An Iterated Local Search Approach to Solve the Milk Collection Problem With Blending

Jorge Villagrán
Departamento de Informática
Universidad Técnica Federico Santa María
Santiago, Chile
jorge.villagran@alumnos.usm.cl

Elizabeth Montero and Germán Paredes-Belmar
Facultad de Ingeniería
Universidad Andres Bello
Viña del Mar, Chile
{elizabeth.montero, german.paredes}@unab.cl

***Abstract*—In this work, we face the vehicle routing problem for the milk collection considering different qualities of milk and blending. This problem can be considered as a multi-product vehicle routing problem with blending. In this version, different products can be mixed, generating an important reduction in traveling costs but a reduced deterioration of milk quality related to milk incoming. To solve this problem, we propose an iterated local search approach. This approach works with unfeasible solutions that are penalized in the evaluation function. Moreover, it uses two movements that allow a strong intensification of the search space during the search process. We test our approach using two sets of problem instances. The first set considers well-known vehicle routing instances in the literature. The second one considers a real case in southern Chile with 500 nodes. From the results, we can demonstrate the abilities of our local search approach to solve small problem instances in reduced times and to find high-quality solutions for real-world problem instances.**

***Index Terms*—Vehicle routing problem with multi-product and blending; milk collection problem; local search; iterated local search**

I. Introduction

Milk transportation from farms to processing plants implies important logistic costs. Milk must be carefully treated due to its perishable nature. Farmers are mostly distributed throughout large rural geographical areas. Moreover, different farms produce different qualities of milk. These different qualities of milk are the raw material for producing different products and, hence, have a different associated income per liter. In most cases, the milk collection of different qualities implies the use of different trucks or trucks with compartments, one for each quality type. Notice that each compartment has a fixed capacity. An alternative procedure is to allow the blending of milk of different qualities. This option reduces the income because the blending reduces the quality of the best milk, but it also reduces the transportation costs of the collection process.

The milk collection corresponds to a vehicle routing problem. Also, if we consider different qualities of milk, it should be considered as a multi-product vehicle routing problem. Moreover, in this work, we study a special case that allows the blending of products.

First and second authors acknowledge the support of FONDECYT project 11150787. Third author acknowledges support of FONDECYT project 11170102.

The main contributions of this paper are: (1) a local search algorithm able to face problem real-world problem instances of the milk collection problem with blending; (2) an evaluation of the main parameters of the proposed iterated local search approach ; and (3) a comparison with the results obtained by several approaches from the literature.

In section II, we present the main variants of the vehicle routing problem related to multi-product, blending, and milk collection problems. In section III, we define the multi-product as a blending version of the vehicle routing problem we face in this work. Section IV describes the local search-based approach we propose in this work. The approach is focused mainly on facing large problem instances that cannot be faced using exhaustive search algorithms. Section V describes the problem instances used to evaluate our approach. We used synthetic problem instances based on well- known instances of the vehicle routing problem and real-world problem instances based on the distribution of 500 milk farms from the south of Chile. Moreover, we present the results obtained and a comparison with the results of an exact method proposed in [15]. Conclusions and future research lines are presented in section VI.

II. Literature Review

The vehicle routing problems comprise a set of combinatorial optimization problems that consider a set of clients that should be visited by a set of vehicles. Those visits can be related to delivery or pick-up of goods (or a combination of them). Trucks have limited capacity. Clients must be visited at most once. The vehicle routing problem corresponds to a generalization of the traveling salesman problem that considers only one vehicle. As the traveling salesman problem, the NP-hard nature of the vehicle routing has already been demonstrated [10].

The classic vehicle routing problem can be extended in several ways, modifying or adding some features. Some changes can be related to features of the truck fleet that can be homogeneous or heterogeneous. Some additional features can be related to time windows for the visits [20]. The time window is usually considered a hard constraint, but some studies indicate that it could be relaxed under complex scenarios [6]. Moreover,

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

some variants consider dynamic requests [16], dynamic transportation times [11], periodic visits [3], simultaneous pickup and delivery [23] and split deliveries [1]. All these studies consider the transportation of only one type of product. The multi-product vehicle routing problem considers the pick-up and delivery of products of different types. A reduction in transportation costs can be obtained when different types of products are transported in the same truck [12]).

Depending on the industry that tries to solve the vehicle routing, new features can be included in the model: Cold chain, time windows, compartments, and blending. In [14], authors propose a vehicle routing problem with multiple-product mixing to reduce costs and risks during the transportation of several hazardous materials. In their work, they propose an integer linear programming model is used to solve a real-world problem instance of 167 nodes divided into 7 zones. For the complex zone (32 nodes), the optimal solution was found using between 300 and 50.000 seconds.

