23rd International Conference on Harbor, Maritime and Multimodal Logistic Modeling & Simulation
18th International Multidisciplinary Modeling & Simulation Multiconference

ISSN 2724-0339 ISBN 978-88-85741-59-1 © 2021 The Authors.
doi: 10.46354/i3m.2021.hms.003

Exact and heuristic static routing algorithms for improving online grocery shopping logistics

Francesco Cepolina [1], Elvezia Maria Cepolina [2] and Guido Ferla [3]

[1]Department of Mechanical Engineering, DIME, University of Genoa, Genoa, 16145, Italy
[2]Italian Center of Excellence on Logistics, Transport and Infrastructure, CIELI, University of Genoa, Genoa, 16135, Italy
[3]School of Sciences and Technology, University of Camerino, Italy

*Corresponding author. fcepolina@hotmail.com

Abstract

The context is online grocery shopping and the paper focuses on the optimization of the related transport logistics that can lead to important economic and environmental advantages. At the beginning of the day, each customer provides: a shopping list, defining the product typologies and the related quantities to be collected from the different shops, the delivery address and the related delivery time window. One vehicle is in charge of serving all the customers by collecting the products from the shops and by delivering them to the provided delivery addresses. The target is to find the shortest path that satisfies the customer's needs.

The proposed routing algorithms could support also logistic processes in supermarket supply. Each supermarket defines the daily freight demand, in terms of product typologies and related quantities, to be collected from the different manufactures/distributors, the delivery address and the related delivery time window. One vehicle is in charge of collecting the products from manufactures/distributors and of delivering them to the provided delivery addresses.

The faced problem is therefore a complex multi commodity pick-up and delivery traveling salesman problem. Many constraints could be taken into account, related, for instance, to ecology and customer satisfaction. One local and four global optimization algorithms are proposed; their advantages and limits are discussed. The algorithms are tested on a basic logistic example, the numerical results are reported. The proposed algorithms use effective and efficient optimization algorithms able to minimize the overall miles necessary to deliver the goods in order to increase business efficiency.

Keywords: Logistics, grocery e-commerce, last mile delivery, pick-up and delivery points, exact and heuristic algorithms.

1. Introduction

In recent years, the retail company has been working with its suppliers to better anticipate and meet consumer demands, thereby reducing shortages, excess inventory and waste, and ensuring the performance and resilience of its supply systems. Production and retail supply chain are therefore expected to follow the dynamic of customer demand and logistics processes need to be more flexible and faster, helping to increase their quality of service and business efficiency.

Customer demand is defined even more frequently online. In online shopping, the orders are characterized by tight order-to-delivery lead times and the frequent and discrete arrival of orders. Customers expect to receive their products anytime and anywhere with excellent service and high convenience. Delivery is therefore becoming always more important issue in logistics.

In order to optimize the delivery, several aspects need to be considered such as: ecology - electric mobility is becoming increasingly popular for freight services in urban areas (Molfino et al., 2015); customer satisfaction - linked also to a good freight handling and preservation and to the extension to which the logistic system is able to meet customer needs in term of time

and location of the delivery (Cepolina, 2016); shortest road; fastest road. What is the best path to satisfy all these needs? The optimization is not trivial. For retail stores, this adds to the problems of store layout analysis, item and shelf management, and store monitoring and control (Bruzzone et al. 2010).

One of the basic human desires is to automate tasks in order to simplify life. Bow, washing machine, robots are just a few examples. With the advent of informatics this wish has been translated into a code. This code represents the mind of the process. Automation is based on the reproduction of several basic commands in a loop.

The first step is to create the loop itself. The idea behind a loop is to replicate the same basic action a number unlimited of times.

Input variables are processed by a loop (while... wend, for... next) of conditions (if... then).

A classic example is an industrial control processes that takes, as an input, data read from a sensor, and corrects real time the process in order to keep the value inside a given threshold.

In order to create a clean code easy to read, programs are often split in subroutines. The same subroutine can be recalled many times from the main program. Open source and commercial libraries are also available; these libraries include a rich collection of subroutines.

Based on these ideas, different approaches, for optimizing the collection and distribution of products in the context of online grocery shopping, are presented and compared.

Four exact methods are presented: the first one is the brute force method that performs exhaustive search. The second, the third and the last exact methods have been obtained by sequential modifications to the brute force method. The apported modifications aim to increase the method efficiency, in terms of computational time reduction.

