Department of Information Engineering and Computer Science

Master's Degree in
Artificial Intelligence Systems

Final Dissertation

DESIGN AND DEVELOPMENT OF AN OPTIMISATION FRAMEWORK FOR WASTE COLLECTION IN LARGE GEOGRAPHICAL AREAS

Supervisors

Roveri Marco

Palopoli Luigi

Student

Bertelli Davide

Academic year 2021/2022

Contents

Abstract

The aim of this work is to address the design and implementation of a framework able to optimize the garbage collection task in large municipal areas minimizing the resources in use and the time needed for completion.

The design phase revolves around the idea to employ free access geographical data which describe the territory of the province of Trent, to then model the problem of garbage collection through the Constraint Programming paradigm providing a Mixed Integer mathematical definition of the task that several off-the-shelf solving software can handle. The focus is on addressing this problem in such a manner that the resulting model would grow as linear as possible, bounding the complexity of the solution searching task. In this phase it has also been decided to employ two different kinds of agents devolving them to the waste collection and disposal tasks, respectively, in an attempt of simulating real companies inner organization. Furthermore a categorization of the geographical elements in use is proposed to better associate the agents to the suited areas where their abilities can be exploited to carry out the garbage management task.

Concerning the implementation phase, the system has been developed taking into account the available data representing urban areas comprehensive of all the geographical elements described through a set of tags and their coordinates expressed in latitude and longitude. A specific algorithm has been written and applied to mine knowledge about the existing buildings, the roads supplying them and the intersections among the streets. Furthermore, the specific characteristics of the roads have been exploited to fully understand the road network capabilities identifying narrow passages or one way only paths. Subsequently, the information regarding the buildings locations and encumbrance has been used to estimate the probable waste pick up points in the road network. This was achieved through a mapping operation of the buildings to the nearest roads. The system exploits this knowledge to build a suitable graph representation of the environment, comprehensive of the road network capabilities and the paths among the buildings, to provide an accurate model that will be used in the planning step of the framework. The graph will be translated into a set of mathematical relations linked to a cost function influenced by the agents, the collection nodes available and the amount of time that each actor can be used. Next, a proprietary Constraint Programming solver and optimizer searches for a solution satisfying all the relations added in the model, thus providing a sequence of steps expressing how the garbage is collected and disposed, and how the agents move across the territory. Throughout the implementation phase, the main performances measure has been the computation time of a solution. For this reason, to speed up the solver operations, it has been chosen to apply hierarchical planning during the model making step and a clustering algorithm over the graph representing the environment. The hierarchical formulation splits the planning process into two independent subsequent tasks in which the solution to the first is exploited as a prior knowledge to simplify the problem tackled by the second one, being most of the operations immutable fixed points in time. Consequently the second problem to solve is simplified, yielding to a smaller search tree that is parsed faster by the solving algorithm. Similarly, clustering allows to drop the computation times by compacting the graph associated to the environment merging strongly connected nodes.

In testing, the employment of these features results into computational time improvements in the vast majority of the cases with few outliers. Although, computational time keeps being the main bottleneck in the system, being the garbage collection task part of those problems which complexity grows exponentially with the number of parameters to handle. In the last part of this work it has been led a comparison with another open source optimization engine in which the pros and cons of both approaches are highlighted.

1 Introduction

1.1 Purpose

In the modern society the waste management task has become an highly important matter in the organization of cities and human livable territory, this due to the astonishing improvement of the urbanization phenomenon that took place in the last decades, together with the growth of the population caused by the migration flows and an improved life expectancy. Consequently, vast areas were transformed for residential use, increasing the population density on the territory and favouring the production of municipal and industrial wastes. An overview of these events is shown in Figures 1.2, 1.4 and 1.3. To sustain the increasing numbers of inhabitants and to ensure the availability of city services to the vast majority of the citizens, the road network supplying the environment has been improved becoming more tangled over the time. Coupling this phenomenon with the poor availability of waste management implants such as incinerators, disposal sites or waste to energy facilities, Figure 1.1, able to recycle, stock and destroy the garbage, it is immediate to understand how the need of an efficient system able to tackle the waste management problem in such a complex environment raised to avoid health and decency issues. Furthermore, such framework would allow the companies, adopting it, to observe an improved quality of service, a more flexible human resources management and a reduction in costs related to vehicle maintenance and fueling, together with dropping the human error in the scheduling tasks and improve their market reliability. A dynamic workforce reallocation would be possible by exploiting an algorithm which plans the best routes to finish the waste disposal jobs in the least possible time and avoiding traffic queues that are known to build up during the the day. On the citizens point of view, the employment of such system would allow to live in a less polluted environment with a perceived better living experience.

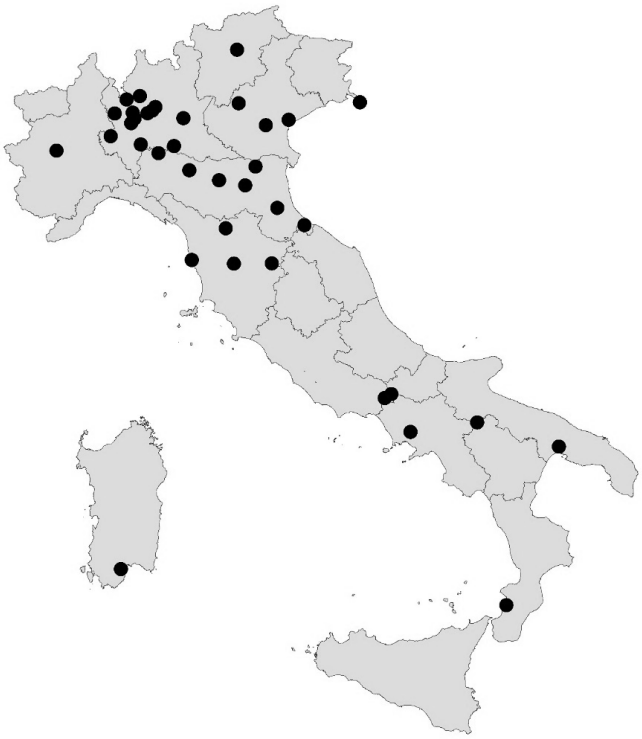


Figure 1.1: Map of the available incinerators for municipal waste, Italy 2022 [5].

The purpose of this work is to design and build a framework optimizing the garbage collection task specifically under the aspects of the system throughput, the computational time needed to find a solution and the amount of resources available. Focusing on the throughput, the framework will ensure that at each time step the garbage collected is the maximum possible one, simultaneously minimizing the time required for completing the task. Forcing the planner algorithm to reduce the amount of resources in use it will be possible to lower the costs of the operations and have at disposal a surplus of agents to rely on, whenever necessary. A tool like this, will allow to reduce the latency in waste management and balance the workload among the actors and to avoid queues at the disposal facilities. At the same time, employing such algorithm will permit to decrease the amount of gas emissions caused by the vehicles in use, which in the city areas is one of the main issue being among the fundamental indicators of breathable air and life quality. In case, companies employ full-electric fleets, the system, in optimizing the total duration of the task, may ensure a longer battery life and a reduction in the recharge costs.

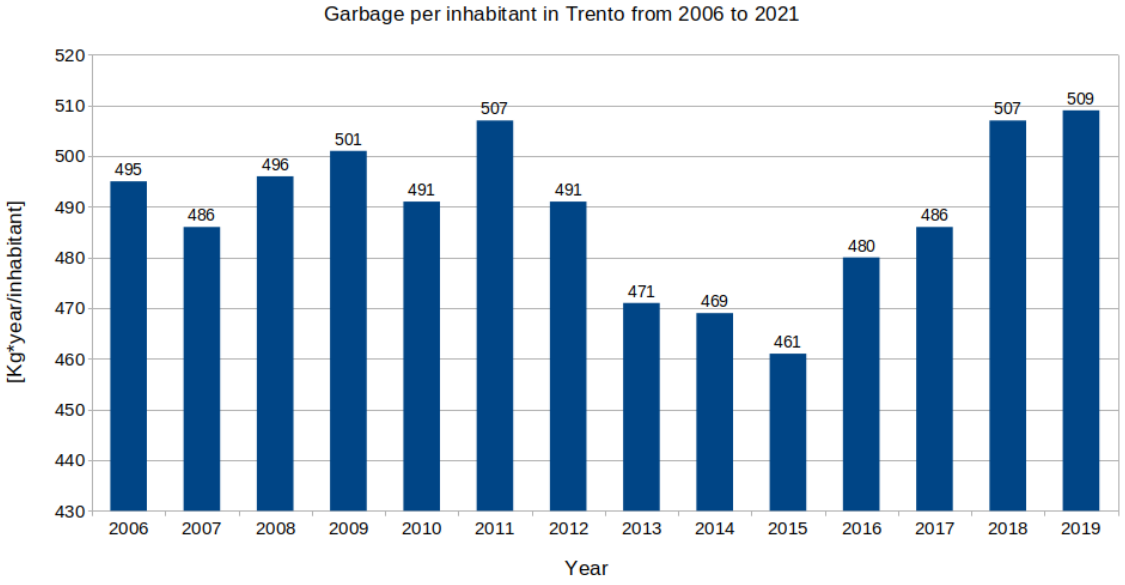


Figure 1.2: Amount of garbage per inhabitant in the province of Trent from 2006 to 2019 [6].

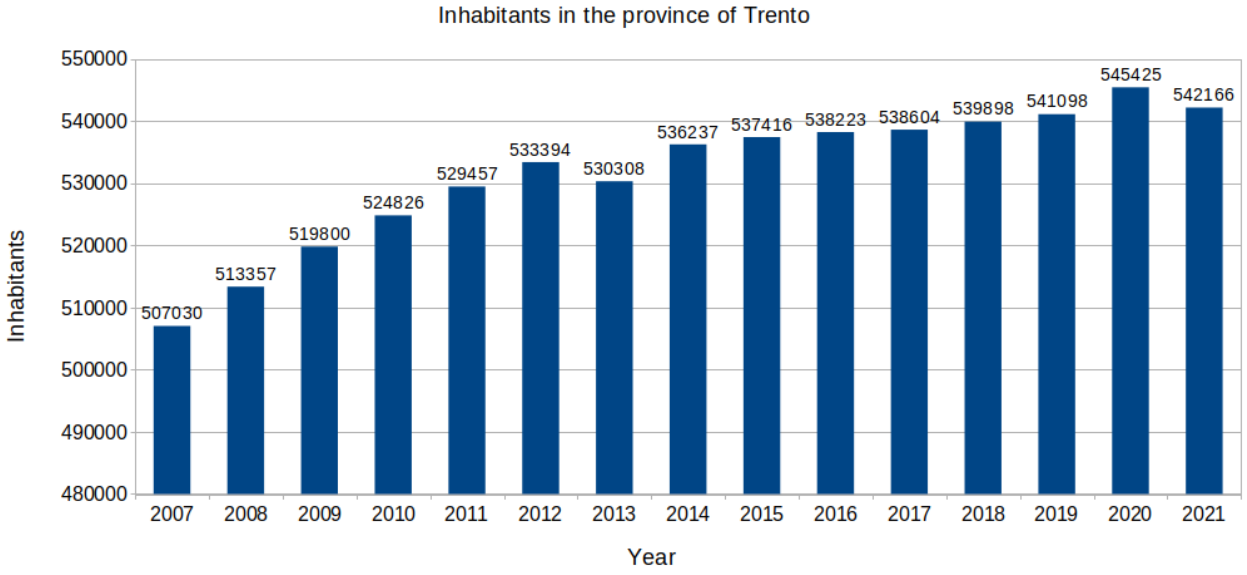


Figure 1.3: Amount of inhabitants in the province of Trent from 2006 to 2021 [3].

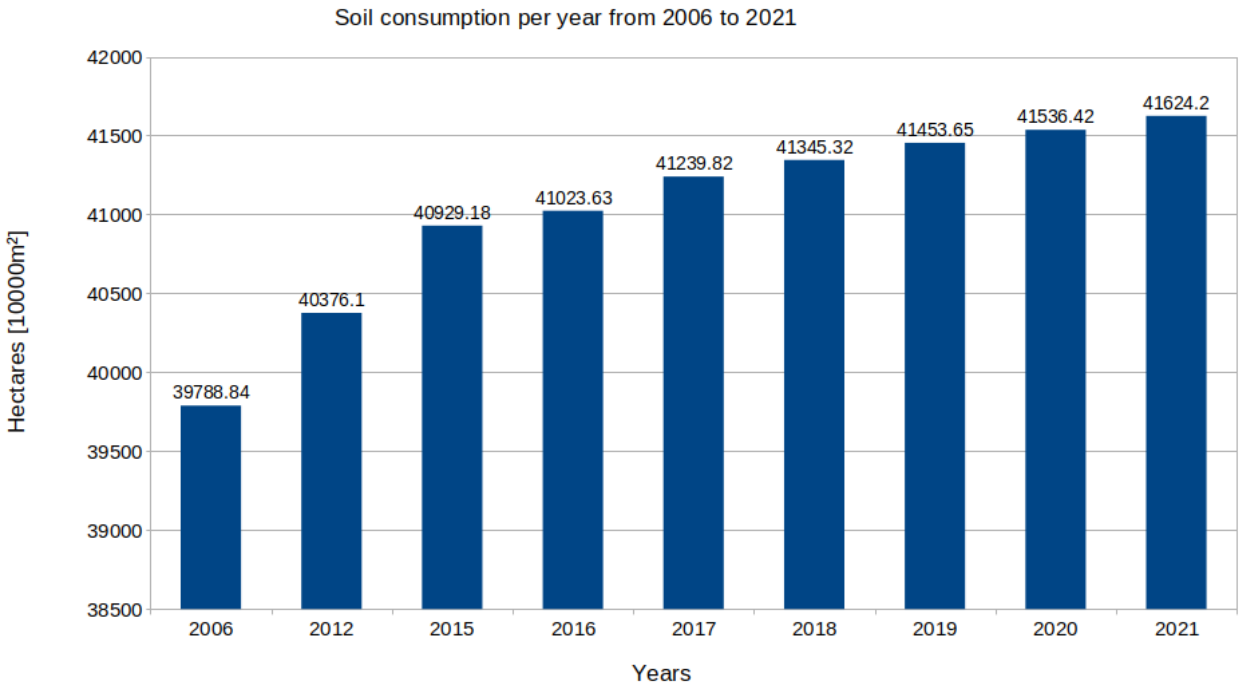


Figure 1.4: Amount of soil consumption, urbanized, in the province of Trent from 2006 to 2021 [4].

1.2 Problem Formulation

The municipal waste collection problem belongs to the family of optimization tasks known to be NP-Hard complex and it is an instance of the Vehicle Routing Problems, VRP, which the well known Travel Salesman Problem, TSP, is part of. In the simplest formulation of this class of problems only one agent and several geographical locations are known. The main task of a solver algorithm, applied to this problems, is to search the sequence of actions which allows the actor in use to visit all the available locations, without moving through the same place twice, or at least minimizing the travel time. Depending on which parameters are considered in the search task and how precisely it has been modelled, it is possible to identify different declination of the VRP. For example, by taking into account the capacity of the vehicles it becames a Capacitated VRP; if there are different ranges of time in which the nodes are available to being visited then the problem is identified as a Time Windowed VRP; if both are considered it is the case of the Capacitated VRP with Time Windows.

In this work we decided to tackle the municipal waste management problem as a Capacitated VRP and to express it as the task of finding an efficient sequence of actions, in our case a combination of movements, collection and discharge operations, performed by a finite set of agents, which allow to clean an arbitrary large geographical area, where the human processes have generated wastes. The garbage produced is assumed to be directly available from the road network supplying the territory. To complete the task, the waste must be collected from the environment and transferred in facilities having at their disposal the necessary equipment and procedures to ensure its disposal in total safety and in full accordance with the laws. Once the collection and discharge operations are completed, all agents involved, as the last step, must reach their depots before the end of their work shifts.

The waste management problem is limited both in execution time, being the agents employed daily for a finite amount of time, and in the quantity of resources available to complete the task. With the term “resources” we identify the number of agents able to accomplish the cleaning operation, the structures available for the disposal of the materials and the depots in which the actors start and end their work shifts. These elements are the main factors influencing the efficiency of the management system. Agents having their depots far from the collection points will yield to solutions performing worse than in the case of actors being located closer. Moreover, having a work shift lasting too less than the optimal amount required to collect the locations may lead to unsolvable problem instances.

In the same way, the amount of vehicles in use to complete the task and the number of disposal sites influence how much waste can be collected and destroyed at each time step from the environment and impact the throughput of the system itself. A solution is found by constraining the behaviour of the agents and deciding how the garbage harvesting operation and the waste shipping to the existing disposal centers must be carried out. Agents employment causes the introduction, in the problem, of bounds which depend on the number of actors available, the amount of hours that any of them can work and the time span at disposal to complete the task itself. A broad view of the issue to solve can be formulated as follows: at the beginning of the work day all agents are in their garages, or depots, distributed across the territory; as time flows by, each actor moves and collects garbage avoiding to overfill, then, at the most appropriate time, they will move towards the disposal facilities to discharge and restore their carrying capacity; once empty, each agent may chose to keep going with the collection and disposal operations or come back to the initial garage ending the work shift and preparing for the next one.

Formally, it is possible to address the problem as having a graph $G(V,\ E)$ where V is the set of vertices available and E is the set of edges linking the vertices together. In our case, V represents the nodes in which the garbage is found and can be collected, the depots and the facilities available. Complementary, E is the set identifying the possible roads connecting a vertex to any other. Exploiting this graph, our framework has to provide a finite set of actions which allows to clean the environment and have the waste delivered to the disposal sites before the end of the available time. The solution is extracted by combining matrices describing the main operations, carried out by the agents, and linking the actors to the actions they undertook in a certain combination of time step and location. These matrices represent the collection and discharge events, and the path of the vehicles in use. All of them are filled up by the solver and satisfy the defined constraints. The optimal solution is identified applying a cost function to the resulting combination of the matrices.

2 Relevant Literature

Even if the municipal waste management problem is of such relevance in the modern society it is not new to the literature, indeed one of its first formulations dates back to 1974 [1]. Following are some works which inspired the design of the developed framework.
First of all, the garbage collection problem may be addressed as an instance of the Vehicle Routing Problem or of the Capacitated Arc Routing Problem which many authors tried to solve exploiting Mixed Integer Programming, Constraint Programming, genetic algorithms and even brand new approaches.

In 2006, Kim et al. [7] proposed to solve the garbage collection task by employing an insertion algorithm which extends Solomon's one [14]. The model that was developed accounted for time windows in servicing the nodes and lunch breaks for the drivers. Furthermore, they took advantage of clustering algorithms to retrieve solutions having compact routes for the vehicles. The environment in use is supposed to have a depot in which the agents start their work, one disposal location in which to empty and several demand nodes where garbage must be collected. The researchers divided those last nodes into three categories depending on the amount of garbage that could be produced.
In 2015, Rodrigues et al. [13] proposed a model similar to the one in this work, with multiple agents and landfill, addressing the waste management problem in one city of Portugal. The model is then translated to a mathematical one and solved with the CPLEX solver.

In 2019, Tirkolaee et al. [18] proposed a solution to this problem by employing the Simulated Annealing algorithm to provide an initial solution to the input use case. In their work the environment was assumed to have one depot area in which the agents starts and ends their journeys, one disposal site and several demand nodes in which the garbage could be collected. It is also enforced that when agents reached their maximum capacity they needed to reach the disposal site to empty. Furthermore it accounts for time constraint by exploiting soft and hard limits on the time windows. The study showed that a genetic algorithm as Simulated Annealing provided a satisfactory solution in less time than an exact solver like CPLEX. Further in 2021, Tirkolaee et al. [17] improved their approach by developing an hybrid algorithm exploiting Multi-Objective Simulated Annealing to retrieve some initial solutions. On top of those they then called an Invasive Weed Optimization Algorithm which refined the solutions until the closest one to the optimal is found.
A similar approach with genetic algorithms has been employed in 2006 by Król et al. [8] where a novel approach for collecting the electrical equipment waste was proposed. In their model they decided to exploit genetic algorithms to both define an heuristic for the collection task and to compute the amount of agents required to accomplish it.

Another case of study in Portugal was addressed by Texeira et al. [16] where the planning process had to account for a periodic management of the waste. They modelled different kinds of garbage to be collected weekly, and provided a division in zones for the territory.
In 2022, Tee et al. [15] proposed a multi goal programming model to provide a solution to the VRP problem using as objective functions the duration of the task, the environmental impact, the expenses and public health. The developed model assumes three different kind of areas in which the agents can perform their work: the depot in which they start and end their job activities, the disposal places in which garbage is removed from the agents and the demand nodes in which the garbage can be collected. It is also assumed that each demand node has a specific time window in which collection can happen and that the collection task is instantaneous. The resulting model is a mixed-integer non-linear programming one.

Rabbani et al. [12] proposed a model for the management of wastes coming from the automotive sector. They employed multi-criteria decision making techniques and optimization modelling to tackle the issue from the perspective of economics and environmental and social sustainability. A linear model is developed in order to obtain the optimal minimum of the cost function solving it through the CPLEX solver. More recently, an attempt to address the waste management problem was made with a full electric fleet of agents [2] through a Mixed Integer Problem representation and an adaptive variable neighbourhood search to solve the issue more efficiently.

