have also introduced a formal definition of independence and proven some properties which form the basis for the correctness of the proposed approach.

Through a large set of numerical experiments we have shown that using moves suggested by a MIP model greatly improves the quality of a basic VRP heuristic. This supports the idea that, notwithstanding its computational burden, plugging our MIP-based neighborhood search in any of the several powerful state-of-the-art methods for VRP might yield significant improvements. We have also shown that, thanks to this idea, a simple-minded heuristic algorithm is capable of discovering solutions whose quality is equal, or very close, to that found by the best, and significantly more complex, methods available. This opens the way towards more specialized implementations for different and more complex variants of VRP. In order to do so, we can either establish different (and more restrictive) independence definitions (e.g., for graphs with asymmetric costs, pairs of nested moves can be forbidden due to the reversal of some arcs), or reformulate the move selection problem to account for the peculiarities of the variant at hand. Another direction which might be promising is to substitute the MIP algorithm with a carefully designed heuristic method, capable of producing good sets of independent moves in a substantially smaller computational time.

Acknowledgment

We are grateful to both reviewers and the associate editor for their stimulating comments on the first version of this paper: answering those comments helped us to significantly improve the quality of this paper.

A Appendix

In this section we will provide the conditions required in order to safely combine sets of legal moves. First we show that, under suitable conditions, each of the moves considered in TSIS is legal.

Proposition 2 *Given a well-formed solution s, a `relocate(v,{y,z})` move m that moves the order v into the edge $\{y,z\} \in E_s$, with $v \neq y$, $v \neq z$, is legal.*

Proof By definition of the relocate operator, $R_m = \{\{u,v\},\{v,w\},\{y,z\}\}$ where u and w are the two nodes adjacent to v, and $I_m = \{\{y,v\},\{v,z\},\{u,w\}\}$. It is easy to verify that $R_m \subseteq E_s$, and $I_m \cap (E_s \setminus R_m) = \emptyset$, even in the case where $u = y$ and/or $w = z$, otherwise s could not be a well-formed solution. It is also trivial to verify that the move preserves the degree of the involved nodes.

Finally, suppose by contradiction that the move creates subtours when applied on a well-formed solution s. Let T be the set edges comprising such subtour. Note that, if $\{y,v\} \in T$, then also $\{v,z\} \in T$, otherwise the degree of v would not be preserved. Then three cases can happen

- $T \cap I_m = \emptyset$. Then also $T \subseteq E_s$, so s is not a well-formed solution.
- $\{u,w\} \in T$. Then consider the set $T' = (T \setminus \{u,w\}) \cup \{\{u,v\} \cup \{v,w\}$. Since this operation replaces the arc $\{u,w\}$ with the pair $\{u,v\}$ and $\{v,w\}$, also T' contains a tour. But we can see that by construction $T' \subseteq E_s$, so s is not a well-formed solution.
- $\{y,v\} \in T$ and $\{v,z\} \in T$. Then consider $T' = (T \setminus \{y,v\} \setminus \{v,z\}) \cup \{y,z\}$. Following the same reasoning as the previous case, we can see that also T' contains a tour, and that $T' \subseteq E_s$, so again s is not a well-formed solution.

Proposition 3 *Given a well-formed solution s, an `exchange(v,y)` move that swaps two orders v and y is legal.*

Proof By definition of the exchange operator, $R_m = \{\{u,v\},\{v,w\},\{x,y\},\{y,z\}\}$ where u,w and x,z are the nodes adjacent to v and y, respectively, and $I_m = \{\{u,y\},\{y,w\},\{x,v\},\{v,z\}\}$. It is easy to verify that $R_m \subseteq E_s$, and $I_m \cap (E_s \setminus R_m) = \emptyset$, even in the case where $w = x$ and/or $u = z$, otherwise s could not be a well-formed solution.