A very simple heuristic method is presented as well. Its advantages, in terms of computational time, and its disadvantages, in terms of no optimality of the obtained solution, that determine the effectiveness of the algorithm, are discussed.

The paper has been structured in the following way: section 2 presents the problem and put it in the literature context. Section 3 describes a simple case study. Section 4 describes the proposed five optimization approaches. The optimization approaches are compared in terms of efficiency in section 5. Discussion and conclusions follow.

2. Problem description and literature review

At the beginning of the day, each customer provides: a shopping list, defining the product typologies and the related quantities to be collected from the different shops, the delivery address and the related delivery time window. This defines the daily customer demand that has to be satisfied.

One vehicle is in charge of collecting the products from the shops and delivering them to the provided delivery address.

It has been assumed that the customer demand is low and therefore, only one trip is enough for satisfying it. The trip starts and ends in the garage.

The vehicle is electrically powered therefore it needs to be recharged in the garage whenever the battery is low. However, we assume that the van leaves the garage with a fully charged battery and its capacity allows satisfying the customer demand without any recharging.

The proposed algorithm aims to define the minimum cost route that satisfies the customer demand.

The road network in represented by a graph. The graph is composed by a set of nodes and a set of links, where each link is defined by an ordered couple of nodes. The graph is therefore oriented. Nodes represent shops, delivery addresses and the garage location. A constant cost is associated to each link; the cost in one direction can be different from the cost in the opposite direction.

According to the literature, the problem faced can be defined as:

- TSP, since the problem consists in determining a minimum cost Hamiltonian cycle in the graph (if any exists) that respects all the problem constraints. There is no vehicle capacity.
- PD-TSP: it is a pickup and delivery TSP since there are pickup nodes, delivery nodes and one garage. Pickup nodes represent shops whilst delivery nodes represent the locations where products have to be delivered. Certainly, a given product has to be collected in a pickup node before been delivered in the related delivery node.
- m-PD-TSP: it is a multi-commodity pickup and delivery TSP since pickup and delivery nodes are paired to form requests. Bread has to be collected in a bakery and not in a butcher's shop and has to be delivered to the customer that ordered it. We don't want to deliver fish to a customer that is waiting for bread!
- TW-m-PD-TSP: in the TSP with time windows customers may choose a time window, which is defined by an earliest delivery time and a latest delivery time, during which the package is delivered.
- The problem is non-preemptive since products cannot be dropped at intermediate locations and picked-up again later.

However, there are other constraints that need to be considered, in order to define a more realistic and customer oriented logistic problem. In the following,

the additional constraints that the proposed algorithms aim to consider (Figure 1) are reported.

- Fresh/Drug: there are specific classes of goods, like "fresh" food and "drugs", which may require prioritized delivery and specific handling.
- Load: the size of the products may impose additional physical constrains to the van "load" and unload operations; for example, it may be necessary to load last, and unload first, a big product.
- Traffic: congestion on the read network changes with the time of the day. Late afternoon delivery may be preferred inside crowded cities, if compatible with customer defined time windows.

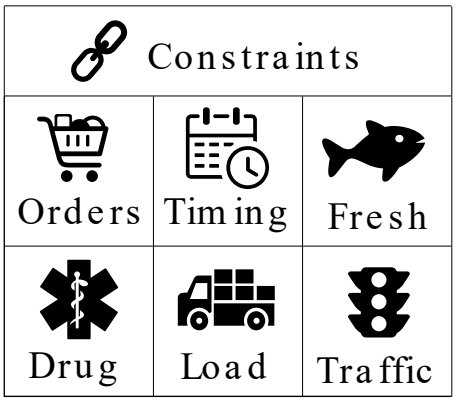

Figure 1. Simulation constraints

Even if the literature about the TSP is very rich and many are the variations of the original TSP that have been approached with exact algorithms or heuristics, as far as we know the proposed problem has not yet being studied.

The PD-TSP has received relatively little attention. While some papers have proposed exact algorithms for the PD-TSP and some of its variants (Ascheuer et al. 2000; Hernández-Pérez and Salazar-González, 2004; Lu and Dessouky, 2004; Ropke et al. 2007), most have focused on heuristic solution methods (Renaud et al., 2002 and Renaud et al. 2000). Branch-and-cut algorithms have been adopted by Ropke, Cordeau and Laporte (2007) for the pickup and delivery problem with capacity and time window constraints, where vehicle routes must also satisfy pairing and precedence constraints on pickups and deliveries.

