Knowledge Intensive Evolutionary Algorithms

François de Bertrand de Beuvron*, Carlos Catania* and Cecilia Zanni-Merk*
ICube laboratory, BFO team,
INSA de Strasbourg, CNRS
France
Email: {debeuvron, catania, merk}@unistra.fr

***Abstract*—In this paper, we show through the resolution of a real problem, how knowledge engineering techniques can be used to guide the definition of Evolutionary Algorithms (EA) for problems involving a large amount of structured data. Evolutionary Algorithms have proven to be very effective in optimizing intractable problems in many areas. Various representations of the fitness functions (multi-objective EA), the genome and mutation / crossover operators adapted to different types of problems (routing, scheduling, etc.) have been proposed in the literature. However, real problems including specific constraints (legal restrictions, specific usages, etc.) are often overlooked by the proposed generic models. To ensure that these constraints are effectively taken into account, we propose a methodology based on the structuring of the conceptual model underlying the problem, creating a domain ontology suitable for optimization by EA. The real-world example, that is detailed throughout the article, belongs to the general field of medical assistance. The project focuses on the logistics involved in the transportation of the patients. Although this problem is a specific case of the heavily studied family of Vehicle Routing Problems (VRP), its specificity comes from the amount of data and constraints: in addition to costs, many legal or health considerations must be taken into account. Our approach is based on the development of a multi-objective genetic algorithm, which has to come up with the best itinerary taking all these constraints into account. We will show that a precise definition of the knowledge model with a domain ontology can be used to describe the chromosome, the evaluation functions, the crossover and mutation operators.**

Keywords–Knowledge engineering, multi-objective optimization problems, evolutionary algorithms

I. INTRODUCTION

In this article, we will show that the use of knowledge engineering can greatly enhance the definition of an evolutionary algorithm for a real case. This study is the result of a collaboration between our team and an Alsatian SME that provides real data to deal with.

The project is developed under a healthcare system environment, specifically oriented to the transportation of patients, normally from or to some healthcare centre. The needs for developing an application arrives because of the fact that the enterprises, which take care of the logistics of the journeys have to manage a big amount of requirements and constrains at the moment of making an itinerary. This logistics affects many enterprise resources, like the employees and vehicles, which should be assigned in an efficient way in order to guarantee, among other things, the conformity of the patient and the satisfaction of certain law regulations.

The problem consists on satisfying the daily requests of the patients minimizing the costs and fulfilling certain constrains. The requests are basically for pick-ups and/or deliveries of the patients to or from their house to some healthcare centre. There are different types of vehicles that can accomplish a journey and each of them has an associated cost. There are also the costs of affecting a crew, meaning a set of one or two employees, to a certain vehicle or to a certain patient.

Many studies concern the Vehicle Routing Problem (VRP) [1]. This large number of studies place themselves along two axes:

- the solving approach; mainly exact algorithms or meta-heuristics (stochastic or nature inspired algorithms) [2].
- the variants of the problem; including time windows constraints [3] or multiple heterogeneous vehicles with pickup and delivery [4], among others.

We have chosen to use Evolutionary Algorithms (EAs) for the specific problem to solve, since we will show in this article that some families of EA are particularly suitable for a knowledge driven definition.

Even if this problem belongs to the family of Vehicle Routing Problems, it soon became clear that the solutions proposed in the literature, as specific as they are [5][6][7], were not intended to take into account all the specific constraints of the problem. In addition, we believe that this situation is found in many optimization problems when the parameters are numerous and varied in nature. Thus, in our example, legal constraints such as the working time of ambulance attendants, or medical constraints such as the disinfection of the vehicles, or the personnel qualification, cannot be overlooked while minimizing costs or distances.

The entire legacy software environment, in particular the conceptual model of the information system, represents a body of knowledge and skills within the company, which should be used in the implementation of the optimization module. But strangely enough, we did not find in the literature any framework taking all of these needs into account. In fact, in the early days of genetic algorithms, much emphasis was put in the domain-independent nature of the basic crossover and mutation algorithms, working on a standard binary coded, fixed length chromosome [8]. The foundation of genetic algorithms relies on the exploitation of similarities (building blocks or schemas) in the chromosome. Using a specific chromosome representation and operators that emphasize meaningful building blocks in the application domain should improve the efficiency, as highlighted by C. Janikow in [9]. This general idea has led to numerous specific genetic algorithms based on domain knowledge. For example, in our problem, the distribution of tasks between ambulances can be seen as a special case of a set partition problem. This kind of problem led to the definition of specialized so-called Grouping Genetic Algorithms, and it is no surprising that some further specialization has produced very efficient genetic algorithms for the pick-up and delivery problem [10]. In addition to the coding of the chromosome, knowledge may be used in other parts of a genetic algorithm: in the the fitness function or the initial population, or to keep collective knowledge among individuals (Cultural Algorithms). A survey about such use of knowledge for the development of highly specialized EAs can be found in [11].

