Towards policies for data insertion in dynamic data driven application systems: a case study sudden changes in wildland fire

Roque Rodríguez, Ana Cortés, Tomás Margalef

Computer Architecture and Operating Systems Department, Universitat Autònoma de Barcelona, Bellaterra, Spain

Abstract

We have applied the Dynamic Data Driven Application System (DDDAS) methodology to predict wildfire propagation. Our goal is to build a system that dynamically adapts to sudden changes in environmental conditions. For this purpose, we are building a parallel wildfire prediction method, which is able to assimilate real-time data to be injected in the prediction process at execution time. This data-injection needs to be intelligent in order not to disturb the simulation process outputs. In this paper, we propose a policy for data insertion using a statistical approach and we design a set of experiments based on California wildfire where Santa Ana winds generate the ideal conditions for sudden changes in fire behavior.

Keywords: Dynamic Data Driven Application System, Parallel computing, Forest fire prediction, HPC, Evolutionary computing, Data insertion policy

1. Introduction

During a wildfire, sudden changes in environmental conditions can result in dramatic changes in fire behavior. These changes represent a significant risk for the people who are fighting against the forest fire. The wildfire in Slovak Paradise National Park in 2000 is a case where people were entrapped by fire (six people’s lives lost) [1]. The technical analysis of this catastrophe concluded that the main factors which determined the mentioned extreme fire behavior were a sudden change of the wind speed combined to an oval shape of the valley with abrupt slopes resulting in an intense growth in the uphill direction. Another example of the tragic consequences of sudden changes in environmental conditions is the wildfire fire produced last summer (21 July 2009) in Horta de Sant Joan - Catalonia (Spain) where four firefighters died. The firefighters were reportedly trapped during a change in wind’s direction, which blocked their escape route on an 1975 acres forest fire in El Port National Park.

Southern California is one of the most prone regions in North America to the development of large wildfires, due to a unique combination of environmental conditions that are favorable to the development of large fires such as volatile fuels, dense fuel growth, steep terrain, and recurrent droughts. The influence of the Santa Ana winds further increases the risk of large fires. In a Santa Ana event [2], the normal onshore marine airflow is replaced by offshore, downslope
winds. The combination of high wind speeds, dry continental air, clear skies, and increased temperatures can rapidly dry fuels. The prevailing conditions during a Santa Ana event, including low relative humidity, high wind speed, and warming, can result in "explosive" burning conditions [3, 4]. This means, the probability of sudden changes in wind conditions is very high.

Dynamic Data Driven Application System (DDDAS) [5, 6] for forest fire spread prediction arises as a relevant tool to approach this kind of catastrophes. Basically, a DDDAS is a paradigm whereby application (or simulations) and measurements become a symbiotic feedback control system. DDDAS entails the ability to dynamically incorporate additional data into an executing application and, in reverse, the ability of an application to dynamically steer the measurement process. Such capabilities improve the classical forest fire spread prediction because it allows guiding the simulation by injecting real time data during the prediction process. However, an arbitrarily data injection into the system could not be always profitable. In order to maximize the improvements provided by real time data injection, data injection policies should be designed to determine an acceptable variability threshold for the input variable, which determines whether to inject the data or not.

2. Related work

Classically, forest fire spread prediction has been performed by executing a plane fire simulator (FS) initialized with the required input data such as initial fire front, vegetation type, meteorological conditions, interval time to be simulated and so on. The prediction provided by this basic scheme usually generates erroneously outputs that do not match the real fire propagation. For this prediction scheme, the interval time \((t_0-t_1)\), where \(t_0\) is the initial instant time and \(t_1\) is the instant time at which the fire spread is going to be predicted, is known as the Prediction phase or Prediction stage. In our research, we included another phase, previous to the prediction one, where the simulator’s input parameters are calibrated depending on the observed fire’s behavior. This phase is called Calibration phase or Calibration stage. Figure 1 shows how this two stages prediction method works. The Calibration stage lasts from \(t_0\) to \(t_1\) and the Prediction stage goes from \(t_1\) to \(t_2\). The Calibration stage needs two real fire fronts to be operable, the initial fire front at \(t_0\) and a posterior real fire evolution at \(t_1\) for comparative purposes. Once the set of input parameters has been calibrated, the system shifts to the next time interval \(t_1-t_2\) and uses the adjusted set of input parameters (called scenario) as inputs to the fire simulator in the Prediction stage. Finally, the prediction provided for the whole system is the fire spread evolution provided as output by the Fire Simulator (FS) in the Prediction stage [7, 8].

