# An Economic Approach for Generation of Train Driving Plans using Continuous Case-based Planning

André P. Borges[1], Osmar B. Dordal[1], Richardson Ribeiro[2], Bráulio C. Ávila[1] and Edson E. Scalabrin[1]

[1]*Programa de Pós-Graduação em Informática, Pontifícia Universidade Católica do Paraná, Rua Imaculada Conceição 1155, Curitiba, Paraná, Brazil*

[2]*Department of Informatics (DAINF), Federal University of Technology-Parana, Via do Conhecimento, Km 1, Pato Branco, Paraná, Brazil*

Keywords: Case-based Planning, Driving Plans.

Abstract: We present an approach for reusing and sharing train driving plans *P* using continuous (or without human intervention) Case-Based Planning (CBP). *P* is formed by a set of actions, which when applied, can move a train in a stretch of railroad. This is a complex task due to the variations in the (i) composition of the train, (ii) environmental conditions, and (iii) stretches travelled. To overcome these difficulties we provide to the driver a support system to help the driver in this complex task. CBP was chosen because it allows directly reuse the human drivers experience as well as from other sources. The main steps of the CBP are distributed among specialized agents with different roles: *Planner* and *Executor*. Our approach was evaluated by different metrics: (i) accuracy of the case recovery task, (ii) efficiency of task adaptation and application of such cases in realistic scenarios and (iii) fuel consumption. We show that the inclusion of new experiences reduces the efforts of both the *Planner* and the *Executor*, reduces significantly the fuel consumption and allow the reuse of the obtained experiences in similar scenarios with low effort.

## 1 INTRODUCTION

For years, different sectors of the economy have been tested in relation to their innovation capabilities and competitiveness, which can be summarized by the expression doing what we already do well, better. These capabilities, in general, aim to create new cash flows for a company. Thus, the use of information technology is imperative for establishing a different and sophisticated way to reduce costs without compromising quality, and still consider the scarcity of resources. In this scenario, the railroad, to be competitive, must minimize transportation costs and capitalize every available resource, such as railroad cars and locomotives.

Although the railroad is one of the most feasible modes for freight transportation, there is still latitude for cost reduction. The use of any technological resource that can reduce expenses, for example, fuel consumption, can represent significant cost reduction in one year of operation. For example, the United States of America railroads consumed 3.600 million of gallons of fuel in 2012, approximated 145.7 thousand gallons per locomotive, which represent a cost 14.285 mi of dollars (of the Assistant Secretary for Research and of Transportation (US DOT), 2014).

The establishment of general train driving policies that derive from important financial returns is difficult because of: (i) the need (or existence) of specialized training for drivers; (ii) variations in train formation (e.g., number of locomotives, railroad cars); and (iii) influence of driving conditions (i.e., climate, constraints). Moreover, the train driver must possess significant knowledge regarding rules and regulations (e.g., driver cab controls, signaling systems, and track safety), traction knowledge (e.g., engine layout and safety systems) and route knowledge, in addition to several hours of practical driver skills. This set of requirements is necessary in order to achieve a feasible safe, fast and cost-effective driving of several trains.

In this context, an approach for generating centralized driving plans that drive with static rules has a small possibility of producing significant results given variations in the use conditions of the track and diversification of the experiences gained. The approach proposed here includes the use of a distributed architecture to increase the availability of resources such as experiences and comprehensiveness of plans be-


P. Borges A., B. Dordal O., Ribeiro R., C. Ávila B. and E. Scalabrin E..
An Economic Approach for Generation of Train Driving Plans using Continuous Case-based Planning.
DOI: 10.5220/0005348504400451
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 440-451
ISBN: 978-989-758-096-3

fore and during execution of such plans. Furthermore, the application of case-based learning allows those involved to become increasingly specialized and autonomous. To achieve autonomy and specialization, the existence of some agents specialized in well-defined tasks is assumed: *Memory* (maintaining a case base), *Planner* (preparing feasible driving plans), and *Executor* (executing and adjusting plans).

The Planner and Executor agents adapt and execute plans, respectively, against different driving conditions and strategies. For example, a given strategy can target specific goals, such as energy efficiency or driving speed. The construction base of each strategy can be purely mathematical, algorithmic, or rule-based (e.g., ⟨IF-THEN⟩) (Sato et al., 2012). These techniques offer few possibilities for adaptation without the intervention of an expert in this field. Such difficulty represents an important limitation when there is diversification of the profiles of the railroads and trains involved.

In this paper, the main motivation is to improve the performance of Executor agents when facing new situations, by exchanging experiences between agents, reducing the necessary efforts to apply a driving plan. The exchange of experiences occurs when the executed driving plans by human drivers, or another sources, is incorporated to the knowledge base of Memory agent and is used by Planner agent to elaborate new driving plans.

The driving plans are elaborated using the Case-Based Planning approach without human interaction. CBP is based on canonical case-based reasoning (Aamodt and Plaza, 1994)(Spalzzi, 2001). This approach is divided in four main steps: *recover* the past case solutions that are similar to the new problem, *reuse* and adapt (if necessary) the most similar case, *revise* the proposed solution to guarantee its applicability and *retain* the new solution for future use. At first step, an Euclidian Distance is used by the Memory agent to calculate the distance between the new problem and the stored cases. A set of most similar cases will be adapted using the Genetic Algorithm, which tries to optimize the new solution. The best adapted solution is revised according to domain specific knowledge, to guarantee the safety during the journey, avoiding situations that damage the train or the railroad, like slipping, lack of moving force or high speed. At end, the executed driving plans are stored by Memory agent when it returns to the station.

The approach was evaluated by the accuracy of the case recovery task, and the efficiency of task adaptation and application of such cases in several scenarios. The applicability of the proposed solution is evaluated in terms of fuel consumption, comparing our best scenario against the fuel consumptions obtained for other approaches with the same configurations of train and railroad. We show that the inclusion of new experiences reduced the efforts of both the Planner and the Executor and reduces significantly the fuel consumption. In addition, the CBR approach allowed the reuse, with low effort, of the obtained experiences in similar scenarios.

The next section introduces some related works. Section 3 presents the developed system. Then, Section 4 shows some of the experiment results. Finally, the last section presents our conclusions.

## 2 RELATED PAPERS

Many researches have applied Case-based reasoning within various problem-solving domains. For example, like a recommender mechanism (Wang and Yang, 2012) or in autonomic systems to minimize human intervention and to enable a seamless self-adaptive behavior in the software systems. (Khan et al., 2011).

