A matheuristic approach for split delivery vehicle routing problems

Aldair Alvarez [a], *Pedro Munari* [b],

[a] *GERAD & HEC Montréal, Montréal, Canada*

[b] *Production Engineering Department, Federal University of São Carlos, Brazil*

aldair.alvarez@hec.ca, munari@dep.ufscar.br

Abstract

We propose a matheuristic approach to solve split delivery variants of the vehicle routing problem (VRP). The proposed method is based on the use of several mathematical programming components within an Iterated Local Search metaheuristic framework. In addition to well-known VRP local search heuristics, we include new types of improvement and perturbation strategies that are tailored for split delivery VRPs. Moreover, we enhance the solution improvement phase by also calling a local search procedure based on a mixed-integer programming model, and resorting to state-of-the-art exact and heuristic libraries, including a tailored branch-and-cut method. To verify the performance of the proposed approach, we run computational experiments using the benchmark instances provided in the 12th DIMACS Implementation Challenge.

Keywords: split delivery, vehicle routing, iterated local search, hybrid method, matheuristic.

1 Introduction

Splitting customer deliveries is a well-known way of reducing travel costs and the number of routes in vehicle routing [7, 5, 11]. With this additional feature, customers can be visited by more than one route if beneficial, which adds more freedom to the design of routes. As reported by Dror and Trudeau [7], who introduced the split delivery vehicle routing problem (SDVRP), significant reductions may be observed in terms of costs and number of vehicles with respect to the traditional (capacitated) vehicle routing problem (VRP) when customer demands are higher than 10% of vehicle capacities. Archetti et al. [5] showed that the savings in travel costs may reach up to 50%, emphasizing the benefits of split deliveries.

The practical advantages of splitting customer deliveries come at a cost from the computational point of view. Indeed, split delivery variants are more challenging to model and solve than the traditional (full delivery) routing problems, as they additionally need to account for multiple visits to customers and consider the decision of how much to deliver at each visit. Regarding formulations and exact methods, the main challenges are how to effectively model multiple visits to customers and how to represent the delivery quantities. In heuristic approaches, the difficulty lies in how to manage the delivery quantities when inserting/removing a customer to/from a route, especially in improvement and perturbations moves. Over the last decades, several researchers have contributed with many clever ideas on how to overcome or mitigate these difficulties in exact and heuristic approaches, as summarized in, e.g., [4, 6, 9, 11]. Yet, the SDVRP remains as a challenging VRP variant, and its inclusion in the 12th DIMACS Implementation Challenge only confirms that.

In this paper, we propose a hybrid solution approach that combines the capabilities of heuristics and mixed-integer programming (MIP) approaches in an effort to effectively solve SDVRP instances. The heuristic component consists of an Iterated Local Search (ILS) metaheuristic [10, 2] that resorts to traditional [7, 8] as well as newly designed improvement and perturbation heuristics. To further enhance the improvement phase of our method, we use a compact MIP formulation to perform local search; and apply the state-of-the-art Hybrid Genetic Search (HGS) library [13] to a modified version of the problem. Moreover, our approach invokes a compact MIP model and a tailored branch-and-cut method [11], which are used to possibly prove optimality or to improve the incumbent solutions.

The remainder of this paper is organized as follows. Section 2 describes the main components of the proposed matheuristic approach. Section 3 presents the results of computational experiments with our matheuristic using benchmark instances. Finally, Section 4 shows the conclusions and future work opportunities.

2 Matheuristic algorithm

Our matheuristic algorithm is based on the incorporation of several mathematical programming components within an ILS metaheuristic framework. An overview of the method is shown in Algorithm 1. In our implementation, we construct an initial set of solutions (line 3) using an adapted version of the classical I1 Solomon heuristic [12], in which we allow split deliveries in the solutions. Then, the method enters its main loop (lines 4-18) until the stopping criteria are met. In this loop, the algorithm applies a series of operations to each solution of the set S as follows. First, a perturbation step (line 6) is applied over the solution. This step aims at changing the split deliveries of the solution to diversify the search by creating solutions with different delivery configurations and also helping the algorithm to escape from local optima. The algorithm follows by applying a local search step (line 7) consisting of a randomized variable neighborhood search heuristic that uses classical vehicle routing movements. The algorithm then updates the solution if a better one is found (lines 8-9) and also the incumbent solution of the search (s^{best}), if improved (line 10). The perturbation and local search heuristics used in the algorithm are described in Subsection 2.1. In addition to them, lines 12 to 19 invoke several exact and heuristic components in an attempt to prevent stagnation. They are detailed in Subsection 2.2