3 Design

3.1 Design

In designing the framework for municipal waste management optimization it has been taken into account the human impact over the exposure of the waste and over the traffic build up in the road network. Secondly, the streets servicing the territory and the amount of agents available together with the time that each of them can be employed have been considered.
To understand the environment available, it has been decided to mine knowledge about its geographical representation exploiting the open source database of OpenStreetMap [11]. In this database all the geographical elements of interest are represented as nodes associated to longitude and latitude coordinates and characterized through a set of tags.

For the purposes of this work, the system focuses on the entities representing the roads and buildings in the territory. In addition to the specified nodes, also the relations between them are listed, allowing to know how an element interacts with the environment by following specific links inside the relations it belongs to. In case of roads, these relations allow to understand which buildings they service, how they intersect and direct traffic, while the tags allow to understand the streets width and the traffic flows through them. In case of buildings, the relations available, express how the nodes are linked together to portrait their boundaries represented by the walls delimiting them, while the tags may specify the purposes of the structure and if it is inhabited or not.

Having this data at disposal and after extracting the information about the roads and buildings available, this knowledge is further processed to obtain a graph structure in which the nodes represent the available buildings and the connections among them identify how the neighbours can be reached through the road network. Thanks to this transformation it is now possible to define a motion model of the agents which relates how certain nodes are reachable from some locations at each time step and if the actors can move towards them. In this manner it is ensured that the available vehicles will use the road network in the same way that any user would. The behaviour of the actors is controlled by exploiting the motion model of the agents and embedding it inside a Constraint Programming representation of the task which is linked to a cost function depending on the time, location of the actors and the amount of garbage left in the environment.
Given this implementation it is now possible to get an exact solution to the problem by relying on a mathematical formulations which bounds the possible operations carried out by the agents.

To solve this kind of optimization problem it has been employed the CPLEX solver developed by IBM which provides a solution by translating the mathematical model into a matrix having as rows the constraints and as columns the decision variables.
A workflow graph comprehensive of all the steps executed by the framework is visible in Figure 3.1.
As observable from the figure, the framework needs only three kinds of data to provide a solution:

1. Geographical Data: It is the knowledge about the environment on a geographical level.
   Specifically, longitude and latitude and relations of each object of interest in the territory. Also the relations among the elements are required to infer the road network structure and the buildings encumbrance.

2. Agents Data: It is the database containing information about the vehicles features such as the maximum load capacity and maximum number of hours of work. Information about the crew are not needed, being a squad planning algorithm not yet supported.

3. Depots/Disposal Data: These are custom information that the user is required to provide to uniquely identify which building nodes, among the ones extracted from the geographical data, have the role of depots for the agents or disposal facilities.

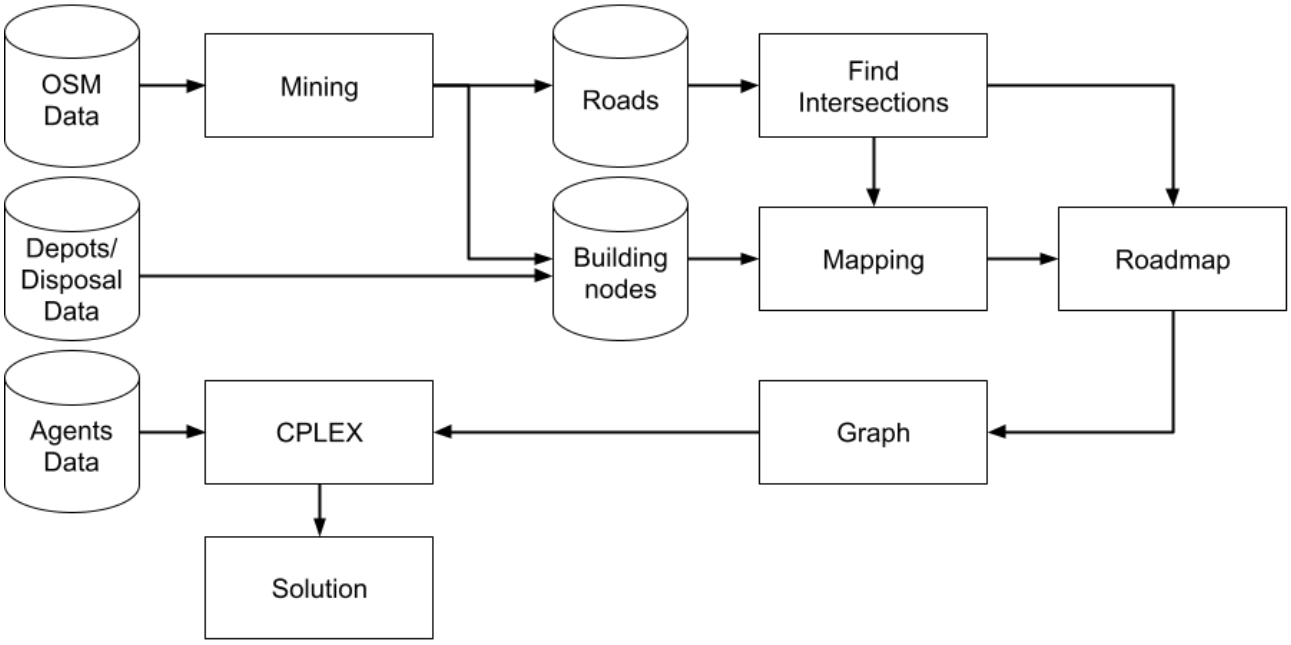


Figure 3.1: Workflow of the optimization framework from raw data to a solution to the waste management task.

3.2 Model

In modelling the waste management task it has been followed the formulation provided in Section 1.2, but slightly changed to fit the hypothetical processes of any company working in the waste collection sector. Specifically, it was said that the agents depart from their depot, or garage, and move to collect those nodes believed to be dirty, due to the presence of garbage, to then reach the disposal sites when full or at the end of the task.
This last step is quite unrealistic, since it would be too much expensive for a company to move the workforce to the disposal sites each time, especially considering that these sites are located far from the living areas, introducing latency in the process. It would be more efficient to force the agents collecting the garbage to discharge their load in an intermediate facility that is entrusted for the transportation to the disposal sites. These locations would have a specific vehicle, called truck, with loading capacity highly exceeding the one of the normal agents. In this way the collection and discharge operations of the agents can be executed in areas nearby one another dropping the time requested for going back and forth to the incinerators, being this last one accomplished by others agent which takes more time to get filled, dropping the amount of time that the incinerators are visited.
Specifically, each normal agent will be allowed to discharge directly into a truck depot thus to simulate the action of directly loading the truck agent. Furthermore, having at disposal such capacious agents allows to select different location in which they can be found, thus providing more flexibility to the agents for their discharge operations. Then, once the collection task is accomplished and all nodes are clean or simply when it is the most suitable time, the trucks will leave their depots and move towards the disposal sites.

As observable in Figure 3.2 the agents in depots 1 and 2 will supervise the collection task of the nodes $A, B,$ and C and then move to discharge in depot 3. Once the agents end their empty operations the truck will move the waste to the disposal facility accomplishing the last step of the waste management task. Normal agents can go through a depot or a disposal node, but no discharge, nor collection operations can be performed, exception made for the empty procedure available only in the truck depots. Symmetrically, the trucks cannot load waste from agents which are in any other node than their garage, thus to avoid mid path loading operation which are highly unrealistic to occur. The municipal waste collection problem has been modelled in such a manner that a solution represents a time-dependent sequence of locations and actions per each agent involved in the task.

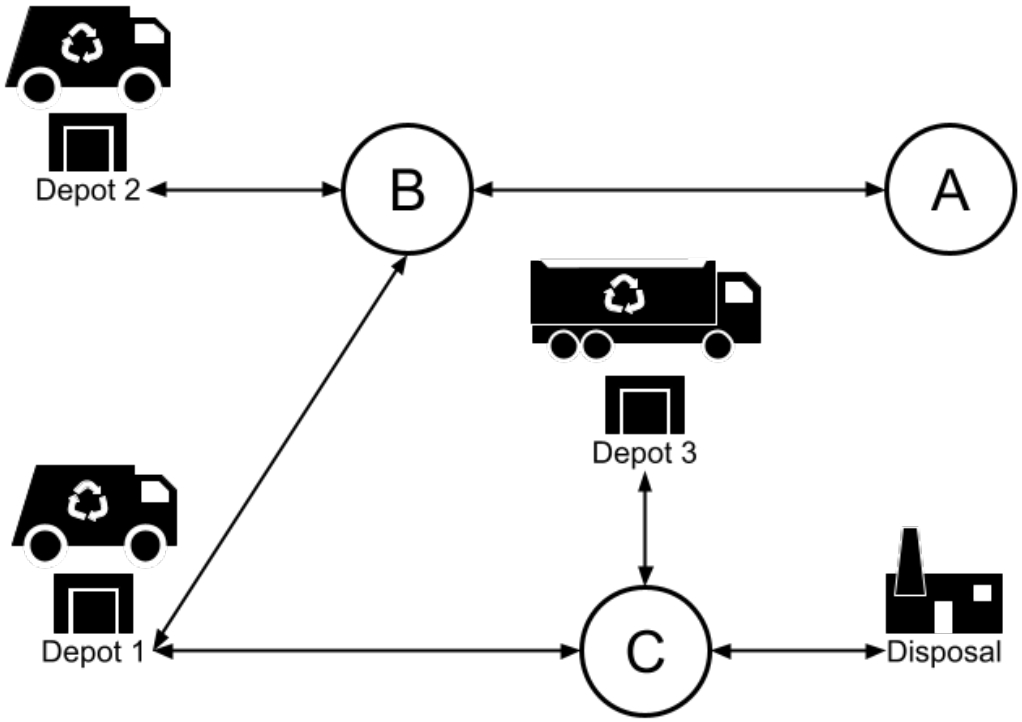


Figure 3.2: Full model of the waste management task.

Under this formulation it is possible to embed the working time of each agent and their motion model directly inside the decision process of the solver in use. This means that at each time step we know where an agent is and which action is performing.
To allow for a well-timed action execution and a detailed overview of the system states, the time at disposal has been split into steps with a granularity in the order of minutes, meaning that an hour of work equals to sixty time steps.
To achieve this representation a set of three dimensional matrices has been employed to represent the aspects of motion, collection and discharge events. Moreover, the geographical area under analysis has been represented as a directed graph representing the collection points, depots and disposal facilities. It has been assumed that the municipal waste is being produced by humans in their living areas and, as such, it has been inferred that the garbage location could be mapped outside of the buildings directly over the road network as any door-to-door garbage collection task would be organized.
Under these assumptions it has been decided to exploit the projection of the buildings over the nearest road as nodes of the graph and estimate the position of the waste to collect directly on the mapped location.
Being the elements constituting the graph representations of real buildings, it is necessary to classify them into categories based on the set of operations that any agent can perform on them. For the purposes of the framework three different types of nodes have been identified:

1. Garage: Garage, or depot, nodes represent the places in which the agents await to start their shift and to which they must return at the end. No waste collection or discharge operations are allowed on these nodes. The garages are further split into garages for normal agents and garages for trucks. The first type can contain multiple agents, while the second one can contain only one truck agent and no normal ones.

2. Harvest: Harvest nodes represent the locations in which waste is made available for the collection operation. If one node of this kind is being collected, then only one agent is allowed to perform such action and once cleaned it stays in this state until the next work day. No collaboration between agents is allowed to avoid partial collection of the nodes.

3. Incinerator: Incinerator nodes, or disposal sites, represent the locations in which there are facilities able to remove the garbage from the environment and proceed to its recycling or destruction.

As previously stated, the nodes are connected in a directed manner aiming to emulate the roads topology of the territory and embedding the presence of one-way traffic streets. Moving from one node to another requires a non constant amount of time which depends by the starting and ending locations, ensuring the correct representation of the network capabilities. Being able to estimate the traffic throughout the day, it would be possible to transform these values into vectors and associate each travel action to a specific amount of steps depending by the time of the day in which the actors move, this ensures that the human dynamics unfolding during the day are considered through the planning process, since they may cause delays in the service.

The agents that can be used by the framework are the vehicles available for the collection task, comprehensive of the humans onboard, however the crew has not been modelled, requiring a dedicated planning process over information specific to the working realities employing the system.
We assume that each agent can work for a specific amount of time per day and no overtime is allowed. Lunch breaks or any other temporary interruption of the service required by law are not embed into the model due to complexity issues.

The decision variables of the model, which ensure the correct management of the agents, are the maximum number of workable hours, expressed in minutes, the maximum amount of garbage that can be carried and the quantity of waste carried at each specific time step, both expressed in kilograms. Depending by the capacity and operations available, the agents have been catalogued into two categories with similar, but different behaviours:

1. Normal Agents: This class of agents represents vehicles which carry small amount of garbage. They can drive through all the roads available and are obliged to discharge the waste only at the locations of truck depots, but passing by incinerators or other depots is allowed. Differently from [15] the collection operation is assumed to last for a certain amount of time which depends by the amount of garbage in the nodes and should be specified by the user.

2. Truck: Truck agents have bigger dimensions with respect to normal ones and even a huge carry capacity. Due to their encumbrance only a subset of roads on the territory is well suited for their movements. They can empty only in incinerator nodes and the discharge operation takes a certain amount of time which depends on the intrinsic characteristics of the vehicles.

As it is possible to infer so far, differently by all the cited papers in Chapter 2, the model here provided supports the existence of several depots, intermediate facilities and disposal sites, even if the tests have been carried out on the same assumptions made by the other researchers, having one instance of each of these specific nodes.

One last remark must be made about the computational time required to find a solution. Differently than other framework employed in industrial scenarios to which high reactivity is the main requirement to avoid damages to the work flow, in the waste management case it is possible to have computational times in the order of hours, due to the garbage collection task being carried out usually between 6AM to 8PM leaving the 10 hours before the next working day available for computing.
Obviously, if the system were to be faster, any company would benefit from the lower computational times.

4 Implementation

In this chapter we explain the different phases that the framework goes through before providing a solution to the user. These operations, shown in Figure 3.1 can be summed up in the following steps:

1. Process OpenStreetMap data.
2. Build a roadmap comprehensive of the buildings.
3. Translate the roadmap into a graph-like representation.
4. Build and solve the CPLEX model.

4.1 OpenStreetMap Data Processing

As said in Section 3.1 the system mines knowledge directly from the OpenStreetMap (OSM) database which is free for use. The chunks of data describing the area of interest for the user must be directly provided in input. The amount of information available in such environment is excessive for the roadmap purposes and, as such, must be filtered out. In the database each geographical object features are described by tags linked to the basic OpenStreetMap data structures being:

- Nodes: which contains latitude, longitude and the id of the elements they represent.
- Ways: which represent objects being, practically, lines on the ground.
- Relations: which are used to describe logical or geographical relations between nodes.

By parsing these three categories and focusing on the different tags available, it is possible to retrieve the information required for the waste management task.
The first chunk of data to collect from OSM input file is the road network structure. To extract it, all the ways objects are parsed and the elements having in their tags the fields “building” or “highway” are retained, being those representative of the buildings walls and roads on the territory, respectively. The ways identified as “building” will be directly saved for further processing, while the ways having the “highway” tag will be subjected to further refinement.
Specifically, the “highway” tag is added to almost any kind of road, but for the task of the framework it is compulsory to filter out roads which are not suitable for vehicle movements. For example, pedestrian areas, hiking tracks or any other path which complicates any four wheeled vehicle movement have been ruled out. Following is a comprehensive list of specifiers for the ”highway” tag that causes the discarding of the street from the road network that will be used by the system.

- “path”: It is the tag which addresses generic roads available to pedestrians, bicycles and other small vehicles.
- “footway”: It is the tag representing those roads which are exclusively used by pedestrians.
- “steps”: It is the tag identifying stairs on paths and footways.
- “pedestrian”: It is the tag in use for large pedestrian areas, usually vehicles cannot go through this places if not for certain times of the day.
- “track”: It is the tag used to address minor roads used mainly for agriculture and which are not considered part of the usual road network.

(a) Aerial view of OSM data. (b) Data of interest.

Figure 4.1: Visual representation of the data contained in the OSM database. Not all buildings have been highlighted due to visualization needs.

Once the selection process ends, the system has at its disposal the location of the buildings walls and the roads itinerary which with high probability allows to four-wheeled vehicles to move with ease.

Due to the employment of trucks agents in the planning, which have a peculiar encumbrance, the tag "maxwidth", in the ways objects, is used to determine whether or not the roads are large enough for them to pass through. The default value of this parameter is zero, so for any value different than that the road is considered too narrow and a specific flag is saved together with the roads information. These specifiers will then be exploited during the model and graph creation to account for a correct motion behaviour of the agents.

4.2 Roadmap Creation

Having at disposal the knowledge about the nodes constituting the safe roads and those which represents the buildings, it is possible to build the roadmap that will be used as a reference in making the graph representation of the environment. The roadmap creation process consists into two phases.

The first one is the identification of the junctions, also known as crossings points, between roads. In practice it is the task of finding those points which are common to two or more ways meaning that from that specific location other roads can be reached.
To perform this operation, the roads will be scanned individually and each pair of nodes composing them will be checked for intersection with the nodes pairs belonging to other roads. If this process is executed by blindly comparing all the possible nodes in a road with all the others, the amount of computation cycles will grow exponentially with respect to the number of roads.

In an attempt to avoid useless runs, and drop the computation time of this step, it has been decided to enclose each road into a bounding box and proceed to the comparison of nodes only in those cases where the bounding boxes of two arbitrary roads intersects or one is contained by the other. In this manner the algorithm completely rules out all the roads that cannot be in contact in any way.
Working with roads representations, means that the system has at its disposal a two-dimensional set of segments which are connected as a chain.

Being the road completely free to unroll for all the territory it makes sense to approximate it with a bounding box expressing the road boundaries in both dimensions. At this point, it can be inferred that any pair of roads can interact only in three cases: if one ends in the starting point of the other, if one starts in the ending point of the other and if one of the two begins or ends in any intermediate point of the other.

All these cases are derived through the bounding boxes interactions, indeed in the first two cases the bounding boxes will be adjacent or intersecting, and the junction point found will be the one transitioning from one road vertex to the other, while the last case may happen when the bounding boxes intersect or one is fully contained in the other and the junction point will belong to the vertex of one road and to the edge of another.
Once the intersection points are found, those will be addressed as junctions between roads and two specific data structure will be made embedding the knowledge that one of them can be reached by the other.
Then, the junctions will be linked in the two roads which had an intersection creating a cross relation between them. In this way moving along a road will allow the system to detect the junction nodes and decide whether to continue up to the end or change the path the algorithm is working on.

The second step in roadmap making is the mapping of the buildings to the road network available. As already stated, the framework knows where the buildings are, but not their access to the streets. Due to the agents being four-wheeled vehicles and assuming that the humans living in the area will expose their garbage in the nearest road outside their living quarters, it is of fundamental importance to estimate a garbage location on the roads available.
To achieve this, the buildings extracted during the parsing of the OSM database are parsed singularly and for each set of nodes composing the structure, the centroid is computed. After that, each building is enclosed by a ring which has radius equal to the maximum distance between the centroid and the nodes it is made up.

Then the intersections between the ring and the bounding boxes of the roads are checked, if an intersection is found then a second intersection query is performed between the ring and the nodes composing the road. Whenever the intersection exists the location of the intersection point is saved into a list and the process repeated until all the roads have been parsed. In case no bounding box intersecting with the ring or no intersection between nodes and the ring is found, the radius of the ring is increased and the process repeated.

At the end the system will have a list of intersection points for the same building, but only the one nearest to the centroid will be kept as an approximation of the waste location on the road network and a new node will be linked to the intersecting road representing this place.
At this point the framework has completely built the roadmap and can proceed in making the graph which will be used for modelling the environment. In the following figures it is possible to observe how the mined data shown in Figure 4.2 have been mapped to achieve the roadmap represented in Figure 4.3. The buildings enumeration are omitted for visualization purposes.

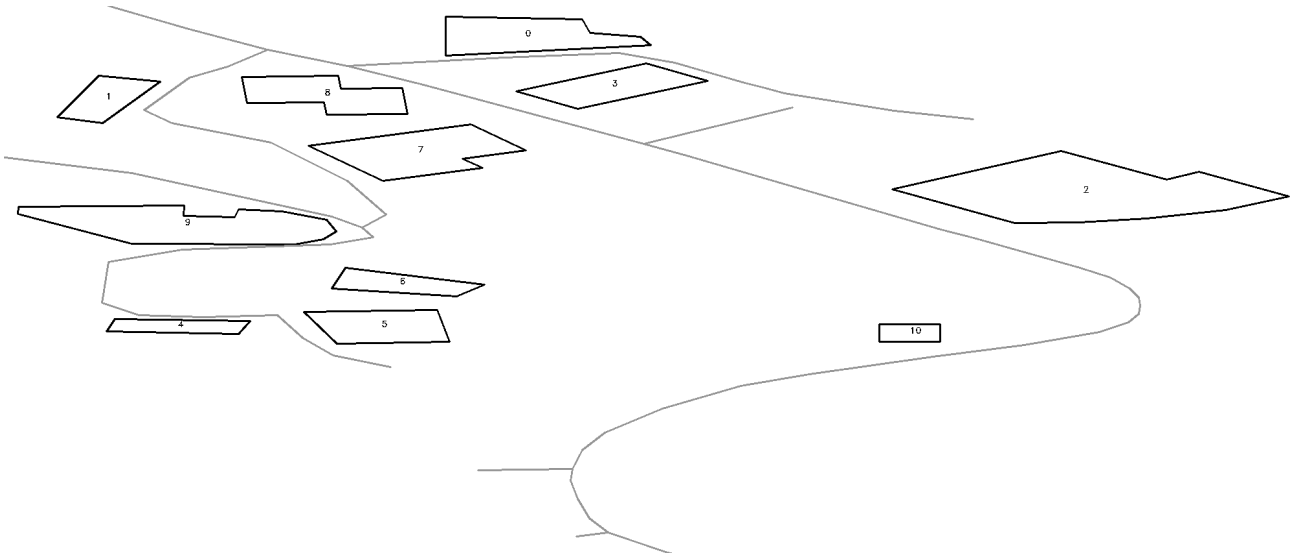


Figure 4.2: Portion of the extracted data from OSM database.