Case-based reasoning has been used successfully in collaborative systems whose missions were to assist people in various tasks, such as selecting the appropriate behavior for an unattended vehicle ride (Bajo et al., 2007) (Vacek et al., 2007) , optimizing industrial processes (Navarro et al., 2012), and establishing rules used to assist experts in detecting environmental changes (Mota et al., 2008). In these systems, the goal was to share plans as a way of enriching experiences by generating shared and interactive plans and execution. In our application, the collaborative driving approach directly considered the experiences of human drivers (adjusting them if necessary), as well as the experiences of automatic driving systems. Also, our main difference lies in the independent execution of the plan, i.e., during its execution, there is no interaction with the Planner (which generated the plan) located in the processing station. The Executor is embedded in the onboard computer of the main locomotive, without a dedicated communication channel with the agents of its origin station. There is no transmission of driving plans because of environmental restrictions, such as the presence of tunnels, distance of transmission towers, and increase in the cost of the operation.

In railroad transportation, several works were developed over the years in order to optimize driving and use of the rail network. In (Gu et al., 2012) the focus was to determine the speeds practiced during the trip, using non-linear programming to avoid abrupt actions from the driver agent. A similar focus

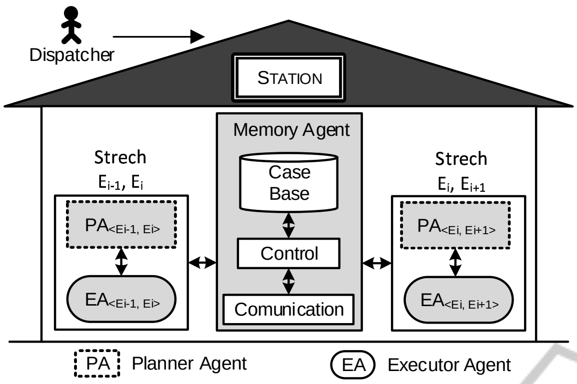


Figure 1: ITDS: Representation of a station, with a Memory agent, two Planner agents, and two Executor agents.

was given by (Hengyu and Hongze, 2012), but with the use of fuzzy neural networks to control actions. In (Fang et al., 2013) the authors present several approaches developed for re-scheduling problem in rails networks. In (D. and Houpt, 2011) the authors present a train control-system to optimize trips that requires the continuously profile update by GPS (Global Positioning System) to control the locomotives, however, due the presence of tunnels the GPS signal may fail and become unavailable. Finally, in (Borges et al., 2012) a single agent, based on CBR, was created to prepare train-driving plans based on the recovery of a single action at a time. The internal organization of the case base was proposed in the composition of the cases and specializations, which required time to be organized, and resulted in lower percentages of success than those presented here. Furthermore, the planning of a single action at a time made the approach computationally costly.

# 3 ARCHITECTURE

The developed system, referred to as the Intelligent Train Drive System (ITDS) utilizes the following resources: rail network, freight train, and software agent (Wooldridge and Jennings, 1995). The rail network is viewed as a graph where the vertices represent physical or logical stations, and the edges have information from the track (i.e., profile). Each station hosts a Memory agent, a Planner, and one or more Executors for each stretch $S_i$. This configuration is illustrated in Figure 1, where the triple $\langle E_{i-1}, E_i, E_{i+1} \rangle$ defines the ends of two stretches $S_1 = \langle E_{i-1}, E_i \rangle$ and $S_2 = \langle E_i, E_{i+1} \rangle$.

The interaction between the different agents is well defined in time. Planner PA1 receives, from an external system, a demand $d$ to drive the train on a stretch $S_i$. PA1 then segments plan $P$ of stretch $S_i$ into $n$ parts $p_1, \ldots, p_n$, according to the vertical profile of the stretch. The cycle starts. For each $p_k$, PA1 forwards to Memory agent MA1 a request for plans applicable in $p_k$. MA1 collaboratively returns to PA1 a set of candidate plans $SP = sp_1, \ldots, sp_m$. PA1 reduces this set SP to a single feasible plan $sp_k$, adapting it to the current situation, and includes $sp_k$ in plan $P$. Then, it selects $p_{k+1}$ and repeats this cycle $n$ times. At the end, $P$ is forwarded to an EA1, which embarks on the train and applies it.

During the application of $P$, there is no communication between the agents, a differential aspect when compared to other approaches (D. and Houpt, 2011)(Gu et al., 2012)(Hengyu and Hongze, 2012). However, because the conditions for plan execution may change, it may be necessary to partially redo parts of plan $P$. The capacity of the Executor to redo parts of plan $P$ ensures feasible and safe driving without the need to receive a new plan from PA1. This is an important aspect for making the use of communication channels less relevant. Existing train-station communication is only used for monitoring and controlling the position of each train. This helps to reduce the costs of freight transport and coupling across system software modules.

The algorithm 1 summarizes the ITDS flow. The input of the algorithm is a base-case $B$ containing past experiences of human drivers or another fonts. While a demand is not emitted by a Dispatcher, the system remain waiting for a demand (lines 1-3). When the order is received, the Planner elaborates a plan $P$ to attend the $d$ using the previous experiences stored in $B$ (line 4). Next, the Executor agent will execute $P$ (line 5). After the execution of $P$, the executed plan $P'$ is incorporated in the case-base of the Memory Agent (line 6).

**Algorithm 1:** Basic ITDS flow.

**Require:** a case-base $B$
1: **while** $d = null$ **do**
2: $\quad d \leftarrow listenDispatcher()$
3: **end while**
4: $P \leftarrow planner(B, d)$
5: $P' \leftarrow execute(P)$
6: $B \leftarrow store(B, P')$

The agents responsible for execute the tasks summarized will be described in the next section.

## 3.1 Agents

The role of the Dispatcher is to globally manage the times and orders for the movement of trains in a rail network, rationally attempting to occupy the spaces and existing resources (Company, 1979). The Dispatcher decides whether to stipulate the maximum allowable speed at certain points of the track, and to restrict or allow the passage of the train in a sector (Company, 1979). This information is sent to the Planner along with the dispatch order, trigger for planning.