2.1 Improvement and perturbation heuristics

Our matheuristic relies on improvement and perturbation heuristics that are described as follows:

- *Local search heuristic*: the local search step of our algorithm consists of a randomized variable neighborhood descent. This heuristic applies a series of local search operators that are randomly selected from a set, to gradually improve the input solution. The set changes according to the success of the operators. An operator is removed from the set if it fails at improving the solution. On the other hand, if an operator succeeds, then the set is reset to its initial configuration, containing all the operators, and the search continues until none of the operators manages to improve the solution. Our implementation uses classical local search operators within the local search heuristic. In particular, we use 2-opt; Or-opt-k, $k \in \{1,2,3\}$; Shift(k), $k \in \{1,2,3\}$; and Swap(k_1, k_2), $k_1, k_2 \in \{1,2\}, k_1 \geq k_2$, where the first two operate over a single route at a time while the rest change two routes simultaneously. These operators are applied using the first improvement strategy.

- *Perturbation step*: the perturbation procedure follows a structure similar to that of the local search heuristic but applies perturbation operators instead of local search moves. In our implementation, we used four perturbation operators that change the split deliveries of the solution in different ways. One is based on the classical k-split method [8], and the other three are new operators that we introduce for the SDVRP. The first operator of the perturbation procedure deletes all the visits to a random customer and split its demand over the routes of the solution using the k-split method. The second operator duplicates a random route, transferring part of the delivery from the original route to the visits in the new route. The third operator chooses a random non-split customer and transfers a fraction of its demand to a random route. Finally, the fourth operator chooses a random customer, deletes all its visits from the routes, and then inserts single-customer routes for this customer. The number of operations applied in every call of the perturbation procedure is given by $\lceil 0.05n \rceil$, where n is the number of customers in the instance.

Algorithm 1: ILS-based matheuristic approach for SDVRPs

```
1  begin
2      $s^{\text{best}} = \emptyset$, $f(\emptyset) = \infty$ and $S = \emptyset$
3      Generate an initial set of solutions $S$
4      while stopping criteria are not met do
5          forall $s \in S$ do
6              $s^* = \text{Perturbation}(s)$
7              $s^* = \text{LocalSearch}(s^*)$
8              if $f(s^*) < f(s)$ then
9                  $s = s^*$
10                 if $f(s) < f(s^{best})$ then $s^{\text{best}} = s$
11             end
12             if $s^{best}$ has not been improved over the past $\alpha_1$ iterations then $s =$
                 MIP-Improvement($s$)
13         end
14         if $s^{best}$ has not been improved over the past $\alpha_2$ iterations then $s^{\text{best}} = \text{HGS}(s^{\text{best}})$
15         if $s^{best}$ has not been improved over the past $\alpha_3$ iterations then MIP-Residual($S$)
16         if $s^{best}$ has not been improved over the past $\alpha_4$ iterations then $s^{\text{best}} = \text{BC}(s^{\text{best}})$
17         Reset worst $s \in S$ if $s^{\text{best}}$ has not been improved over the past $\alpha_5$ iterations
18     end
19     $s^{\text{best}} = \text{MIP-Improvement}(s^{\text{best}})$
20 end
```

2.2 Hybrid components

Our algorithm incorporates several exact and heuristic components into the ILS framework. The first is a MIP-based improvement heuristic, named as *MIP-Improvement(·)*, which resorts to a MIP formulation that models the removal and insertion of visits to the routes in a solution given as input, ensuring that the total demand of each customer is fully delivered. The idea is to improve the solution by performing two operations: (a) split the customer deliveries by inserting new visits into the routes of the solution; and (b) remove potential unnecessary split deliveries. Similar ideas have been used in a related problem [3, 1].