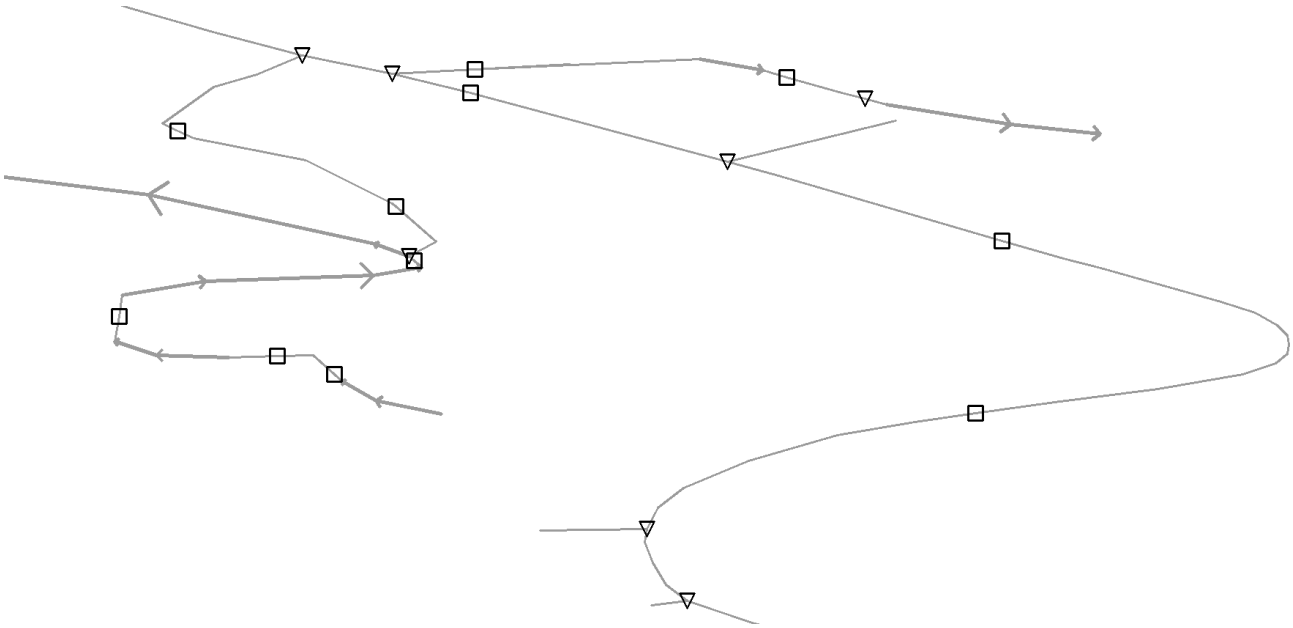


Figure 4.3: roadmap built with the extracted data. It is possible to observe the mapped buildings (squares), the road junctions found (triangles) and the one-way only roads (bold arrowed lines).

4.3 Graph Creation

The graph creation process is carried out with the aim of portraying the buildings and how each of them is connected to the neighbouring ones through graph edges. To this purpose the nodes in the graph will represent the mapped buildings in the roadmap, while the edges through which they are linked represent the paths across the road network servicing them. The graph making operation works by parsing all roads in the roadmap and for the nodes constituting them it undertake specific actions depending by their types.

Specifically, each time a building node is found, if it has not been discovered yet, the algorithm will create a node instance inside the graph representation and will start recording all the nodes that will be observed after. In such a manner the building discovery and the connection making operations are carried out with the same graph scanning step.
If, while the nodes are being recorded, another building is found, an edge to the new element is made describing the nodes needed to be traversed before reaching it and the list of recorded nodes gets reset starting anew with the last building found. The path exploration process continues until the end of the road is reached.

If a junction node were to be observed while searching for a building then a recursive algorithm is called to execute the search task over the roads that are reachable from the intersection point. This recursive call will build a new set of recorded nodes describing the path over the reached road and relate them to the ones found before the call. In this manner, when a building is observed on the connected roads, the recursive algorithm stops and updates the reachable destination of the last building with the newly discovered one.

The nodes of the path describing this connection will be the composition of the recorded ones from before the recursive call, plus the ones after the call. In this way it is ensured that the paths are consistent with the available road network. The recursive algorithm allows to find the paths connecting buildings on completely different roads passing through several junctions, but, at the same time, this procedure may retrieve edges with different lengths connecting the same objects.
For this reason, once the graph have been fully built, the links found will be parsed again and whenever multiple streets to the same destinations are identified, only the one covering the smallest distance will be preserved while the others deleted.

In building the edges, each connection is assumed to be bi-directional, meaning that the starting point and destination can be reached even if the roles were inverted. Practically, the road network not always allows for such links, and the chunks of the roads being one way only have been found during the OSM data parsing.
When adding links to the building, the system will exploit this knowledge allowing both ways travel if all the nodes constituting the link do not have been flagged as one-way only. Obviously, if one of these node exists in a link towards a building then the the entire path has to be handled as one way since it is impossible to traverse the edge backwards and reach the starting point.
At the end of this process the built graph is a directed one, where the majority of connections are bi-directional.
An example of such graph is provided in Figure4.4 which shows the result of the algorithm described so far to the data shown in Figure 4.3.

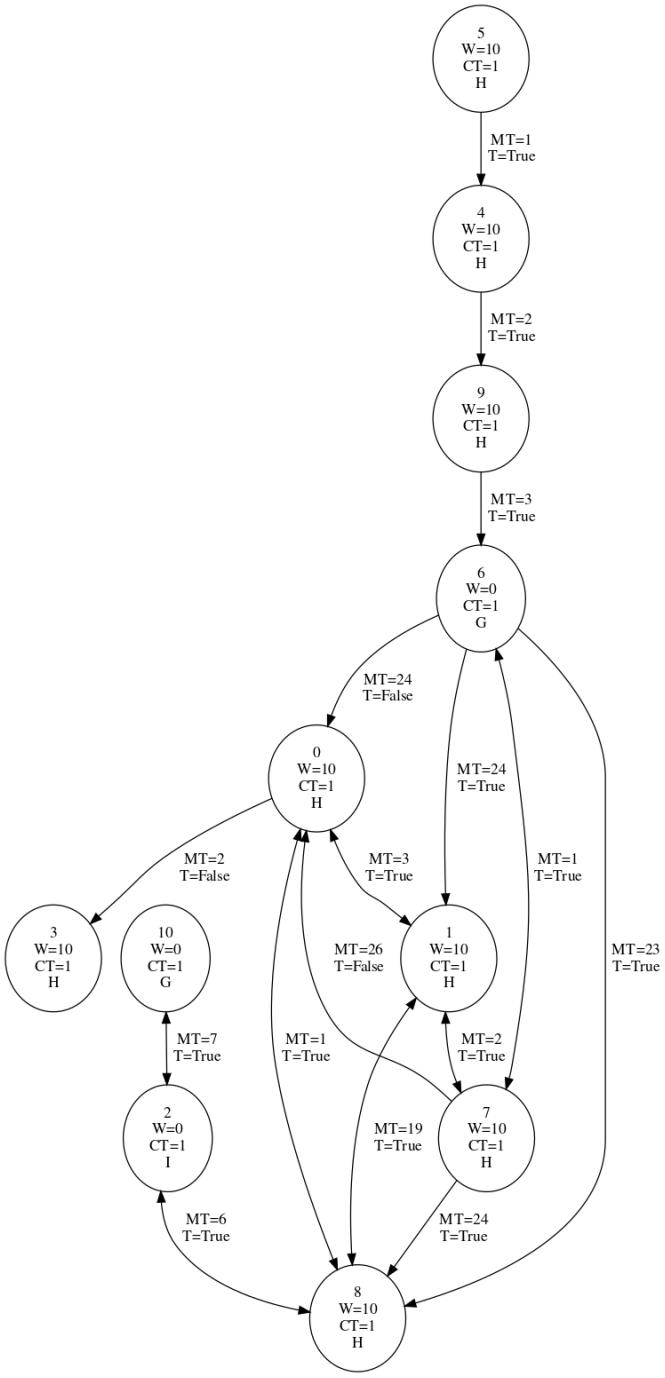


Figure 4.4: Graph generated from a roadmap instance.

In the shown graph, it is possible to notice some information nearby the edges, specifically the labels "MT" and "T". Those are the acronyms indicating "Movement Time" and "Truck" which are used to specify, the amount of time steps needed to reach a pair of nodes and if the link can be traversed by the truck agents, respectively. This information is extracted during the link creation phase.

The movement time is determined by computing the distance between successive pairs of nodes in the link and dividing it by the speed limit on the way, while the truck flag is chosen to not allow the travel for the trucks if there is at least one node belonging to a road which width has been identified as too narrow for a truck agent.

A remark has to be made about the computation of the distance between nodes. Such operation is not performed exploiting the euclidean distance but the distance of two points on the surface of a sphere through the Haversine Formula 4.1.

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cdot \cos\phi_2 \cdot \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{4.1}$$

Where r is the earth radius, ϕ_1, ϕ_2, λ_1 and λ_2 are the latitude and longitude, expressed in radians, of the first and second node, respectively.

As observable by the graph in Figure 4.4 the one way roads generate chains of nodes which can only be visited in one direction, while the other links stay bi-directional, thus potentially leading to dead ends causing the unfeasibility of the problem. Furthermore, no cyclical relations are tolerated, meaning that any path starting from a node and leading back to itself is directly discarded being unnecessary. Indeed, allowing the agents to move through a road that leads to the same point would only result into a more complex model without providing any advantages.

One last remark has to be made about the graph edges creation. Specifically, anyone reading this work may object the choice of connecting each building to only those following or preceding it. In real life scenarios, having a generic road, each building belonging to it should be connected to all the other ones on the same road . The system directly ignores this setting for two main reasons: the first one is to reduce the amount of constraints bounding the agents motion, the other regards the possible solutions allowed. Allowing the representation that connects mutually all the nodes on the same two ways roads, means to permit solutions in which any agent can perform U turns at any point of the collection task or even partially collect portions of roads which are not interrupted by any junctions which could justify such behaviour. In conclusion, the choices of not connecting all the nodes of a road one another and of avoiding multiple paths between the same pair of nodes are enforced to obtain, during the model solving phase, a search tree as compact as possible to drop the computational time required.

A visual representation of how the search tree could be in both cases is presented in Figure 4.5, which shows the hypothetical search trees over a time span of three steps starting in node 0. The cyclic branches allowing the agents to stay in the nodes are omitted for visualization purposes.

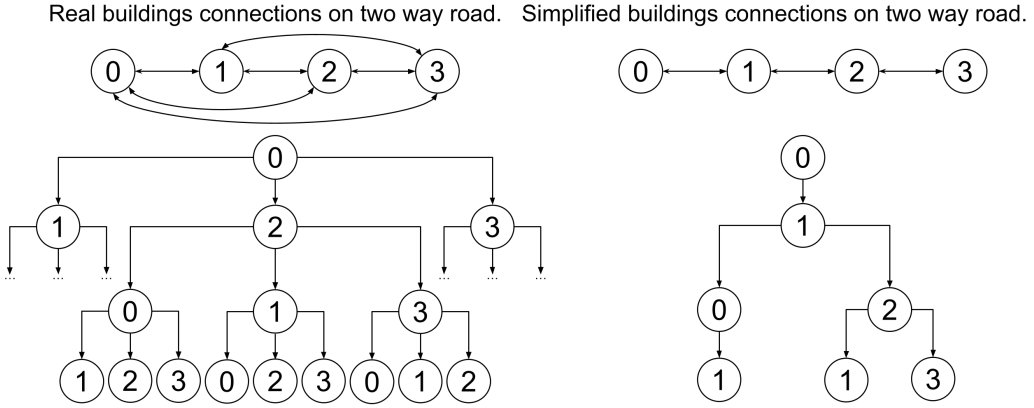


Figure 4.5: Hypothetical search tree for mutual and neighbouring nodes connections. Those express the possible locations moving from node 0 for three time step in the future.

4.4 Considerations Before Modelling

Now that the framework has built the graph, the last steps left to perform, are the creation of the mathematical model of the environment, its linking to a cost function and solving it using the CPLEX solver. However some considerations about the model and the parameters in use must be done.

4.4.1 Time

As anticipated in Section 3.2, the solution to the municipal waste collection problem is a finite sequence of steps which results into an end state having all the nodes in the graph cleaned and the empty agents in their starting garages.
Being the time necessary to complete the task the most important measure of efficiency, the solutions found are expressed using a time granularity of one minute, since we believe that it is the most suited level of resolution to capture step by step the actions decided by the solver for the agents.
Formally, it is possible to define T as the vector containing all the time steps available for completing the task and t as a specific time step in the vector. Consequently to address the total number of elements contained in the vector it can be used the modulo operator applied as $|T|$. All elements inside T are positive quantities in the range $[0, \inf)$.

In Section 3.2 was said that the employment of agents introduced limits both for the amount of garbage that could be collected and for the maximum working hours of each agent. Concerning this last point, it has been decided to reduce the full time span T to the maximum time that any available agent can work, being impossible to solve the problem if no agent is able to remain operative longer. This reductions takes effect directly in the model representation associating the agents to vectors of time having length exactly equal to their amount of working hours. At the same time, the agent working the longest is the one dictating the maximum time span available for task completion.
Thanks to this approximation the system reduces the number of decision parameters by implicitly exploiting the working time of the agents. So, in modelling the problem it will be used T_{Aa} to address the maximum time in $|T|$ that an arbitrary agent a can work.
An objection that can rise about this reduction may be that there are cases in which the task can be solved in the amount of time provided, but not in the maximum workable time of the agents requiring two working shifts.

In those cases the algorithm could be adapted in two different ways to tackle the issue.
The first should be to double the working time of the agents, thus the second shift is implicitly embed, and the actions are planned ahead. To not introduce errors in the time needed to move over the road network, a modifier can be summed to the motion time directly in the constraints ruling the second shift.
The second strategy is to increase the amount of working agents, in this way the workload will be split and a solution could be found without exceeding the maximum working time steps. In this latter case the end-user will not lose time for fruitless computations being the model set in such a manner that if the problem is unsolvable it will stop the computations immediately.

4.4.2 Nodes

The nodes constituting the graph have been represented using an identifier and collected into a vector for ease of use. Each element in this vector has been portrayed in the mathematical model through a vector of boolean decision variables with length equal to the number of available nodes.
The strategy followed is the one-hot encoding. In this manner the nodes being visited by an agent at a certain time step, will be identified by setting the decision variable corresponding to their index in the nodes vector to 1, while all other variables will be set to 0.
The vector containing the graph nodes has also been ordered in such a manner that the garage ones are the first, while the incinerators, or disposal sites, are the last, leaving the collection points in between. Specifically, the depots have been further ordered having the normal agents garages come first the truck agents ones.

Formally, it can be stated that if N is the ordered vector containing the available graph nodes, then h can be used as the index addressing a specific node in the vector.

For simplicity of explanation we will use the variables NG, NT and NI to identify the subsets of the vector containing the elements associated to the garages of the agents, the trucks available, and incinerator facilities respectively.
Even tough the model does not use this parameter, the nodes in which garbage is available for collection are called harvest nodes and are represented by the subset HN.
All the indexes related to the nodes vector, including its subset, have values in the interval $[0, \text{inf})$.

4.4.3 Nodes Connection

The connection among the nodes could have been easily implemented through a connection matrix, but this would imply the management of an object which grows quadratically with the number of nodes in the graph available. To avoid major memory consumption and lowering the time needed to parse the data contained in the connection matrix, each node in the graph has a list of buildings reachable from them and the time needed to traverse the connection, which can be obtained rapidly exploiting hash maps.
Thanks to this representation it has been possible to reduce the maximum number of parameters representing the connections from $|N|^2$ to $|N| \times max\_conn(N)$ where $max\_conn(N)$ identifies the maximum number of connections for any node in the graph.

4.4.4 Agents

The agents involved must be provided in input by the user and will be represented using a specific data structure. In this way, each agent contains the maximum amount of its working hours (in minutes), the maximum weight (in kilograms) of garbage that it can carry and a vector of decision variables expressing the amount of garbage loaded inside the agent at each time step.
All these parameters are used in the algorithm through integer decision variables which simplify the solver computations and account for a certain degree of safety introduced by overestimating the amount of garbage to collect through a rounding operation translating it into a discrete amount.
Lastly, the vector representing the agents have been ordered in such a manner that the normal agents come first with respect to the truck ones.

Nodes	One-Hot
A	[1, 0, 0, 0]
B	[0, 1, 0, 0]
C	[0, 0, 1, 0]
D	[0, 0, 0, 1]

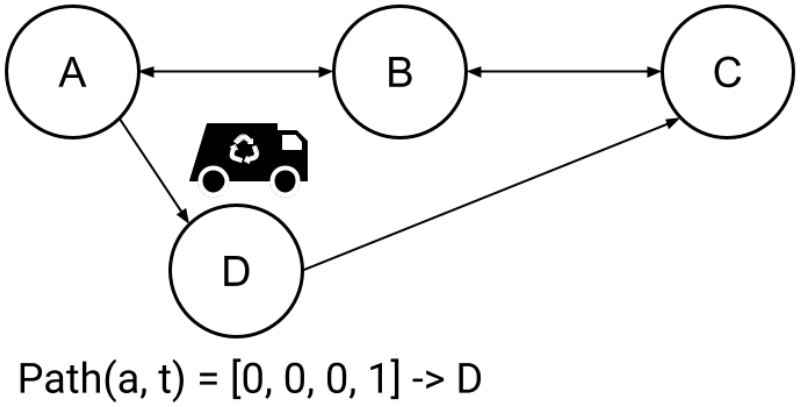


Figure 4.6: Visual representation of one-hot encoding strategy.

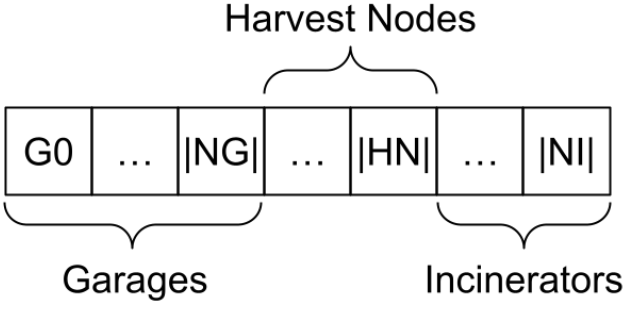


Figure 4.7: Visual representation of the ordered vector for the graph nodes.

4.4.5 Garbage in Nodes

The garbage located in each node inside the graph throughout the available time steps has been represented exploiting a two dimensional matrix of decision variables which rows are the time steps at disposal and columns are the nodes suitable for the collection operations.
The amount of rows is ruled by the agent which works for the longest time span, while the amount of columns is reduced to the minimum including only nodes that can be collected, hence reducing the parameters to add to the model from $max\_time \times |N|$ to $max\_time \times |N| - |NG| - |NI|$, where max_time indicates the time steps available for the most hard working agent.
Formally, to express that at time step t the collectable node h has a non-zero amount of garbage g it is possible to state that:

$$nodes\_garbage(t,\ h) = g \quad \text{with} \quad t \in [0,\ max\_time), \quad h \in [0,\ |N| - |NG| - |NI|), \quad g \in \mathbb{N}^{+}.$$

4.4.6 Motion Representation

The sequence of steps that each agent performs in completing the cleaning task is represented using a three dimensional matrix. In it, each agent is associated to a table representing the actor location, on the column, and the specific time step on the rows.
Formally this matrix states that an agent a is in a node h at a time step t if the corresponding element in the motion matrix $path$ is equal to 1:

$$path(a,\ t,\ h) = 1 \quad \text{with} \quad t \in [0,\ T_{Aa}) \text{ and } h \in [0,\ |N|)$$

where T_{Aa} identifies the workable time for agent a.

4.4.7 Collection Events

Given that more than one agent can be in a dirty node in a certain time step and that the node will be cleaned in the next step, it is necessary to find a way to identify which agent has collected the garbage.
For this reason we introduced a three dimensional matrix of boolean decision variables representing the collection events for the available agents.
Practically this matrix associate to each agent a table having the time steps as rows and the location as columns. In the same way done for representing the garbage in the nodes, even this matrix uses a number of location indexes equal to the nodes that support the collection operation.
Formally, the collection event carried out by an agent a in a time step t over a collectable node h is represented as: $collects(a,\ t,\ h) = 1 \quad \text{with} \quad h \in [|NG|,\ |N| - |NI|)$

4.4.8 Discharge Events

As done for the collection events, another aspect to be solved is which agent empties in the garage of a truck, so we introduced a three dimensional matrix representing the discharge events and associating to each agent a table with the location on the columns and time steps on the rows. The number of columns in this latter matrix has been set equal to the number of the depots available.
Despite what may have been said up to now, the number of nodes in the columns have not been reduced to those in which the empty action is supported due to empirical results highlighting how the presence of the normal agents depots led to a solution faster than a model employing only the truck ones.