The move preserves the degree of the involved nodes, and does not create subtours. The proof is trivial and follows the exact same structure as the one for the relocate operator. □

Proposition 4 *Given a well-formed solution s, a `relocate-pair({v,w},{y,z})` move m that relocates the edge $\{v,w\} \in E_s$ into the edge $\{y,z\} \in E_s$ is legal.*

Proof Let us denote by $u \neq w$ the other node adjacent to v in E_s (i.e., $\{u,v\} \in E_s$) and analogously $x \neq u$ is adjacent to w. Then this move is defined through

$$R_m = \{\{u,v\},\{w,x\},\{y,z\},\{v,w\}\}$$
$$I_m = \{\{u,x\},\{y,v\},\{v,z\},\{v,w\}\}$$

and the proof proceeds similarly to the previous one. □

Notice that in the definition of this move we chose to insert edge $\{v,w\}$ both in R_m and in I_m. This will prove useful in order to to ensure that the edge $\{v,w\}$ remains in E_s when combining this move with other ones, as we will see later.

Proposition 5 *Given a well-formed solution s, an* `exchange-pairs({v,w},{y,z})` *move that swaps two edges $\{v,w\} \in E_s$ and $\{y,z\} \in E_s$ is legal.*

The proof is omitted, as trivial and similar to the previous ones. As before, we assume that both edges $\{v,w\}$ and $\{y,z\}$ appear both in R_m and in I_m in order to ensure that they are not removed by other moves, when we will combine them.

Theorem 1 *A set M of legal moves of the type* `relocate`*,* `exchange`*,* `relocate-pair`*,* `exchange-pairs` *with no edge overlap over a well-formed solution s is independent in the sense of Definition 3.*

Proof We need to show that the solution obtained after applying all the moves in M does not contain subtours. To do so, consider any sequence of moves in M, in any order. The first move m_1 can be applied to s, and $m_1(s)$ is still well-formed, by hypothesis and hence does not contain subtours.

Consider the n-th move in the sequence and consider the $n-1$ moves applied before. Let s_{n-1} denote the solution obtained after the application of these moves, and assume that it does not contain subtours. The n–th move can be applied since the edges affected by m_n are, by hypothesis, non-overlapping with all the other ones, so the edges in R_{m_n} belong to s_{n-1} and I_{m_n} do not. Then by Propositions 2–5 move m_n is legal, s_n is well-formed, so it does not contain subtours. The claim follows by induction. □

A.1 Extension to `2-opt` moves

In the previous subsection we have proved that several classes of moves can be safely combined. More specifically, it is always safe to combine non-overlapping `relocate`, `exchange`, `relocate-pair`, and `exchange-pairs` moves, due to the fact that the only requirements for them to be legal is that the edges in R_m are in the solution they are applied to (and those in I_m are not). More complex moves require some additional pre-conditions in order to avoid sub-tours. For `2-opt` moves, these conditions depend on the relative order of the nodes in the tour.

Let us define a *path* p as an *ordered* sequence of nodes which are pairwise adjacent in the solution s. A path is *simple* if no node appears more than once in it. For any two nodes u, v in the same route, the path p induces a partial ordering $\prec_p$ such that $u \prec_p v$ if u precedes v in the path p. Observe that u and v are not required to be adjacent.

Proposition 6 *Given a well-formed solution s, a* `2-opt({u,v},{y,z})` *move m over s, defined as $R_m = \{\{u,v\},\{y,z\}\} \subset E_s$ and $I_m = \{\{u,y\},\{v,z\}\} \not\subset E_s$, with u,v,y,z belonging to the same tour (route), is legal if and only if there is a simple path p where $u \prec_p v \prec_p y \prec_p z$.*

Proof If a simple path p with $u \prec_p v \prec_p y \prec_p z$ exists, then cutting the edges $R_m = \{\{u,v\},\{y,z\}\}$ in the tour creates two disconnected components: a path P_1 that contains u and z at the two ends, and a path P_2 with v and y at the two ends. The nodes u,v,y,z have degree 1 after the cut. Inserting the edges $I_m = \{\{u,y\},\{v,z\}\}$ fulfills the degree condition, and it reconnects the two components, creating a single tour. Thus the move is legal.