The MIP model used in this component uses two types of binary variables to model the insertion and removal operations from the routes of the input solution as follows. Let γ_{ik} be a binary variable equal to one if and only if customer i is inserted into the route of vehicle k; and let δ_{ik} be a binary variable equal to one if and only if customer i is removed from the route of vehicle k. There is an insertion cost and a saving associated with γ_{ik} and δ_{ik}, respectively, which are recomputed every time we invoke this routine. The MIP then defines the delivery quantities q_{ik} to each customer i by every vehicle k given the input solution as well as the insertion and removals operations. The routine MIP-Improvement(·) is invoked when the number of iterations without improving the incumbent solution reaches α_1 (line 12), and right after the main loop finishes (line 19).

Another component used in our approach is based on the HGS framework [13], a state-of-the-art metaheuristic implementation for the capacitated VRP. This method is applied to a modified problem in which, given a feasible SDVRP solution, we replicate nodes with split deliveries by creating a new customer for each partial delivery, such that the demand of the new customer is equal to the corresponding delivery quantity. Hence, the resulting problem becomes an instance of the capacitated VRP, and the application of the heuristic aims at improving the routes, keeping the same delivery quantities as in the original solution. This method is called when the number of iterations without improving the incumbent solution reaches α_2, and it replaces the best incumbent if an improved solution is found.

A MIP model is also used in the third component of our approach. The routine *MIP-Residual(·)* in line 15 uses the alternative formulation introduced in [11], to model a residual optimization problem defined as follows. Given the solutions of S, the arcs that are not used in any of these solutions are removed from the network of the residual problem (i.e., their corresponding variables are fixed at zero). The resulting model is then solved by a general-purpose MIP solver, using a relatively short time limit. The solution obtained replaces the solution in S with the largest objective value, even when it is worse. This helps with the diversification of the search procedure, using a different source of feasible solutions. The incumbent solution is updated only in case of improvement. This routine is called when the number of iterations without improving the incumbent solution reaches α_3.

The last MIP-based component used in our matheuristic is the state-of-the-art BC framework for the SDVRP proposed in [11]. This BC is initialized with the incumbent solution s^{best}, and a time limit is imposed to its execution. This routine seeks to prove the optimality of the incumbent, if possible within the time limit; and to find an improved incumbent, if any, with the help of the branching tree and the powerful heuristics currently embedded in general-purpose MIP solvers. The BC framework is called every α_4 iterations without improvement of the incumbent solution.

Finally, to further prevent stagnation, the heuristic generates a new solution with the I1 heuristic to replace the solution in S with the largest objective value when the number of iterations without improving the incumbent solution reaches α_5.

3 Computational results

In this section, we present the results of computational experiments with our matheuristic approach using the 95 SDVRP instances provided in the 12th DIMACS Implementation Challenge and available at `http://dimacs.rutgers.edu/programs/challenge/vrp/vrpsd/`. Distances are taken to be rounded Euclidean distances, computed as

$$trunc(\sqrt{(xi-xj)^2+(yi-yj)^2)}+0.5),$$

where $trunc(.)$ is the function that truncates a number to an integer. We coded our approach in C++, resorting to the following packages: the IBM CPLEX Optimization Studio v.20.1 (to solve MIP models); the HGS solver [13]; and the tailored BC framework described in [11]. The ILS metaheuristic was fully coded by us in C++. The experiments were run on a Linux PC with Intel Core i7-8700 3.2 GHz processor and 32 GB of memory, and using one single thread. We impose the time limit of 1349 seconds for the execution of each instance. For each run of HGS($\cdot$), MIP-Improvement($\cdot$) and MIP-Residual($\cdot$) we use the time limit of 20 seconds, while for the BC, the time limit is 300 seconds. Finally, parameters α_1 to α_5 were empirically set as $\alpha_1 = 10{,}000$; $\alpha_2 = 15{,}000$; $\alpha_3 = 60{,}000$; $\alpha_4 = 249{,}000$; and $\alpha_5 = 99{,}000$.