Our goal is different: we want to use domain knowledge to assist in the definition of the algorithm. That is why we propose a methodology centred on an extended domain ontology for the definition of evolutionary algorithms whose parameters and constraints represent a huge volume of structured data (this is what we call *Knowledge-Intensive Evolutionary Algorithms*).

The rest of the paper is structured as follows. In Section II, we present the main steps of the methodology we propose. In Section III, we detail the way of building an extended conceptual model in order to link the domain ontology with the EA own constructions. We then see how this structuring of the conceptual model can be used to define the evaluation functions (Section IV) then the chromosome, and associated mutation/crossover operators (Section V), ensuring that all the specific constraints of the problem are taken into account. Finally, we present in Section VI the preliminary results on the ambulance routing problem, before giving some conclusions and perspectives.

II. Methodology overview

We propose a methodology in three phases (Figure 1).

A. Analysis

This phase consists in defining precisely the function to be optimized through a specific labelling of the domain ontology. The project objectives are identified by a team of domain experts, and a comprehensive list of constraints and costs related to the realization of these objectives is identified. A team of domain experts, incorporating expertise in Knowledge Engineering and in Evolutionary Algorithms is then formed. The aim is to link the goals / costs / constraints in the project with the generic concepts of EAs (evaluation functions, representation of the genome, mutation, and crossover) through a specific labelling of the domain entities. Four general, overlapping categories of domain entities are defined:

1) *Entities to Optimize*: entities whose values are to be determined by the optimization algorithm (e.g., which ambulance will take care of each client). The structure of the chromosome will largely depend on these entities and their relationships. It prefigures the output data structure of the evolutionary algorithm.
2) *Parameter Entities*: entities whose values act as costs or constraints (e.g., cost/km for a vehicle, legal maximum number of daily driving for an ambulance attendant). The calculation of the fitness functions depends on these entities. They prefigure the structure of the input data.
3) *Evaluation Entities*: entities whose values are objectives to optimize (e.g., minimize the total number of kilometres travelled by all ambulances or maximize the benefits). It must be ensured that each of these goals is represented by one or more fitness functions in the evolutionary algorithm.

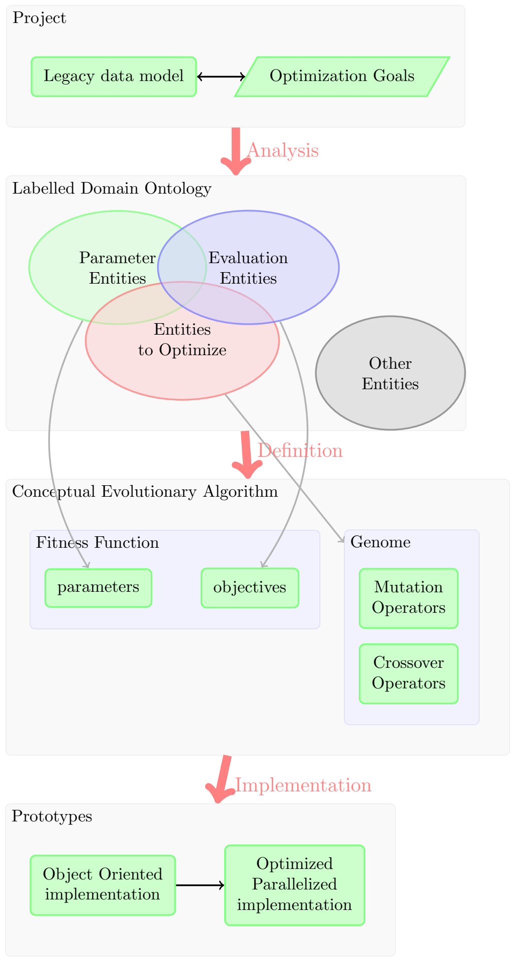


Figure 1. Methodology overview

4) *Other Entities*: entities that are not involved in the optimization (e.g., history of customer calls to the customer call centre).

The first three categories overlap. In fact, the membership of an entity to a category depends mainly on the usage of the values of its properties in the optimization problem. Some of the entities in the application domain may be “heterogeneous" when not all their properties serve the same purpose.

In the frequent case where the function to be optimized must be integrated into a larger existing system, most entities already exist in its data model. Most often, however, new entities will have to be introduced to have a more fine-grained representation of the entities to optimize.

We describe in Section 3 a specific labelling of the domain ontology to specify these categories.

B. Definition

From the labelled domain ontology defined in the analysis phase, the fitness functions, the structure of the chromosome and the associated evolutionary operators must now be defined. This process will be described in Section 4 for the fitness functions and in Section 5 for the genome and operators of the genetic operators.