The first approaches to milk collection consider only one type of milk [18], [22]. Moreover, when different qualities of milk were considered, the compartment approach was used [4], [8]. In 2015, Lahrichi [9] used tabu search to solve the milk collection problem. In [5] and [19] authors used genetic algorithms to solve the milk collection with different qualities of milk, but without milk blending. In [15], the authors present the milk collection problem with blending. Moreover, they propose a three-step heuristic to solve real-world problem instances. Moreover, it considers the blending of three different qualities of milk and 100 trucks for collection. The approach starts dividing the farms into clusters using clustering algorithms, and then milk quotas and trucks are assigned to each cluster. Finally, each cluster is solved using a branch-and-cut approach. Best results were obtained when dividing the farms into 13 geographic zones. It required around 16 hours to find that solution. In section V, we compare our results with those obtained in [15].

III. Problem definition

The milk collection problem with blending involves determining routes for a set of trucks. Each route starts and ends in the processing plant. During a route, each truck visits a set of farms that produce milk of different quality. The milk income obtained by each truck is proportional to the final quality of the blended milk collected. When two types of milk are mixed, the final quality of the total milk corresponds to the worst milk type in the blend of the truck. The blending reduces the quality of milk but is allowed because it reduces in greater proportion the transportation costs. Each truck has a limited capacity. In each visit, the truck collects all the milk produced by the farm.

A quota of each type of milk must be satisfied with the collection process. It is possible to mix different qualities of milk both in the trucks and in the processing plant. Independent of milk quotas, all milk should be collected because the company works as a cooperative business.

The objective of the problem is to find the set of routes for the trucks that satisfy the quotas of each type of milk and represents the biggest difference between the total income related to the milk collected and the transportation costs.

Fig. 1 shows an example of the milk collection problem with blending.

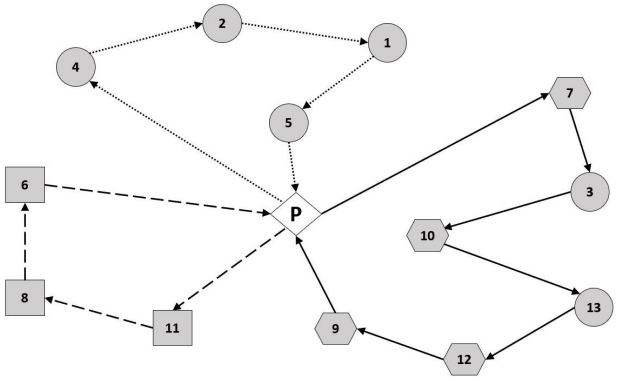

Fig. 1. Small instance with 13 farms, three trucks, three types of milk, and one processing plant.

The instance considers 13 farms that produce three different qualities of milk. Circles represent farms that produce milk of the best quality (A). Hexagons represent farms that produce milk of good quality (B). Squares represent farms that produce milk of regular quality (C). The diamond in the center of the figure represents the processing plant. Truck routes are displayed using arrows between nodes. Three trucks are used in this problem instance; hence, three routes are displayed.

The first truck (dotted arrows) starts its path in the processing plant, visits farms 4, 2, 1 and 5. At the end of its route, it returns to the processing plant. This truck collects only milk A. The second truck (solid arrows) starts its path in the processing plant, visits farms 7, 3, 10, 13, 12 and 9, and then returns to the processing plant. Farms 7, 10, 12 and 9 produce milk B and farms 3 and 13 produce milk A. The milk produced by farms 3 and 13 is mixed in the truck with milk B and, hence, degraded to this type of milk. Farms 3 and 13 are geographically distant from the other farms that produce milk A. Hence they are visited by the truck that visits nearby farms, generating a deterioration of their milk quality but a more important reduction of transportation costs. The last truck (dashed arrows) starts its path in the processing plant, visits farms 11, 8 and 6, and then returns to the plant. Only milk C is collected by this truck.

The mathematical model of the milk collection problem with blending studied in this work can be found in [15].

IV. Iterated Local Search approach

In this work, we propose an *iterated local search* approach to find high-quality solutions in reduced times to real-world problem instances. The search process starts generating random initial solutions. These initial solutions are allowed to be infeasible regarding the capacity of trucks and the quotas of each type of milk required by the processing plant. Then,

a local search process is performed until a local optimum is found. A new initial solution is then created, applying perturbations to the current best solution. Next, we explain the main components of our iterated local search algorithm. First, we explain the representation, the evaluation function, and the initialization process. Then, we present the structure of the iterated local search approach, and we explain in detail its main processes.

A. Representation

Each solution considers a set of routes, one route for each truck. Each route contains a sequence of farms to visit. Fig. 2 shows the representation of the solution of the problem instance shown in Fig. 1. Each farm is represented by its identification number, as shown in Fig. 1.

Truck 1	4	2	1	5		
Truck 2	7	3	10	13	12	9
Truck 3	11	8	6			

Fig. 2. Representation of solution presented for problem instance in Fig. 1.

In order to reduce the stored information, the plant is not included in the representation because it is always at the beginning and end of each route.

B. Evaluation function

The quality of solutions is measured according to the objective function presented in [15] and shown in equation (1). It considers the difference between the income of the total collected milk and the route costs. The income depends on the resulting quality of each liter of milk after truck and plant blending.

$$Max \sum_{t \in T} \sum_{r \in T} \alpha^r v^{tr} - \sum_{(i,j,k) \in AK} c_{ij}^k x_{ij}^k \qquad (1)$$

Here c_{ij}^k represents the travel cost between nodes i and j using a given truck k. α^t represents the income by liter of milk of quality t. Moreover, x_{ij}^k is a binary variable that represents that truck k travels from node i to node j and v^{tr} represents the volume of milk of quality t delivered to the plant that used it as quality r.