Tables 1 and 2 present the detailed results of the proposed matheuristic approach on the benchmark instances. For each instance, the columns in these tables give the instance name, number of customers (n) and vehicle capacity (Q); lower bound obtained by the BC component (LB); objective value of the initial solution obtained by the I1 heuristic (Initial); objetive value of the best solution obtained by the matheuristic (Obj); total number of iterations (Iter) and total computing time (T) in seconds corresponding to the execution of the matheuristic; and iteration number (Iter$_{best}$) and exact time (T$_{best}$) at which the best solution was found. The character '–' in column LB means that no lower bound was obtained, and the acronym 'tl' in column T means that the time limit was reached. The results for instance set 3 are given in Table 2, and for the remaining instance sets are given in Table 1.

As the results indicate, the metaheuristic is able to find feasible solutions for all the instances. The construction heuristic of the method already provides good starting solutions, which are further refined by the different components of the matheuristic. The method finds 15 provably optimal solutions within the time limit and provides relatively tight lower bounds for a total of 41 instances. The results also show that the time to find the best solution represents, on average, about half of the total computing time, which indicates that the method is effective at escaping from local optima. It is also possible to observe that, as expected, the total number of iterations that are performed within the time limit tends to decrease with the number of customers of the instances.

Tab. 1: Results of the matheuristic approach on instance sets 1, 2 and 4.

Instance	n	Q	LB	Initial	Obj	Iter	T	Iter$_{best}$	T$_{best}$
SD1	8	100	228.28	228.28	228.28	249000	25.10	0	0.00
SD2	16	100	708.28	708.28	708.28	249000	47.51	0	0.00
SD3	16	100	430.60	430.60	430.60	249000	37.68	0	0.00
SD4	24	100	631.08	631.08	631.08	249000	75.45	0	0.00
SD5	32	100	1390.59	1390.62	1390.59	250760	216.69	1760	0.80
SD6	32	100	831.20	859.92	831.20	392635	202.60	143635	92.42
SD7	40	100	3638.28	3640.00	3640.00	690715	tl	0	0.00
SD8	48	100	5008.28	5068.28	5068.28	498825	tl	0	0.00
SD9	48	100	2025.80	2155.66	2044.24	563475	tl	168507	391.87
SD10	64	100	–	2828.45	2693.95	265065	tl	177924	960.42
SD11	80	100	12712.00	13280.00	13280.00	249130	tl	0	0.00
SD12	80	100	–	7280.08	7219.78	206060	tl	196060	1223.31
SD13	96	100	–	10110.68	10110.50	105120	tl	85120	1091.84
SD14	120	100	–	10919.94	10741.43	70230	tl	53580	967.30
SD15	144	100	–	15151.00	15150.97	38580	tl	10000	318.44
SD16	144	100	–	3383.07	3383.07	41360	tl	0	0.01
SD17	160	100	–	26560.11	26519.25	33220	tl	33220	1328.05
SD18	160	100	–	14419.85	14268.47	33745	tl	33742	1347.05
SD19	192	100	–	20220.39	20124.16	17830	tl	17830	1328.17
SD20	240	100	–	39839.88	39733.26	9695	tl	9695	1328.44
SD21	288	100	–	11295.31	11293.13	4520	tl	0	0.07
S51D1	50	160	458.00	513.00	458.00	249305	160.36	301	0.16
S51D2	50	160	693.46	761.00	703.00	764560	tl	513981	1020.02
S51D3	50	160	924.27	990.00	945.00	670765	tl	16281	12.12
S51D4	50	160	1525.77	1662.00	1553.00	445110	tl	401664	1225.99
S51D5	50	160	1303.85	1426.00	1328.00	610830	tl	111811	183.70
S51D6	50	160	–	2281.00	2166.00	265460	tl	55456	129.67
S76D1	75	160	592.00	718.00	592.00	296965	626.75	47961	87.06
S76D2	75	160	1030.61	1169.00	1082.00	512845	tl	32841	55.97
S76D3	75	160	–	1537.00	1422.00	485385	tl	425383	1133.87
S76D4	75	160	–	2208.00	2078.00	279660	tl	199659	979.46
S101D1	100	160	706.96	843.00	716.00	320540	tl	11682	34.99
S101D2	100	160	–	1478.00	1368.00	317040	tl	98891	424.75
S101D3	100	160	–	2009.00	1869.00	232340	tl	37976	216.97
S101D5	100	160	–	2942.00	2778.00	141665	tl	108007	965.43
eil22	21	6000	375.00	414.00	375.00	249000	49.53	0	0.00
eil23	22	4500	569.00	688.00	569.00	249000	44.72	3	0.00
eil30	29	4500	503.00	576.00	503.00	249000	63.72	2	0.00
eil33	32	8000	835.00	905.00	835.00	249035	74.50	34	0.01
eil51	50	160	521.00	580.00	521.00	249145	199.27	141	0.08
eilA76	75	140	786.24	941.00	818.00	526120	tl	27453	60.20
eilB76	75	100	951.00	1059.00	1002.00	502095	tl	33118	73.47
eilC76	75	180	709.53	882.00	733.00	562970	tl	120971	232.51
eilD76	75	220	662.18	852.00	682.00	500020	tl	1347	1.88
eilA101	100	200	794.94	933.00	814.00	304790	tl	20669	79.14
eilB101	100	112	1011.24	1218.00	1064.00	282440	tl	19689	80.94