Let us now assume that there is no simple path p over the considered route with $u \prec_p v \prec_p y \prec_p z$. Since u must be adjacent to v, and y to z, it is easy to verify that there must be a path q such that $v \prec_q u \prec_q y \prec_q z$. If such path exists, removing the edges $R_m = \{\{u,v\},\{y,z\}\}$ creates two disconnected components: a path p_1 with endpoints v and z, and a path p_2 with endpoints u and y. Inserting the edges $I_m = \{\{u,y\},\{v,z\}\}$ fulfills the degree condition, but it does not reconnect the components, creating two disconnected subtours. Then the precedence must hold, and the claim follows. □

When combining `2-opt` moves, then, we must pay particular care to maintain the necessary and sufficient condition in Proposition 6. To show how this can be achieved, let us start with the definition of nested `2-opt` moves.

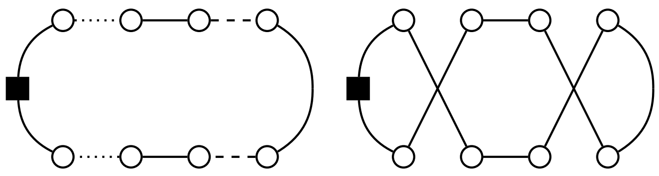


Fig. 3 The move m_i (that removes the dashed edges) is nested in m_j (that removes the dotted edges). Applying both moves on s (left) does not introduce subtours (right).

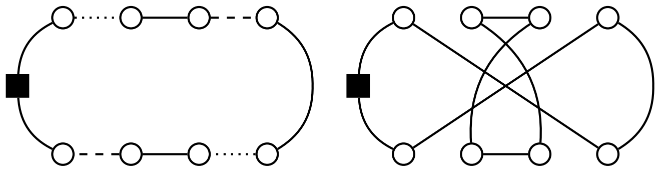


Fig. 4 Applying intertwined moves on a solution s (left) does not guarantee the absence of subtours (right). In this example, two intertwined moves create a disconnected subtour (the four nodes in the middle), although both of them are legal if applied singularly.

Definition 4 Given two `2-opt` moves $m_i = \texttt{2-opt}(\{u_i, v_i\}, \{y_i, z_i\})$ and $m_j = \texttt{2-opt}(\{u_j, v_j\}, \{y_j, z_j\})$ over the same route of a solution s, let $\bar{p}_i$ be the set of nodes in the simple path that connects v_i to y_i not passing through the depot, and let $\bar{p}_j$ be the set of nodes in the path that connects v_j to y_i not passing through the depot. Three cases can occur:

- if $\bar{p}_i \subset \bar{p}_j$, we say that m_i is *nested* into m_j;
- if $\bar{p}_i \cap \bar{p}_j = \emptyset$, we say that m_i and m_j are *disjoint*;
- if $\bar{p}_i \cap \bar{p}_j \neq \emptyset$, but neither is nested in the other, we say that m_i and m_j are *intertwined*.

Two `2-opt` moves are *intertwined* if neither is nested into the other. Applying simultaneously two intertwined `2-opt` moves does not guarantee the absence of subtours, as shown in Figure 4.

Definition 5 Let $\bar{E} \subseteq E_s$ be a set of edges belonging to the same route. Consider any partial ordering on the nodes of the edges in $\bar{E}$ and a simple directed path following that order that connects all the nodes in the route.

We say here that a move *preserves the relative order* of $\bar{E}$ if, after its application to s, the edges in $\bar{E}$ still belong to the same tour, and it is still possible to find a path over such tour so that the chosen partial ordering of the nodes of the edges in $\bar{E}$ is preserved.

Note that no partial ordering can be defined on edges that belong to different tours.

Two preliminary results are needed before we can finally prove how `2-opt` moves can be safely combined with the others.