Moreover, our evaluation function introduces two penalties. The first penalty (P_q) ponders each unsatisfied liter of milk for the plant quotas (l_q). The second penalty (P_c) ponders each liter of milk that exceeds the capacities of trucks (l_c). These penalties are incorporated in the evaluation function to allow the algorithm to search infeasible areas of the search space. Equation (2) shows the evaluation function used in our approach.

$$f(s) = \sum_{t \in T} \sum_{r \in T} \alpha^r v^{tr} - \sum_{(i,j,k) \in AK} c_{ij}^k x_{ij}^k - P_q \cdot l_q - P_c \cdot l_c \quad (2)$$

In order to ponder the infeasibility with respect to plant quotas, blending is first checked. To fulfill the quotas of milk of lower quality, leftovers of milk of higher quality are used. When the quota of the best quality of milk is not satisfied, no possible blend is checked.

C. Generation of initial solutions

The generation of initial solutions is focused on assigning farms to trucks randomly. Thus, farms that produce the same quality of milk are grouped. For this purpose, we first order the trucks randomly in each solution. Then, the procedure tests the allocation of each farm to each truck. There is a $slack$ parameter that controls the tolerance of the truck's capacity to the allocation of farms. It is initialized in zero, and it is increased when farms do not fit in any truck. The $slack$ parameter allows the initialization procedure to generate high-quality solutions in terms of quotas compliance but reducing its feasibility in terms of truck capacities.

Algorithm 1 – *Generation of initial solution*

```
1:  procedure RANDOM_SOLUTION()
2:      solution ← empty
3:      trucks ← randomly sorted trucks
4:      farms ← farms sorted by type
5:      slack ← 0
6:      while farms > 0 do
7:          fr ← select_random (farms)
8:          allocated ← 0
9:          for tr in trucks do
10:             if fits(fr, tr, slack) and same(fr, tr) then
11:                 add in tr a visit to fr;
12:                 remove fr from farms;
13:                 allocated + +;
14:                 break;
15:             end if
16:         end for
17:         if allocated == 0 then
18:             slack + = 10
19:         end if
20:     end while
21:     return solution
22: end procedure
```

D. Iterated local search

Iterated local search [13] is a heuristic-based algorithm that, starting from an initial random solution, searches on neighborhood structures to find better quality solutions. This process is repeated until no neighbor has better quality, i.e., a local optimum is found. At this point, an iterated local search algorithm performs a perturbation to the current solution and repeats the process. The general structure of our approach is shown in algorithm 2.

The algorithm considers two stopping criteria: the maximum quality and the maximum execution time. Moreover, it requires the setting of the penalties of the evaluation function (P_q and P_c according to equation (2)) and the perturbation factor (pf) that we will describe in section IV-D3.

The procedure starts defining the variables $global\_local$ to detect stagnation and $extra\_local$ and $intra\_local$ to store the best solutions obtained from each local search process. A random feasible solution is generated in line 2 according

to the procedure previously explained in section IV-C. The best solution to the search process is initialized as the first initial solution. An iterative process is performed between lines 4 and 2. In this cycle, two local search procedures are performed. In line 7 an extra routes swap procedure and in line 8 and intra swap routes procedure. From each procedure, the best solution is returned. The better procedure replaces the current solution in line 14. Once this cycle is finished, the quality of the local optimum found is compared with the best solution found so far. Finally, in line 19, a perturbation to the current solution is performed. This perturbation generates a new solution performing random changes to the current one. The perturbation level is controlled by the parameter perturbation factor (pf).

Algorithm 2 – *Iterated local search*

```
1: procedure ILS(max_Q, max_T, P_q, P_c, pf)
2:     solution ← random_feasible_solution();
3:     best_solution ← solution;
4:     while f(solution) < max_Q and t < max_T do
5:         global_local ← false;
6:         while ! global_local do
7:             extra_local ← extra_routes_ls(solution);
8:             intra_local ← intra_routes_ls(solution);
9:             best ← best(extra_local, intra_local);
10:            if f(best) > f(solution) then
11:                solution ← best;
12:            else
13:                global_local ← true;
14:            end if
15:        end while
16:        if f(solution) > f(best_solution) then
17:            best_solution ← solution;
18:        end if
19:        solution ← perturbation(best_solution, pf);
20:    end while
21:    return best_solution
22: end procedure
```

When one of the termination criteria is reached, the algorithm returns the best solution found during the whole search process (line 21).

1) Extra routes local search: This local search procedure aims to generate new routes by swapping farms between routes. At each step, a random farm is selected, and its visit is changed to all possible positions in other trucks. The set of all possible re-allocations to other routes is considered the neighborhood of this movement. This process has $O(n^2)$ in the worst case due to the use of a first improvement strategy.