4 Conclusions

We proposed a matheuristic approach that combines an ILS metaheuristic algorithm with MIP-based approaches and other heuristic strategies. The ILS relies on traditional as well as novel improvement and perturbation heuristics. The improvement phase is further enhanced by a MIP-based local search, and the use of the HGS library applied to a modified problem. One tailored BC framework was also incorporated in the proposed approach, in an attempt to prove optimality and improve the best incumbent solution. Therefore, the proposed approach seeks to explore the main advantages of the combined heuristic and exact approaches, in an attempt to effectively solve SDVRP instances.

The results of computational experiments using benchmark instances provided in the 12th DIMACS Implementation Challenge indicate that the proposed hybrid approach is effective and resulted in a powerful solution strategy. Feasible solutions were obtained for all instances

Tab. 2: Results of the matheuristic approach on instance set 3.

Instance	n	Q	LB	Initial	Obj	Iter	T	Iter_{best}	T_{best}
p01_00	50	160	521.00	580.00	521.00	249145	197.50	141	0.08
p01_1030	50	160	740.35	813.00	753.00	826840	tl	327333	339.32
p01_1050	50	160	979.34	1089.00	998.00	731335	tl	232089	290.20
p01_1090	50	160	1455.48	1605.00	1481.00	563450	tl	313612	642.66
p01_110	50	160	458.00	518.00	458.00	249050	160.10	47	0.04
p01_3070	50	160	1452.15	1608.00	1474.00	578940	tl	312013	596.27
p01_7090	50	160	–	2277.00	2151.00	328640	tl	138640	562.15
p02_00	75	140	783.46	941.00	818.00	539585	tl	59583	145.86
p02_1030	75	140	1052.09	1169.00	1109.00	463070	tl	300840	959.81
p02_1050	75	140	1441.74	1614.00	1497.00	387200	tl	369354	1249.87
p02_1090	75	140	–	2391.00	2291.00	207125	tl	194229	1255.71
p02_110	75	140	602.03	771.00	612.00	530150	tl	31455	63.96
p02_3070	75	140	–	2343.00	2221.00	266500	tl	246498	1215.81
p02_7090	75	140	–	3372.00	3220.00	176925	tl	176921	1323.51
p03_00	100	200	795.09	933.00	814.00	300315	tl	20304	78.13
p03_1030	100	200	–	1581.00	1453.00	305845	tl	185841	786.74
p03_1050	100	200	–	2151.00	1997.00	229000	tl	136616	837.40
p03_1090	100	200	–	3274.00	3087.00	121841	tl	71841	682.60
p03_110	100	200	736.01	874.00	749.00	301175	tl	17484	70.94
p03_3070	100	200	–	3137.00	2985.00	133015	tl	133013	1315.41
p03_7090	100	200	–	4514.00	4374.00	96045	tl	93232	1282.20
p04_00	150	200	–	1230.00	1012.00	134765	tl	46171	443.78
p04_1030	150	200	–	2161.00	2012.00	96695	tl	33721	428.34
p04_1050	150	200	–	2975.00	2861.00	68040	tl	49541	893.97
p04_1090	150	200	–	4754.00	4555.00	46970	tl	46970	1328.07
p04_110	150	200	–	1106.00	910.00	136145	tl	16145	164.85
p04_3070	150	200	–	4486.00	4363.00	46950	tl	46950	1328.10
p04_7090	150	200	–	6559.00	6453.00	34430	tl	32047	1227.11
p05_00	199	200	–	1481.00	1266.00	63945	tl	55739	1147.10
p05_1030	199	200	–	2610.00	2491.00	42690	tl	24147	721.58
p05_1050	199	200	–	3635.00	3498.00	35705	tl	35705	1328.24
p05_1090	199	200	–	5681.00	5638.00	21265	tl	17258	1053.99
p05_110	199	200	–	1260.00	1059.00	60030	tl	38327	834.72
p05_3070	199	200	–	5591.00	5499.00	21310	tl	21310	1328.12
p05_7090	199	200	–	8314.00	8223.00	14295	tl	14295	1328.14
p10_00	199	200	–	1481.00	1266.00	61270	tl	50354	1085.08
p10_1030	199	200	–	2610.00	2491.00	40755	tl	24147	753.85
p10_1050	199	200	–	3635.00	3498.00	34515	tl	34515	1328.28
p10_1090	199	200	–	5681.00	5638.00	21200	tl	17258	1077.52
p10_110	199	200	–	1260.00	1059.00	63310	tl	38327	790.72
p10_3070	199	200	–	5591.00	5476.00	22340	tl	22340	1328.29
p10_7090	199	200	–	8314.00	8223.00	14940	tl	14940	1328.30
p11_00	120	200	–	1097.00	1026.00	240635	tl	89279	468.34
p11_1030	120	200	–	3044.00	2894.00	151665	tl	142067	1230.56
p11_1050	120	200	–	4343.00	4221.00	93050	tl	64467	815.47
p11_1090	120	200	–	6939.00	6865.00	69987	tl	59987	1135.73
p11_110	120	200	–	1142.00	1030.00	249530	tl	180548	934.18
p11_3070	120	200	–	6751.00	6646.00	62665	tl	62665	1346.85
p11_7090	120	200	–	10341.00	10218.00	43630	tl	43630	1328.08