Lemma 1 *Given a well-formed solution s, a move m of the type `relocate`, `exchange`, `relocate-pair`, `exchange-pairs`, preserves the relative order of any pair of edges $\{u, v\}$, $\{x, y\}$ not affected by the move.*

Proof Consider a legal `relocate` move m that moves the order b with adjacent nodes a, c into the edge $\{f, g\}$. Assume that a simple path p induces the ordering $u \prec_p v \prec_p x \prec_p y$. We can distinguish two cases:

- p does not include the edges in R_m. Then the same path also exists in $m(s)$, so the order is trivially preserved.
- p includes $\{a, b, c\}$. Assume that p is defined as $p = (u, v, \ldots, a, b, c, \ldots, x, y)$. Then consider in $m(s)$ the path $q = (u, v, \ldots, a, c, \ldots, x, y)$, where the edge $\{a, c\}$ has been inserted by applying m. Over q, $u \prec_q v \prec_q x \prec_q y$.

- p includes $\{f, g\}$. Assume that p is defined as $p = (u, v, \dots, f, g, \dots, x, y)$. Then consider in $m(s)$ the path $q = (u, v, \dots, f, b, g, \dots, x, y)$, where the edges $\{f, b\}$ and $\{b, g\}$ have been inserted by applying m. Over q, $u \prec_q v \prec_q x \prec_q y$.

Analogous arguments can be applied to show the claim holds also for the other types of moves. □

Lemma 2 *Given a well-formed solution s, a legal `2-opt` move m preserves the relative order of any two edges $\{u, v\}$, $\{x, y\}$ if they belong to the same of the two connected components in $E_s \setminus R_m$.*

Proof Assume that a simple path p induces the ordering $u \prec_p v \prec_p x \prec_p y$. Taking into account the move m, we can distinguish two cases:

- p does not include the edges in R_m. Then the same path also exists in $m(s)$, so the order is trivially preserved.
- p include both edges in $R_m = \{\{a, b\}, \{c, d\}\}$. Assume w.l.o.g. that p is defined as $p = (u, v, \dots, a, b, \dots, c, d, \dots, x, y)$. Then consider in $m(s)$ the path $q = (u, v, \dots, a, c, \dots, b, d, \dots, x, y)$, where the nodes between a and d are reversed due to the application of the move. Over q, $u \prec_q v \prec_q x \prec_q y$.

The case where p includes only one of the edges in R_m is excluded by the hypothesis that $\{u, v\}$, $\{x, y\}$ lie in the same connected component of $E_s \setminus R_m$. Then the claim follows. □

We can now show that, if we restrict ourselves to nested moves, as in Figure 3, the following result holds:

Theorem 2 *Given a set M of legal moves of the type `relocate`, `exchange`, `relocate-pair`, `exchange-pairs`, `2-opt` with no edge overlap over a well-formed solution s, if all pairs m_i, m_j of `2-opt` moves are non-intertwined, then the set is independent.*

Proof We need to show that the solution obtained after applying all the moves in M does not contain subtours. To do so, consider any sequence of moves in M, in any order. The first move of the sequence m_1 can be applied to s, and $m_1(s)$ is still well-formed, by hypothesis, thus no subtour exists. Moreover, by Lemmas 1–2 the relative order of any two edges affected by any other `2-opt` move m_k with $k > 1$ is preserved (since all `2-opt` are pairwise nested in s), so all pairwise non-intertwined `2-opt` moves are still so.

Let m_n be the n-th move in the sequence, and let s_{n-1} denote the solution obtained after the application of the first $n-1$ moves. By inductive assumption, s_{n-1} does not contain subtours, the relative order of any two edges affected by any `2-opt` move m_k with $k > n - 1$ is preserved, and all pairs of 2-opt moves that were non-intertwined on s, are still so in s_{n-1}. Then:

- If the n–th move is not a `2-opt`, it can be legally applied since the edges affected by m_n are, by hypothesis, non-overlapping with all the other moves in M, so the edges in R_{m_n} still belong to s_{n-1}, and those in I_{m_n} do not. The solution s_n does not contain subtours. By applying Lemma 1, the order of any two edges not affected by m_n is preserved and all pairwise non-intertwined `2-opt` moves are still so.
- If on the contrary the n–th move is a `2-opt`, it can be applied without introducing subtours, since the relative order of the edges in R_{m_n} is preserved in s_{n-1} by the inductive assumption. We must now show that the order is preserved also in s_n. Consider any `2-opt` move m_k with $k > n + 1$. By the inductive assumption, m_n and m_k are still non-intertwined in s_{n-1} so the move m_n preserves the order of the two edges in R_{m_k} by Lemma 2. By applying the same lemma, all non-intertwined moves are still so in s_n.