2) Intra routes local search: This local search procedure is focused on improving the quality of routes by changing the order of their visits. The set of all possible orders using the *2-opt* movement is considered as the neighborhood of this movement. This process has $O(n^2)$ in the worst case. However, it implements a first improvement strategy and a delta evaluation approach that compute only the changes in the current solution to compute the improvement/deterioration of neighbor solutions.

Algorithm 3 shows the pseudocode of the extra and intra-routes local search procedures. In each case, the current solution is received as a parameter. While not stagnation is detected, the procedure generates and evaluates the neighborhood of the current solution. If a better quality solution is found the current solution is replaced, and the corresponding search process is stopped.

Algorithm 3 – *Extra/Intra local search*

```
1: procedure LS(solution)
2:     local ← false;
3:     while ! local do
4:         local ← true;
5:         N ← neighborhood(solution);
6:         for neighbor in N in random order do
7:             if f(neighbour) > f(solution) then
8:                 solution ← neighbour;
9:                 local ← false;
10:                break;
11:            end if
12:        end for
13:    end while
14: end procedure
```

3) Perturbation: The perturbation process of iterated local search aims to allow the search process to escape from local optima. These perturbations are usually performed as random modifications to the current solution. In our algorithm, perturbations are performed by selecting a random farm and changing it to any position on any route. Thus, this modification can produce external or internal changes in routes. Moreover, it can generate infeasible solutions. Parameter pf controls the number of perturbations and was implemented as a percentage of the number of farms of the instance. Algorithm 4 shows the pseudocode of the perturbation process of our approach.

Algorithm 4 *Perturbation*

```
1: procedure PERTURBATION(solution, pf)
2:     perturbations ← farms_lenght * fp
3:     for pf do
4:         r1 ← random route from solution
5:         r2 ← random route from solution
6:         node ← random node from r1
7:         pos ← random position from r2
8:         solution ← move(solution, r1, r2, node, pos)
9:     end for
10:    return solution
11: end procedure
```

V. Experiments and Results

In this section, we present the experiments and results obtained from the evaluation process of our algorithm. The main objectives of our experiments are:

- Evaluate the relevance of the main parameter of our algorithm: perturbation factor.
- Compare the results obtained with the results published in the literature.
- Analyze the convergence of or approach when solving real-world problem instances.

A. Test instances

For the experiments, we consider two test sets. The first one considers 37 well-known instances from the vehicle routing problem adapted to the milk collection problem. This set of problem instances is called *small instances set*. All these instances consider three trucks. The second set was generated from the data of 500 farms located in the south of Chile. This set is called *real-world instances set*. All problem instances consider three qualities of milk.

1) Small instances set: This test set is composed of 37 problem instances that were used by Paredes et al. in [15]. Augerat instances from *a33* to *a80* obtained from [2]. Taillard instances from *tai75A* to *tai75D* and instances *c50* and *c75*, obtained from [21]. Fisher instances *f45*, *f71* and *f72* obtained from [7]. Reinelt instances *eil22* to *eil76* and *att48*, obtained from the TSPLib [17]. Details of these problem instances can be found in [15]. According to [15] most of Augerat's instances can be solved using one truck for each quality of milk. In a few cases, it is necessary to blend milk of different qualities. Taillard and Fisher problem instances show higher quotas than productions of low qualities milk. Hence, the blending is mandatory in these cases. Reinelt's instances show low quotas compared to productions and capacities.

2) Real-world instances set: This set was built with the information of 500 farms distributed along $9.600\ km^2$ in southern Chile. Fig. 3 shows the distribution of farms and the processing plant. Circles, triangles, and squares differentiate farms according to the quality of milk they produce. Daily

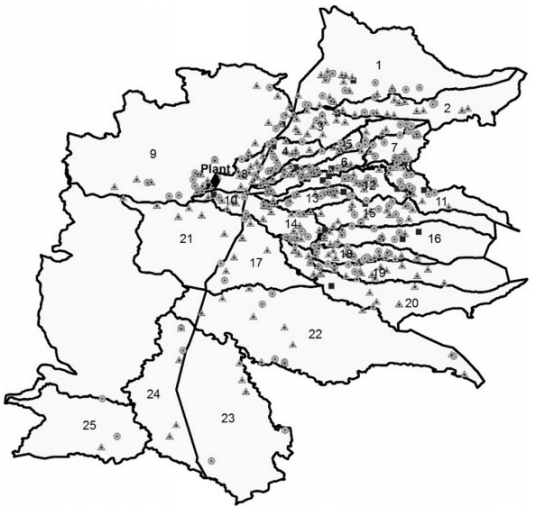

Fig. 3. Distribution of 500 farms in the south of Chile.

production of farms varies from 57 to 25.000 liters as shown in the histogram in Fig. 4. From 500 farms, 313 produce milk A (1.430.715 liters), 159 produce milk B (268.564 liters), and 28 produce milk C (100.000 liters of milk of quality C). The income of milk A is $0,015$ per liter, $0,0105$ per liter for milk type B, and $0,0045$ per liter of milk of quality C.