C. Implementation

A first prototype using the EASEA platform [12], [13] was developed to test the feasibility of the approach. However, it soon become clear that it will not be sufficient to solve real problems. Limitiations come from:

- the total memory footprint ($genomeSize \times populationSize$) is too large to be handled by a single computer
- the convergence time to an acceptable solution is too long.

For our application, the convergence time is critical: ambulance itineraries cannot be fully calculated off-line, because waiting times and consultation durations are largely unpredictable, and new journeys may be requested at any time, some of them urgent. The system must be able to take these changes into account and propose a new schedule in about one minute. Only a massively parallel implementation as proposed also by the EASEA platform can achieve such an efficiency.

Section 6 presents the evaluation of the the first prototype. Notice that the highly specialized technical details concerning the parallel implementation are not discussed in this article.

III. Structuring knowledge for evolutionary algorithms

The data model needs to deal with a huge quantity of information, taking into account all the constraints. Given the complexity of the data model, we have decided to formalize it as a domain ontology. The ontology will guide the definition of the evolutionary algorithm thanks to the formal relationships that appear among the entities in it.

An ontology is a knowledge representation of a domain or a field that provides conceptual resources for knowledge-based systems (KBS). It gathers and defines the set of objects that are known as belonging to the domain [14].

In general, an ontology is composed of entities sometimes called *concepts* or *classes* and relationships between these entities usually called *roles*, *properties*, or *attributes* if they are mono-valued. Within this paper we used *entity* and *property*. In fact, ontologies provide the conceptual and notional resources needed for knowledge formulation and for making knowledge explicit. Our domain ontology formalizes the main concepts concerning our problem, such as vehicles, crews, patients, addresses, journeys and so on (see entities below *DomainEntity* in Figure 2).

One of the essential components of the routing problem is $PlannedElement$. It is an event to be held at a given location ($Address$), which should start at a desired time ($requestedDate$), and should last a known or estimated time ($duration$). There are many subtypes of $PlannedElement$, some of them are shown in Figure 2:

- $PlannedEmployeeElement$: events related to an employee. For example, such employee shall be home by noon. On the same principle, there are also $PlannedVehicleElement$ events related to a vehicle: the vehicle must be revised or disinfected, etc.
- $PlannedFleetElement$: an event that is not directly dependent on a vehicle or on a particular employee. The optimization algorithm will have to determine which vehicle driven by which employees ($Crew$) will be assigned to the event. There are again two subtypes of $PlannedFleetElement$:
    - $BusinessFleetElement$: this is the the most classical event: a patient must be collected or deposited somewhere. These events are paired within a $Journey$, which includes the collection, the ride and the deposit of the patient. Of course, a patient who was picked up by an ambulance must be deposited by the same one. The optimization will obviously have to take this basic constraint into account. The $Journey$ is associated to $PlannedElement$ and not directly to a $BusinessFleetElement$ since constraints related to the vehicle may be associated with $Journey$: for example, for some infectious patients, the ambulance must be disinfected immediately after the ride.
    - $InternalFleetElement$: these are constraints that are not directly related to the patients (e.g., fetching a document in a hospital).

The assignment of the events to individual vehicles is the main goal of the optimization. A $PlanningLine$ represents the list, ordered by increasing time, of the events supported by a given vehicle. A complete $Planning$ is simply the set of the $PlanningLine$ for all the vehicles.

To optimize itineraries, notions of distance and travel time are crucial. They are represented by the $Distances$ and Time for the Next (TFN) entities. These data are huge: the ambulances of a large company may have to visit several thousands of addresses per day. As evolutionary algorithms may randomly test any path, the distance between any two addresses must be known or estimated. This is even more critical for the travel time, which usually depends on the time of day (see Section IV for more details).

Finally, the algorithm must take into account many additional constraints. For the sake of simplicity, only two appear in Figure 2:

- $CostCustomerVehicule$: vehicles are more or less suitable for patients. A patient in a wheelchair is easier to take care of (soft constraint) in a adapted vehicle. A patient who must travel lying down requires (hard constraint) a real ambulance.
- $CostCustomerEmployee$: common language, personal preferences, among others.