The claim follows by induction. □

A.2 Extension to `2-opt`* moves

The classes of moves that can be combined can be further extended to include the `2-opt`* operator.

Proposition 7 *Given a well-formed solution s, a `2-opt`* move m over s, defined as $R_m = \{\{u,v\},\{y,z\}\} \subset E_s$ and $I_m = \{\{u,y\},\{v,z\}\} \not\subset E_s$, with $\{u,v\}$ and $\{y,z\}$ belonging to two different tours (routes), is legal.*

Proof Let us consider only the subset of E_s that contains the two tours with the nodes u,v and y,z, respectively. Cutting the edges $R_m = \{\{u,v\},\{y,z\}\}$ creates a tree with the depot as the root node, and four linear branches with the nodes u,v,y,z as leaves. Inserting back in the solution the edges $I_m = \{\{u,y\},\{v,z\}\}$ reestablishes the degree condition and reconnects the dangling branches, thus obtaining two tours, connected by the depot, which contain the nodes u,y and v,z, respectively. The solution is well-formed. □

The additional assumption here is that the edges in R_m must be in two *different* routes. To safely combine them, then, it is sufficient to ensure that no route is affected by more than one `2-opt`*. Concerning the interaction with `2-opt` moves, similar steps to what is shown in the previous subsection can be followed.

Definition 6 Given a `2-opt` move $m_i =$ `2-opt`$(\{u_i,v_i\},\{y_i,z_i\})$ and a `2-opt`* move $m_j =$ `2-opt`$^*(\{u_j,v_j\},\{y_j,z_j\})$ with $\{u_i,v_i\},\{y_i,z_i\},\{u_j,v_j\}$ belonging to the same route of a solution s, and $\{y_j,z_j\})$ belonging to a different one, let $\bar{p}_i$ be the set of nodes in the simple path that connects v_i to y_i not passing through the depot. Two cases can occur:

- if $\{u_j,v_j\} \subset \bar{p}_i$, we say that m_j is *nested* into m_i;
- if $\{u_j,v_j\} \not\subset \bar{p}_i$, we say that m_i and m_j are *disjoint*.

Lemma 3 *Given a well-formed solution s, a legal `2-opt`* move m preserves the relative order of any pair of edges $\{u,v\}$, $\{x,y\}$ in the same tour in s, if there is a tour in $m(s)$ that still contains them both.*

Proof Consider the path $p = (u,v,\ldots,x,y)$ that induces the ordering $u \prec_p v \prec_p x \prec_p y$. Let m be a legal `2-opt`* move. The set R_m consists of two edges, belonging to different routes. If $\{u,v\}$, $\{x,y\}$ still belong to the same tour in $m(s)$, then p does not include the edge in R_m – otherwise they would be in two different routes. This means that p also exists in $m(s)$, and the order is preserved. □

Theorem 3 *Given a set M of non-overlapping legal moves over a well-formed solution s, if:*

- *all pairs $m_i, m_j \in M$ of `2-opt` moves are non-intertwined (either one is nested in the other, or they are disjoint)*
- *for each route in s, there can be at most one `2-opt`* affecting any edge of that route, and it must be disjoint from all the `2-opt` moves over that route*

then the set is independent.

Proof We need to show that the solution obtained after applying all the moves in M does not contain subtours.

An induction proof that follows the same idea used in Theorem 2 can be used. For sake of readability, we will omit the details. To prove the thesis, it is sufficient to guarantee, applying Lemmas 1, 2 and 3, that at each step of the induction:

- the relative order of any two edges affected by any `2-opt` is preserved
- all pairwise non-intertwined `2-opt` moves are still so
- for any `2-opt`* move $m \in M$, the edges R_m are still in different routes
- all pairs of `2-opt`* and `2-opt` moves are still disjoint.