The Planner generates a driving plan $P$ to meet the demand of the Dispatcher, which is to move train $T$ from end $E_i$ to $E_{i+1}$. Each action of $P$ can take one of the following behaviors: accelerate, maintain, or reduce the speed of train T. Each behaviour is adjusted according to a specified power. Accelerate and reduce correspond, respectively, to increase and decrease acceleration points. In a locomotive, each acceleration point, typically from 1 to 8, generates a positive power capable of moving the train. Speed reduction can be accomplished by generating a power less than the sum of the resistances or by applying the brakes. Braking involves applying pressure in the brake pipe, measured in *psi* (pressure in pounds per square inch) and is represented here by acceleration point -1. To move the train, an appropriate acceleration point for a given position of the railroad stretch to be covered must be planned in order to avoid sliding and overcome the sum of resistances (Loumiet et al., 2005).

To ensure train movement in all points of the track, it is also necessary to know the minimum required power. This power is calculated by the Dispatcher and reported to the Planner in the dispatch order. The calculation considers the minimum tractive force of all the locomotives intended to move a train against the high resistance of the stretch to be covered. Moreover, the resulting set of actions should meet the objectives, which are in opposition in the first two at the base, of performing a quick trip, reducing fuel consumption, and complying with the security restrictions. Thus, one should plan speeds near the cruise speed, for example, 5 km/h below the maximum speed. Within this criterion, actions should reduce fuel consumption and travel time. The resulting set is the driving plan $P$ for train $T$ in a stretch $S$.

The role of the Executor is to apply plan $P$, received at end $E_i$, performing the following basic tasks: testing the applicability of ak based on the current train conditions, adjusting the parameters of $a_k$ (if necessary), and applying $a_k$. Until the complete execution of $P$, $P$ may undergo several $\Delta$ adjustments. For example, in the case of non-applicability of an action $a_k$, the Executor may adjust ak based on driving skills. Adverse conditions may represent, for example, a climate change (changes the friction coefficient), changes in the maximum allowable speed, and others. Such conditions perceived by various sensors

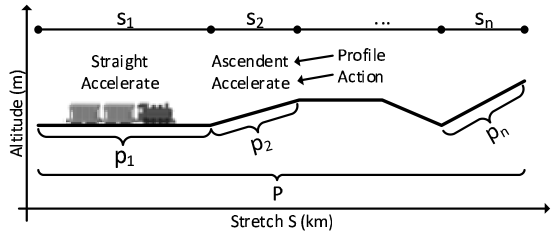


Figure 2: Example of segmentation of a stretch S into parts $s_1,\ldots,s_n$ according to the vertical profile and a set of cases $p_1, p_2,\ldots, p_n$, which when ordered form a plan $P$ for the movement of trains.

are read in predetermined time intervals.

The Memory agent has two basic functions: (i) providing the Planner with a set of plans applicable to each part of a given stretch and (ii) maintaining a base of plans in a dynamic memory structure (Schank, 1983). Each Memory, located in a station, maintains only the plans applied in the stretches from ends $E_i$ to $E_{i+1}$ and from $E_i$ to $E_{i-1}$. Maintaining only the plans of the stretches connected to the station allows specialization in these stretches. Each plan executed $P+\Delta$ is returned to the Memory of the origin end point to be integrated into the local base of the station plans. Such structure allows the inclusion of a new plan that can activate simple internal processes of plan reclassification and/or more sophisticated optimization of data structures and indexing of plan contents (Schank, 1983).

## 3.2 The Plan Elaboration

The elaboration of a plan requires: information from the train (i.e., position, number of locomotives and railroad cars, and weight) and from the track stretch (i.e., maximum speeds, friction coefficients, and vertical profile). The vertical profile is critical for defining the relevant portions of a stretch $S = s_1,\ldots,s_n$ (see Figure 2).

For each part $s_i$, a plan is prepared that corresponds to a new problem to be solved. Each part must become specialized. Figure 2 illustrates plan $P$ with $p_1,\ldots,p_n$. Each $p_i$ describes a set of actions with more predictable behaviours (maintain, accelerate, and reduce) for a vertical profile (rising, falling, and plain/plateau).

The behaviours that must be undertaken for proper driving are shown in Figure 3. Such behaviours will vary depending on current speed $s$, initial speed $s_i$, cruise speed $s_c$, and maximum speed $s_s$ (Pinto et al., 1985). Proper driving is obtained when the current speed is greater than the initial speed and less than the maximum speed, remaining close to the cruis-

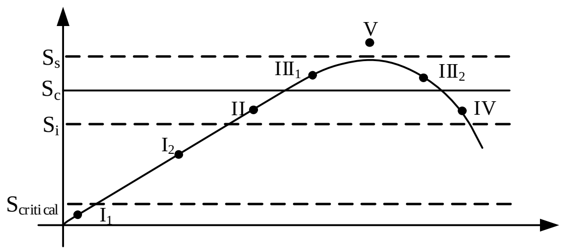


Figure 3: Possible states of train driving (Pinto et al., 1985).

c1 = <EAC, Accelerate>, <KM, 330.000>, <NL, 3>, <NW, 58>, <IS, 30>, <FS, 32>, <MS, 40>, <%R, 0>, <CL, 60>, <J, <0,5><5,3><25,4><35,6><50,8>>

Figure 4: Example of a retrieved case.

ing speed. States indicate actions that must be performed to achieve proper driving. State $I$ suggests ACCELERATING the train because the current speed is less than the initial speed. States $II$ and $III_1$ suggest ACCELERATE or MAINTAIN speed. On the other hand, state $III_2$ must REDUCE the speed because of the proximity to the current speed with the maximum speed. In state $IV$, it must ACCELERATE to approximate the current speed to the speed regimen. The BRAKING action is recommended in state $V$ because the train can exceed the maximum allowable speed.

Pragmatically, case $C$ is a representation of a real-world object or episode in a particular representation scheme (Kolodner, 1993). A case represents a finite set of $n$ attribute/value pairs that is a snapshot of the situation executed by a train conductor during one trip.

Figure 4 shows an example of a case recovered in the form of pairs $\langle attribute, value \rangle$ formed by the following attributes: action performed (EAC), initial kilometre (KM), number of locomotives (NL), number of railroad cars (NW), initial speed (IS) in km/h, final speed (FS) in km/h, maximum speed (MS) in km/h, ramp percentage (%R), and total displacement (CL) in meters. Although such attributes are presented as an illustration, they represent the dominant features in the movement of a train.

A case also has a solution formed by a set

$$J = \langle m_1, AP_1 \rangle, \langle m_2, AP_2 \rangle, , \langle m_j, AP_j \rangle$$

of ordered pairs. Each pair contains an acceleration point (AP) and an application position (M) defined in meters. Figure 5 shows an instance of $P$, and an application of each $AP$, from left to right, must move from 0 km to 60 km. The determination of each element of $P$ should allow the Executor to move the train safely and efficiently.