The ontology for Evolutionary Algoritms (EA) ontology provides generic, domain independent, entities and properties allowing to define what use will be made of entities and properties of the domain in the genetic algorithm. The EA ontology is therefore a characterization of the entities and properties of the domain ontology seen as individuals. A precise modelling would require for the EA ontology to be defined at a meta-model level over the domain ontology. The separation of entities into four categories as proposed in Section II should be represented as specializations of the meta-entity "Entity" itself. Similarly, a parameter used in the fitness function should be explicitly linked to some entities or properties in the application domain. For example, in the application domain of ambulances, the specific evaluation function "minimizing travel distances" should be connected

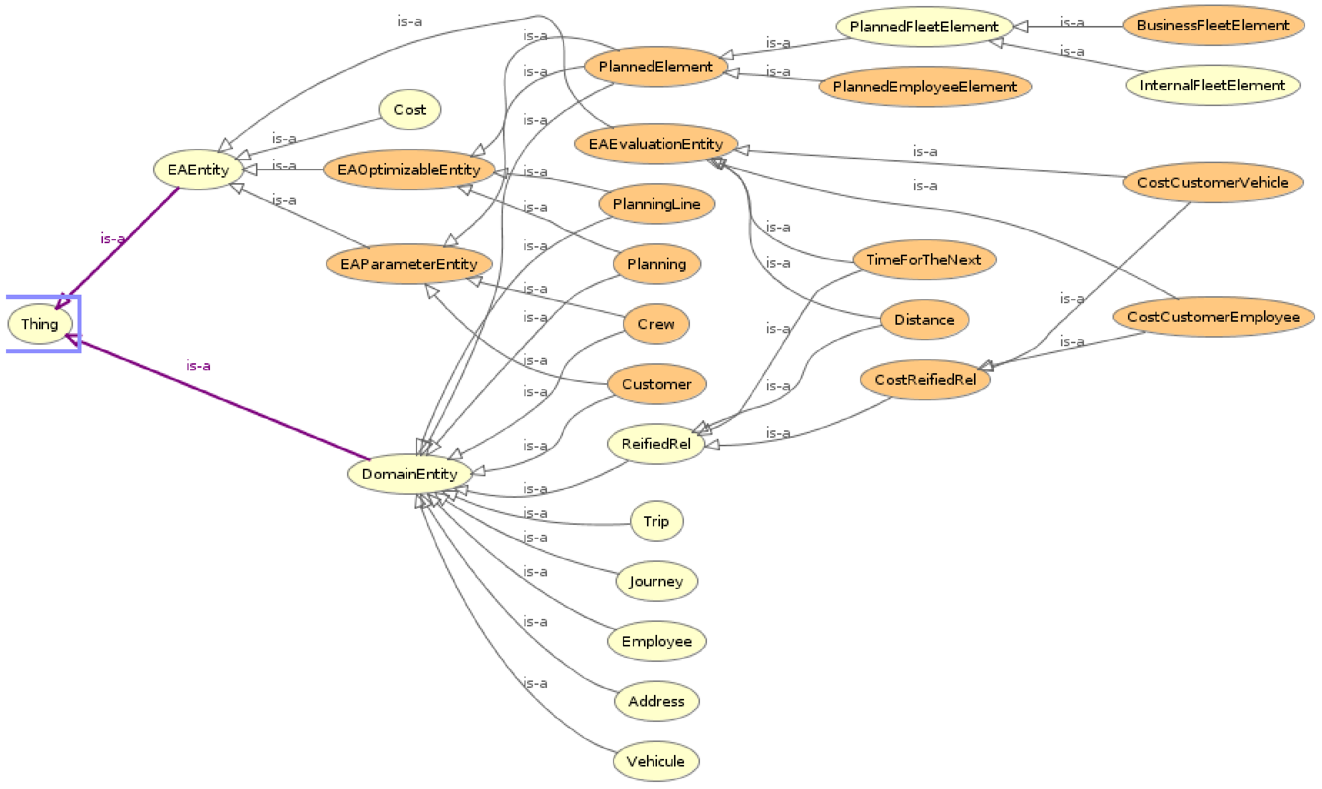


Figure 2. extract of the domain and EA ontology

(among others) to the property "address" of the "patient" entity. Unfortunately, not many formalisms allow the representation of relationships between different semantic levels (model and meta-model). In the ontology description language OWL 2 [15] for example, the possibility for a single object to be seen both as an entity and as an individual is reserved to the OWL-full expressivity, which is undecidable, and although theoretical studies have been conducted [16], few tools or reasoners, if any, allow to take into account this kind of structures.

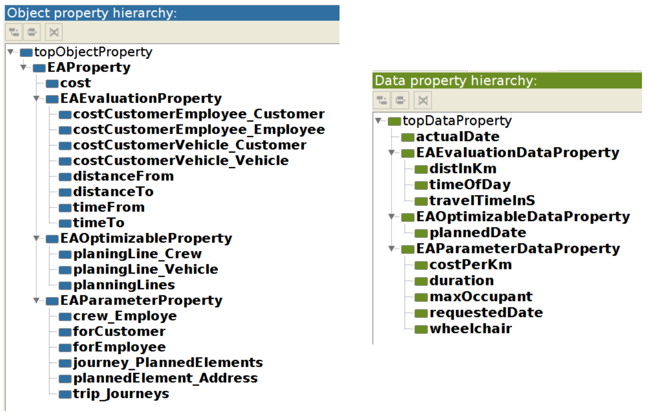
We therefore chose a simpler model for the current version of our methodology in OWL-DL:

- two generic entities, $DomainEntity$ and $EAEntity$ are defined. All the domain entities are defined as subclasses of $DomainEntity$
- we define a generic object property $EAProperty$ and three mutualy exclusive sub-Properties $EAEvaluationProperty$, $EAOptimizableProperty$, $EAParameterProperty$. All object properties of the domain ontology involved in the genetic algorithm must be subproperties of one of these three specific properties depending on whether they are involved for optimization, in the evaluation, or as a parameter.
- For the same purpose, but for atomic domain knowledge, we also define a generic data property $EADataProperty$ and three mutually exclusive sub-dataproperties. The top level EA object and data properties, and their specialization for the ambulance example can be seen in Figure 3.
- The entity $EAEvaluationEntity$ is defined as a subclass of $EAEntity$ having at least one evaluation property :

$EAEvaluationEntity$ ⊑ $EAEntity$

$EAEvaluationEntity$ ≡ ($EAEvaluationProperty$ some Thing) or ($EAEvaluationDataProperty$ some (boolean or dateTime or integer or real or ...))

$EAParameterEntity$ and $EAOptimizableEntity$ are defined similarly.



```
Object property hierarchy:
▼ topObjectProperty
  ▼ EAProperty
      cost
    ▼ EAEvaluationProperty
        costCustomerEmployee_Customer
        costCustomerEmployee_Employee
        costCustomerVehicle_Customer
        costCustomerVehicle_Vehicle
        distanceFrom
        distanceTo
        timeFrom
        timeTo
    ▼ EAOptimizableProperty
        planingLine_Crew
        planingLine_Vehicle
        planningLines
    ▼ EAParameterProperty
        crew_Employe
        forCustomer
        forEmployee
        journey_PlannedElements
        plannedElement_Address
        trip_Journeys
```

```
Data property hierarchy:
▼ topDataProperty
    actualDate
  ▼ EAEvaluationDataProperty
      distInKm
      timeOfDay
      travelTimeInS
  ▼ EAOptimizableDataProperty
      plannedDate
  ▼ EAParameterDataProperty
      costPerKm
      duration
      maxOccupant
      requestedDate
      wheelchair
```

Figure 3. Object and data properties

The use of domain entities in the evolutionary algorithm is shown at two levels of granularity in the EA ontology: properties are classified according to their use within the EA, and every entity involved in the realization of the EA will be automatically classified into one or more sub-classes of $EAEntity$ based on its properties. Thus, one can see in Figure 2 that $PlannedElement$ plays a role both as a parameter and as an entity to optimize.

IV. Multi-objective Optimization and Evaluation Functions

As stated in the previous sections, our problem is not a single-objective optimization problem, but a multi-objective one, because optimal decisions need to be made in the presence of trade-offs between two or more, eventually conflicting, objectives. Multi-objective approaches appear clearly as a possibility to solve our problem after a careful analysis of the constraints coming from the underlying data model. Moreover, beyond the classical advantages of these approaches, they are efficient in limiting a concentrated convergence of the solutions in a small subset of the Pareto front, which is very interesting for knowledge intensive evolutionary algorithms.

To be able to define a set of objectives, we first have to present a macro-objective and then break it down into a series of micro-objectives. The macro-objective is defined as: *Based on a set of required elements, vehicles, patients and employees, we have to generate a planning of itineraries that these vehicles will take so that the crews achieve the requested elements of the customers, minimizing the cost for the transport companies, maximizing service quality and observing all the restrictions that may appear in the context.*

Several micro-objectives have been deduced from this macro one, including:

1) Generate itineraries without delays or idle time between two different requests.
2) Generate itineraries that minimize the cost associated to the length of the trip.
3) Optimize the working time of the employees to avoid paying overtime or that they work less time that the legal number of hours per week.
4) Minimize the cost of the employees.
5) Minimize the cost of the vehicles.
6) Optimize the quality of service.
7) Balance the number of requests served by each vehicle.

These micro-objectives are, of course, in relation with the entities in the ontology in Figure 2.

For the two fitness functions detailed below, we will denote by V the set of vehicle, by PE the set of $PlannedElement$, and to each vehicle v_i, we associate the ordered list $P_i = [p_1^i, ..., p_{n_i}^i], p_j^i \in PE$ of $PlannedElement$ assigned to the vehicle.

1) Minimizing the cost associated with arrivals to a certain point in delay or in advance: Each $Journey$ between two $PlannedElements$ p_j and p_{j+1} is represented in a temporal line (Figure 4) with three values, $RequestedDate$ (RD), $Duration(D)$ and the estimated time to arrive to the next point (TFN). The duration estimates the time needed in a point to take care of the patient. This time depends on multiple factors, for example, the size of the wheelchair if the patient needs one or the fact that the patient is in a stretcher or not.