It is important to remember that, differently by the normal agents, the truck ones must be in their own depot alone, this causing the number of trucks being equal to the number of garages for truck agents.
Formally, having an agent a discharging at a time t in a truck depot h the discharge operation is represented as:

$$discharge(a,\ t,\ h) = 1 \quad \text{with} \quad h \in [|NG| - |NT|,\ |NG| + |NT|) \quad \text{and} \quad t \in [0,\ |T_{Aa}|)$$

4.5 Decision Variables

Given the implementation above and assuming that A represents the vector of the available agents, with indexes having values in $[0,\ |A|-1]$, the number of decision variables that will be added, to the mathematical representation of the municipal waste collection problem, can be expressed as following:

- $\sum_{a \in A} T_{Aa} \times |N|$ boolean location variables.
- $max(T_A) \times |N \setminus \{NG, NI\}|$ garbage amount in the nodes for maximum working time.
- $\sum_{a \in A} T_{Aa}$ carried garbage by an agent during its working time.
- $\sum_{a \in A} T_{Aa} \times |N \setminus \{NG, NI\}|$ collection events of the agents.
- $\sum_{a \in A} T_{Aa} \times |NG|$ discharge events of the agents.

In conclusion the amount of decision variables in use can be determined by the following equation:

$$\begin{aligned} num\_variables = & \sum_{a \in A} (T_{Aa} \times |N|) \quad + \\ & max(T_A) \times |N \setminus \{NG, NI\}| \quad + \\ & \sum_{a \in A} T_{Aa} \times |N \setminus \{NG, NI\}| \quad + \\ & \sum_{a \in A} T_{Aa} \times |NG| \quad + \\ & \sum_{a \in A} T_{Aa} \end{aligned} \tag{4.2}$$

The above relation can be developed into a more compact form:

$$num\_variables = \sum_{a \in A} (T_{Aa} \times (2|N| - |NI| + 1)) + (max(T_A) \times |N \setminus \{NG, NI\}|) \tag{4.3}$$

Analyzing the equation it is noticeable how the model has a number of decision variables which grows linearly depending by the number of agents and time steps available.

4.6 Hierarchical Planning

Due to the preliminary tests over a simple problem, the time required to find a solution was observed to grow exponentially with the amount of nodes in the environment. To reduce the computation time and contain the size of the resulting mathematical representation, it has been decided to split the municipal waste management problem into two parts.

The first one, will address the collection and transport task of the garbage carried out by the normal agents, while the second part will focus on moving the collected garbage to the incinerators, role reserved to the truck agents.

Once a solution to the first part has been computed, the knowledge it carries will be transferred to the model of the second part in an attempt to limit the amount of unbound decision variables.
This is accomplished by crossing the information contained in the results of the discharge events performed by normal agents with their carried amount of garbage and the decision variables of the truck agents. Indeed, the second problem works over the optimal collection path and knows which agent empties in which location at each time. The solver will, then, chose the movement actions of the trucks knowing that their carried capacity increases only in the garages having one or more normal agents emptying in them.

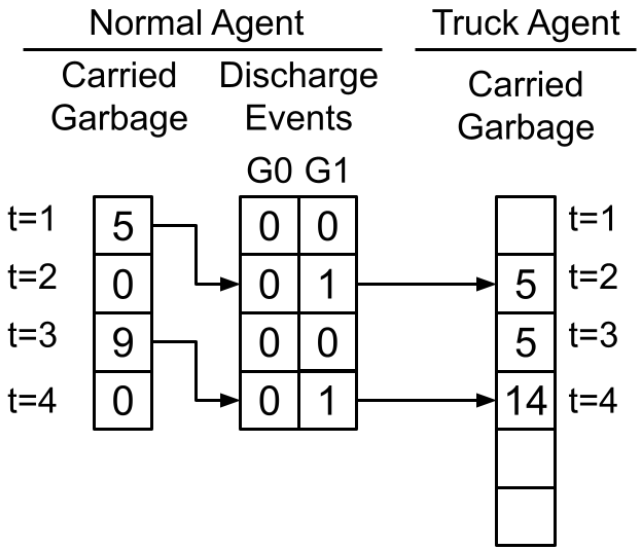


Figure 4.8: Embedding of normal agent solution to truck agent knowledge base.

In conclusion, the hierarchical planning allows us to have an optimal management of the normal agents in the first phase and a reduced model to manage the truck agents in the second one.

In Figure 4.8 is shown an example of how hierarchical planning is carried out. The example is made over an environment in which there are only two agents, a normal one and a truck. The actors have four and six working time steps respectively.
As observable, the solution to the collection and unloading of the normal agent, left side, is exploited to determine the carried amount of garbage in the truck.
As will be enforced by the constraints explained in the next section, the carried garbage of a normal agent goes to 0 whenever a discharge action is being executed and the quantity contained previously is transferred in the truck agent.
This means that if at time step t the normal agent performs a discharge operation, then a quantity of garbage g equal to the agent carried garbage at $t-1$ is enforced through the constraints of the truck agent for its carried capacity at time t. Formally:

$$carried\_truck(at,\ t)\ =\ carried\_agent(a,\ t-1) \times discharge(a,\ t,\ garage(at)) \tag{4.4}$$

where $a \in [0, |A| - |NT|)$ is the index of the normal agent in use, $at \in [0, |NT|)$ is the index of the truck agent and $garage(at) \in [0, |NG|)$ identifies the index of the garage of the agent provided.
In this way it is possible to directly inform the model of fixed actions and values in time which highly decrease the amount of possible states.
To be completely clear, the carried amount for the truck agent at time $t = 3$ should be $>= 5$ due to the constraints formulation. However being only one normal agent in the example and assuming that reaching the disposal, or incinerator, site requires two time steps, it can be enforced that the value can only be 5.

This approach yields a huge saving into computational times as shown in Figure 4.9, where a comparison in computational time is proposed for the environments employing or not the hierarchical structure.

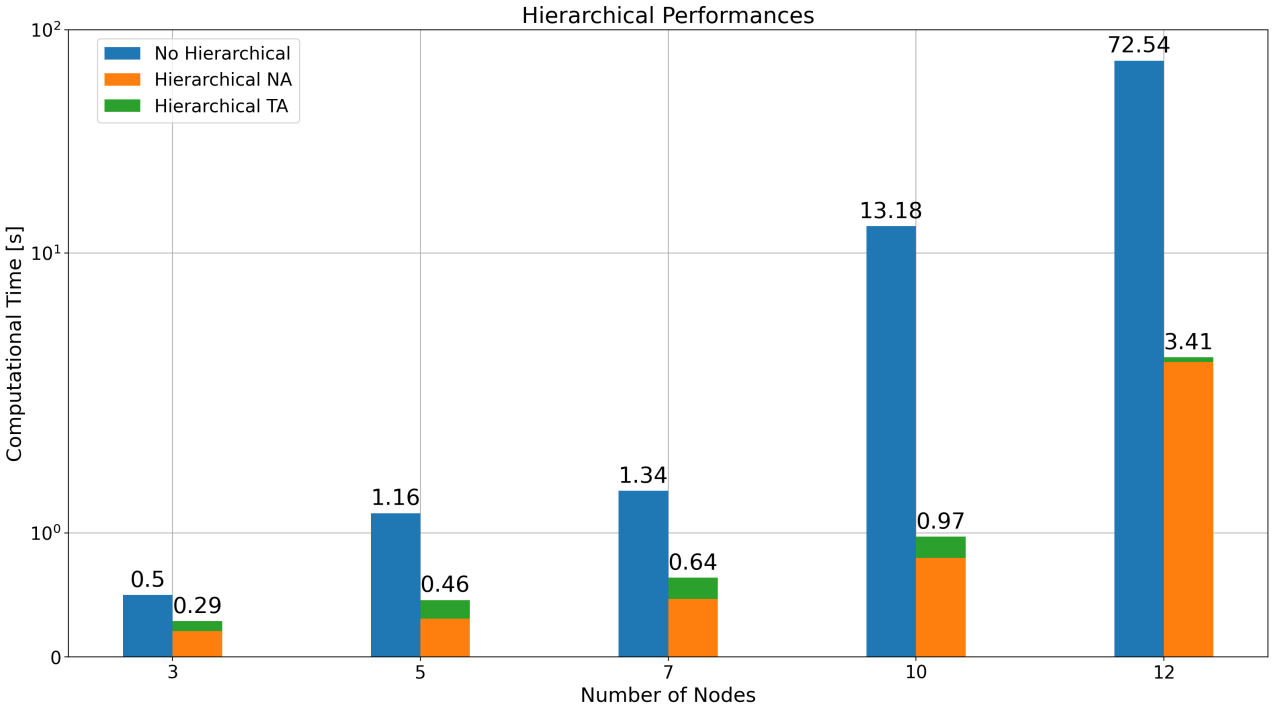


Figure 4.9: Computational time required by the algorithm with and without the hierarchical planning. The logarithmic scale is used to enhance the differences with small values. In Blue are shown the amount of time needed without the hierarchical approach, while Orange and Green portraits the planning time respectively for the Normal agents and Truck ones.

4.7 Clustering Planning

Thanks to the hierarchical planning it was possible to reach good time reductions in searching for a solution, but the first part of the planning, with the normal agents, still took an unsatisfactory amount of time. For this reason it was implemented a new part of the algorithm allowing it to improve its performances.

In an attempt to emulate the human dynamics in act during the waste collection and to further decrease the amount of computational time needed, it was decided to employ clustering algorithms. This change into the model solving process allows us to simplify the normal agents collection task by grouping some harvest nodes together.

Exploiting clustering algorithms is justified by the possibility of group of citizens exposing their garbage on the same collection point rather than outside their buildings. At the same time, there exist the case in which many buildings on the same road may have an in between distance coverable in less than a time step. Such cases are treated in the framework by rounding the motion time to the next integer, this to account for a safety error in time scheduling.
Using clustering algorithms the aim of the system is to identify the nodes which are highly connected among them and merge them within a new node. This new entity will have a total waste to collect equal to the sum of the garbage contained by the nodes composing it.
Furthermore, the motion time needed to reach it is fixed equal to the sum of the maximum motion time needed to traverse the pairs of nodes merged, the same happens to the collection time of the node. In this way the algorithm, in finding a solution, will reserve a window of time steps enough large for further refining computations.

Indeed, after the clusters have been identified and a solution to the planning problem is found, the algorithm expands the available clusters and enforces the solution knowledge over the new problem. What the framework is doing is to compress the available environment into a more compact one, solve this reduced representation and exploit the solution to solve the original configuration.
Practically, the system relaxes the environment, by generating artificial nodes with the suitable characteristics allowing for a plausible solution.

Then, it generates a solution applying the constraints of Subsections 4.8.3, 4.8.4 and 4.8.5 to the task. After that, if a solution is found, its knowledge is exploited, while the clusters are expanded and the environment restored to its original settings.

The discharge events matrix is directly enforced in the new model solution, being it completely unaffected by the clustering due to the amount discharged, and considering the fact that the amount of time required is the same with or without clustering.
The collection events, the location of the agents and the carried amount of garbage, together with the waste in the nodes, have been set up in the new solution only if the nodes involved at a certain time step do not belong to any of the ones constituting the clusters.
In this way the system fixes the environment spatially outside the clusters and temporally before and after traversing them, while the decision variables representing what happens inside are left to be found allowing for a continuity in the solution.

Thanks to this relaxation it has been possible to reach a reduction in computational times, as shown in Chapter 5, with the same solutions as the clusterless settings.
In Figure 4.10 we show an example of the computation of a solution to the waste management problem employing the clustering technique. In the example we assume that the clustering operations affect the nodes 1, 2 and 3 which will be then merged into an abstract node that will keep the id of the last node absorbed, 3 in this case. Due to the number of nodes grouped and the movement time among them, it is assumed that a minimal window of three time steps is required to visit all of them and, in case, perform the collection operation. Also the collection time is updated for the abstract node and set to three, assuming that all of the nodes need one time step to be collected.
That said, the algorithm will search for a solution on the clustered graph.

Let's suppose that the solution is $[0, 3, 3, 3, 5, 6, 4]$, then time window reserved for refining the solution to a compatible version for the original graph is given by the sequence of 3 which denotes the time needed to collect the cluster node. Consequently, all the locations neighbouring the merged nodes will be enforced on the model for the expanded graph, being, those, fixed points in time and not affected by the visit order inside the cluster.
At this point the constraints ruling the behaviour of the environment will be enforced on the expanded model and a solution will be searched. The solving process, can be supposed as the one shown in the figure, where the time steps in the reserved window are kept as decision variables, and knowing the locations before and after visiting the clustered nodes, the solution is almost immediate.

First it must be decided which node to visit after being in node 0, in this case the only answers are either nodes 1 or 5 which are connected to 0, but choosing 5 means that the visit to the cluster is delayed by at least one time step, making the resulting plan violate the constraint over the maximum number of working time steps. Then the only plausible action is to chose 1 as the next node to visit. Repeating this process for all the other time steps in the window allows to reach a suitable solution without needing to exploit more time than needed.

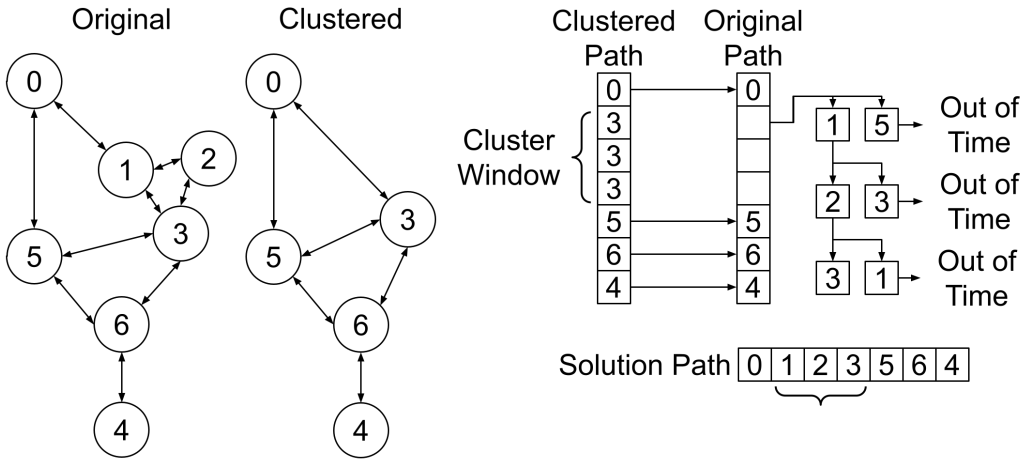


Figure 4.10: An example of solution search with clustering algorithm.

Specifically, once the agent reaches 1 the choice is about the nodes 2 or 3, but choosing 3 now means to be needing one more time step to visit 2, go back and reach 6 so the solution exploiting the node 3 now is unfeasible, leaving 2 as the only option. In this last step the choice is between 1 and 3, but it is known that after the next two step the agent must be in 6 leaving as the only option moving to 3. The refining step of the initial solution for the expanded graph is now complete and the final solution is given to the user.

4.7.1 Clustering Algorithm

In the literature many algorithms exist to perform clustering and graph partitioning which can be employed to reduce the number of nodes in the problem, yielding to faster computation results.
In this framework it has been decided to exploit the Markov Clustering Algorithm which is an unsupervised algorithm detecting clusters of nodes exploiting the random walking principle.

The idea behind this approach is to simulate a random exploration of the graph computing the probabilities that each node can be visited from any other location, then to inflate the resulting values to rule out low scoring elements by increasing the weight of high ones. Performing this operation until convergence results into a set of nodes having very high probabilities of transitioning among them, and as such they identify a cluster in the graph.

The search of the cluster works by first filling up a square matrix representing the probability of reaching from a node in the rows any node in the columns. This results into having only those combinations of nodes connected together to be non zero and with value equal to $\frac{1}{conn(n)}$ where n is any node on the rows and $conn(n)$ is the number of connections that the node n establishes with the other nodes in the graph.
After that, the matrix is raised to a power defined by the user which simulates the random walking operation for a number of steps equal to the power in use.
At this point an inflation operation is performed to push the small values towards zero and rise the values of strong probabilities.

Inflating is made rising each element in the matrix to a power chosen by the user and normalize each column by dividing the powered elements in the column by the sum of the values inside itself. Assuming that $Rows$ is the total number of rows in the matrix and $Cols$ is the total number of columns in the matrix, the operation to perform is:

$$\forall r \in Rows,\ \forall c \in Cols \quad matrix[r][c] = \frac{matrix[r][c]}{\sum_{c2 \in Cols} matrix[r][c2]} \tag{4.5}$$

The process is then repeated until convergences or until a specific stop condition is met. The Markov clustering algorithm allows also to rule out those probabilities which values are extremely low, setting them to 0, this assuming that they will eventually be zeroed and saving iterations in the process. At the end of the cycle, the elements in the matrix which are non zero are representative of the clusters and the matrix must be parsed to extract this information.

The cluster forming process starts by parsing the rows of the matrix and merging each non zero element in the columns which has not been already grouped. In the end, it is possible to get as many clusters as the independent rows, with none associated values on the columns.

4.8 Constraints

4.8.1 Constraint Programming

As anticipated into the Design Section 3.1, the framework builds a mathematical representation of the environment and constraints the possible values of the decision variables, ensuring that the solutions found satisfies all these bounds.

This is simply the definition of Constraint Programming, in which an objective function is provided as the measurement of the goodness of the solutions and the bounds rule out unsatisfactory results.
A solver algorithm will then search the combination of values which associated to the decision variables will minimize the objective, or cost, function.

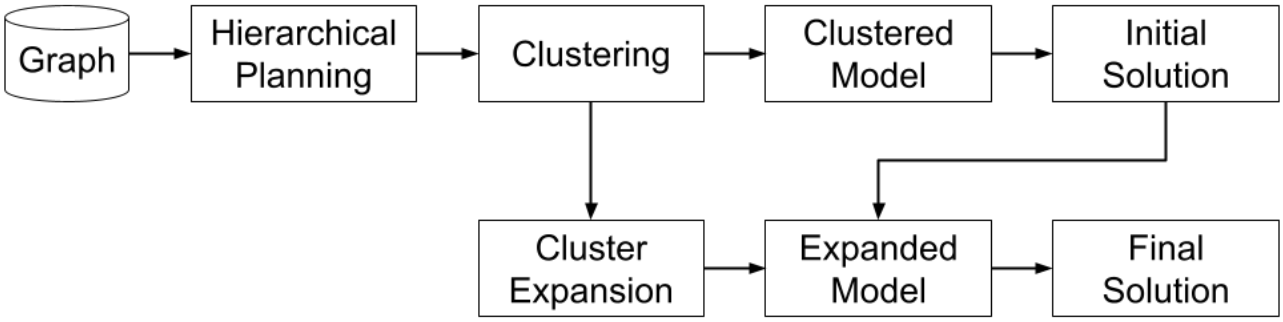


Figure 4.11: Flowgraph of the planning process.

Formally the model built by the framework can be expressed by Equation 4.6.

$$\begin{aligned} & min_{a,t,h}\ obj(A,\ T,\ N) \\ \text{subject to:}\ & constraint1 \\ & constraint2 \\ & \dots \end{aligned} \tag{4.6}$$

Where a, t and h are the elements composing the decision variables and belonging to the sets A, T and N representing respectively the agents, nodes and time steps available.

4.8.2 Decision Variables Definition

To clarify the objects and decision variables employed in the constraint making process, we now list the main definitions needed to correctly understand the bounds introduced to the model.

Recalling what have been said in the Model Section 3.2 and before discussing the constraints, notice that T identifies the vector of time steps available globally, so the maximum time in which the waste management problem has to be completed; T_{Aa} is the element of T representing the amount of time steps that an agent a has at its disposal to carry out its tasks.
N is the vector of the available nodes comprehensive of garage, harvest and incinerator nodes.
Specifically NG is the subset of nodes representing the garages in N, while NI is the subset of nodes representing the incinerators.

The agents at disposal of the framework are represented through the vector A which is comprehensive of both normal and truck agents. Furthermore, it will be used $garage(a)$ to identify a function which returns the index in the vector N addressing the garage of the provided input agent a; the function $garbage(h)$ is used to return the amount of garbage of a node h at $t = 0$; the function $conn(h)$ is exploited to identify the sets of nodes reachable from an arbitrary node h.

There are four different decision variable matrices in the model, namely, *path* which represents the motion actions undertaken by the agents, *collects* which describes the collection events, *discharge* which addresses the discharge events and *nodesGarbage* which shows the amount of garbage that each collectable node has in a specific time step. All those matrices are three dimensional due to the association of the agents available to the time and location of the specific operation. The final solution will then be extracted by crossing the information contained in their elements.

In addition to those matrices, each agent contribute to the model with a vector of length $|T_A|$ expressing the amount of garbage carried at each time step. This vector is addressed by $agent\_garbage(a, t)$ specifying the amount of waste that agent a carries at time t. Moreover, an agent a maximum carrying capacity can be exploited in the constraints by using the wording $a.max\_capacity$.
Furthermore, note that $|vector|$ is used to address the number of elements in a vector and that all parameters addressing indexes in the vectors have values in the interval $[0,\ |vector|)$.

Lastly, due to the introduction of the hierarchical planning there are two sets of constraints ruling first the behaviour of the normal agents, and then the behaviour of truck ones, but some of the constraints are in common and will be explained together even though the framework practically encodes them differently.

4.8.3 Agents Shared Constraints

Following are the constraints shared both by normal and truck agents.