As it concerns the m-PD-TSP, the literature contains many interesting articles on the delivery of different commodities (objects or persons) with a single vehicle. When the commodities are persons, these problems are non-preemptive, consider time-window constraints and are typically termed Dial-a-Ride problems (Psaraftis, 1983). Rodriguez-Martin and Salazar González (2011) describe some strategies to solve these problems based on a branch- and-cut procedure. When instead of a vehicle having to pick up people and take them to their destinations, it is the person who picks up a vehicle and leaves it at his or her destination, the system is called one way car sharing and several solutions have been proposed to solve the problem related to the vehicles rebalancing in the service area (Cepolina et al., 2015).

Complex problems are usually solved finding a "good solution" with local optimization (heuristic methods). The global (exact) optimization techniques, necessary to find the optimal solution are often discarded because are too slow.

The paper compares the performance of both the optimization techniques for the faced problem (Figure 2).

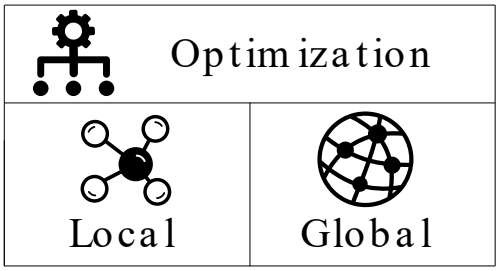

Figure 2. Optimization techniques

A single variable optimization is performed: it is possible to minimize the route length or the route time (Figure 3).

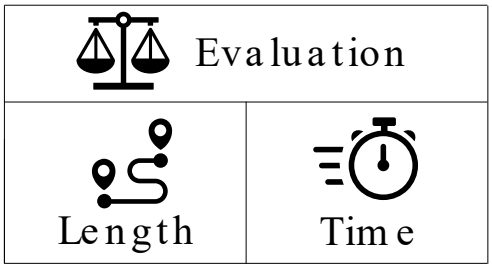

Figure 3. Single variable optimization: minimization of the trip length or of trip time

3. Testing example

In order to test the proposed algorithms, a simple example is taken into account. The customer demand requires to deliver to Mary one kilogram of fruit and to send to Tom a fresh cake. Mary and Tom have two different delivery addresses (Figure 4).

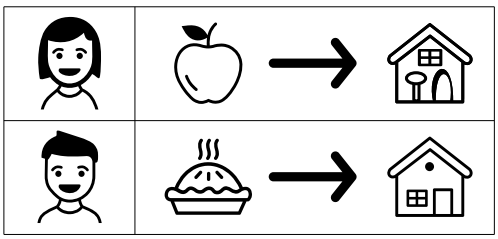

Figure 4. Basic Example

The van can make a stop in one of these five nodes

(Figure 5); the garage where the charging facility is (A), the house of Tom (B), the house of Mary (C), the bakery (D) and the greengrocer (E). The distance matrix related to these locations is known.

As a preliminary approach, the trip distance is minimized.

Then the van will make overall 5 stops, included the garage.

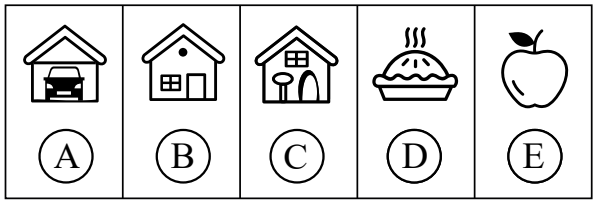

Figure 5. Basic example – the 5 nodes

The dimension of the search space is 5^5. Only a subset of routes is feasible since they satisfy the constraints. The target is to find among these, the shortest one. Some local and global optimization strategies are now analyzed and described.

4. Optimization algorithms

The search space must be explored. Exploration can be exhaustive (in exact methods) or limited (in the heuristic method).

Exploration is performed by generating solutions. In all the proposed methods, the generation is performed starting from an empty solution and, iteratively, at each step, adding new elements to the solution, according to a criterion, until a complete solution is defined.

The new added element could be feasible (in the pre-treated, embedded and recursive exact methods and in the heuristic method) or not (in the brute force exact method). In the first case, the new element is selected only among the feasible ones: at each iteration, partial solutions satisfy the problem constraints and only feasible complete solutions are assessed. In the second case, the new element is any node. Only after the complete solution generation, the solution feasibility is checked.