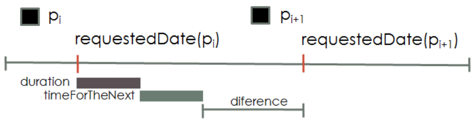


Figure 4. Temporal line between points p_j and p_{j+1}

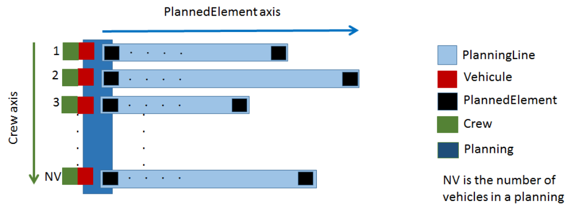


Figure 5. Structure of the chromosome

The TFN data are represented as a three-dimensional matrix (called $cubeTFN$), where each element represents the travel times between two addresses for a given (discrete) time in the day. Therefore, $cubeTFN_{ijt}$ is the time in seconds it takes to go from the point i to point j at time t.

The cost is represented through a piecewise function $f_0(x)$, where $x = RD_{j+1} - (RD_j + D_j + TFN_j)$ is the temporal difference shown in Figure 4. If this difference is negative (arrival in delay), the cost is quadratic; otherwise (arrival in advance), the cost is linear.

$$f_0(x) = \begin{cases} x^2 & , \; x < 0 \\ x & , \; x \geq 0 \end{cases}$$

Therefore, the objective can be formalized as:

$$min \sum_{v \in V} \sum_{p \in P_i} f_0(RD_{j+1} - (RD_j + D_j + cubeTFN(p_j, p_{j+1}, RD_{j+1})) \quad (1)$$

2) Minimizing the cost associated to the length of the trip: Ideally, the vehicles should attend points that are close to each other, in order to prevent the crews from driving long distances between two successive points in the itinerary. The cost to go from point p_j to point p_{j+1} is estimated from the distance matrix $cubeTFN$ and the associated cost per kilometre of each $Vehicle$ v_i. Therefore, if $x = (p_j, p_{j+1}, t)$ where t is the time of the day when the trip needs to be made, this objective can be formalized as:

$$min \sum_{v \in V} \sum_{p \in P_i} costKm\,(v_i, cubeTFN\,(p_j, p_{j+1}, RD_{j+1})) \quad (2)$$

V. Definition of the Chromosome and of the Evolutionary Operators

When the genetic algorithm finishes its execution, it returns a planning based on the ontology, to which the structure of the chromosome is adapted (Figure 5). For the set of all the vehicles, the chromosome associates then a crew and a list of points ($PlannedElements$) that the vehicle needs to attend (this list of $PlannedElements$ is the $PlannedLine$ associated to each vehicle). All list of $PlannedElements$ are sorted by

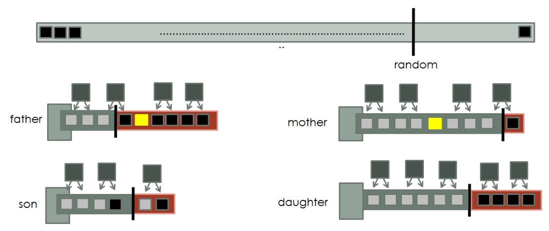


Figure 6. The $PlannedElementCrossover$ operator

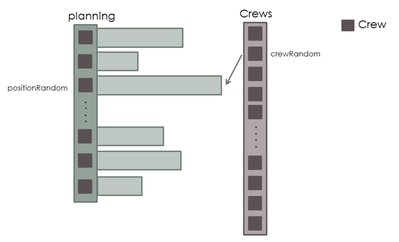


Figure 7. The $CrewMutation$ operator

ascending requested Date (object property $requestedDate(p)$ in model of Figure 3).

For the population to evolve, it is necessary to define a set of evolution operators. These should take into account all the aspects that are necessary for the planning to tend towards a possible final solution. There are two major axes that differ in the structure of a chromosome, the distribution of the $Crews$ and the $PlannedElements$ on the set of vehicles.

Making evolutionary changes in the two axes can affect all costs associated with the objectives, generating new populations with individuals of better quality. Several operators for the two axes have been defined, including crossover, mutation and swap. Only two of them are detailed below.

A. PlannedElementCrossover

As noted in Section III, $PlannedElements$ may be regrouped in a $Journey$. Each $PlannedElement$ belongs to at most one $Journey$. We denote by $inJ(p)$ the set containing p and all the $PlannedElements$ in the same $Journey$ as p.