Starting, Ending and Connection constraints

1. All agents begin the working shift in their garage and must return to it by the end of the available time.

$$\forall a \in A \quad path(a,\ t = 0,\ garage(a)) = 1 \tag{4.7}$$

$$\forall a \in A \quad path(a,\ t = |T_{Aa}| - 1,\ garage(a)) = 1 \tag{4.8}$$

2. Each agent must be only in one node per time step, which translated in the model means that per each agent the row of locations associated to a time step must always have only one decision variable set to 1.

$$\forall a \in A, \forall t \in T_{Aa} \quad \sum_{h \in nodes} path(a, t, h) = 1 \tag{4.9}$$

3. Ensure that all solutions do not require more time steps than the available ones. This is more of a safety constraint, due to the capacity of the CPLEX solver to allocate further decision variables if needed. By adding this bound the resulting solutions of such behaviour became illegal. So, what is checked is that summing each variables in the motion matrix the result must be equal to the number of time steps available for the agent.

$$\forall a \in A \quad \sum_{t \in T_{Aa}} \sum_{h \in N} path(a, t, h) = |T_{Aa}| \tag{4.10}$$

Motion Model

The following equation describes how the connection among the nodes is embedded inside the agents decisions.

1. Agents are allowed to move only through connected nodes. Which can be interpreted as: if an agent a at time t is in a node h, then at time $t+1$ it can either be in the same node or in one of the nodes connected to h.

   $\forall a \in A, \forall t \in T_{Aa} \setminus \{|T_{Aa}|\}, \forall h \in N, \forall h2 \in N$

$$path(a, t, h) = 1 \implies path(a, t+1, h) = 1 \vee \bigvee_{n \in conn(h)} path(a, t+1, n) = 1 \tag{4.11}$$

   Where $\bigvee$ identifies the OR operation over a set of elements.

Shared Capacity Constraints

1. In the beginning and in the end all agents must be empty. They are in their garage.

$$\forall a \in A, t \in \{0, |T_{Aa}| - 1\} \quad agent\_garbage(a, t) = 0 \tag{4.12}$$

2. An agent cannot carry more garbage than its maximum capacity. It is possible to discard the first time step being it set previously.

$$\forall a \in A, \forall t \in T_{Aa} \setminus \{0\} \quad agent\_garbage(a, t) \leq a.max\_capacity \tag{4.13}$$

4.8.4 Constraints Ruling The Amount Of Garbage In Nodes

In this subsection we list the constraints defining how the waste in the graph is updated depending by the agents actions.

1. Harvest nodes at the start of the working day are dirty.

   $\forall h \in N \setminus \{NG, NI\}$

   $$nodesGarbage(t = 0,\ h) = garbage(h) \tag{4.14}$$

2. If an agent is in a node either the node has garbage or is clean. This constraint allows any agent to pass by any nodes without enforcing any collection action and avoiding to limit movement only on clean nodes.

   $\forall t \in [1,\ max(T_A)),\ \forall a \in A \setminus \{NT\},\ \forall h \in N \setminus \{NG, NI\}$

   $$path(a, t, h) = 1 \implies nodesGarbage(t, h) = 0 \quad \vee \quad nodesGarbage(t, h) = garbage(h) \tag{4.15}$$

3. If a node is clean at time t, then either one or more agents are in the node or the node was already clean. This constraint directly enforces that nodes can be cleaned only if at least one agent is in them.

   $\forall t \in [1,\ max(T_A)),\ \forall h \in N \setminus \{NG, NI\}$

   $$nodesGarbage(t, h) = 0 \implies \sum_{a \in A \setminus \{NT\}} path(a, t, h) \geq 1 \quad \vee \quad nodesGarbage(t-1, h) = 0 \tag{4.16}$$

4. At any time step a node contains either its initial value of garbage, or none. This bound acts as a reduction of possibilities in the search tree limiting the nodes garbage values only to two cases.

   $\forall t \in [1,\ max(T_A)),\ \forall h \in N \setminus \{NG, NI\}$

   $$nodesGarbage(t, h) = 0 \quad \vee \quad nodesGarbage(t, h) = garbage(h) \tag{4.17}$$

5. If the amount of garbage in a node is less than the global minimum set it to 0. This constraint acts as a safety one, being the solver approximating values spanning 64 bit it is possible that, due to some approximation, some illegal values are set. Hence the reason of this limit to exist.

   $\forall t \in [1,\ max(T_A)), \forall h \in N \setminus \{NG, NI\}$

   $$nodesGarbage(t, h) \leq min(nodesGarbage(0, :)) - 0.1 \implies nodesGarbage(t, h) = 0 \tag{4.18}$$

   Where : is the symbol used to say that all the columns are being spanned. Furthermore the 0.1 ensures the error margin of acceptability of the value.

6. Once cleaned a node stays clean.

   $\forall t \in [1,\ max(T_A)) \setminus \{0, |T_A|\}, \forall h \in N \setminus \{NG, NI\}$

   $$nodesGarbage(t, h) = 0 \implies nodesGarbage(t+1, h) = 0 \tag{4.19}$$

7. Once the environment has been cleaned then it keeps being clean. This is a more general implementation of the previous constraint.

   $\forall t \in [1,\ max(T_A))$

   $$\sum_{h \in N \setminus \{NG, NI\}} nodesGarbage(t, h) = 0 \implies \sum_{h \in N \setminus \{NG, NI\}} nodesGarbage(t+1, h) = 0 \tag{4.20}$$

8. At the end all Harvest nodes must be cleaned. This is the constraint representing the end state of the municipal waste collection problem.

$$\sum_{h \in N \setminus \{NG, NI\}} nodesGarbage(max(T_A), h) = 0 \tag{4.21}$$

4.8.5 Normal Agent Constraints

Capacity and Collection Constraints

1. The agents are not allowed to collect any waste at the beginning of their shift and at its end, this because it is required they to be in their garage where no operation if not movement can be performed.

$$\forall a \in A \setminus \{NT\},\ t \in \{0,\ |T_{Aa}| - 1\} \quad \sum_{h \in N \setminus \{NG,\ NI\}} collects(a,\ t,\ h) = 0 \tag{4.22}$$

2. Any agent can collect at most once per time step.

   $\forall a \in A \setminus \{NT\},\ \forall t \in [0, T_{Aa})$

$$\sum_{h \in N \setminus \{NG, NI\}} collects(a, t, h) = 0 \quad \vee \sum_{h \in N \setminus \{NG, NI\}} collects(a, t, h) = 1 \tag{4.23}$$

3. If a Normal agent collects from an node, then it is in the node.

   $\forall a \in A \setminus \{NT\}, \forall t \in [1, |T_{Aa}|), \forall h \in N \setminus \{NG, NI\}$

$$collects(a, t, h) = 1 \implies path(a, t, h) = 1 \tag{4.24}$$

4. If a Normal agent is not in a Harvest node then the agent cannot collect that node.

   $\forall a \in A \setminus \{NT\}, \forall t \in [1, |T_{Aa}|), \forall h \in N \setminus \{NG, NI\}$

$$path(a, t, h) = 0 \implies collects(a, t, h) = 0 \tag{4.25}$$

5. A Normal Agent staying in the same node keeps its carried capacity unchanged. This is possible even if more than one time step is required for collecting the node, because at the time of the collection event the carried amount is updated and the agent is forced to stay in position for the required time.

   $\forall a \in A \setminus \{NT\}, \forall t \in [1,\ |T_{Aa}|), \forall h \in N$

$$\begin{aligned} &(\ path(a, t, h) = 1 \quad \wedge \quad path(a, t-1, h) = 1\ ) \implies \\ &agent\_garbage(a,\ t) = agent\_garbage(a,\ t-1) \end{aligned} \tag{4.26}$$

6. If a node gets collected, then only one agent is collecting it.

   $\forall t \in [1,\ |T_{Aa}|),\ \forall h \in N \setminus \{NG,\ NI\}$

$$\begin{aligned} (\ nodesGarbage(t,\ h) = 0 \quad \wedge \quad &nodesGarbage(t-1,\ h) = garbage(h)\ ) \implies \\ &\sum_{a \in A \setminus \{NT\}} collects(a,\ t,\ h) = 1 \end{aligned} \tag{4.27}$$

7. If a normal agent collects garbage, then its carried amount is equal to the previous one plus the quantity in the node collected.

   $\forall a \in A \setminus \{NT\} \forall t \in [1, |T_{Aa}|), \forall h \in N \setminus \{NG, NI\}$

$$
\begin{aligned}
&collects(a,t,h) = 1 \implies \\
&\quad agent\_garbage(a,t) = \quad agent\_garbage(a,t-1) \quad + \\
&\qquad\qquad\qquad\qquad\quad nodesGarbage(t-1,h)
\end{aligned}
\tag{4.28}
$$

8. If a Normal agent does not collect any Harvest node in a time step, then its carrying capacity is either 0 or the previous one.
   $\forall a \in A \setminus \{NT\}, \forall t \in [1, |T_{Aa}|)$

$$
\begin{aligned}
&\sum_{h \in N \setminus \{NG,NI\}} collects(a,t,h) = 0 \implies \\
&\qquad agent\_garbage(a,t) = \quad agent\_garbage(a,t-1) \quad \vee \\
&\qquad\qquad\qquad\qquad\quad agent\_garbage(a,t) = 0
\end{aligned}
\tag{4.29}
$$

9. Each node can be collected only once and by one agent.
   $\forall h \in N \setminus \{NG, NI\}, \forall a \in A \setminus \{NT\}$

$$
\sum_{t \in [1,|T_{Aa}|)} collects(t,a,h) = 1 \tag{4.30}
$$

10. If a normal agent collects garbage, then the agent is in the node and the node gets cleaned.
    $\forall t \in [1, |T_{Aa}|,\ \forall a \in A \setminus \{NT\},\ \forall h \in N \setminus \{NG, NI\}$

$$
\begin{aligned}
&collects(a,t,h) = 1 \implies \\
&path(a,t,h) = 1 \quad \wedge \\
&nodesGarbage(t,h) = 0 \quad \wedge \\
&nodesGarbage(t-1,h) = garbage(h)
\end{aligned}
\tag{4.31}
$$

Constraints To Empty Normal Agents

1. If a normal agent discharges in a truck garage then its carried garbage goes to 0.
   $\forall a \in A \setminus \{NT\},\ \forall t \in [1, |T_{Aa}|)$

$$
\begin{aligned}
&\sum_{at \in NT} path(a,t,garage(at)) = 1 \quad \wedge \sum_{at \in NT} discharge(a,t,garage(at)) = 1 \implies \\
&agent\_garbage(a,t) = 0 \vee agent\_garbage(a,t) = agent\_garbage(a,t-1)
\end{aligned}
\tag{4.32}
$$

2. If a normal agent empties the carried garbage, then it must be in an truck garage.
   $\forall a \in A \setminus \{NT\}, \forall t \in [1, |T_{Aa}|)$

$$
\begin{aligned}
&(\ agent\_garbage(a,t) = 0 \ \wedge \ agent\_garbage(a,t-1) \neq 0\ ) \implies \\
&\sum_{at \in NT} path(a,t,garage(at)) = 1 \quad \wedge \sum_{at \in NT} discharge(a,t,garage(at)) = 1
\end{aligned}
\tag{4.33}
$$

3. If an agent is not in a truck garage, then its carried garbage can only increase.
   $\forall a \in A \setminus \{NT\}, \forall t \in [1, |T_{Aa}|)$

$$
\begin{aligned}
&\sum_{at \in NT} path(a,t,garage(at)) = 0 \implies \\
&agent\_garbage(a,t) \geq agent\_garbage(a,t-1) \ \wedge \sum_{at \in NT} discharge(a,t,garage(at)) = 0
\end{aligned}
\tag{4.34}
$$

Discharge Events Constraints

In the following section, the constraints ruling over the discharge event matrix are defined.

1. At the beginning and at the end of the work shift no discharge events are allowed. Agents must be in their garages and no discharge can be done.

$$\forall a \in A \setminus \{NT\}, \forall t \in \{0,\ |T_{Aa}| - 1\} \quad \sum_{h \in NG} discharge(a,\ t,\ h) = 0 \tag{4.35}$$

2. normal agents are allowed to empty only in truck agents garages.

$$\forall a, a2 \in A \setminus \{NT\},\ a \neq a2,\ \forall t \in [1,\ |T_{Aa}|) \quad discharge(a,\ t,\ garage(a2)) = 0 \tag{4.36}$$

3. If a normal agent discharges in a truck garage then the agent is in the garage.
   $\forall a \in A \setminus \{NT\}, \forall t \in [1, |T_{Aa}|), \forall at \in NT$

$$discharge(a, t, garage(at)) = 1 \implies path(a, t, garage(at)) = 1 \tag{4.37}$$

4. A Normal agent which is not in a truck garage cannot discharge in it.
   $\forall a \in A \setminus \{NT\}, \forall t \in [1, |T_{Aa}|), \forall at \in NT$

$$path(a, t, garage(at)) = 0 \implies discharge(a, t, garage(at)) = 0 \tag{4.38}$$

5. If an agent is in a truck garage it either discharges or not. This constraint allows the agents to move across the truck garages without enforcing a discharge action which may rule out an optimal solution.
   $\forall a \in A \setminus \{NT\}, \forall t \in [1, |T_{Aa}|), \forall at \in NT$

$$\begin{aligned} &path(a, t, garage(at)) = 1 \implies \\ &discharge(a, t, garage(at)) = 1 \quad \vee \\ &discharge(a, t, garage(at)) = 0 \end{aligned} \tag{4.39}$$

6. At each time step any Normal agent can either discharge or not. Which translated means that an agent can empty only once per time step.
   $\forall a \in A \setminus \{NT\}, \forall t \in T_{Aa}$

$$\sum_{h \in NG} discharge(a, t, h) = 0 \quad \vee \quad \sum_{h \in NG} discharge(a, t, h) = 1 \tag{4.40}$$

4.8.6 Truck Agent Constraints

Following are constraints specific for the truck agents. The need of having different constraints is due to the trucks working on separate objects and exploiting the knowledge of the normal agents routing to find a suitable solution. Due to this constraints working over the solved matrices of the normal agents, it will be used *solCarried* and *solDischarge* to address the solution vector of the carried amount of garbage and the matrix regarding the discharge events of the normal agents, respectively.

Capacity Constraints

1. If more than one Normal Agent discharges into the garage of a truck the truck will carry the sum of the discarded waste.

   $\forall at \in NT,\ \forall t \in [1, |T_{Aat}|),\ \forall a \in A \setminus \{NT\},\ \forall t2 \in [1, |T_{Aa}|) \mid t2 < t$

$$\begin{aligned} \sum_{a} solDischarge(a, t2, garage(at)) \geq 1 \implies \\ agent\_garbage(at, t) = agent\_garbage(at, t-1) + \\ \sum_{a} (solCarried(a, t2-1) \times solDischarge(a, t2, garage(at))) \wedge \\ path(at, t, garage(at)) = 1 \end{aligned} \tag{4.41}$$

Discharge Constraints

1. If a Truck agent is in an Incinerator node, either it empties or passes by.

   $\forall at \in NT, \forall t \in [1, |T_{Aat}|)$

$$\begin{aligned} \sum_{h \in NI} path(at, t, h) = 1 \wedge agent\_garbage(at, t-1) \neq 0 \implies \\ agent\_garbage(at, t) = 0 \vee agent\_garbage(at, t) = agent\_garbage(at, t-1) \end{aligned} \tag{4.42}$$

2. If a Truck agent is empty, either was empty before or it is in an Incinerator node.

   $\forall at \in NT, \forall t \in [1, |T_{Aat}|)$

$$agent\_garbage(at, t) = 0 \implies agent\_garbage(at, t-1) = 0 \vee \sum_{h \in NI} path(at, t, h) = 1 \tag{4.43}$$

3. If a Truck agent is not in an Incinerator node, then its carried garbage amount can only increase.

   $\forall at \in NT, \forall t \in [1, |T_{Aat}|)$

$$\begin{aligned} \sum_{h \in NI} path(at, t, h) = 0 \wedge agent\_garbage(at, t-1) \neq 0 \implies \\ agent\_garbage(at, t) >= agent\_garbage(at, t-1) \end{aligned} \tag{4.44}$$

4. If the garbage inside a Truck agent changes, either it is in an incinerator or it is in its garage with one or more normal agents.

   $\forall at \in NT, \forall t \in [1, |T_{Aat}|)$

$$\begin{aligned} agent\_garbage(at, t) \neq agent\_garbage(at, t-1) \implies \\ \left(\left(\sum_{a \in A \setminus \{NT\}} path(a, t, garage(at)) \geq 1 \right) \wedge path(at, t, garage(at)) = 1 \right) \vee \\ \sum_{h \in NI} path(at, t, h) = 1 \end{aligned} \tag{4.45}$$

4.8.7 Working Time Constraints

Following we present the constraints ruling the time spent by the agents during movement, collection an discharge actions.

1. If an agent collects a node, then it stays in the node for the amount of time necessary.
   $\forall a \in A \setminus \{NT\}, \forall h \in N \setminus \{NG, NI\}, \forall t \in [1, |T_{Aa}|)$

   $$collects(a,t,h) = 1 \implies \sum_{t2 \in [t, t+h.HT) | t2 < |T_{Aa}|} path(a, t2, h) \geq h.HT \tag{4.46}$$

   Where $h.HT$ define the time needed to collect the node h.

2. If a truck agent discharges into an Incinerator node, then it stays in the node for the amount of time necessary.
   $\forall at \in NT, \forall h \in NI, \forall t \in [1, |T_{Aat}|)$

   $$discharge(at,t,h) = 1 \implies \sum_{t2 \in [t, t+at.DT) | t2 < |T_{Aat}|} path(at, t2, h) \geq at.DT \tag{4.47}$$

   Where $at.DT$ define the time needed by a truck agent to discharge.

3. If an agent moves, then the waiting time must satisfy the following constraint.
   $\forall a \in A, \forall t \in [1, |T_{Aa}|), \forall h, h2 \in N | h \neq h2$

   $$\begin{aligned} path(a, t-1, h) = 1 \wedge path(a, t, h2) = 1 \implies \\ \sum_{t2 \in [t - conn(h,h2), t).time \mid t2 \geq 0 \wedge t2 < T} path(a, t2, h) \geq conn(h, h2).time \end{aligned} \tag{4.48}$$

   Where $conn(h, h2).time$ defines the time needed by an agent a to move from node h at time $t-1$ to node $h2$ at time t.

4.9 Cost Function Normal Agents

Having defined the constraints under which the mathematical model of the normal agents works, it is now necessary to declare what objective function is being used in solving the waste management problem. Specifically the objective function for the normal agents accounts for the amount of agents in use, the time of completion and the garbage left in the nodes. In this way the minimization of the function yields a solution with the minimal number of actors resulting in savings in the hypothetical company resources. At the same time minimizing the amount of nodes still dirty at a certain time t allows to maximize the throughput of the collection task. Lastly, enforcing a penalty proportional with the time in which the nodes are collected ensures that the algorithm will search the solution requiring the least number of steps. Such a function is presented formally in Equation 4.49.

$$\begin{aligned} cost = \sum_{t \in T} \left(|A| - \sum_{h \in NG} \sum_{a \in A \setminus \{NT\}} path(a,t,h) \right) \times t + \\ \left(\sum_{h \in N \setminus NG, NI} nodesGarbage(t,h) / graph(h).garbage \right) * (t+1) \end{aligned} \tag{4.49}$$

The first line will oblige all the agents to reach their garage as soon as possible attributing a cost for each agent not in it at each time step. In such a manner the agents are forced to stay in their garage whenever their employment does not benefit the solution, while the needed actors are forced to end their task and go back in the least time possible. The second line will oblige the solver to collect all the nodes as soon as possible by attributing a cost equal to the number of nodes not yet cleaned.

4.10 Cost Function Trucks

In the environment of the Truck Agents, the cost function is simpler with respect to the normal agents. Indeed the major requirement to this cost function is to force the solver to find a solution which will minimize the time that the truck agents are outside their garage. So what is desired is for it to move the trucks only when necessary, i.e., when agents are full, or when an effective advantage exist.
Such function is formally expressed by Equation 4.50.

$$cost = \sum_{t \in T} \left(|NT| - \sum_{at \in NT} path(at, t, garage(at)) \right) \times t \tag{4.50}$$

Adding a multiplicative factor depending by the time, t, ensures that the solver will try to reach the solution with minimal time steps. It will also avoid leaving trucks wondering around the graph, being those the major causes of the cost to be added.

5 Tests

To understand the capabilities of the system and its performances we decided to run a set of tests varying the amount of agents, their working time and the number of nodes to collect. Those are, indeed, the main parameters influencing the framework goodness.
Thanks to the mathematical modelling described in Chapter 4 it was possible to reach a Mixed Integer Problem formulation, MIP, which allows to find a solution easier with respect to the general Constraint Programming problems having continuous values decision variables.
Furthermore, the constraints introduced in the problem allow to account for a number of parameters which grows linearly with the input variables and to simplify the solving process with respect to problems having non-linear constraints.

5.1 Dataset Creation

Carrying out the performance measurement process requires different test environments which ensure that a solution can be found. For this reason exploiting real data for an unsupervised creation of the environment, was almost impossible due to the unpredictability of the road network connections, but given the need to test the system over different cases, an artificial dataset has been created starting from the real data of the OSM database.
The dataset building process works through two main steps: random sampling and rewiring.

The first action performed in creating the artificial data consists into randomly sampling nodes from a graph representing real data. In this way the information regarding the amount of garbage available and the collection time needed are preserved.