It is expected that, as the Memory agent has in its case base, a number of cases applied in stretches leav-

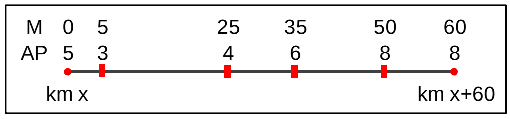


Figure 5: Example of the solution of an applicable case in kilometre (km) $x$ to the maximum kilometre $x+60$.

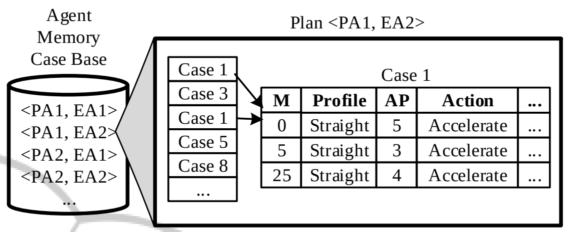


Figure 6: Representation of the local Memory (left) case base and example of part of a stored plan (right).

ing the station where it resides, there are greater possibilities of recovering an applicable case. If the case is not fully applicable, it is believed that the adaptations are in reduced percentages. Figure 6 shows the cases organized according to the actions taken (*AP* and *Action*) and the constituents of the track profile. This organization allows recovery of cases more similar to the stretch of the track in question and the action to be applied.

## 3.3 Inter and Intra Agents Data Flow

Figure 7 shows the basic flow of inter and intra agent collaboration. In this figure, the Dispatcher has been omitted. Thus, the flow is started by the Planner, whose first activity is to receive the dispatch order generated by the Dispatcher and to generate a new problem. As a result, two collaboration cycles are established: the first interactively involves the Planner agent and the Memory agent, and the second sequentially involves the Planner, Executor, and Memory. Henceforth, PA1 is used to designate a Planner agent, EA1 an Executor agent, and MA1 a Memory agent.

### 3.3.1 Planning Flow - Planner and Memory Interaction

Planning starts when PA1 receives a dispatch order from the Dispatcher.

(inform :sender Dispatcher :receiver PA1
:content (locomotives=3, wagons=58, [...]))

Then, the perception of the environment is performed, resulting in a new case $c1$ to traverse stretch

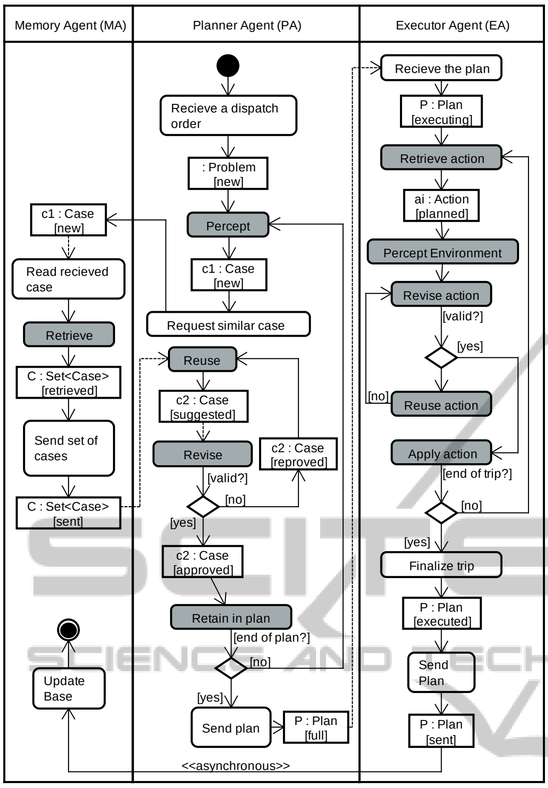


Figure 7: Basic ITDS flow.

$S_i$. The instantiation of $c1$ is managed by a dedicated module that performs two basic functions: (i) read sensors for speed, position, traction, and others, of the train; (ii) calculate, in this case, the values for resistance, effective tractive force of the pulling locomotives, acceleration force, adherent tractive force, etc. The data set (perceived and derivatives) defines the values used to move the train (Loumiet et al., 2005).

PA1 passes on the new case instantiated by means of a request to MA1.

(request :sender PA1 :receiver MA1 :content
(⟨attributes of the problem⟩))

Internally, the message is received and managed by the Memory. The recovery of similar cases is done using the Euclidian distance. The advantage of this division of labor is to allow PA1 to perform another task during the recovery phase, for example, to process another dispatch order.The recovered cases are then sent to PA1.

(inform :sender MA1 :receiver PA1 :content
(⟨set of similar cases⟩))

The most similar case $c_i$ is selected and adapted according to the current perception (if necessary) by the Reuse activity. This task involves replacing the values of the $J$ pairs (e.g., Figure 5), which are, respectively, ⟨application position, acceleration point⟩. The adaptation step of the Planner used a genetic algorithm (Baeck et al., 2000) (Mitra and Basak, 2005). In a genetic algorithm the case base form the initial population of genotypes. Firstly, the algorithm retrieves partial matching cases from case base with specified design requirements. In this paper, each individual is composed by the recovered case solution $J$ and the initial population consist of the 50 most similar cases recovered from the Memory.

Next the retrieved cases are mapped into a genotype representation, so, the solution $J$ of the each recovered case is mapped into a genotype of an individual, using integer numbers. For example, the individual of Figure 5 is mapped as

Individual Number: i0|
Evaluated: T
Fitness: f1138065562|427.0047|
i0|i5|i5|i3|i25|i4|i35|i6|i50|i8|i60|i8

to obey the standard used by the ECJ framework.

Later crossover and mutation operators are applied. The mutation and crossover rates are 50% for both. We used one-point crossover since this technique have been good results when compared with others (Magalhães Mendes, 2013). Finally, newly generated genotypes are mapped into corresponding phenotypes/cases by inferring values for the attributes and adding the context of the new design.