In Figure 6, the big long top rectangle represents the list of all the $PlannedElements$ sorted by ascending time. To perform the crossover between two individuals, named mother and father in the figure, a $PlannedElement$ p_r is randomly chosen in this list. For each $PlanningLine$ i, we have the list P_i^f (resp. P_i^m) of $PlannedElements$ affected to vehicle i in the father (resp. mother) individual. The set P_i^s of $PlannedElement$ affected to vehicle i in the new son individual is defined by :

$$[l]P_i^s = \{p^f \in P_i^f; \exists p \in inJ(p^f)\ RD(p) <= RD(p_r)\} \bigcup \\ \{p^m \in P_i^m; \forall p \in inJ(p^m)\ RD(p) > RD(p_r)\}$$

Informally speaking, the son has the assignation of its father for early $PlannedElements$, and of its mother for later ones. We can also create a second derived individual (daughter in Figure 6) by reversing the role of the father and mother. The corresponding operations are schematically shown for one $PlanningLine$ at the bottom of Figure 6.

B. CrewMutation

This operator is implemented by taking a random $Crew$ of the list of all possible crews and make a simple swap between it and a randomly selected $Crew$ in the individual, only if the number of employees is the same in both crews (Figure 7).

VI. Evaluation

For a preliminary evaluation of the resulting algorithm, we consider the two evaluations functions described in Section IV. Evaluation is carried out under three different scenarios. In each scenario, we select an increasing number of $Vehicles$, which have to satisfy the constrains of an also increasing number of $Journeys$. The input data of the algorithm consists of a number of $Journeys$ randomly generated according previous information provided by a partner service medical company. As mentioned in Section III, each $Journey$ includes constrains regarding the time, the cost, the number of occupants, etc.

All the three scenarios are evaluated using a population of 1000 individuals during 400 generations. The probability of the genetic operators are 0.8 for the crossover and 0.1 for the mutation.

Figure 8 shows the best and average fitness values for the three different scenarios along the different generations. The results considering the cost of the delay (equation (1)) are shown on the left side of the figure, whereas results for the cost for the length of the trip (equation (2)) are shown on the right. Notice that fitness values are shown in the y-axis while the generation number is shown in the x-axis.

As can be observed, for the three scenarios the algorithm has been capable of minimizing the values of both objective functions along the evolution process.

In the case of the delay function on the two first scenarios, we can observe that the improvement tends to slow down considerably beyond the generation 100. A different situation is shown on the third scenario, where the algorithm requires more generations to find a good solution. Clearly, as a consequence of having a larger problem, more evolution time seems to be required.

When considering the cost of the trip, the best values curve shows a more variable behaviour. However, even with this non monotonic behaviour, an improvement along the evolution process is still observed. In the second and third scenario the difference between average and best fitness values is bigger than the first scenario. Therefore, it could be possible that more evolution time be required.

Finally, we can observe that in those scenarios with the largest number of journeys to satisfy, average and best fitness values are also considerable larger. In particular, in the case of the cost fitness function we can observe best values starts at 70 (K€) for the third scenario, whereas best values start at 45 (K€) and 17 (K€) for the second and first scenarios, respectively. A similar pattern is also observed in the delay fitness function.

VII. Conclusions and perspectives

This article has shown, through the resolution of a real problem, how knowledge engineering techniques can be used to guide the definition of EA for problems involving a large amount of structured data.

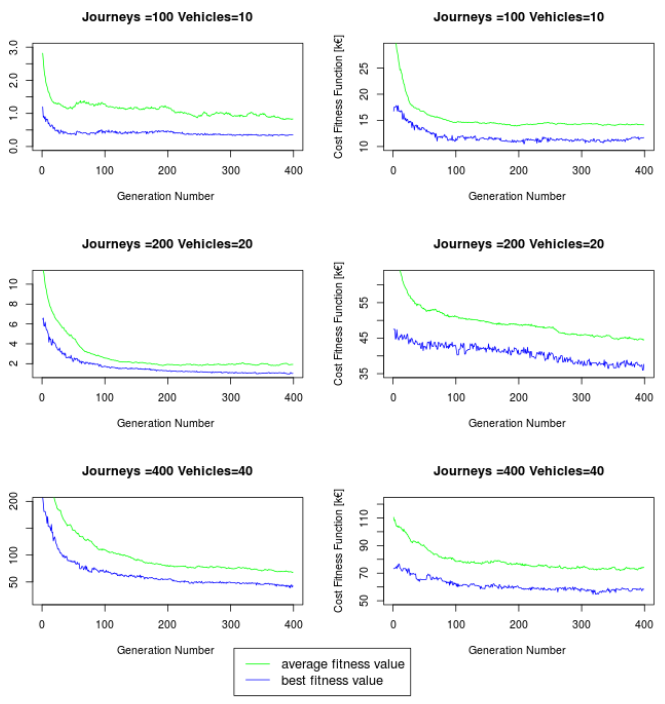


Figure 8. Fitness function values for different problem sizes

The first prototype, using the EASEA platform, has been evaluated. The idea behind this evaluation is just to demonstrate the feasibility of the methodology. Therefore, we have not fine tuned the EA parameters in order to get a solution suitable for the real-life scenario. Notice that such fine tuning will be carried out on a parallel version of the EA, which is currently being developed.