and, for many of them, the value of the obtained solution is significantly improved with respect to the initial solutions obtained by construction heuristics. Tight lower bounds and proven optimal solutions were also obtained by the approach for a few instances.

References

[1] A. Alvarez, J.-F. Cordeau, R. Jans, P. Munari, and R. Morabito. Formulations, branch-and-cut and a hybrid heuristic algorithm for an inventory routing problem with perishable products. *European Journal of Operational Research*, 283(2):511–529, 2020.

[2] A. Alvarez, P. Munari, and R. Morabito. Iterated local search and simulated annealing algorithms for the inventory routing problem. *International Transactions in Operational Research*, 25:1785–1809, 2018.

[3] C. Archetti, L. Bertazzi, A. Hertz, and M. G. Speranza. A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24(1):101–116, 2012.

[4] C. Archetti and M. G. Speranza. *The Split Delivery Vehicle Routing Problem: A Survey*, pages 103–122. Springer US, Boston, MA, 2008.

[5] C. Archetti, M. G. Speranza, and M. W. Savelsbergh. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science*, 42(1):22–31, 2008.

[6] S. Chen, B. Golden, and E. Wasil. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks: An International Journal*, 49(4):318–329, 2007.

[7] M. Dror and P. Trudeau. Savings by split delivery routing. *Transportation Science*, 23(2):141–145, 1989.

[8] M. Dror and P. Trudeau. Split delivery routing. *Naval Research Logistics (NRL)*, 37(3):383–402, 1990.

[9] S. Irnich, M. Schneider, and D. Vigo. Chapter 9: Four variants of the vehicle routing problem. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 241–271. SIAM, 2014.

[10] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search: Framework and applications. In *Handbook of metaheuristics*, pages 129–168. Springer, 2019.

[11] P. Munari and M. Savelsbergh. Compact formulations for split delivery routing problems. *Transportation Science*, 2022. DOI: 10.1287/trsc.2021.1106.

[12] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[13] T. Vidal. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140:105643, 2022.