The cost function can be assessed for the complete solutions (in the pre-treated exact method and in the heuristic method) or for the partial solutions (in the embedded and recursive exact methods).

4.1. Global optimization: the brute-force exact method

The problem is solved by an exhaustive search.

In order to find the optimal solution, it is necessary to create 5^5 possible trips, apply all the constraints considered, calculate the cost function value for each resalting feasible trip, and finally compare the results.

While the problem is correctly set, this "brute force" approach is time consuming; it takes time to explore all these solutions. Depending on the size of the problem, the execution time risks to make this solution not viable.

The simulation program is written in Visual Basic and Microsoft Excel for portability reasons. The 5^5 solutions are generated thanks to a 5 levels nested (For.. Next) cycle. The parameters a,b,c,d and e represent the van stops (a=stop-1, b=stop-2 etc.). A simplified program for the creation of the simulation tree is reported:

```
For a=1 to 5
  For b=1 to 5
    ......
      For e=1 to 5
      Create path=f(a,b,c,d,e)
      Next e
    .......
  Next b
Next a
```

It is interesting to notice that the size of the code grows with the size of the problem: a map with 20 stops needs a code with 20 nested levels of iterations.

The optimization steps (Figure 6) highlight that the program needs to do 3 series of iterations in sequence.

Steps	Loops
Data input	1
Tree creation Loop A(B(C(D(E))))	5^ 5
Apply constraints Loop A(B(C(D(E))))	5^ 5
Comparison of all the solutions	5^ 5
Output results	1

Figure 6. Global optimization v1 – brute force

The simulation tree represents all the possible paths that are evaluated. For example, the path a-a-a-a-a is a trip where the van enters 5 times into the garage. This trip is not feasible since it doesn't satisfy the customer demand constraint. The graphic representation of the tree is schematic; in order to allow a clear representation, only a few nodes are represented (Figure 7).

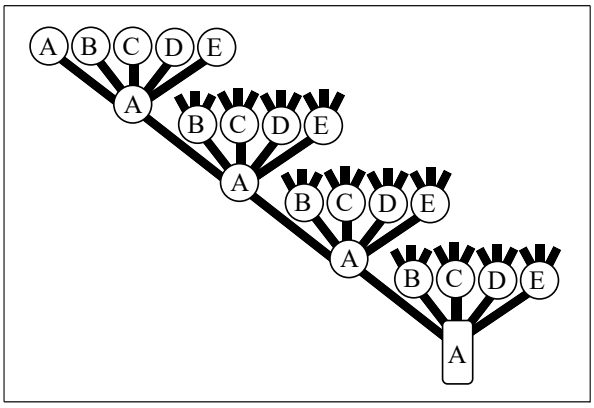

Figure 7. Simulation tree

4.2. Local optimization – heuristic method

Complex problems are usually solved thanks to heuristic methods. The speed gained by the solver is paid by the partial loss of view of the problem; the found solution is "good enough" but we do not know any more if we have found the best solution. The estimation of the error is also not always trivial.

The implemented algorithm belongs to the family of constructive heuristics: it provides a solution starting from an empty solution and, iteratively, at each step, new elements are added to the solution according to a predefined expansion criterion, until a complete solution is defined.

The implemented algorithm is a very simple greedy algorithm that adopts a local expansion criterion, where the choice is the one which seems to be best choice in that moment, also taking into account the constraints of the problem: at each iteration, the element to add to the current solution is the one that provides the best improvement to the objective function (i.e. the closest one). Only feasible elements that satisfy the problem constraints, could be added.

The van, going out of the garage, will select the closest node representing a pickup location (stop-1). Once the van will arrive to its first stop (stop-1), it will select (stop-2) using the same "minimum distance" criteria among the nodes representing the other pickup points + the delivery address of products already uploaded, and so on.

Only one solution is generated.

Local optimization is blazing fast, compared to global optimization; the number of loops is really limited (Figure 8). However, local optimization judges only using local information.

Steps	Loops
Data input	1
Tree + constraints + evaluation Loop A	5
Tree + constraints + evaluation Loop B	5
Tree + constraints + evaluation Loop C	5
Tree + constraints + evaluation Loop D	5
Tree + constraints + evaluation Loop E	5
Output results	1

Figure 8. Local optimization

A simplified code showing the optimization strategy is reported.

For a=1 to 5

Create the path=f(a)

Appy the constrains on the path

Evaluate the path

Next a

For b=1 to 5

Create the path=f(b)

Appy the constrains on the path

Evaluate the path

Next b

....