Applying the genetic algorithm to case adaptation also requires the identification of a fitness function. In this paper, two of the main objectives are to reduce the fuel consumption and optimizing the speed practised by Executor agents. So, the fitness function $g(x)$, illustrated in Equation 1, was defined to minimize such objectives, although they are opposites. These attributes have a great impact on the generation of individuals and will determine their utility in the population. Each attribute has a factor which represents his weight in the problem: fuel consumption ($flgtt$) and speed ($fv$), where $flgtt = 0.02$ e $fv = 0.01$. The values of fuel consumption ($lgtt$) and speed ($v$) are obtained by simulating the application of the individual in the problem. During the simulation, the following values are calculated for each action: consumption, travel time, resistance, force to drive the train, and others. Therefore, the fitness function considers calculations that result in the forces required to move the train considered (Borges et al., 2012).

$$Min.g(x) = \sum_{j=0}^{J}((flgtt \times lgtt_j) + (fv \times v_j)) \quad (1)$$

The values of $flgtt$ and $fv$ were calculated by a genetic algorithm made specific for it, where the individuals are composed by pairs of $< flgtt, fv >$. Each parameter can vary according the interval $[0.01; 1.00]$, totalling 100 possible values for each factor, being this the value of each population generated by the algorithm. For each individual a journey was simulated, all with the same train and railroad configurations. The fuel consumption obtained in each journey was used as fitness function value. The crossover and mutation techniques used the same configuration developed for this paper. The stopping criteria was set to 100 generations, because there is a 10000 possible factor combinations.

Returning to the adaptation step of this paper, the crossover and mutation operations creates individuals that consists of solutions for the actual problem, called proposed solutions. Each solution generated by the genetic algorithm $J$ is composed by acceleration points and places of its applications, according to Figure 5. Each proposed solution $J$ is simulated, applying the acceleration points at the positions specified in $J$. The resulting fuel consumption corresponds to the fitness of the individual. If the proposed solution is not applicable, because it results in slipping or lack of movement force, the individual is penalized with a high value for the fitness and this individual is discarded. The stopping criterion of the strategy was the number of generations, equal to 10. At end, the individual with minor value of fitness will be chosen for the adapted case. The fitness function maximizes some value by default, so, to minimize the fuel consumption we convert the fitness function to $1/g(x)$.

The adapted case $c'i$ (corresponds to case $c2$ in Figure 8) is feasible if and only if all parts of $J$ meet the following situations: (i) does not result in sliding, (ii) have sufficient force to move the train, and (iii) if the acceleration point can be reduced and continue to move the train, which indicates unnecessary fuel consumption of the action. If the case is not approved, the values of $J$ are again submitted to the Reuse step, and this is repeated until a valid solution is found and the case is approved. A valid solution must comply with the criteria (i) and (ii) described in the previous paragraph, but not necessarily with criterion (iii), added only to reduce fuel consumption.

After adding the valid adapted case in $P$, PA1 verifies whether the application of the plan results in the arrival at the destination. If not, the planning cycle is repeated for the next part of stretch $s_i$. If so, $P$ is sent to EA1.

c2 = <EAC, Maintain>, <KM, 335.000>, <NL, 3>, <NW, 58>, <IS, 20>, <FS, 20>, <MS, 40>, <%R, 0>, <CL, 520>, <J, <0,8><5,3><25,3><35,7><50,8>>

Figure 8: Adapted Case.

(execute :sender PA1 :receiver EA1 :content (the plan $P$))

EA1 is embedded in the onboard computer of the main locomotive and begins to command it. If later, another train passes through the same station, another Executor EA2 is created with a specific plan for the new train.

#### 3.3.2 Flow of Execution of the Plan - Executor

The plan execution cycle is similar to the planning cycle. Each Executor starts its activities upon receiving the trip plan $P$. In possession of $P$, EA1 starts the trip. During the trip, EA1 Recovers an action $a_k$ of $P$ and evaluates its preconditions. For such, it perceives the environment through data read from the onboard computer of the lead locomotive (e.g., maximum speed, friction coefficient, and others). If no precondition is violated at the time of reading, the $a_k$ action is applied, resulting in new information (i.e., speed, position, and others), which becomes the precondition for the next action $a_{k+1}$. On the other hand, if one or more preconditions are invalid, for example, because of some unforeseen event (e.g., rain, fog, or changing the speed limit), EA1 corrects the action of the plan based on its knowledge of driving (Reuse). Such adjustment is made by evaluating the reason for the failure, which can be: sliding, lack of force to move, or stopping unexpectedly. It is expected for each change (when required) to cause as insignificant an impact as possible, i.e., the shortest distance between the planned and the applied acceleration point, thus resulting in a reduced state space search.

Once the task of EA1 driving the train from end $E_i$ to end $E_{i+1}$ is complete, the Finalize trip activity is executed. In it, plan $P$ and its $\Delta$ adjustments are processed, resulting in a plan $P$. $P$ is returned to the origin station $E_i$ by an Executor whose destination is $E_i$.

(inform :sender EA1 : receiver MA1 :content(an executed plan $P$ + $\Delta$

The transport of the modified plan to the origin by means of another train is assumed because, during the trip, no communication channel is used to transmit the plans between the train and the station. If the stretch has only one direction, the plan is stored at the station and forwarded to the next station that has a connection with the plan origin station. Communication between train and origin station is limited to sending information related to the conditions of the railroad and the position of the train because of the cost of data transmission and limitations of the means of communication, most of the time, against geographical conditions.

#### 3.3.3 Update Case Base - Memory

Memory agent MA1, upon receiving plan $P$, does the following: for each profile $p_i$ of $P$, identify the action taken (accelerate, maintain, or reduce) based on the speed variation practiced. Each separate and identified part of the plan is compared with other existing cases in the base. If it is already present, its reputation receives a backup. Otherwise, a new event is inserted. The reputation is a piece of information that allows management of the case base (if necessary, thus reducing the number of cases).

## 4 EXPERIMENTS

The approach was evaluated by different metrics: (i) fuel consumption ($LGTT$), (ii) accuracy of the case recovery task, and (iii) efficiency of task adaptation and application of such cases in synthetic scenarios which were created inspired in real-world scenarions. The last two indicate the efficiency of the collaboration of the Memory and Planner agents.

The experiments were conducted in a simulated driving environment, and field equations (Loumiet et al., 2005) were implemented in Java (Luke et al., 2014) and validated using several experiments. The profiles of the railroads, trains, and the initial case base are derived from real situations.

Table 1 defines the configurations of the trains used in the experiments. To complete the experiments, two different stretches of real railroads were included: $S_1$ and $S_2$, both with the same length (approximately 64 km), but with different profilesvertical and horizontaland maximum speed restrictions.

To evaluate the learning curve of each agent and the performance of the collaboration in terms of sharing and reusing plans, four scenarios were defined (see Table 2).