The initial design of this scheme was done using as a working hypothesis the idea that the environmental conditions keep quite similar during both the Calibration stage and the Prediction stage. Two alternatives have been developed to approach this two stages prediction scheme: the unique solution and the multiple solution approach also called statistical integration. The two stages prediction scheme based on a unique solution basically consists of applying an optimization algorithm during the Calibration stage, which provides as input to the Prediction stage a unique combination of the input variables. Genetic Algorithm (GA) has been demonstrated to be the most efficient scheme for this kind of problem. However, the main drawback of this approach is the loss of reliability when the working hypothesis is broken. A first attempt to overcome this problem was the design of a multiple solution scheme.
called Statistical Integration. This approach is based on analyzing a probability map obtained by overlapping the fire evolution provided for a huge number of scenarios. Then, the best probability is chosen for prediction purposes. This probability value implicit includes the information of multiples scenarios what produces a positive effect in the prediction provided by the entire system. However, the main drawback of such a scheme is the execution time. In order to be useful, the statistical approach must execute hundreds of thousands of scenarios becoming a very time consuming task. Consequently, the fire spread prediction could be provided in an outdated time what have no sense if we want to deal with a real time prediction system.

Therefore, new techniques could be designed to overcome the above mentioned disadvantages. We propose an alternative two stages prediction scheme called SAPIFE3 - this is the Spanish acronym for Adaptive System for Fire Prediction Based in Statistical-Evolutive Strategies (Sistema Adaptativo para la Predicción de Incendios Forestales basados en Estratégias Estadístico-Evolutivas) [9, 10], which joins the advantages of the two previous described fire spread prediction methods (GA and Statistical Integration). This new approach is able to reduce the number of total scenarios to simulate, from a number such as hundreds of thousands to some hundreds, by optimizing the set of scenarios through the use of a Genetic Algorithm. Although the first design of SAPIFE3 did not consider real time data injection into the system, this feature was immediately included in the next version referring to this prediction scheme as SAPIFE3rt. In figure 2, we see the main components of the SAPIFE3rt, which are going to be described with more details in the next section.

3. SAPIFE3rt

SAPIFE3rt has been implemented using the master/worker programming paradigm. The Master node is responsible for running the main tasks of the two method’s phases, the GA and the Statistical Integration. Furthermore, the Master receives data from the external sources at run time, such as data from Surface Weather Stations. The Worker nodes run f(FireSim), our black-box simulator, which is in turn the most compute-intensive part of our method. Following, we will describe in more detail how each component of SAPIFE3rt is implemented:

1. Genetic Algorithm: At the initial time, the master receives data from external sources: fire propagation maps, topographical data and meteorological variables. Once all this initial data has been properly set up, the master
starts running the GA and generates the scenarios’ population. Afterwards, the initial real fire propagation map and these scenarios are distributed among the Workers using a Factoring Scheduling policy [11], avoiding communication bottlenecks. Workers start running \( f(FireSim) \) over the received scenarios and compare their outputs to the real propagation map obtaining an error value for each scenario simulated. The workers then send back this error to the master. As the master receives all scenarios, it proceeds to run the evolutive operations (elitism, selection, crossover and mutation). This first phase is repeated over a number of \( n \) generations. The final output of this phase is a population of Best Scenarios (BSc).