The local optimization simulation tree, shows the possible sequences of stops, starting from the garage; the possible values of each stop are contained into an elongated circle (Figure 9). An example of output for a specific map is: A-E-D-B-C.

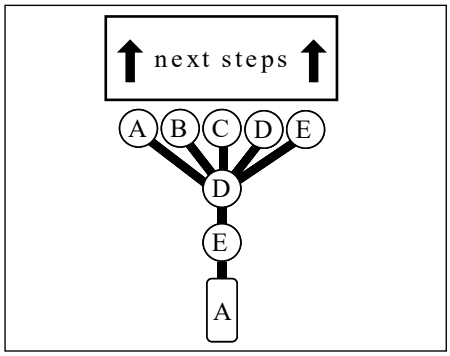

Figure 9. Local simulation tree

4.3. Global optimization – the pre-treated exact

method

Like for the local optimization strategy already described, the global optimization process can be highly accelerated imposing the constraints during the creation of the population. Using this approach, whole families of units are eliminated in one shot.

Imagine judging an hair style competition. Each people joining the competition has several attributes that need to be evaluated (by the constraints of the target function): hair color, hair length, sex, age, skin color, look, eye color, height, weight. Assuming that half of the people are bold, they can be directly excluded from the competition, without doing any other evaluation. The first basic question: "Are you bold?" allows saving a lot of time.

Table 1. Global optimization steps

	N	Paths	Ⓐ	Ⓑ	Ⓒ	Ⓓ	Ⓔ
	01	A.(D,E)					
	02	A.D.(B,E)					
	03	A.D.B.E					
*	04	A.D.B.E.C					
	05	A.D.E.(B,C)					
*	06	A.D.E.B.C					
*	07	A.D.E.C.B					
	08	A.E.(C,D)					
	09	A.E.C.D					
*	10	A.E.C.D.B					
	11	A.E.D.(B,C)					
*	12	A.E.D.B.C					
*	13	A.E.D.C.B					

The example is now discussed. The steps necessary for the pre treated global optimization are now analyzed (Table 1). When the van goes out from the garage (A), it can only buy goods (D, E) because the constraint related to customer demand satisfaction forbids reaching the delivery addresses (A, B) without the food. The target function also forbids the van going to the garage (A); the van starts the trip with a fully charged battery. This step 01 is represented with: A.(D,E). Now the van can go to buy the apples or the cake, so the tree is split in two branches A->D (garage -> cake shop) and A->E (garage -> apples shop). Following the first branch, now two further options arise: deliver the cake to Tom D->B (cake shop -> Tom house) or buy the apples D->E (cake shop -> apples shop). So far the journey is A.D.(B,E). The exploration of the paths continues until all the options are considered. The evaluation of 13 steps gives 6 possible solutions, highlighted with (*). From the comparison of these 6 solutions the best path is found; the shortest path is the best one.

The pre treated approach, imposing the conditions during the creation of the tree, allowed to reduce the algorithms loops from 3x(5^5) (Figure 6) to 13x5+6 (Figure 10). The algorithm execution speed is directly related to the number of loops.

During the first step the option A, B and C have been discarded, the removal of the "bold" population gave a big time saving; 3 branches (3*5^5 solutions) of the general three (Figure 7) are cut at the starting of the process.

Steps	Loops
Data input	1
Tree + constraints creation Loop A(B(C(D(E))))	13x5
Comparison of all the solutions	6
Output results	1

Figure 10. Global optimization v2 – pre-treated

The graphic representation of these six paths has the shape of a tree (Figure 11) really small compared to the unconstrained one (Figure 7).

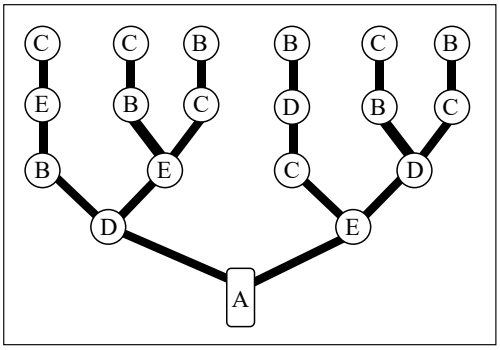

Figure 11. Global optimization tree

4.4. Global optimization – the embedded exact method

Sometimes it is annoying not to know how much it is lost doing local optimization; an estimation of the error may be advisable. It is a pity to eat a small slice of cake in the sitting room, while a whole big cake is waiting in the kitchen. The desire to seek "big cakes" has led to look for cleaner and faster global optimization techniques.