The initial case base of the Memory agent, in all tested scenarios, contains actual trip plans and trips executed in simulators (Sato et al., 2012).

Table 1: Train configuration used in the experiments.

| Train | Locomotives | Railway cars | Weight (tons) |
|---|---|---|---|
| 1 | 3 | 58 | 6278 |
| 2 | 4 | 100 | 6342 |
| 3 | 4 | 58 | 6541 |
| 4 | 2 | 31 | 3426 |
| 5 | 3 | 47 | 5199 |
| 6 | 2 | 31 | 3441 |
| 7 | 4 | 59 | 6579 |
| 8 | 2 | 28 | 3118 |

Table 2: Simulated scenarios in the experiments.

| Scenario | Train (Table 1) | Reuse plans | Stretch |
|---|---|---|---|
| A | 1 | No | $S_1$ |
| B | 1 | Yes | $S_1$ |
| C | 1 | Yes | $S_2$ |
| D | [1;8] | Yes | $S_1$ |

The scenarios are evaluated according to the efficiency (%) of the recovery and adaptation steps of the cases. This percentage indicates the success of recovery or adaptation of a case. For example, at any given time, the Memory agent recovers, for the Planner agent, a case with a set of actions $A = \{3,4,4,4\}$. This set is adapted by the Planner, resulting in $A = \{4,4,4,4\}$. Thus, the recovery task has an accuracy of 75%. Then, $A$ is passed to the Executor agent and applied without any changes, resulting in a 100% fitting accuracy. Soon, the adaptation effort is 25% and the execution effort, in terms of adaptation, is null.

In scenario A, the Memory agent uses only the initial case base. Ten trips were planned and all of them for train 1 in stretch $S_1$. All plans were executed by an Executor agent. At the end of each trip, the new experiences (new plans) were not incorporated to the case base of the Memory agent. Figure 9 presents the efficiency (%) of the recovery and adaptation steps of all the cases in the ten trips. It is observed that the recovery of cases is less effective than the adaptation of cases. In percentage, the difference between the recovery and adaptation task corresponds to the contribution of the adaptation task to make the plan applicable to a given case. The average of this difference is 8%, with a standard deviation of 1%. Compared to a method of satisfaction of constraints, the effort obtained is less in terms of memory used, execution time, and number of states. Regardless of the highest peaks of success of the recovery and adaptation task being 46% and 51%, respectively, the generated and executed plans are similar at 86%. The similarity is calculated by the Cosine distance.

In scenario B, the same train configuration and the same stretch from scenario A was used. However, at every trip made by the Executor agent, the applied plan was incorporated to the case base of the

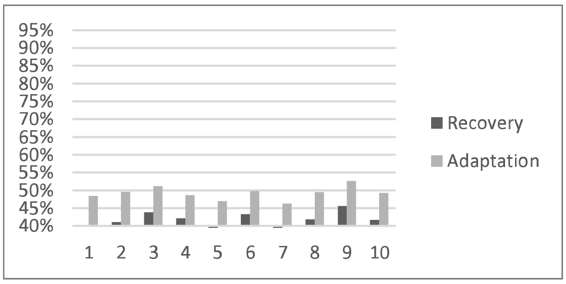


Figure 9: Results of scenario A. Ten trips with the same train configuration (configuration 1), in stretch $S_1$ and without reusing plans.

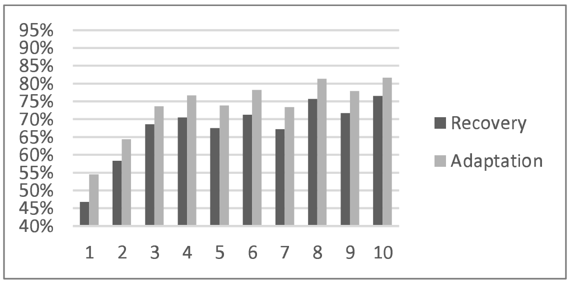


Figure 10: Results of scenario B. Ten trips with the same train configuration (configuration 1), in stretch $S_1$ and reusing implemented plans.

Memory agent. Figure 10 shows that the inclusion of new plans in the case base of the Memory agent increased by approximately 25% the efficiency of the recovery and adaptation tasks. The similarity between what was planned and what was executed is on average 90%. The addition of new cases to the experience base broadens the efficiency of the planning task. It is observed that between trips 1 and 3, there is an increasing linear trend of 20%. Moreover, from trip 3, there is a slightly increasing stability, with average variation of 4% for both tasks, and with standard deviation of 2%. The variation is justified because the trips followed speeds similar to each other, but with differences in driving plans at certain times. This difference occurs because of adaptations made by the genetic algorithm, which in some places suggested different acceleration points. This results in variation in the power used, and consequently, variation of the practiced speeds. This fact is inherent to the natural behavior of the genetic approach (e.g., mutation).

In scenario C, as in scenario B, on each new trip made by the Executor agent, the applied plans are incorporated into the experience base of the Memory agent, and thus reused by the Planner agent. The case base of the Memory agent began with the experiences generated in scenario B. The steps for recovery and adaptation have an average efficiency of 80% and 86%, respectively (see Figure 11).

In percentage, the difference between the recovery and adaptation task was 9% at the beginning of the experiment, and then immediately fell to an average of 5%, with a standard deviation of 1%. Despite increased effort because of unfamiliarity with the environment, the results are significant. These results encourage the use of a collaborative approach between agents, located at different stations, to exchange plans.

Figure 12 shows the results of scenario D, where the trips always occur in the same track, but with eight different train formations. In this scenario, the Memory agent initiates the experiences of scenario B. In

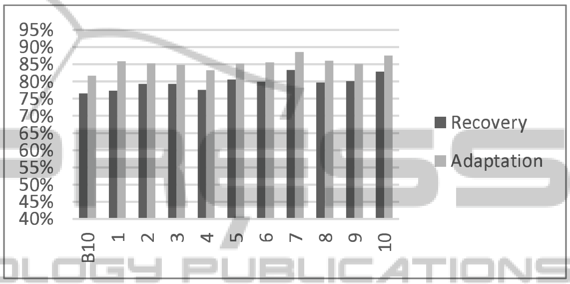


Figure 11: Results of scenario C. Same train configuration (configuration 1) and initial plans of scenario B, but in stretch $S_2$.