We also intend to use machine learning techniques in order to increase the efficiency of the EA. In fact, most of the $PlannedElements$ are recurring in the same geographical area; the development of a module taking profit of the past experience is being undertaken to attempt guiding the population of the EA to more accurate and close-to-reality itineraries.

As evoked in Figure 4, each $PlannedElement$ has an internal duration associated with it. Whether the point is a collection point or a deposit point, this internal duration depends specifically on the conditions in which the next ride needs to be made. This internal duration represents the amount of time taken for the patient to get in or out of the vehicle and depends on certain specificities of the patient, such as the need of a wheeling chair (that can have different sizes), or of a stretcher or of crutches, oxygen mask or perfusion. And, of course, this duration depends also on the age of the patient and on his general health state.

For the moment, some tests with linear regressions using WEKA [17] have been made, yielding encouraging results; although some occasional missing values in the input data (due mainly to oversights of the staff in the call centre that receives the requests) induces the need of using other techniques [18].

References

[1] B. Golden, R. Raghavan, and E. Wasil, Eds., The Vehicle Routing Problem : Latest Advances and New Challenges, ser. Operations Research/Computer Science Interfaces Series,. Springer, 2008, vol. 43.

[2] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed. Pearson Education, 2009.

[3] A. Garcia-Najera and J. A. Bullinaria, “An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows,” Computers & Operations Research, vol. 38, no. 1, 2011, pp. 287 – 300.

[4] A. S. Tasan and M. Gen, “A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries,” Computers & Industrial Engineering, vol. 62, no. 3, 2012, pp. 755 – 761.

[5] J. Berger and M. Barkaoui, “A parallel hybrid genetic algorithm for the vrp with time windows,” Computers & Operations Research, vol. 31, no. 12, 2004, pp. 2037 – 2053.

[6] O. Bräysy and M. Gendreau, “Vehicle routing problem with time windows, part ii: Metaheuristics,” Transportation Science, vol. 39, no. 1, Feb. 2005, pp. 119–139.

[7] N. Carrasquero and J. A. Moreno, “A new genetic operator for the travelling salesman problem,” in Proceedings of the 6th Ibero-American Conference on AI: Progress in Artificial Intelligence, ser. IBERAMIA ’98. London, UK, UK: Springer-Verlag, 1998, pp. 315–325.

[8] J. H. Holland, Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, 1975.

[9] C. Z. Janikow, “A knowledge-intensive genetic algorithm for supervised learning,” Machine Learning, vol. 13, no. 2-3, 1993, pp. 189–228.

[10] G. Pankratz, “A grouping genetic algorithm for the pickup and delivery problem with time windows,” OR Spectrum, vol. 27, no. 1, Jan 2005, pp. 21–41.

[11] R. Giraldez, J. Aguilar-Ruiz, and J. Riquelme, “Knowledge-based fast evaluation for evolutionary learning,” Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 35, no. 2, May 2005, pp. 254–261.

[12] P. Collet, E. Lutton, M. Schoenauer, and J. Louchet, “Take it EASEA,” in Parallel Problem Solving from Nature PPSN VI, ser. Lecture Notes in Computer Science, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.-P. Schwefel, Eds. Springer Berlin Heidelberg, 2000, vol. 1917, pp. 891–901.

[13] O. Maitre, F. Kruger, S. Querry, N. Lachiche, and P. Collet, “EASEA: specification and execution of evolutionary algorithms on GPGPU,” Soft Computing, vol. 16, no. 2, 2012, pp. 261–279.

[14] S. Staab and R. Studer, Eds., Handbook on Ontologies, ser. International Handbooks on Information Systems. Springer, 2004.

[15] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, “OWL 2: The next step for OWL,” Web Semantics: Science, Services and Agents on the World Wide Web, vol. 6, no. 4, 2008, pp. 309 – 322.

[16] B. Motik, “On the properties of metamodeling in owl,” J. Log. Comput., vol. 17, no. 4, 2007, pp. 617–637.

[17] I. H. Witten, E. Frankc, and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques. 3rd Editio. Morgan Kaufmann Series in Data Management Systems, 2011.

[18] M. Saar-Tsechansky and F. Provost, “Handling missing values when applying classification models,” Journal of Machine Learning Research, vol. 8, 2007, pp. 1217–1250.