The pre-treated model is already faster than the brute force model. Somehow local optimization is a linear problem (Figure 9), while the global optimization is an exponential problem (Figure 7); for this reason, it is necessary to create really efficient algorithms, mainly when the size of the problem grows. In the example, the size of the problem is given summing the number of shops and the number of houses to be reached by the van.

The pre treated strategy can be further improved, replacing the final comparison of solutions (Table 1)

with a real time selection of the temporary best solution (Figure 12). For example, the length of the path A.D.E.C.B (step 07, Table 1) can be compared with the length of the temporary winning path A.D.B.E.C (step 04, Table 1).

Following this approach, the number of loops is decreasing (good), while for each loop, the program loses time comparing the new solution with the last best solution found (bad). The balance is positive? There is a real gain in the execution time? Yes, the van simulator shows that embedded simulation is faster than the pre-treated one.

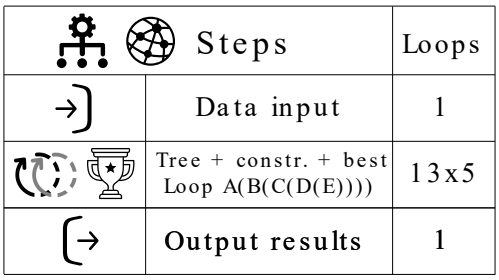

Figure 12. Global optimization v3 – embedded

The size of the "embedded solution" code is shorter. This embedded approach can also stop the exploration of any not finished path, if its length is already longer than the shortest path found. For example, if the length of the path A.D.E.(B,C) of (step 05, Table 1) is already longer than the winning path, all the sons of this tree branch (A.D.E.B.C and A.D.E.C.B.) do not need to be explored. Rejecting a single branch may erase even 1000 sons in one shot. This branch pruning makes the algorithm blazing fast.

4.5. Global optimization – the recursive exact method

A further step is now performed working on the algorithm structure; the code of the algorithm is now dramatically changed introducing the use of recursive subroutines (Figure 13).

All the local and global optimization models presented so far, feature a nested (for.. next..) structure. This implementation has a big limit. If the number of problem parameters changes, it is necessary to implement a custom code. A problem with 1 garage, 10 houses and 10 shops will need a specific code having 21 nested (for.. next) loops.

The number of parameters of the problem changes the search optimization code size; a big code is difficult to clean, maintain and update. Moreover, it runs slowly.

The recursive solution is implemented with a code made by three elements; "data input", "subroutine X" and "data output" (Figure 13). This special subroutine, can generate a new instance of itself. The process can be recursively iterated. For example the "subroutine X(1)" solves the step 01 sub-problem (Table 1) and then creates "subroutine X(2)" for step 02 and "subroutine X(3)" for step 08. Each subroutine stays open until its sons have fully explored the rest of the branch. A hill formed by hundreds of subroutines may stay open at the same time. At least during our tests, every time a large simulation runs, we needed to reset the system because, once the simulation is finished, the large quantity of RAM used is not fully released by the operating system.

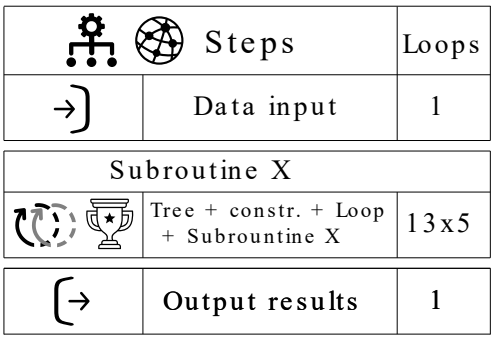

Figure 13. Global optimization v4 – recursive

The embedded solution code is clean and short. In our example (2 houses and 2 shops) the embedded algorithm size is like 70% smaller than the recursive algorithm; this solution allows a faster execution and an easier maintenance.

The recursive algorithm can be defined as a self replicating machine; according to its needs, the algorithm can create an unlimited number of copies of itself to dig into a big data structure

5. Numerical results

The solver is written, for portability reasons, with a Visual Basic macro of Microsoft Excel. The performance of the local optimization solver and the global optimization recursive solver, have been tested on a MacBook Pro Late 2013 (processor 2,3 Ghz, Quad-Core, Intel(R) I7, 8 GB RAM). The results are reported in the Table 2.