The second step deletes all the previous connections of the sampled nodes and then performs a rewiring process which will establish new links among the existing nodes. To add randomness in the making of the connections, the number of routes that a node has at its disposal is being decided randomly in the interval [1, $nmax$] where $nmax$ is the maximum number of connections that any node can have and it is provided by the user.

As it is possible to notice, each node must have at least one link with any other one in the graph, which ensures that the input environment does not have orphan nodes which may result into an unsolvable problem. Even though a cleaning function exists to remove the nodes without connections, as an intermediate step before building the mathematical model, avoiding such case is of primary importance to reach the desired size of the artificial graph for the tests in act.
To further ensure the feasibility of the solution all the new links established in the rewiring process are forced to be bi-directional, thus to avoid nodes which exploration results into dead ends for the solver.

Concerning the travel time between nodes, it has been decided to set it as a constant amount for all the links. This choice has been taken due to the needs of obtaining an environment which was feasible to clean given the workforce and its maximum working time.
Having the collection time of the nodes uncertain due to the random sampling, it was not possible to manage also the uncertainty rising from the connection among the nodes, so setting it to a constant amount of time allowed to keep the problem solvable. To further understand how much it would have impacted the feasibility of the problem to keep the distances between the nodes, imagine what the random sampling operation is doing: it randomly picks nodes from all across the graph until the maximum number of nodes is not reached. If the system during the rewiring operation were to account for an approximation of the travel time between connected nodes probably it would have used the Haversine formula 4.1. Thus, the algorithm may have connected two nodes far away, resulting in prohibitive motion times, which in turn would have required an increase in the working time of

the agents and consequently an increase in the number of parameters in the model. As noticeable by these observations, account for an estimate of the travel time would have caused the model to became immediately massive in time with consequently huge computational times.

Being those tests created only to evaluate the performances of the system without waiting too much for the algorithm to end its computations, we decided for an artificial rewiring discarding any roads approximation. Indeed, in this phase of the framework, we are only interested in the performance of the system majorly depending by the number of agents and the nodes in use, so keeping the time strictly to the bare needed amount plus some error margin, allows to have tests results which do not extremely penalize the model with an excessive number of constraints.

5.2 CPLEX Tests

Following are the results obtained with the CPLEX solver over the mathematical model described so far. The employment of CPLEX has been chosen due to being one of the leading software for solving constraint programming and known to be fast and efficient. The tests settings available is a combination of a number of collection nodes, agents and their working time. In the test environment it has been decided to have one garage per all the normal agents, one truck agent with its own garage and one incinerator. These settings have been chosen to align to the literature which has usually one agent, one disposal site and a variable number of harvest nodes. However, we purposely vary the value of the parameters to show the behaviour of the framework under an increasing number of agents and nodes. In those tests the truck in use has been given a carrying capacity highly above the needed one, to simulate an infinity carrying capacity. The test environment settings are as follows:

- Number of collection nodes $\in \{4, 6, 8, 10, 12\}$. To these, other 3 nodes have to be added in the environment being 2 garages for the agents and 1 for the incinerator. At the end, the total number of nodes is $\{7, 9, 11, 13, 15\}$.
- Number of agents $\in \{1, 2, 3, 4\}$ plus one agent being the truck one, so in total the test had $\{2, 3, 4, 5\}$ agents.
- Number of working time steps $\in \{20, 25, 30\}$. These have been kept low and unreal due to the need of understanding how the system behaves depending by the nodes and agents in use, so such a coherence to reality was not needed.

Figure 5.1, 5.2 and 5.3 show how the model behaves when varying the number of agents and collection nodes, respectively. In particular, we focus on the number of constraints that are created by the model and hence how this enlarges.
Observing the plots, it is possible to understand how the formulation of the mathematical model into a linear one, allows to keep the number of constraints under control, without having it to grow exponentially and become rapidly intractable.
The only plot which shows a different behaviour is the one where we vary the number of nodes, as the models mainly depends on the connections between them. So, having randomly connected nodes may generate environments in which some nodes are strongly linked to others explaining why in some combinations of time steps and agents the number of constraints peaks.

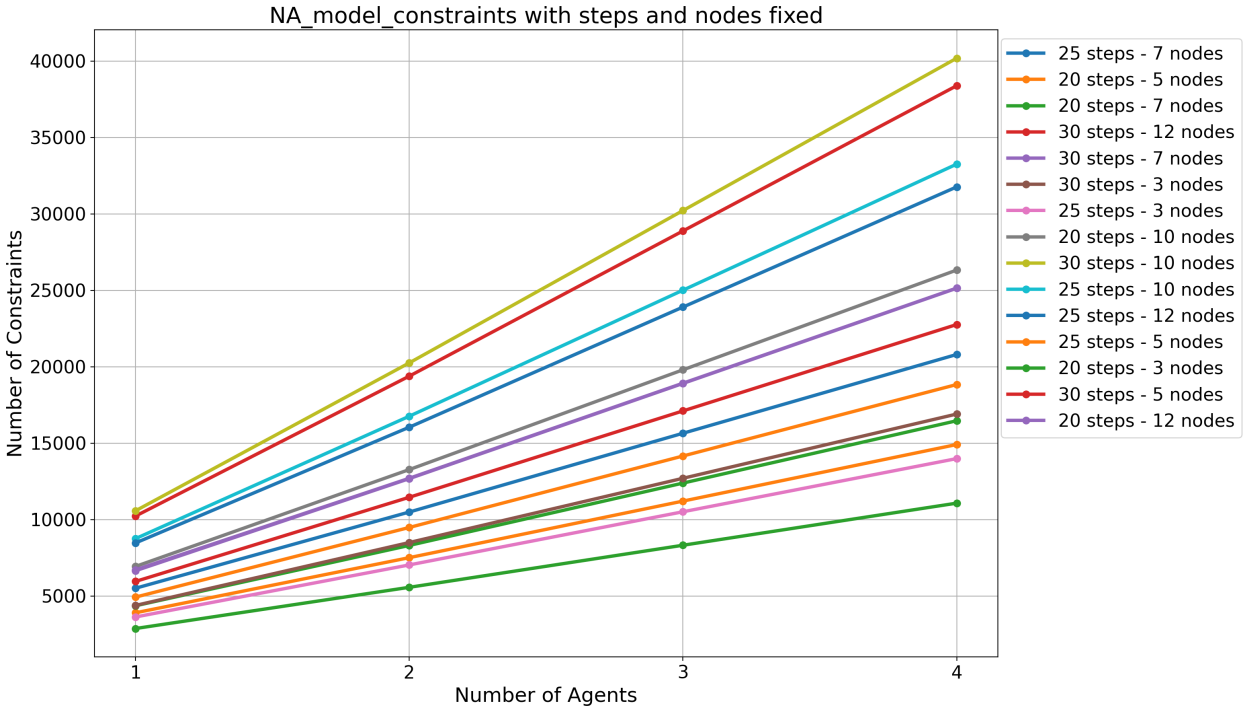


Figure 5.1: Number of constraints, varying the number of agents and with time steps and nodes fixed.

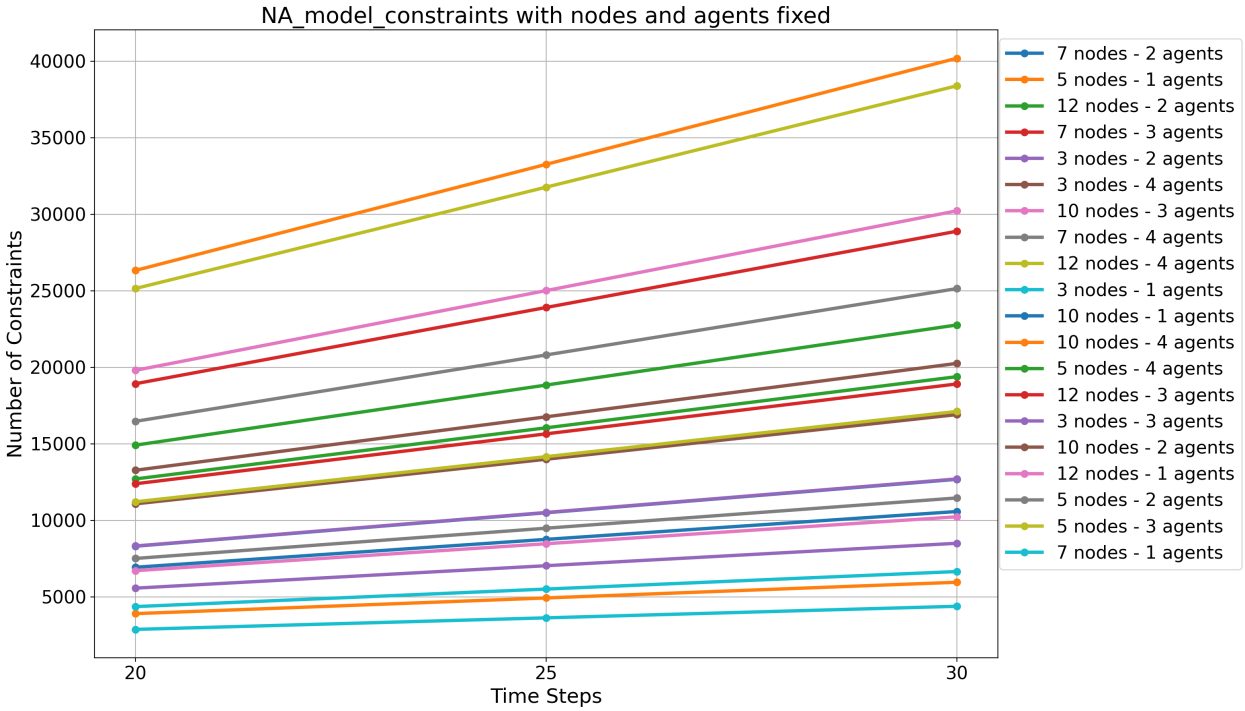


Figure 5.2: Number of constraints in the model, varying the number of nodes, but keeping the agents and working time fixed.

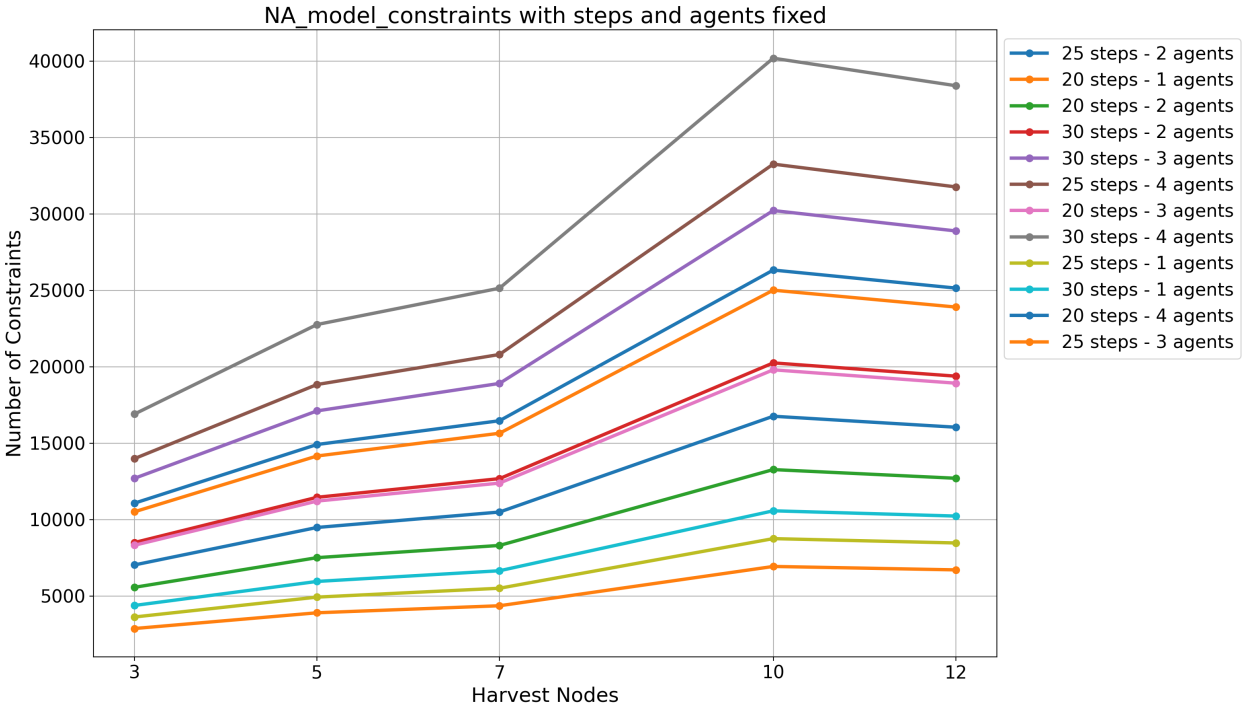


Figure 5.3: Number of constraints in the model, varying the number of nodes to collect, but keeping the number of agents and working time steps fixed.

We then considered the computational times and created tests varying the number of agent, of working time steps and of collection nodes. The results are shown in Figure 5.4, 5.5 and 5.6 respectively.
The plots, in response to the increasing number of agents and working time, exhibit no peculiar emergent behaviours, thus suggesting that the variation of a single one of these elements does not directly or heavily affect the system performances. Indeed the rise of computational time depends by the combination of the parameters. On the other hand, Figure 5.6 showing the computational time with respect the number of nodes, reveals an exponential behaviour with its highest peaks having 12 nodes to collect with 4 agents working respectively 25 and 30 time steps.

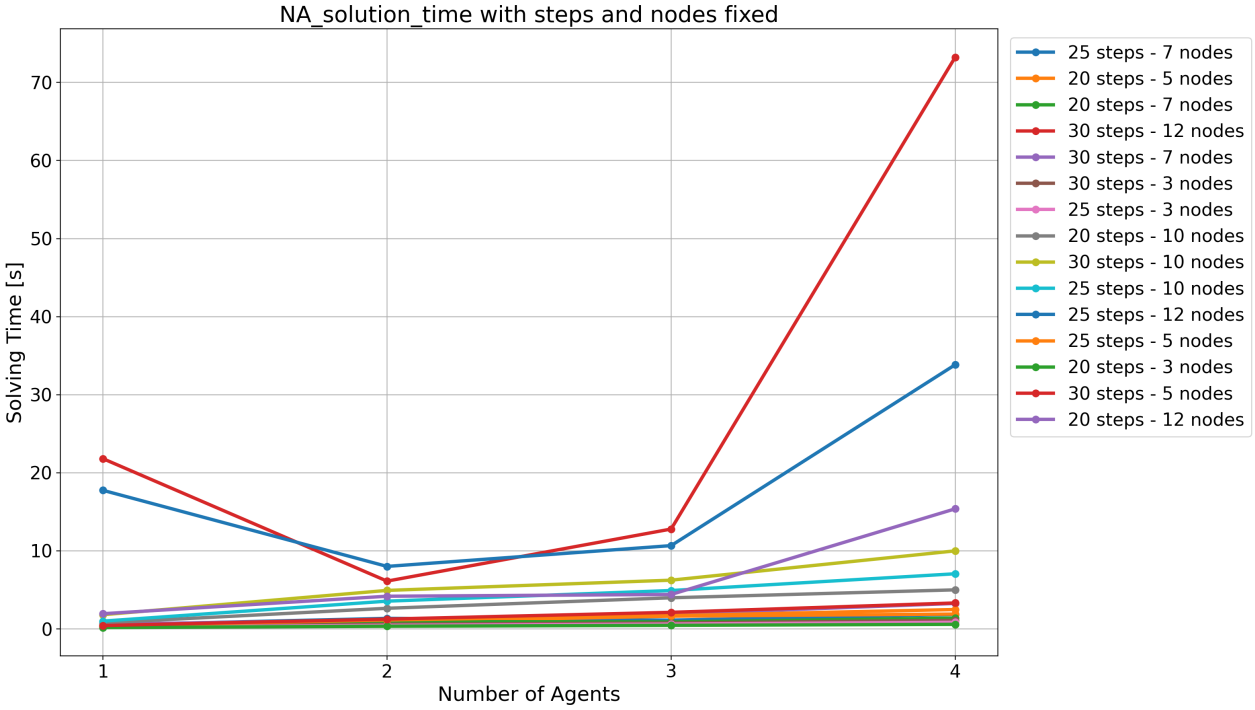


Figure 5.4: Solution time dependently by the number of agents keeping the number of collection nodes and working time steps fixed.

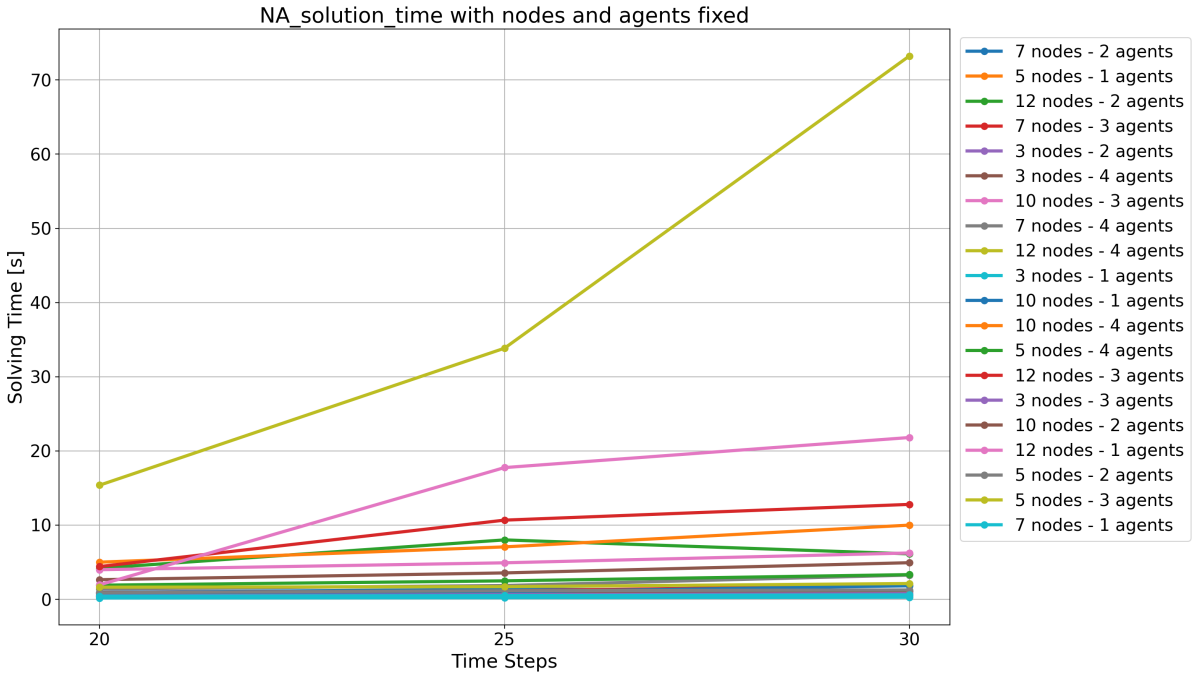


Figure 5.5: Solution time dependently by the number of working time steps keeping the number of nodes and agents fixed.

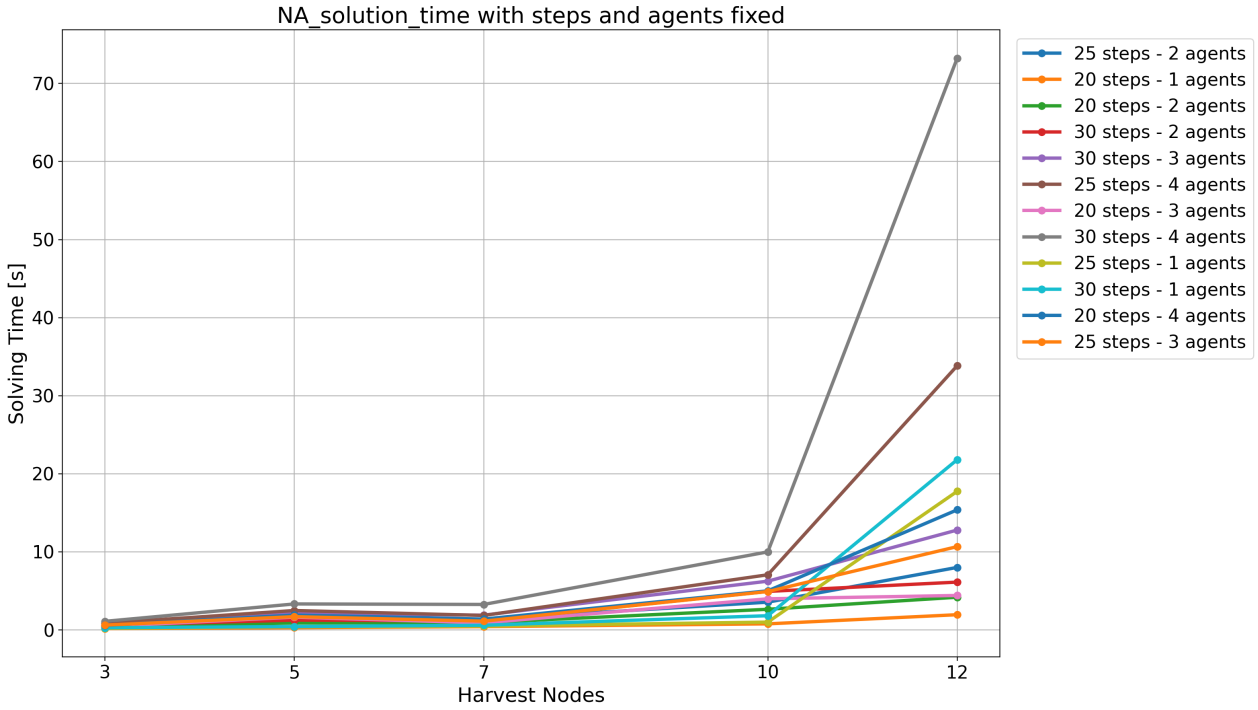


Figure 5.6: Solution time dependently by the number of collection nodes while keeping the number of working time steps and agents fixed.