2. **Statistical Integration**: The input of the second phase is the BSc from the previous phase. These scenarios are distributed to the workers. They can be modified according to the existence of a sudden change in wind’s condition. If there is a sudden change either in wind direction or wind speed, the population of scenarios will be modified prior to their delivery to the workers. In each new scenario, the wind values are changed according to the new data obtained from the Surfaces Weather Stations and a dispersion factor around this new data. This new population is called Best Scenario Modified (BScM) and is then sent to the workers. The workers will then run \( f(FireSim) \) over this population and return back the simulated propagation maps. Finally, the master integrates all these maps and, according to the real map, it generates a probability map that will be used as prediction for the next time frame.

3. **Data Collection System component**: This component is responsible to gather all information regarding the fire’s environment, such as weather, topography and terrain composition data (the fuel) and must work alongside GIS (Geographical Information System) tools. This module must also be well connected to a network of weather stations such as the Network of Automatic Weather Stations of Catalonia’s government. This module will also be in charge of injecting data at real-time. If a sudden change is detected, it will be introduced in the form of replacing the worst individual who came from the GA.

4. **Policies for data insertion**

As we have previously mentioned, the initial working hypothesis of the two stages prediction methods required that the environmental conditions keep quite similar both in Calibration stage and Prediction stage. Therefore, we need to check the degree of variability of the environmental conditions before the Prediction stage takes place. In the next section, we shall explain the methodology to detect sudden changes in environmental conditions in order to decide whether the data collected should be inserted into the system or not.

4.1. **Sudden changes detection**

To determine whether a data should be inserted into the system or not, first of all, we need to study the frequency of reading data from the sources. This value is significantly different depending on the type of data we are dealing with. Subsequently, we introduce a simple classification of SAPIFE\(_3\) inputs according to their variability.

1. **Fixed-data**: This data usually is stored in databases and is read previous to execution time and their values keep constant throughout the simulation process. In other words, we need to read this data from databases only once before beginning the simulation process.

2. **Dynamic-data**:

   a. **Remote sensing and pre-processed by Geographical Information Systems (RSGIS)**: these data are satellite images of the fire evolution. They are taken many times when the hazard occurs. These data give information about the behavior of the forest fire and permit us to calibrate the SAPIFE\(_3\) outputs.

   b. **Weather Stations (WS)**: these data have the most dynamic behavior of all the simulator inputs. Their values depend on the environmental conditions and they need real-time monitoring during the simulation process. SAPIFE\(_3\) used these parameters to determinate if a sudden change has occurred and steered the simulation process. For instance, the wind direction and wind speed are read from weather stations near the fire zone.

3. **Data-estimated**: this kind of input data corresponds to parameters whose values are unknown prior to the simulation process and they could not be obtained during the simulation. Those values should be either calculated or estimated before the execution.
In this paper, we have used the FireLib [12] simulator (a C library based on BEHAVE fire behavior algorithms [13]) as a simulation core. Table 1 lists all input data needed for this simulator, as well as their corresponding type and sources. FireLib is a cellular automata simulator, where the terrain is a matrix of cells and, as output, it generates the average time that each cell is burned, as well as the average flames’ height. To evaluate the fire simulator outputs we used the error formula shown in equation (1). This equation calculates the differences in the number of cells burned, both missing or in excess, between the simulation and the real fire.

\[
Error = \frac{(\text{Cells} \cup \text{InitCells}) - (\text{Cells} \cap \text{InitCells})}{\text{RealCells} - \text{InitCells}}
\]  

(1)