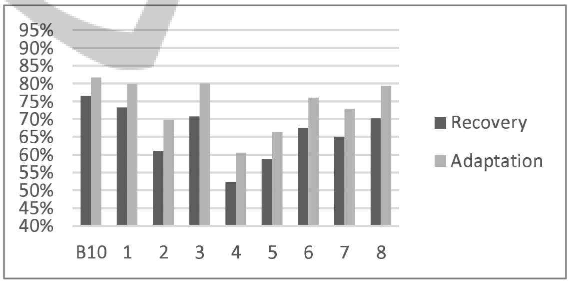


Figure 12: Results of scenario D. Different train configurations (configurations 1 to 8), initial plans of scenario B, stretch $S_1$.

terms of efficiency of recovery and adaptation tasks, the adaptation task proves superior. There is an expected drop in success because the train configurations are different for each trip. However, even in this scenario without repetition of train configuration, it is possible to note that, to the extent that new cases are included in the base, the overall efficiency improves. Hopefully, with a greater number of trips with similar configurations, efficiency rates move rapidly towards scenario B.

In the scenarios on which we worked, it can be observed that without reusing plans as past solutions, the average success rates in the recovery and adaptation tasks remain low, 42% and 49%, respectively. However, when we start to reuse the plans as past solu-

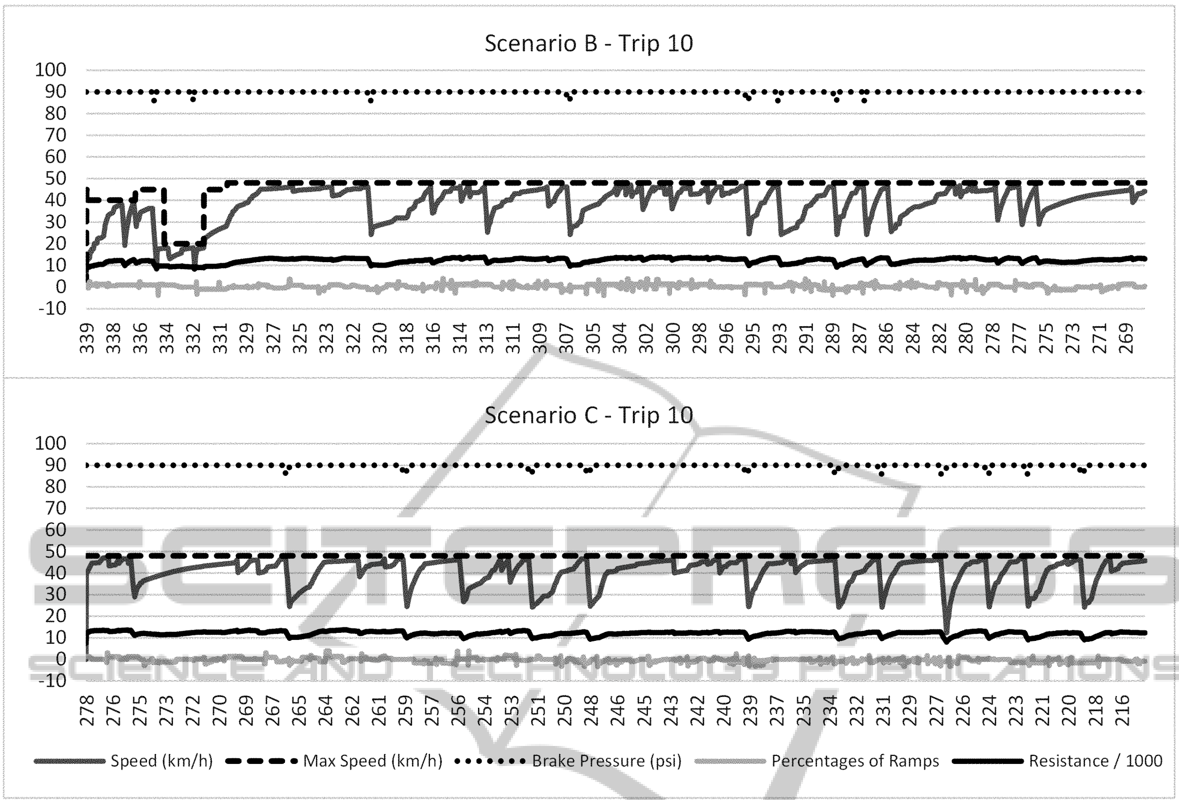


Figure 13: Data of trips made for the same train configuration in different stretches.



tions, the average success rates of recovery and adaptation tasks increase to 64% and 74%, respectively. In terms of complexity, raising the efficiency of such tasks reduces the effort of searching for the problem solution. Finally, the contribution of the adaptation task, in the reuse of shared experiences of different stretches, remained at the same level of scenario B. This suggests that the accumulation of cases in a given scenario proves to be useful in another scenario.

Figure 13 shows graphically the execution of the ten trips for scenarios B and C. It can be noted that there is a significant difference for each profile. Such differences can be observed in the maximum speeds practiced, percentages of ramps, and resistances. The evolution of the speeds shows a significant difference in driving style. The decrease in speeds is related to the applications of the brakes, and to reflections of the driving policy that attempts to maintain the speed of the train near the maximum allowable speed. Such heuristics are applied to attempt to reduce the duration of a trip. It should be indicated that the first application of the brake follows a default value and must heavily influence train speed. A smoother speed reduction without exceeding the safe speed limit is desired. Such a study is not part of this paper, but it can be incorporated into the Planner agent, thus giving it a look ahead mechanism.

In general, in all observed situations, the effort on adaptation is present, but efficient, and rises over time. For the field in question, as the trains and the environment change, the plans used in past situations are not easily applicable to others; this effectively requires an adaptive and efficient approach. This finding goes to the direction of what has classically been understood as an advantage of the CBR approach in view of a rule-based approach (Kolodner, 1993). The latter approach requires explicit knowledge models regarding the application domain. This demand is difficult to execute in a complex environment. It also fails when there is no rule that can be applied in the field. Furthermore, if the environment changes, there is a need to update the rule base to be able to derive a solution. As in CBR, knowledge of the field is represented in the form of cases. It exempts an explicit representation of the application domain. This allows dynamically maintaining and learning new knowledge as new cases are incorporated into the case base.