To have a better understanding of the system time performances, we can look at the plots showing the computational time employing the Markov Clustering algorithm per agents available. In Figures 5.7, 5.8, 5.9, and 5.10 the time taken by the framework, to produce a solution, has been color coded and a logarithmic scale has been adopted for better visualization. Specifically the blue color is used to represent the time needed by the hierarchical planning without the clustering algorithm, the orange represents the time needed for hierarchical planning over the clustered environment and the green one is the time needed to refine the solution over the expanded environment.

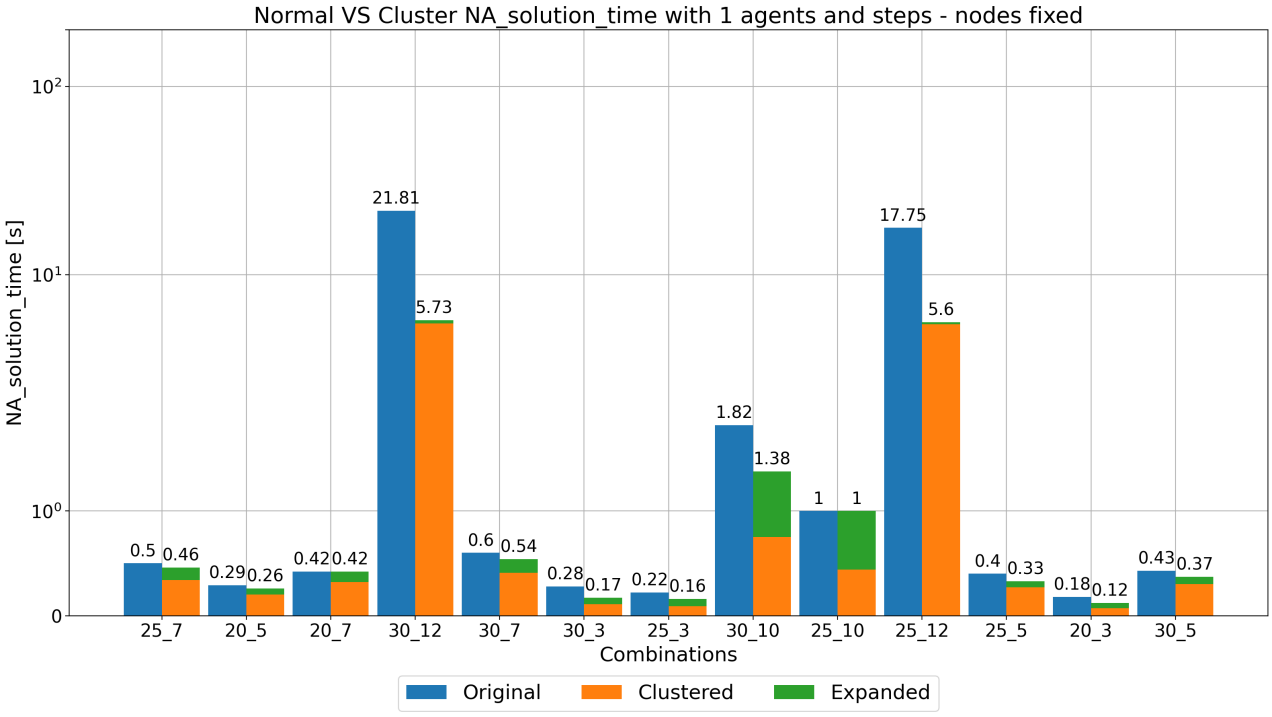


Figure 5.7: Markov clustering results having 1 agent in the environment.

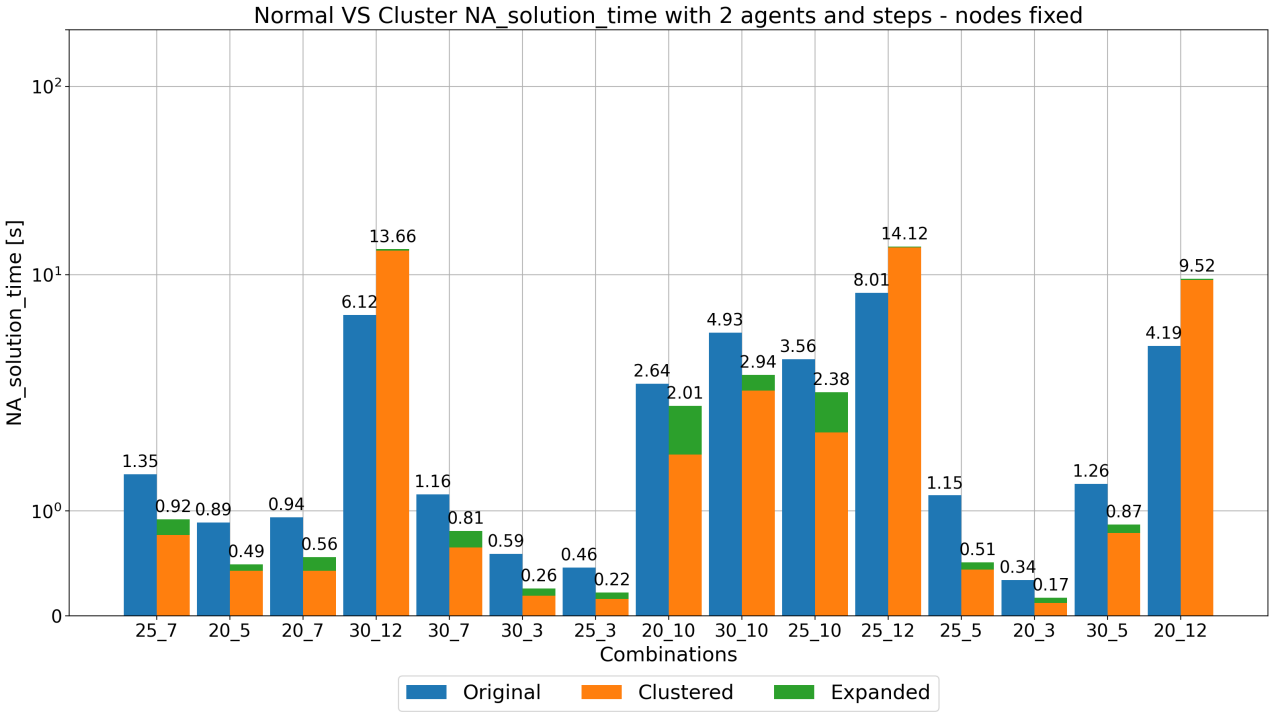


Figure 5.8: Markov clustering results having 2 agent in the environment.

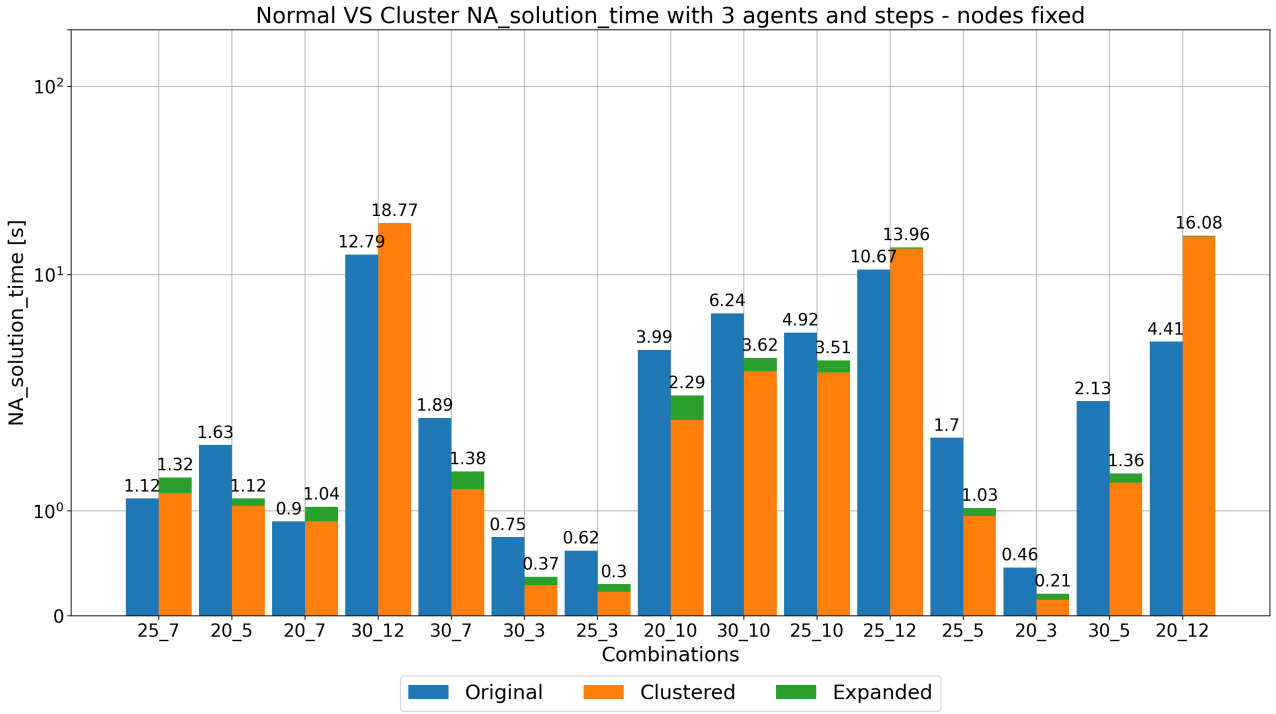


Figure 5.9: Markov clustering results having 3 agent in the environment.

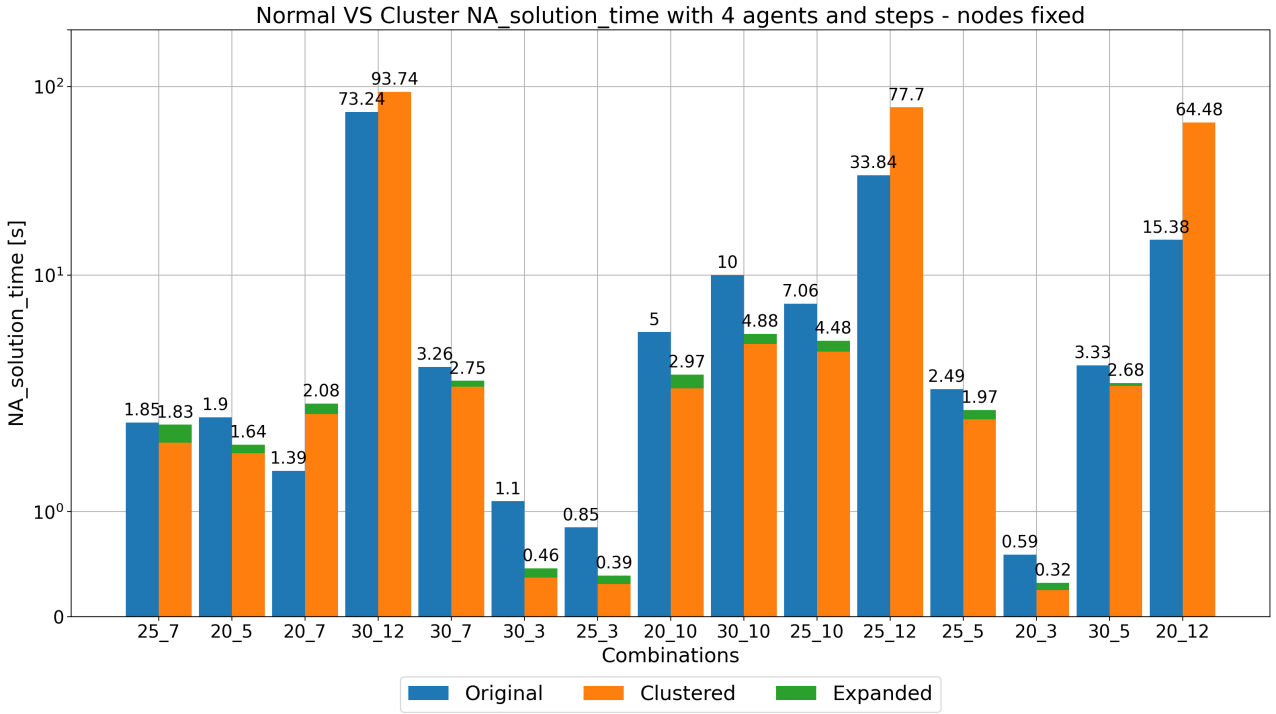


Figure 5.10: Markov clustering results having 4 agent in the environment.

Focusing on Figure 5.7 having just one agent, we can notice how the introduction of the clustering algorithm allows to have great improvements in the computational time, with the worst case never exceeding the clusterless hierarchical planning performances, and in the optimal case providing a reduction in time of over half the time needed by its previous implementation.

Moving to Figures 5.8, 5.9 and 5.10 having more than one agent, it is possible to notice some outliers cases in which the computational time of the clustering implementation dramatically exceeds the computational time required by just the hierarchical planning, while all the other settings show a sensitive reduction in computational time.

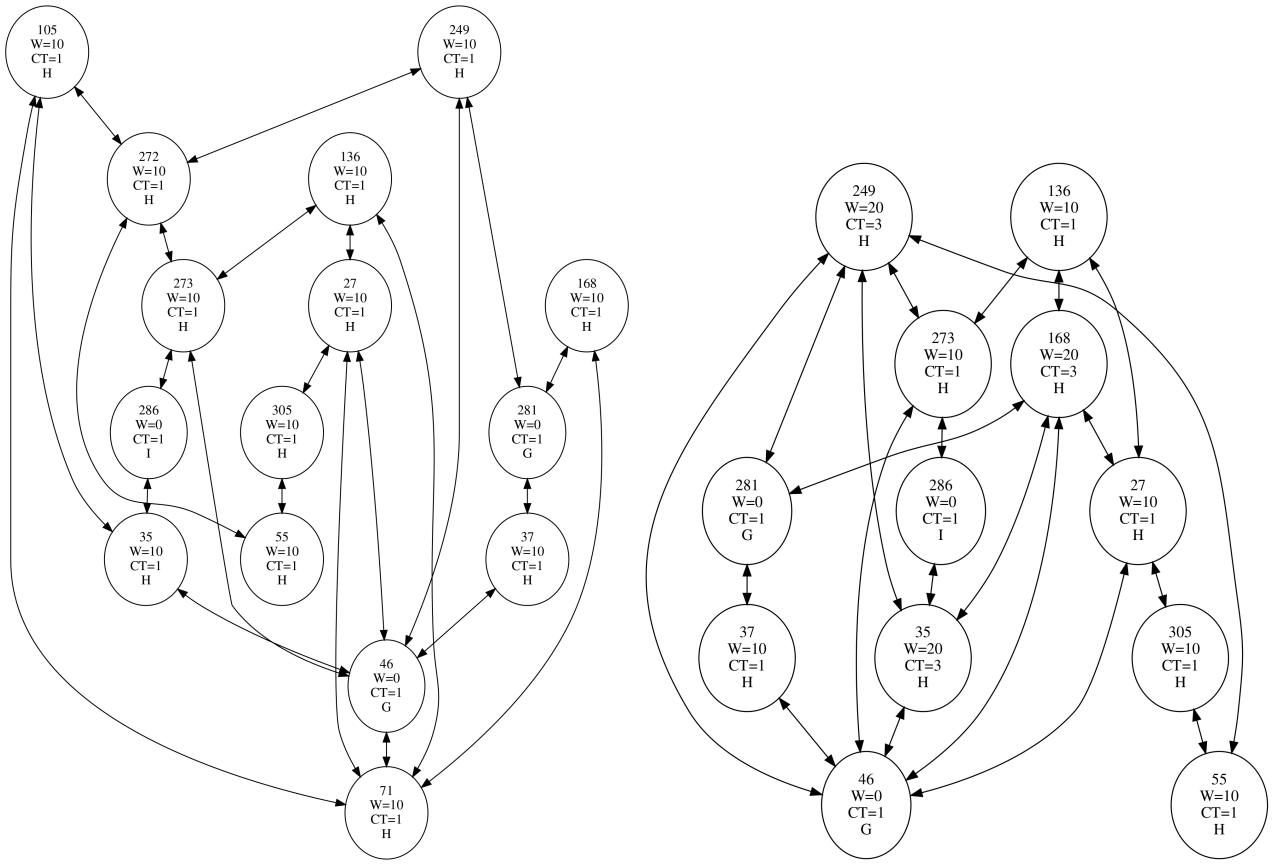
The reason behind this behaviour may be attributed to the graphs of the environments. For example, in the following diagrams, 5.11a and 5.11b, we compare the graphs for the hierarchical and clustering planning in the setting of the 12 collection nodes, which is the one exhibiting the outliers.

By comparing these graphs it is immediate to notice how the clustered one is more internally connected than its original version.

Consequently we can suppose that the clustering algorithm not knowing the resulting connectivity of the graph would merge nodes into new ones, that inheriting their links will result into a more connected environment.

This will then influence the solver by generating a search tree incredibly level-wide causing the rise of computational times.

Furthermore, having multiple agents in the scene, would additionally worsen the search tree and the solver performances being multiple locations for different agents, and group of agents, considered.



(a) Original graph for the 12 collection nodes case. (b) Clustered graph for the 12 collection nodes case.

6 VROOM Model

To validate the system and the model proposed, it was decided to confront it against another framework which implements the Capacitated Vehicle Routing Problem with and without Time Windows.
For such task the VROOM [10] open source framework has been chosen since it is able to support three different routing engines used to compute the costs to move from one node to another, and to support different cost values depending by the type of vehicles. Indeed the framework allows to define vehicle profiles to which associate custom cost matrices.
Furthermore it permits to define different tasks carried out by agents, addressed as jobs. There are three different categories of jobs:

1. Pickup: these jobs have several locations in which a certain amount of one or more goods has to be picked up by the agents.

2. Delivery: these jobs have agents that at the beginning of time have a certain amount of one or more goods loaded and must deliver the correct quantity to precise locations.

3. Shipment: These jobs consists in having an available agent pick up a certain amount of goods from a set of places and deliver it to the provided locations.

As already said one of the core capabilities of VROOM is that it works on top of several routing engines, among them it can uses OSMR [9] which is a routing engine queried by the algorithm to extract costs and travel times among locations directly from the OSM data, making it quite similar to our proposed system.

Before diving into the test results of VROOM, few clarifications have to be made.
First of all VROOM provides to the user the steps that have to be carried out in order to complete the jobs supplied, it checks consistency with the provided time windows, if any, but does not allow to plan for the agent to come back to their depots.
Furthermore, the environment modelled in VROOM is slightly different with respect to ours, due to the translation of the collection and discharge tasks into jobs elements. Specifically we used jobs of the type "shipment" to account for these operations, but no hierarchical planning was possible.
Indeed, employing our hierarchical planning implies knowing when the agents would discard and where, due to the time being unknown.
The only way to reach some results to compare was to discard the hierarchical planning and force the normal agents to pickup the garbage and empty directly into the truck agents, or incinerator, available as the ending points.
Even though this translation is not one-to-one, it still allow some degree of comparison between the systems.
To allow for a correct comparison between the models, it has been decided to provide to VROOM the cost matrix of the graph which will be used for inferring the structure of the environment, in accordance with our framework.

Following are shown the graph representing the average time taken by VROOM to provide a solution to the same batch of CPLEX, described previously in Section 5.2.