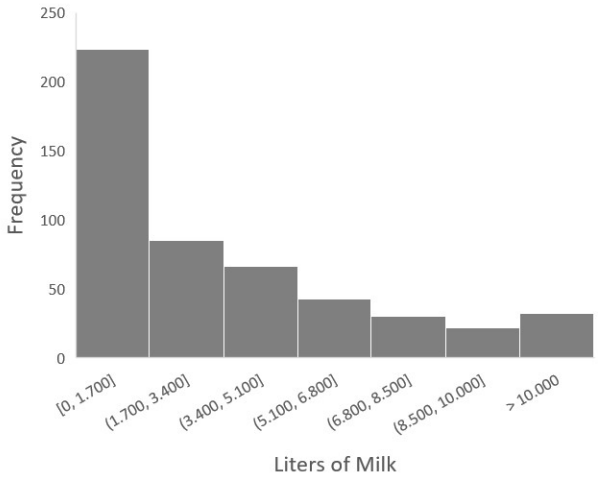

Fig. 4. Histogram of milk production in liters.

Table I summarizes the features of problem instances generated from the data of the 500 farms. For each problem instance, we show its name, number of nodes, number of trucks, and aggregated capacity of trucks. In [15] authors considered 100 trucks for this problem instance. This case corresponds to *r500-100* in table I. Based on the results obtained there, we also studied three new versions of the instances varying the number of trucks (72 trucks, 77, and 88 trucks). To generate these problem instances, we discarded the trucks of lower capacity.

TABLE I
REAL-WORLD PROBLEM INSTANCES FROM PAREDES-BELMAR [15].

Instance	Size	Capacities	Number of trucks
r500-100	500	2.500.000	100
r500-88	500	2.320.000	88
r500-77	500	2.115.000	77
r500-72	500	2.015.000	72

B. Experimental setup

Since the stochastic nature of the proposed algorithm, all the experiments were executed 30 times using different random seeds. Tests were performed in an Intel Core i5-4690K 3.50GHz, with 16GB of RAM at 2.400GHz.

In all our experiments we fixed the termination criteria based on the quality of the best solution and the time required by Paredes et al. in [15]. Even though the times are not comparable, these can limit the execution effort of our approach. Moreover, parameters P_c and P_q were both set to 10. Using these values, the unfeasible solutions that are generated in some cases quickly become feasible when submitted to the local search process.

C. Components evaluation

Here, we analyze the relevance of the perturbation factor parameter on the performance of our iterated local search approach.

Tables II, III and IV show the performance of the iterated local search algorithm on the small instances set, considering four different values for pf: $0,05$; $0,10$; $0,15$ and $0,20$. In each table, we show the name of the problem instance, the best objective function and time (in seconds) obtained in [15]. The time is displayed only for reference because these experiments were performed in a different experimental environment. Moreover, for each value of the perturbation parameter, we show the gap for the objective function and the time (in seconds). A "-" in the gap column, indicates that the best-known solution was reached in these cases. A "-" in the time column, indicates that the time required in these cases to reach the optimum was much lower that one second.

TABLE II
RELEVANCE OF PERTURBATION FACTOR ON AUGERAT INSTANCES.

Instance	[15]		$pf = 0,05$		$pf = 0,10$		$pf = 0,15$		$pf = 0,20$	
	Objective	time	gap	time	gap	time	gap	time	gap	time
a33	29.417	62	-	-	-	-	-	-	-	-
a34	30.496	40	-	-	-	-	-	-	-	-
a36	29.233	110	-	-	-	-	-	-	-	-
a37	24.837	45	-	-	-	-	-	-	-	-
a38	28.596	570	-	1	-	1	-	1	-	15
a39	30.808	110	-	-	-	-	-	-	-	-
a44	38.771	101	-	-	-	-	-	-	-	2
a45	40.282	136	-	-	-	-	-	-	-	-
a46	40.696	66	-	-	-	-	-	-	-	-
a48	39.800	230	-	-	-	-	-	-	-	-
a53	46.662	183	-	-	-	-	-	-	-	-
a54	22.414	304	-	-	-	-	-	-	-	-
a55	24.694	270	-	-	-	-	-	-	-	3
a60	25.041	3,5	1,8	3,5	1,8	3,5	1,8	3,5	1,8	3,5
a61	60.644	561	-	-	-	-	-	-	-	-
a62	22.917	1.022	-	-	-	-	-	-	-	1
a63	24.447	2.930	-	-	-	1	-	2	-	1
a64	24.100	5.395	-	-	-	13	-	69	-	141
a65	28.046	478	-	-	-	-	-	1	-	3
a69	25.822	1.552	-	-	-	1	-	1	-	15
a80	29.977	5.626	-	-	-	-	-	2	-	21

As can be observed, the only problem instance that cannot be solved in Augerat set was *a60*. No matter, the value of pf used, the approach was unable to reach the optimum value. For the other problem instances, it can be observed that increasing the value of pf only increases the time required to solve it.