Table 3 is a comparison table that contrasts the performance of human drivers (Actual column) driving a simulator where the actions applied are determined by a constraint satisfaction system (DCOP column) (Sato et al., 2012) and by the approach pre-

Table 3: Consumptions obtained in scenario C.

| Train | Consumption (LGTT) Actual (A) | Consumption (LGTT) DCOP (B) | Reduction Our (C) | Reduction C-A (%) | Reduction C-B (%) |
|---|---|---|---|---|---|
| 1 | 6.19 | 4.16 | 3.36 | 50% | 26% |
| 2 | 5.68 | 4.18 | 4.22 | 30% | -5% |
| 3 | 6.23 | 4.09 | 3.95 | 41% | 10% |
| 4 | 6.49 | 4.51 | 3.88 | 46% | 23% |
| 5 | 6.29 | 4.22 | 3.31 | 49% | 24% |
| 6 | 6.17 | 3.99 | 3.69 | 40% | 8% |
| 7 | 6.26 | 4.07 | 3.86 | 42% | 11% |
| 8 | 5.68 | 4.41 | 4.00 | 34% | 6% |

sented (Our column). The DCOP column represents the best values obtained in this approach. It is emphasized that for all consumption values (measured in LGTT), our approach is higher than for the other competitors, except on a single opportunity (train 2), where the DCOP is higher by 5%.

The feasibility of an automatic train driving system seems significantly important. For example, for a fuel consumption expenditure of approximately 250 million dollars per year, any cost savings above 6% can have a significant impact on the competitiveness of the freight transport sector.

# 5 CONCLUSIONS

This paper presented a collaborative approach for sharing experiences in generating plans for driving trains. The results obtained showed that the adopted approach can be generalized and deployed at various stations of a rail network. We showed that the efficiency of recovery and adaptation tasks increases as new cases are obtained. Such efficiency generates a tendency to reduce efforts in planning and re-planning driving plans. Obviously, if conditions change significantly, planning efforts increase, at least initially.

Finally, in terms of domain application, two results are important: in monetary terms, the generated driving plans can produce significant gains; and in terms of reuse of experiences, the approach suggested that good drivers should be used to drive trains in several different stretches of a railroad, for a certain time, in order to generate experiences. Such experiments can then be used to generate good plans for less experienced drivers. This helps rationalize the expertise capable for driving trains efficiently. Future work should follow the following directions: avoiding unnecessary stops (Dordal et al., 2011), and ensuring the certification of the information exchanged.

# REFERENCES

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59.

Baeck, T., Fogel, D., and Michalewicz, Z. (2000). *Evolutionary Computation 1: Basic Algorithms and Operators*. Basic algorithms and operators. Taylor & Francis.

Bajo, J., Corchado, J., and Rodríguez, S. (2007). Intelligent guidance and suggestions using case-based planning. In Weber, R. and Richter, M., editors, *Case-Based Reasoning Research and Development*, volume 4626 of *Lecture Notes in Computer Science*, pages 389–403. Springer Berlin Heidelberg.

Borges, A., Dordal, O., Sato, D., Avila, B., Enembreck, F., and Scalabrin, E. (2012). An intelligent system for driving trains using case-based reasoning. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 1694–1699.

Company, G. R. S. (1979). *Elements of railway signaling*. General Railway Signal.

D., E. and Houpt, P. (2011). Trip optimizer for railroads. Technical report, IEEE Control Systems Society.

Dordal, O., Borges, A., Ribeiro, R., Enembreck, F., Scalabrin, E., and Avila, B. (2011). Strong reduction in fuel consumption driving trains in bi-directional single line using crossing loops. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 1597–1602.

Fang, W., Sun, J., Wu, X., and Yao, X. (2013). Rescheduling in railway networks. In *Computational Intelligence (UKCI), 2013 13th UK Workshop on*, pages 342–352.

Gu, Q., Cao, F., and Tang, T. (2012). Energy efficient driving strategy for trains in mrt systems. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 427–432.

Hengyu, L. and Hongze, X. (2012). An integrated intelligent control algorithm for high-speed train ato systems based on running conditions. In *Digital Manufacturing and Automation (ICDMA), 2012 Third International Conference on*, pages 202–205.

Khan, M. J., Awais, M. M., Shamail, S., and Awan, I. (2011). An empirical study of modeling self-management capabilities in autonomic systems using case-based reasoning. *Simulation Modelling Practice and Theory*, 19(10):2256 – 2275.

Kolodner, J. (1993). *Case-based Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Loumiet, J., Jungbauer, W., and Abrams, B. (2005). *Train Accident Reconstruction and FELA and Railroad Litigation*. Lawyers & Judges Publishing Company.

Luke, S., Panait, L., Balan, G., and Et (2014). Ecj 21: A java-based evolutionary computation research system.

Magalhães Mendes, J. (2013). A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS Transactions on Computers*, 12(4):164–173.

Mitra, R. and Basak, J. (2005). Methods of case adaptation: A survey: Research articles. *Int. J. Intell. Syst.*, 20(6):627–645.

Mota, J. S., Câmara, G., Fonseca, L. M. G., Escada, M. I. S., and Bittencourt, O. O. (2008). Applying case-based reasoning in the evolution of deforestation patterns in the brazilian amazonia. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 1683–1687, New York, NY, USA. ACM.

Navarro, M., De Paz, J. F., Julián, V., Rodríguez, S., Bajo, J., and Corchado, J. M. (2012). Temporal bounded reasoning in a dynamic case based planning agent for industrial environments. *Expert Syst. Appl.*, 39(9):7887–7894.

of the Assistant Secretary for Research, O. and of Transportation (US DOT), T. O.-R. . U. D. (2014). Table 4-17: Class i rail freight fuel consumption and travel. Eletronic.

Pinto, B., Scheneebeli, H., Borba, J., Amaral, P., and A.B., F. (1985). Microcomputador de bordo para controle de potência de locomotivas em tração múltipla. In *II Congresso Nacional de Automação Industrial*, volume 1, pages 125–129, São Paulo, SP, Brazil.

Sato, D., Borges, A., Leite, A., Dordal, O., Avila, B., Enembreck, F., and Scalabrin, E. (2012). Lessons learned from a simulated environment for trains conduction. In *Industrial Technology (ICIT), 2012 IEEE International Conference on*, pages 533–538.

Schank, R. C. (1983). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, New York, NY, USA.

Spalzzi, L. (2001). A survey on case-based planning. *Artificial Intelligence Review*, 16(1):3–36.

Vacek, S., Gindele, T., Zollner, J., and Dillmann, R. (2007). Using case-based reasoning for autonomous vehicle guidance. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 4271–4276.

Wang, C.-S. and Yang, H.-L. (2012). A recommender mechanism based on case-based reasoning. *Expert Syst. Appl.*, 39(4):4335–4343.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152.