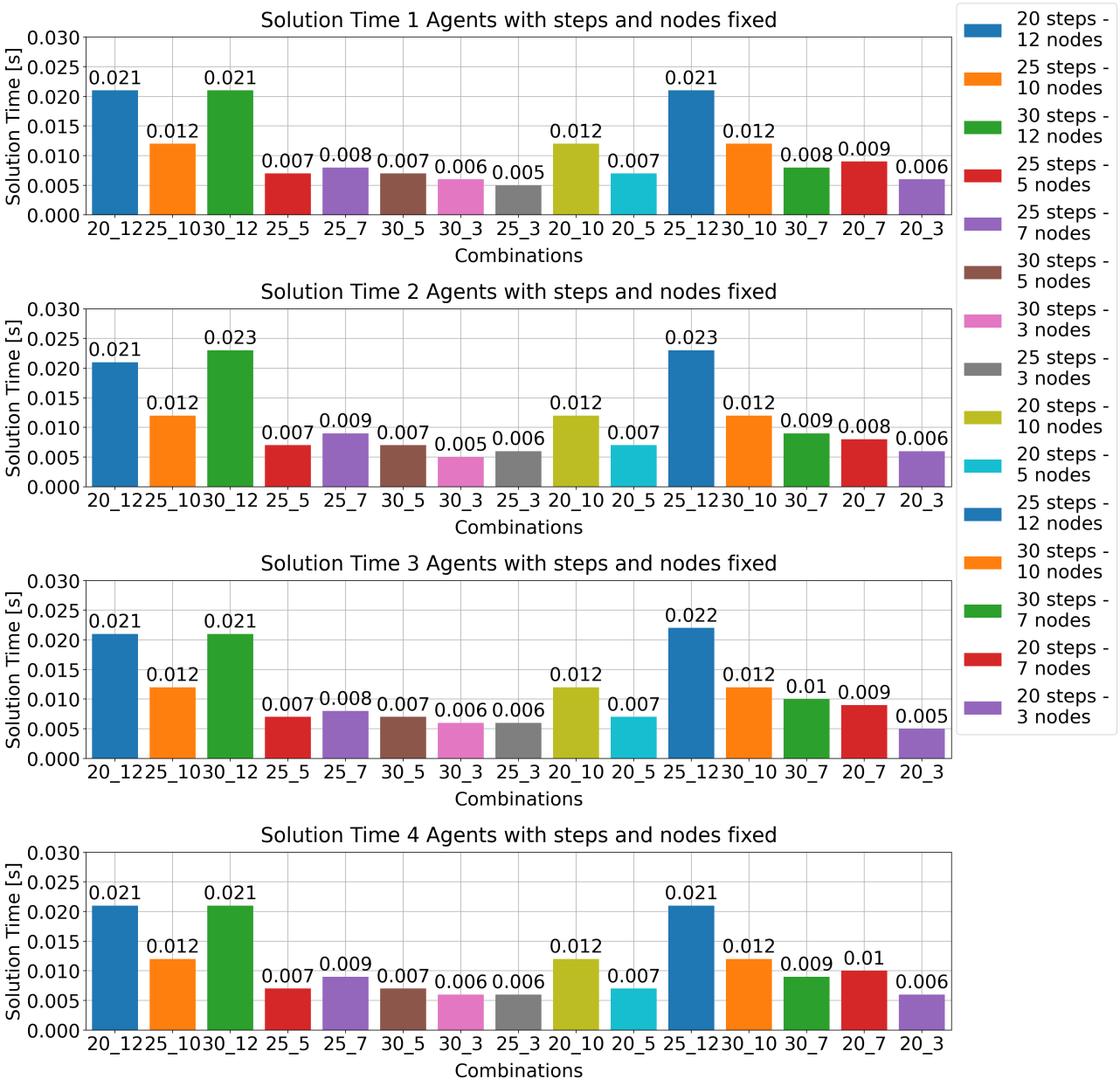


Figure 6.1: Solution time varying the agents, while keeping the working hours and nodes fixed.

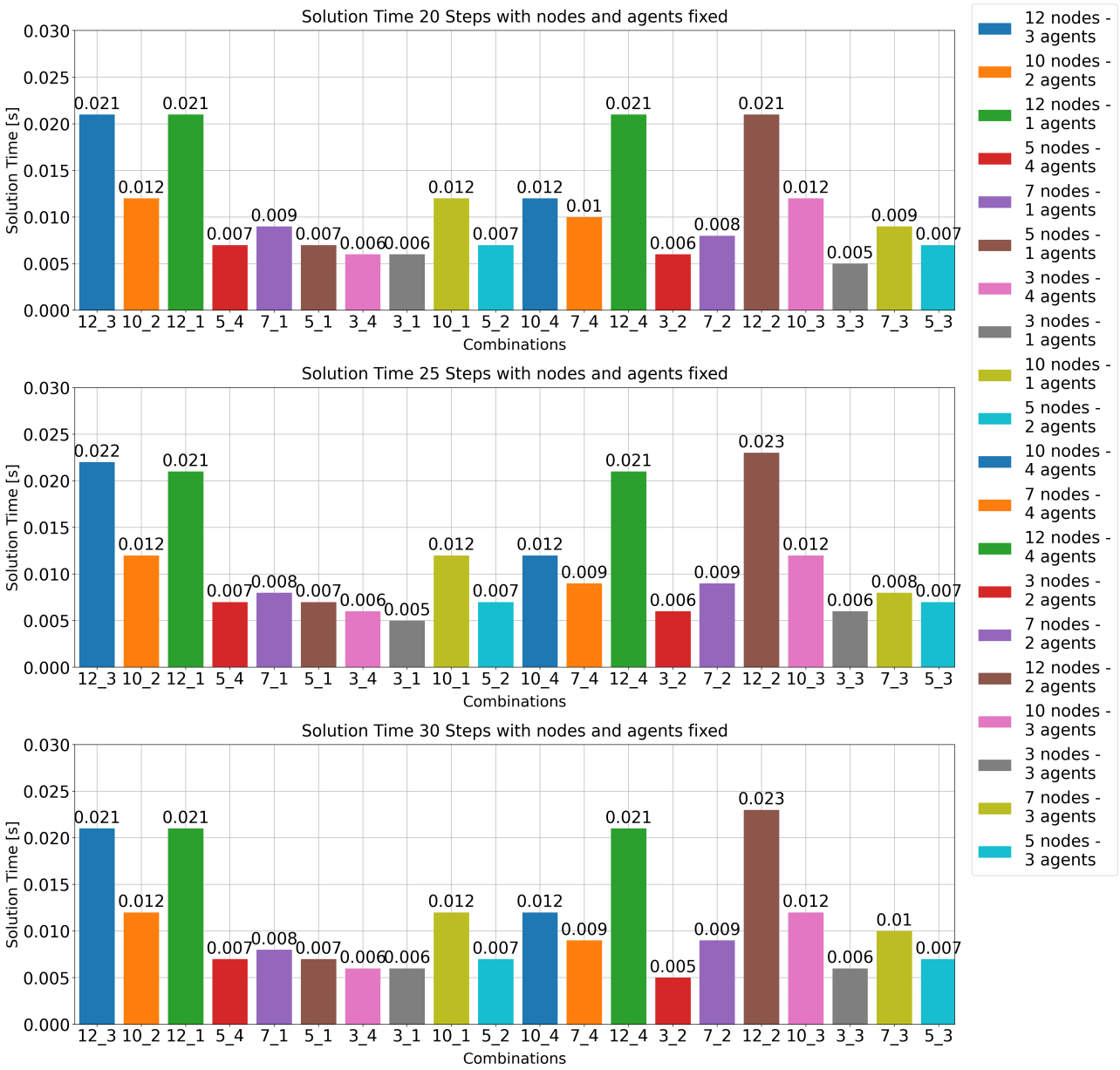


Figure 6.2: Solution time varying the amount of working hours and keeping the amount of nodes and agents fixed.

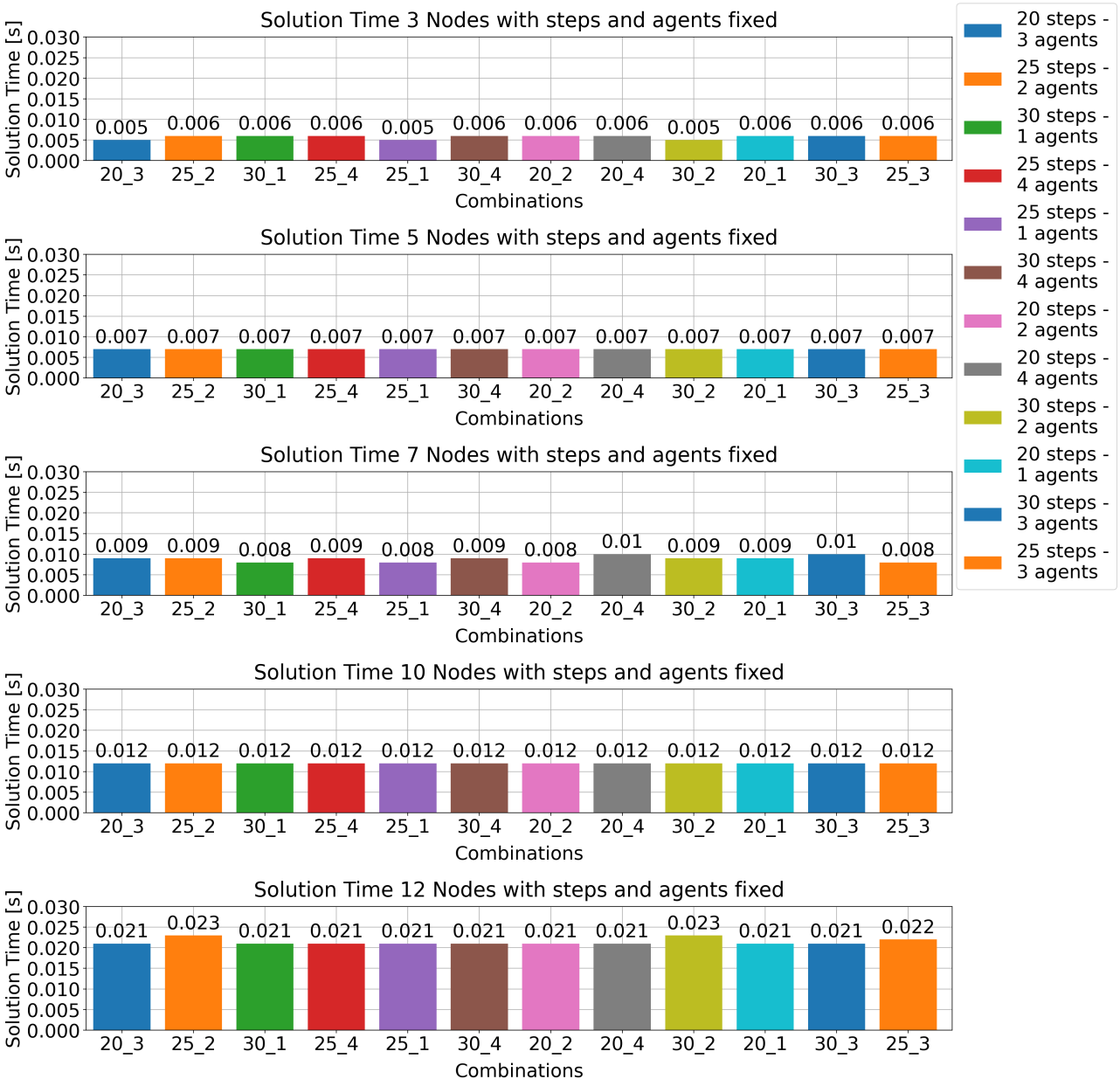


Figure 6.3: Solution time varying the nodes, while keeping the number of working hours and agents fixed.

As observable in Figures 6.1 and 6.2 the time needed to get to a solution express almost the same behaviour varying the number of agents and the working hours, respectively. The changes in time are due to the different combinations in the test environment, having the values repeating across some of them, specifically when the same amount of nodes is being used. Thus we can infer that in VROOM, the amount of time steps and agents partially affect the time required to find a solution. On the other hand, Figure 6.3, shows an increasing demand of computational time when varying the amount of nodes to be collected, while keeping the same amount of collection points result into the same time requirements across all the available combinations. Furthermore, we can notice how the increase in time resembles an exponential curve, being the values almost doubling when the number of collection nodes exceeds 7. This behaviour is line with the CPLEX model as previously shown in Figure 5.6. However, even tough the response of both system in dependency on the nodes available is the same, the computational time is incredibly lower with VROOM than CPLEX, indeed comparing the time demand for the 12 nodes environment we get that VROOM requires around 0.022 seconds, while CPLEX is between a minimum of circa 2 seconds to a maximum of 21 in the case with one agent.

Even though huge savings in time are achieved, the model of VROOM does not optimize the workload between the available agents, being empirically proven that it tends to exploit the same agent instead of splitting the collection operations.

Indeed, by inspecting the results retrieved from the test batch data, the solutions obtained exploits the same agent over and over while our system balances the work load among the available actors to end the task as soon as possible, as portrayed in Table 6.1. Specifically, the table shows how CPLEX splitting the workload is able to end the collection task with consequent come back travel in at most 6 time steps, while VROOM needed 11.

Furthermore, VROOM provides the steps to perform in order to complete the task correctly while our framework returns to the user the expected timing of start and end of the operations and the locations of the agents at each step. Specifically, looking at the elements in Table 6.1, we can notice that CPLEX provides in output a sequence of steps consistent with the graph, while VROOM limits itself in stating the visiting order of the nodes, thus lengthening the task if we consider the movements in between the nodes. For example, the first action undertaken by VROOM is to move from node 167 to 66, but looking at the graph and to the behaviour of agent C in CPLEX, we can notice how 66 is reachable only from nodes 101, 180 and 43 meaning that the path suggested by VROOM is at least 1 time step longer.

In conclusion, VROOM is a very fast tool which can be useful in understanding which operations to perform to complete the tasks efficiently, but for a system as ours some fine tuning is still needed. Precisely, a change into the evaluation of the best value for the cost function has been made, to allow VROOM to reduce the amount of travels back and forth the disposal sites. This behaviour was due to the internal cost function of VROOM for which getting back to the disposal time a certain amount of times leads to the same cost of getting there in the least number of times possible.

CPLEX			
Agents	Time	Operation	Capacity
A	0	"167"	0/100
	1	C("43")	10/100
	2	C("180")	20/100
	3	"43"	20/100
	4	D("303")	0/100
	5	"101"	0/100
	6…19	"167"	0/100
B	0…19	"167"	0/100
C	0	"167"	0/100
	1	C("101")	10/100
	2	C("66")	20/100
	3	"101"	20/100
	4	D("303")	0/100
	5	"221"	0/100
	6…19	"167"	0/100
D	0	"167"	0/100
	1	C("221")	10/100
	2	D("303")	0/100
	3	"101"	0/100
	4…19	"167"	0/100

VROOM			
Agents	Time	Operation	Capacity
A	0	"167"	0/100
	1	C("66")	10/100
	2	C("221")	20/100
	3	C("180")	30/100
	4	C("101")	40/100
	5	D("303")	30/100
	6	D("303")	20/100
	7	C("43")	30/100
	8	D("303")	20/100
	9	D("303")	10/100
	10	D("303")	0/100
	11…19	"167"	0/100
B	0…19	"167"	0/100
C	0…19	"167"	0/100
D	0…19	"167"	0/100

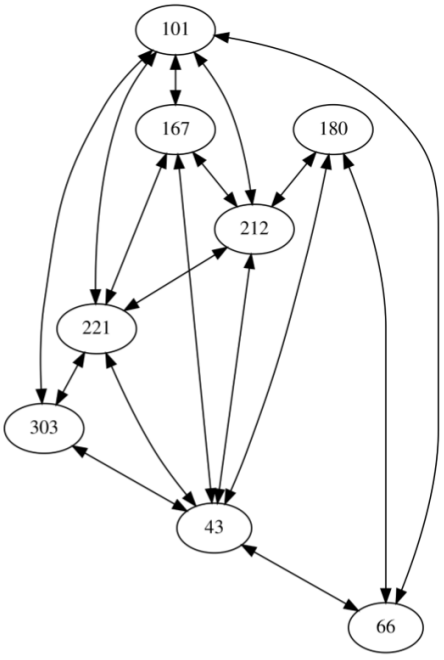


Table 6.1: Comparison between the plans of CPLEX and VROOM in a setting having 4 agents, 5 collection nodes and 20 time steps. The collection operations are identified with a "C" before the node id, while the discharge ones are represented by a "D". Whenever an agent moves on a node without performing any action, only the id of the node is written.

7 Conclusion

In this work it was addressed the design and implementation of a framework able to optimize the municipal waste management task focusing to optimize the available resources and the time needed to find a solution.

First of all we discussed about the data available and how to exploit them to represent the environment for the collection task. For this purpose the Open Street Map data were chosen being completely free to use and updated through a community, accounting for reliability in tackling the road network updates.

Exploiting this data, we were able to infer the location of the garbage by relating its position to the buildings on the territory and mapping them to the road network. Having the buildings locations a graph was built exploiting the garbage locations, garages and incinerators as nodes, while the roads supplying them were interpreted as the edges. Knowing the graph representation and some custom settings provided by the user, a complete mathematical model of the waste management task has been built, taking into account the agents motion pattern, the different depots for the agents, the several disposal points, the collection time for the nodes having garbage and the discharge time in the disposal locations.

The model was built exploiting the Constraint Programming paradigm, aiming at generating a Mixed Integer Problem with a number of constraints linearly growing with the input parameters. Due to the number of constraints in use and the large amount of collection nodes in the environment, it was immediately clear that the computational time required to provide a solution would have been the main challenge in the system and the first measure of efficiency.
To avoid such drawback it was decided to exploit the hierarchical planning strategy to split the waste collection and disposal operations into two independent tasks.
In an attempt to further decrease the computational time, also the Markov clustering algorithm was employed to narrow down the amount of collection nodes, being it highly focused over the internal connectivity of the graph.
Those changes allowed the framework to provide solutions in lesser computational time than its original implementation, although comparing our system with an open source general purpose optimization engine like VROOM showed that its competitor runs in lesser time, but provides solutions without balancing the workload, and which does not account for return travels to the initial depots.
In conclusion, our system has proven to optimize the path of the input agents and the resources available, but with high computational requirements. The time need has been observed being exponential in the number of nodes to collect.

We are confident enough, that our model with some future fine tuning in the mathematical representation and in the algorithm for clustering may become a useful tool providing better solutions to all the municipal waste collection problems.

8 Future Work

In the future we plan to improve our framework under several aspects.
First of all we aim at achieving a model fully aware of the environment in use and of the consequences of the planned actions. In doing so, we will need to develop a part of the system which, provided in input the data about the vehicles operators, is able to determine the crew onboard of the agents depending by the weekly and monthly working hours.

Furthermore we will devise a tool able to forecast the amount of garbage on the territory and the traffic amount in the road network, both depending on the year period and on the previous data available to the company. For this branch of the framework it will be needed to employ machine learning and deep learning techniques aiming at dropping the prediction error and in correctly generating an estimate fully exploiting both the past data and the near future events.
Reaching such level of accuracy, will allow the system to manage the agents, crew and plans in such a way to avoid traffic queues either due to maintenance work on the road networks, or to pubblic events, or just due to to an increased amount of people coming to the city for tourism in specific periods of the year. Indeed, it is well known how the garbage production increases around the festivity days, in the summer time, and when public events occur. In the same way, traffic queues rise easily in almost the same periods of the garbage production peaks, stimulated by the improved amount of people in the core areas of the cities and in the tourist facilities spread across the territory.

Lastly, we must improve the framework computational capabilities, and to achieve this we may employ genetic algorithms. Specifically, these will be applied over an initial solution provided by CPLEX, working on a relaxed problem, or even from the output of VROOM.
By employing this family of algorithms, we aim at dropping the computation time by reaching a solution that, will be as close as possible to the best one, even if it will not be optimal. Practically we will trade off the optimally of the solution for reduced computation times.
Another option to improve the time in processing, may also be to exploit VROOM to get an initial ordered list of nodes to visit, which may be translated into a new set of constraints for the CPLEX model, stating a sort of priority among the nodes to cut the vast majority of the search tree, reducing the time needed to parse it.

Bibliography

[1] E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94, 1974.

[2] Mehmet Erdem. Optimisation of sustainable urban recycling waste collection and routing with heterogeneous electric vehicles. *Sustainable Cities and Society*, 80:103785, 2022.

[3] ISPAT. Statistiche popolazione. `http://www.statistica.provincia.tn.it/statistiche/societa/popolazione/`.

[4] ISPRA. Consumo di suolo. `https://groupware.sinanet.isprambiente.it/uso-copertura-e-consumo-di-suolo/library/consumo-di-suolo/indicatori`.

[5] ISPRA. Presentazione del rapporto rifiuti urbani edizione 2022. `https://www.isprambiente.gov.it/it/archivio/eventi/2022/12/presentazione-del-rapporto-rifiuti-urbani-edizione-2022`.

[6] ISPRA. Produzione rifiuti urbani. `https://annuario.isprambiente.it/sys_ind/434`.

[7] Byung-In Kim, Seongbae Kim, and Surya Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006. Part Special Issue: Recent Algorithmic Advances for Arc Routing Problems.

[8] Aleksander Król, Piotr Nowakowski, and Bogna Mrówczyńska. How to improve weee management? novel approach in mobile collection with application of artificial intelligence. *Waste Management*, 50:222–233, 2016.

[9] Dennis Luxen and Christian Vetter. Real-time routing with openstreetmap data, 2011.

[10] Vehicle Routing Open Source Optimization Machine. `https://github.com/VROOM-Project/vroom`.

[11] OpenStreetMap. `https://wiki.osmfoundation.org/wiki/Main_Page`.

[12] Masoud Rabbani, S. Amirhossein Sadati, and Hamed Farrokhi-Asl. Incorporating location routing model and decision making techniques in industrial waste management: Application in the automotive industry. *Computers & Industrial Engineering*, 148:106692, 2020.

[13] Ana Maria Rodrigues and José Soeiro Ferreira. Waste collection routing—limited multiple landfills and heterogeneous fleet. *Networks*, 65(2):155–165, 2015.

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

[15] M. L. Tee, K. Y. Wong, and D. E. Cruz. Goal programming approach of a multi-vehicle routing problem on waste collection considering economic, environmental, time, and health objectives. In *2022 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 0906–0910, 2022.

[16] João Teixeira, António Pais Antunes, and Jorge Pinho de Sousa. Recyclable waste collection planning—a case study. *European Journal of Operational Research*, 158(3):543–554, 2004.

[17] Erfan Tirkolaee, Alireza Goli, Selma Gütmen, Gerhard-Wilhelm Weber, and Katarzyna Szwedzka. A novel model for sustainable waste collection arc routing problem: Pareto-based algorithms. *Annals of Operations Research*, pages 1–26, 01 2022.

[18] Erfan Babaee Tirkolaee, Parvin Abbasian, Mehdi Soltani, and Seyed Ali Ghaffarian. Developing an applied algorithm for multi-trip vehicle routing problem with time windows in urban waste collection: A case study. *Waste Management & Research*, 37(1_suppl):4–13, 2019. PMID: 30761957.