In the remaining problem instances, the same behavior can be observed. There is a set of problem instances where the increase of the perturbation factor only generates an increase in solving time. Moreover, there is a set of problem instances that the iterated local search cannot solve consistently. Table V summarizes the gaps and execution times of these three sets. Total gaps are equal in all cases showing that a change in pf parameter does not affect iterated local search performance. Moreover, no statistical differences were detected between different values of pf tested. Concerning the execution times, the best parameter value was $pf = 0,05$ which generates enough changes on solutions that allow the process to escape from local optimum efficiently.

TABLE III
RELEVANCE OF PERTURBATION FACTOR ON TAILLARD AND FISHER INSTANCES.

Instance	[15]		$pf = 0,05$		$pf = 0,10$		$pf = 0,15$		$pf = 0,20$	
	Objective	time	gap	time	gap	time	gap	time	gap	time
c50	84	66,2	1,1	84	1,1	84	1,1	84	1,1	84
c75	13.760	46,2	8,1	13,8	8,1	13,7	8,1	13,7	8,1	13,7
tai75A	4.806	43,0	5,8	4,8	5,8	4,8	5,8	4,8	5,8	4,8
tai75B	15.056	48,2	-	4	-	21	-	123	-	203
tai75C	4.537	26,0	-	1	-	4	-	3	-	8
tai75D	2.645	43,9	3,2	2,6	3,2	2,6	3,2	2,6	3,2	2,6
f45	80	23,7	-	-	-	-	-	-	-	-
f71	3.483	72,9	-	19	-	348	-	418	-	994
f72	3.659	72,1	-	2	-	1	-	9	-	21

TABLE IV
RELEVANCE OF PERTURBATION FACTOR ON REINELT INSTANCES.

Instance	[15]		$pf = 0,05$		$pf = 0,10$		$pf = 0,15$		$pf = 0,20$	
	Objective	time	gap	time	gap	time	gap	time	gap	time
eil22	12	15,9	-	-	-	-	-	-	-	-
eil23	6	7,2	-	-	-	-	-	-	-	-
eil30	99	7,1	-	-	-	-	-	-	-	-
eil33	58	20,4	-	-	-	-	-	-	-	-
eil51	154	50,1	-	-	-	-	-	-	-	-
eil76	1.700	91,5	-	1	-	6	-	20	-	31
att48	284	17,4	-	-	-	-	-	-	-	-

TABLE V
SUMMARY OF PERMUTATION FACTOR EVALUATION.

Perturbation factor	Total gap	Total time
$pf = 0,05$	19,9	**24,9**
$pf = 0,10$	19,9	25,3
$pf = 0,15$	19,9	25,5
$pf = 0,20$	19,9	26,3

D. Real-world problem instances

We first present an analysis of the performance obtained using the iterated local search approach proposed. Moreover, in this case, we compare the performance of the algorithm with the performance obtained using a hill-climbing algorithm. For this, we replaced the perturbation step of the iterated local search by a restart procedure. Each algorithm was executed 40 times using different random seeds. The maximum time considered for these tests was set to 5.959 seconds. This time was the maximum time used in [15] to find a local maximum without using a clustering approach.

Fig. 5 shows the convergence of 40 executions of search processes performed by both, hill-climbing and iterated local search. Orange squares show the convergence of the hill-climbing approach, and blue circles show the convergence of the iterated local search. The black horizontal line in the upper area shows the maximum objective value found in [15] using the clustering approach. The gray line shows the maximum objective value found without applying the clustering approach. It is clear from the plot that the performance of the iterated local search algorithm is strongly superior to the performance of the

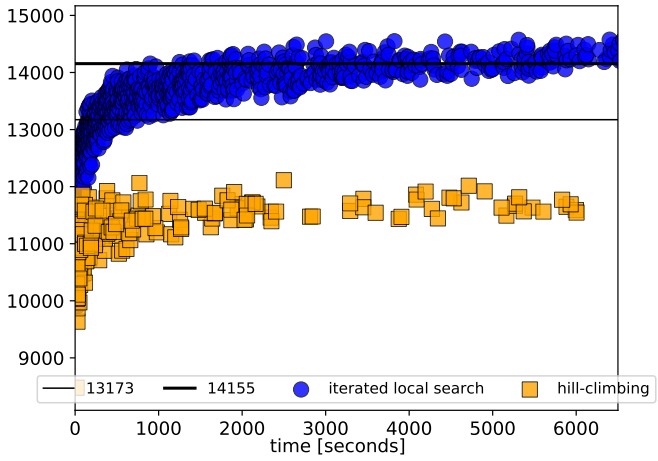

Fig. 5. Convergence of problem instance *r500-100* using hill-climbing (Orange squares) and iterated local search (blue points).