Table 2. Simulation results

				Error
N	N	mins	mins	%
2	2	0.03	0.08	61.5
2	3	0.05	0.26	48.5
2	4	0.06	0.83	13.2
2	5	0.08	3.18	22.2
3	5	0.09	11.35	8.2
4	5	0.11	48.45	0

For the heuristic method and the global recursive exact method, the computational time increases with the complexity of the problem. Local optimization is really fast; the algorithm takes a few seconds to run any problem. The table shows also the errors affecting

the result of the local optimization, for the specific solved problem. The complexity of the problem grows exponentially with the number of its parameters (Figure 7).

The results of the last two simulations are now analyzed (last two rows in table 2). A problem with 1 garage, 3 houses and 5 shops, has a complexity of 9^9. The global recursive exact method took about 11 minutes to solve it. The last problem with 1 garage, 4 houses and 5 shops, has a complexity of 10^10 and an execution time of 48 minutes.

While the complexity of the problem increases exponentially 26x (9^9 → 10^10), the global optimization time increases slowly 4x (11 → 48). This positive result shows that the "branch pruning" approach saves more time on big trees; the pruning is able to compensate well the increase of the problem size.

The increase of speed given by an optimized algorithm allowed running 20 times faster the same task (comparison between the global optimization v1 and the global optimization v4); this means that it is better to focus on code optimization, before to buy a new more performing PC.

6. Discussion

Today the technology is giving us new opportunities. The number of objects generating digital data is growing. Plenty of sensors (cameras, GPS, accelerometers) contribute to create the "Big data" environment. All these sensors are interconnected and create the "Internet of things".

During the last years the price of technology, such as data storage, data flow and computation power is steadily decreased. It is relatively easy to record, store and share big amount of data.

The problem still open is to deeply understand and use this data. Artificial intelligence is moving now the first steps in this direction. A better web crawler could understand better the web. Smarter algorithms may analyze efficiently financial data. A full deep understanding of the DNA code may open new frontiers. Long term weather forecasts are open topic of research for the next future. The algorithm can also be useful in an Industry 4.0 environment because the required automation technology must be improved by the introduction of methods of self-optimization, self-configuration, self-diagnosis, cognition and intelligent support of workers in their increasingly complex work.

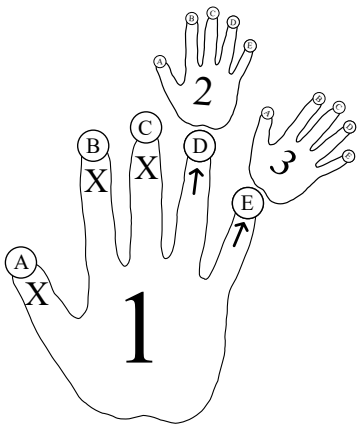

Figure 14. Hands example

It may be interesting to apply the recursive approach to the cited problems. Recalling the delivery example, the subroutine of the algorithm is like a hand (Figure 14); each of the 5 fingers representing the evaluation of a different place (shop or house).

Each finger that satisfies all the constraints, and is not the last place to visit, creates another "smaller" hand. The mission of each hand is to explore all its branch of solutions; the places of all its fingers need to be fully evaluated, including the places of the fingers of their "smaller" hand sons.

Once a hand accomplishes its exploration mission, it communicates the results of its quest to its father hand and disappears.

For example, this approach may be applied to a maze (Figure 15).

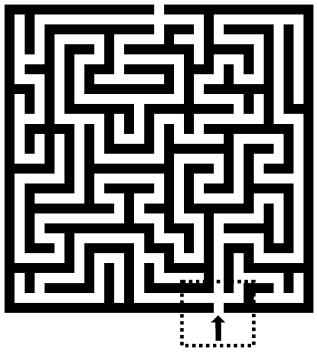

Figure 15. Maze example

The maze is segmented in small portions (Figure 16). In the case of logistics, the hand has five fingers each representing a place to visit. In the case of the maze, the hand has four fingers and each finger describe a possible movement inside the maze (North, South, East West).

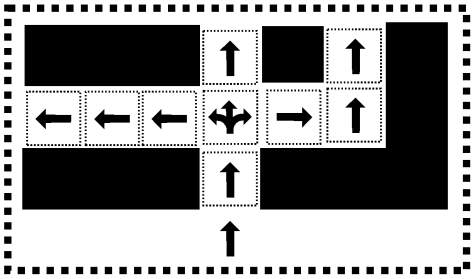

Figure 16. Maze segmentation

7. Conclusion and future work

According to De Giovanni (2018), it is not always possible or appropriate to apply exact solution methods, due to basically two concurrent issues: the inner complexity of the combinatorial optimization problem, and the time available to provide a solution, which may be limited.