□

References

1. Boschetti, M., Maniezzo, V.: A set covering based matheuristic for a real-world city logistics problem. International Transactions in Operational Research **22**, 169–196 (2015)
2. Bosco, A., Laganà, D., Musmanno, R., Vocaturo, F.: A matheuristic algorithm for the mixed capacitated general routing problem. Networks **64**(4), 262–281 (2014)
3. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, part i: Route construction and local search algorithms. Transportation Science **39**(1), 104–118 (2005)
4. Congram, R.K., Potts, C.N., van de Velde, S.L.: An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. INFORMS Journal on Computing **14**(1), 52–67 (2002)
5. Corman, F., Voß, S., Negenborn, R.R. (eds.): An Ant Colony-Based Matheuristic Approach for Solving a Class of Vehicle Routing Problems. Springer International Publishing, Cham (2015)
6. Dayarian, I., Crainic, T.G., Gendreau, M., Rei, W.: An adaptive large neighborhood search heuristic for a multi-period vehicle routing problem. Tech. Rep. CIRRELT-2014-55, CIRRELT (2014)
7. De Franceschi, R., Fischetti, M., Toth, P.: A new ILP-based refinement heuristic for vehicle routing problems. Mathematical Programming **105**(2-3), 471–499 (2006)
8. Ergun, Ö., Orlin, J.B., Steele-Feldman, A.: Creating very large scale neighborhoods out of smaller ones by compounding moves. Journal of Heuristics **12**(1), 115–140 (2006)
9. Foster, B.A., Ryan, D.M.: An integer programming approach to the vehicle scheduling problem. Journal of the Operational Research Society **27**(2), 367–384 (1976)
10. Gurobi Optimization, I.: Gurobi optimizer reference manual (2016). URL `http://www.gurobi.com`
11. Kelly, J.P., Xu, J.: A set-partitioning-based heuristic for the vehicle routing problem. INFORMS Journal on Computing **11**(2), 161–172 (1999)
12. Koç, Ç., Bektaş, T., Jabali, O., Laporte, G.: A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. Computers & Operations Research **64**(0), 11 – 27 (2015)
13. Mancini, S.: A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. Transportation Research Part C: Emerging Technologies pp. 100–112 (2016)
14. Nemhauser, G.L., Wolsey, L.A.: Integer programming and combinatorial optimization. Wiley & Sons, Inc. (1988)
15. Pillac, V., Guéret, C., Medaglia, A.L.: A parallel matheuristic for the technician routing and scheduling problem. Optimization Letters **7**(7), 1525–1535 (2013)
16. Potts, C.N., van de Velde, S.L.: Dynasearch–Iterative local improvement by dynamic programming. Part I. The traveling salesman problem. Tech. rep., University of Twente (1995)
17. Riise, A., Burke, E.K.: On parallel local search for permutations. Journal of the Operational Research Society **66**(5), 822–831 (2014)
18. Rochat, Y., Taillard, É.D.: Probabilistic diversification and intensification in local search for vehicle routing. Journal of heuristics **1**(1), 147–167 (1995)
19. Rousseau, L.M., Gendreau, M., Pesant, G.: Using constraint-based operators to solve the vehicle routing problem with time windows. Journal of heuristics **8**(1), 43–58 (2002)
20. Schmid, V., Doerner, K.F., Hartl, R.F., Savelsbergh, M.W., Stoecher, W.: A hybrid solution approach for ready-mixed concrete delivery. Transportation Science **43**(1), 70–85 (2009)
21. Subramanian, A., Uchoa, E., Ochi, L.S.: A hybrid algorithm for a class of vehicle routing problems. Computers & Operations Research **40**(10), 2519–2531 (2013)
22. Toth, P., Vigo, D.: Vehicle Routing: Problems, Methods, and Applications, second edition edn. SIAM/MOS, Philadelphia (2014)
23. Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A.: New benchmark instances for the capacitated vehicle routing problem. European Journal of Operational Research **257**(3), 845–858 (2017)