TABLE VI
WILCOXON TEST: ITERATED LOCAL SEARCH VERSUS HILL-CLIMBING

Comparison	Neg. Ranks	Pos. Ranks	Ties	p-value
ILS vs HC	0	40	0	0.00

hill-climbing approach. Moreover, the hill-climbing approach was unable to reach none of the maximum found in [15]. In contrast, the iterated local search reaches the first maximum (13.173) in around 100 seconds and the second maximum (14.155) in around 1.000 seconds. Table VI shows the pair-wise Wilcoxon tests comparing the performance obtained by our iterated local search approach and the performance of the hill-climbing algorithm. Negative ranks show the number of times hill-climbing found a better solution than iterated local search, positive tanks the number of times iterated local search found a better solution than hill-climbing. For this, we compare the quality of the best solution found in the 40 executions previously displayed in Fig. 5. Here we can observe that our iterated local search approach shows a significantly better performance than the hill-climbing algorithm.

1) Real-world instances comparison: The number of trucks considered for a problem instance strongly affects the size of the search space traversed by the algorithms. From the results presented in [15], we observed that it was possible to find feasible solutions using only 72 trucks. From this, we created the set of problem instances shown in table I. Fig. 6 shows the convergence of these problem instances. Yellow circles show the convergence when solving the instance that uses at most 72 trucks; green circles show the instance that considers 77 trucks; violet circles the instance that considers 88 trucks and blue circles the instance that considers 100 trucks. From these results, is it clear that reductions in the number of trucks allow the iterated local search approach to find high-quality solutions in a short time. In this case, the instance that considers only 72 trucks reaches the best objective function in around 500 seconds.

TABLE VII
WILCOXON TEST: ITERATED LOCAL SEARCH FOR REAL-WORLD PROBLEM INSTANCES.

Comparison	Neg. Ranks	Pos. Ranks	Ties	p-value
r500-100 vs r500-88	20	20	0	0.662
r500-100 vs r500-77	28	11	1	0.006
r500-100 vs r500-72	20	20	0	0.762
r500-88 vs r500-77	24	16	0	0.256
r500-88 vs r500-72	23	17	0	0.793
r500-77 vs r500-72	24	16	0	0.256

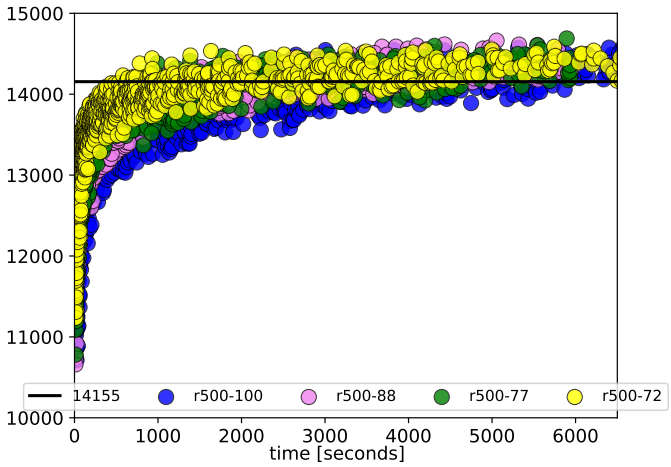

Fig. 6. Convergence of real-world problem instances using iterated local search.

Table VII shows the pair-wise Wilcoxon tests comparing the performance obtained by our iterated local search approach when solving the four variants of the real-world case studied. For this, we compare the quality of the best solution found in the 40 executions previously displayed in Fig. 6. Here we can observe that there is only one comparison presenting a significant difference. The iterated local search solving the instance that uses 77 trucks shows a significantly better performance the base case solution. Moreover, in most cases, there is no important difference between the performance of our approach when solving these four different problem instances.

2) Long term convergence: Since the optimum value of real-world problem instances is unknown, we performed a convergence study of the performance of iterated local search in two problem instances considering an extended execution time of 26.500 seconds. Fig. 7 shows the convergence of instances *r500-72* (yellow circles) and *r500-100* (blue circles). In our experiments, the best value obtained for the instance *r500-72* was 14.844 after 23.361 seconds, and for the instance *r500-100* was 14.846 after 26.432 seconds. Even when these execution times can be considered large, the time required in [15] to find its best value of 14.155 was 57.962 seconds.

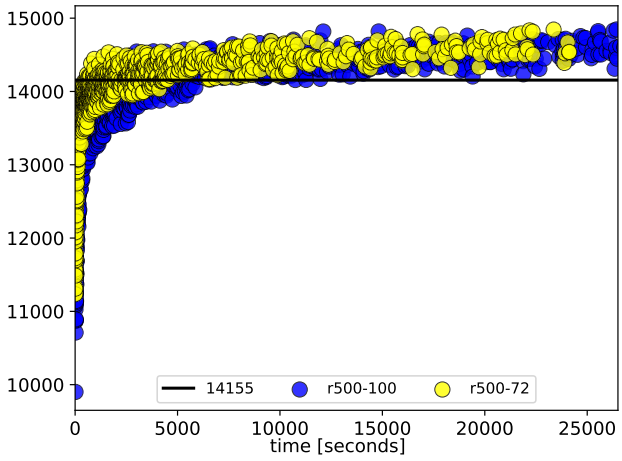

Fig. 7. Convergence of problem instances *r500-72* and *r500-100* using iterated local search with a maximum execution time of 26.500 seconds.