The use of a heuristic method instead of an exact one must be well motivated by the inner complexity of the problem together with consideration on the opportunity of implementing exact methods (which may require, for example, considerable implementation resources), the available computational time, the size of the instances to be solved etc.

With reference to the online grocery shopping, four exact methods and a very simple heuristic method have been proposed; their efficiencies have been compared on the testing example.

The heuristic method is very fast. The solution given by this algorithm could be not the optimal one. This method could result efficient in dynamic (real-time) system, where it is required that a "good" feasible solution is provided within a limited amount of time.

Among the exact methods, the recursive one is the most efficient and shows promising performances. It is a disruptive powerful tool that enables to dig into a wide set of big data problems. This is the reason why the proposed method also seems suitable for improving logistic processes in supermarket supply where the retail company aims to reduce shortages, excess inventory and waste.

The exact recursive method provides the optimal solution but it is still relatively slow comparing with the heuristic method. A new version of the exact recursive method is still under development (Cepolina et al., 2021), with the aim of further increase the execution speed. The plan is to run the final refined algorithm for solving dynamic problem. In this context, the data input can be real-time updated and the time available to provide a solution is limited.

Acknowledgements

This paper was supported by European Union's Horizon 2020 research and innovation program under grant agreement number 875530, project SHOW (SHared automation Operating models for Worldwide adoption).

References

Ascheuer, N., Jünger, M., Reinelt, G. (2000). A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints. *Comput. Optim. Appl.* 17:61–84.

Bruzzone, A.G., Longo, F. (2010). An advanced system for supporting the decision process within large scale retail stores. *Simulation*, 86(12), 742-762

Cepolina E.M., Farina A., Holloway C., Tyler N. (2015). Innovative strategies for urban car-sharing systems and a simulator to assess their performance. Transportation Planning and Technology. 38:375-391.

Cepolina, E.M. (2016). The packages clustering optimisation in the logistics of the last mile freight distribution. *International Journal of Simulation and Process Modelling*, 11:468-478

Cepolina, E.M., Cepolina, F., Ferla, G. (2021). On line grocery shopping: a fast dynamic vehicle routing algorithm for dealing with information evolution. In E. Bottani, A. G. Bruzzone, F. Longo, Y. Merkuryev, M. A. Piera (Eds.). *Proceedings of the International Conference on Harbor, Maritime and Multimodal Logistic Modeling & Simulation (HMS 2021)*. DIME Università di Genova, DIMEG Università della Calabria: Publisher.

De Giovanni, L., Gastaldon, N., Losego, M., Sottovia, F. (2018). Algorithms for a Vehicle Routing Tool Supporting Express Freight Delivery in Small Trucking Companies. *Transportation research procedia*. 30:197-206.

Hernández-Pérez, H., Salazar-González, J.-J. (2004). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Appl. Math.* 145:126–139.

Lu, Q., Dessouky, M. (2004). An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Sci.* 38:503–514.

Molfino, R., Zoppi, M., Muscolo, G.G., Cepolina, E.M., Farina, A., Nashashibi, F., Pollard and E., Dominguez, J.A. (2015). An electro-mobility system for freight service in urban areas. *International Journal of Electric and Hybrid Vehicles*, 7:1-21.

Psaraftis, H. (1983). An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*. 17:351–357.

Renaud, J., Boctor, F.F., Laporte, G. (2002). Pertubation heuristics for the pickup and delivery traveling salesman problem. *Comp. Oper. Res.* 29:1129–1141.

Renaud, J., Boctor, F.F., Ouenniche, J. (2000). A heuristic for the pickup and delivery traveling salesman problem. *Comp. Oper. Res.* 27:905–916.

Rodriguez-Martin, I. and Salazar González, J. J. (2011). The Multi-Commodity One-to-One Pickup-and-Delivery Traveling Salesman Problem: A Matheuristic. In Pahl J., Reiners T., Voß S. (Eds.). *Network Optimization. INOC 2011. Lecture Notes in Computer Science*. 49:258–272. Springer, Berlin, Heidelberg.

Ropke, S., Cordeau, J.-F., Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks.*