VI. Conclusions & Future work

Vehicle routing problems have been studied in specialized literature and practitioners for a couple of decades. Today it is still a research goal because of its high relevance in the supply chain of companies. The milk collection is one type of vehicle routing problem that has a special feature related to the multiple-product collection. Moreover, in this article, different qualities can be blended in the same truck, generating losses in income but a greater reduction in transportation costs, resulting in better profits for the company. Some exhaustive approaches have been formulated to face this problem. These approaches have solved small instances in quite reasonable times but can not solve real-world large size problem instances.

In this work, we proposed a local search-based algorithm to find high-quality solutions to large size problem instances of the milk collection problem with blending. Our iterated local search procedure implements two local search procedures allowing changes intra and inter routes generating the diversification and intensification the search process requires. Also, it implements a perturbation process able to generate variations repairable by both types of local search. We tested our approach using two sets of problem instances. From small instances, we can conclude about the abilities of our approach for solving these small instances in very reduced times. Moreover, from large size instances, we emphasize the abilities of our approach in finding better quality solutions than an exhaustive approach. A large size instance can be run in a short time if a requirement is changed (e.g., milk production, available trucks, costs, incomes, etc.)

As future work, we plan to improve our local search-based approach to explore more specialized approaches that can reach better solutions for large size problem instances. Moreover, we are interested in studying other versions of blending, were the result of mix depends on the concentration of microorganisms in milk and their proportions in the blend.

Acknowledgments

Authors want to thank to FONDECYT Initiation into Research project numbers 11150787 and 11170102.

References

[1] Claudia Archetti and Maria Grazia Speranza. The split delivery vehicle routing problem: a survey. *The vehicle routing problem: Latest advances and new challenges*, pages 103–122, 2008.

[2] Ph Augerat, Jose Manuel Belenguer, Enrique Benavent, A Corberán, D Naddef, and G Rinaldi. *Computational results with a branch and cut code for the capacitated vehicle routing problem*. IMAG, 1995.

[3] Ann Melissa Campbell and Jill Hardin Wilson. Forty years of periodic vehicle routing. *Networks*, 63(1):2–15, 2014.

[4] Massimiliano Caramia and Francesca Guerriero. A milk collection problem with incompatibility constraints. *Interfaces*, 40(2):130–143, 2010.

[5] AE Dooley, WJ Parker, and HT Blair. Modelling of transport costs and logistics for on-farm milk segregation in new zealand dairying. *Computers and electronics in agriculture*, 48(2):75–91, 2005.

[6] Miguel A Figliozzi. An iterative route construction and improvement algorithm for the vehicle routing problem with soft and hard time windows. 2008.

[7] Marshall L Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations research*, 42(4):626–642, 1994.

[8] A Hoff and A Løkketangen. A tabu search approach for milk collection in western norway. In *6th Triennial Symposium on Transportation Analysis, TRISTAN, Pukkett, Thailand*, 2007.

[9] Nadia Lahrichi, Teodor Gabriel Crainic, Michel Gendreau, Walter Rei, and Louis-Martin Rousseau. Strategic analysis of the dairy transportation problem. *Journal of the Operational Research Society*, 66(1):44–56, 2015.

[10] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[11] Tsai-Yun Liao and Ta-Yin Hu. An object-oriented evaluation framework for dynamic vehicle routing problems under real-time information. *Expert Systems with Applications*, 38(10):12548–12558, 2011.

[12] Shuguang Liu, Lei Lei, and Sunju Park. On the multi-product packing-delivery problem with a fixed route. *Transportation Research Part E: Logistics and Transportation Review*, 44(3):350–360, 2008.

[13] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

[14] Germán Paredes-Belmar, Andrés Bronfman, Vladimir Marianov, and Guillermo Latorre-Núñez. Hazardous materials collection with multiple-product loading. *Journal of cleaner production*, 141:909–919, 2017.

[15] Germán Paredes-Belmar, Vladimir Marianov, Andrés Bronfman, Carlos Obreque, and Armin Lüer-Villagra. A milk collection problem with blending. *Transportation Research Part E: Logistics and Transportation Review*, 94:26–43, 2016.

[16] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[17] Gerhard Reinelt. Tspliba traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

[18] Jayaram K Sankaran and Rahul R Ubgade. Routing tankers for dairy milk pickup. *Interfaces*, 24(5):59–66, 1994.

[19] Kanchana Sethanan and Rapeepan Pitakaso. Differential evolution algorithms for scheduling raw milk transportation. *Computers and Electronics in Agriculture*, 121:245–259, 2016.

[20] Marius M Solomon and Jacques Desrosiers. Survey papertime window constrained routing and scheduling problems. *Transportation science*, 22(1):1–13, 1988.

[21] Éric D Taillard. A heuristic column generation method for the heterogeneous fleet vrp. *RAIRO-Operations Research*, 33(1):1–14, 1999.

[22] CD Tarantilis and CT Kiranoudis. A meta-heuristic algorithm for the efficient distribution of perishable foods. *Journal of food Engineering*, 50(1):1–9, 2001.

[23] A Serdar Tasan and Mitsuo Gen. A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers & Industrial Engineering*, 62(3):755–761, 2012.