Table 1: SAPIFE3 inputs

<table>
<thead>
<tr>
<th>Inputs</th>
<th>type</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M) Fuel model</td>
<td>Fixed-data</td>
<td>Databases</td>
</tr>
<tr>
<td>(W_{\text{spd}}) Wind speed</td>
<td>Dynamic-data</td>
<td>Weather Stations</td>
</tr>
<tr>
<td>(W_{\text{dir}}) Wind direction</td>
<td>Dynamic-data</td>
<td>Weather Stations</td>
</tr>
<tr>
<td>(S) Slope</td>
<td>Fixed-data</td>
<td>Databases</td>
</tr>
<tr>
<td>(\arctan\left(\pi \times \frac{S}{180}\right)) Aspect</td>
<td>Fixed-data</td>
<td>Databases</td>
</tr>
<tr>
<td>(M_1) 1-hr dead fuel moisture</td>
<td>Data-estimated</td>
<td>SAPIFE3</td>
</tr>
<tr>
<td>(M_{10}) 10-hr dead fuel moisture</td>
<td>Data-estimated</td>
<td>SAPIFE3</td>
</tr>
<tr>
<td>(M_{100}) 100-hr dead fuel moisture</td>
<td>Data-estimated</td>
<td>SAPIFE3</td>
</tr>
<tr>
<td>(M_{\text{herb}}) Live herbaceous fuel moisture</td>
<td>Data-estimated</td>
<td>SAPIFE3</td>
</tr>
<tr>
<td>Mapini Initial map</td>
<td>Dynamic-data</td>
<td>RS GIS</td>
</tr>
<tr>
<td>Mapcivo Evolution map</td>
<td>Dynamic-data</td>
<td>RS GIS</td>
</tr>
</tbody>
</table>

The time interval of the Calibration stage should be the same as for the Prediction stage. This period of time is determined by the frequency of the read map. Therefore, if we read the first map at time \(t_0\) and the second map at time \(t_i\), the period for calibration stage is \(P = t_i - t_0\). Usually the frequency of reading data from weather stations is greater than the frequency of reading a map. Furthermore, we can evaluate the sample of dynamic data for each Calibration stage. In particular, if we can obtain data from the weather stations every \(t_i\) and the evolution map each \(t_m\), the sample size for studying the sudden changes is: \(N = t_m/t_i\).

To illustrate how the detection of sudden changes has been implemented, we have used as a representative example the information collected at the weather station KCAYORBA4 [14] located at the Complex Fire of California in November 2008. In figure 3 we see the samples of wind speed taken from the weather station on Calibration stage \(i\) and prediction stage \(i + 1\). This differences in the wind speed dispersion can cause that the prediction could get away from the real fire propagation, therefore, we must define a method for detecting this environmental conditions variation between \(stage_i\) and \(stage_{i+1}\).

To measure sudden changes, we use statistical tools as Arithmetic Mean (\(\bar{X}\)) and Standard deviation (\(\sigma\)). The ratio between mean and deviation is the index of variation of one variable throughout two stages. Therefore, in figure 4 we observe that \(\bar{w}_{i}\) is the arithmetic mean of wind speed data in the calibration stage \(stage_i\); and \(\sigma_i\) is the standard deviation of wind speed data in the calibration stage \(stage_i\). Consequently, \(\bar{w}_{i+1}\), \(\sigma_{i+1}\) are the arithmetic mean and standard deviation in the prediction stage \(stage_{i+1}\). For that reason, if we use this data for marking two points in the Cartesian coordinate system where the “\(y\)” axis represents the wind speed mean and the “\(x\)” axis shows the standard deviation, it allows us to calculate the distance between these two points using the Pythagorean formula for the distance. The distance \(\alpha\) between these points is the Change Factor of a given Variable (CFV) through two stages. If \(\gamma = \bar{w}_{i+1} - \bar{w}_{i}\) and \(\beta = \sigma_{i+1} - \sigma_{i}\), the method for distance calculation is

\[
CFV = \sqrt{(\bar{w}_{i+1} - \bar{w}_{i})^2 + (\sigma_{i+1} - \sigma_{i})^2}
\]  

(2)
The α distance is the change factor of variation of each variable (CFV) through two stages. Using this factor we can measure the differences in the variable behavior between Calibration stage and Prediction stage. Therefore, we should set a threshold to determine if a sudden change has occurred and if it affects the system outputs in a negative way.

As we have previously mentioned, to calculate the threshold, we carried out a set of tests using data gathered from KCA YORBA4. The objective of these tests is to determine the relationship between CFV and the error level. To facilitate the study, we take into account only wind speed and we configure SAPIFE3rt without data insertion at execution time. So, we focus only on one variable to calculate the CFV to study how it affects SAPIFE3rt outputs. Accordingly, we can study each variable in an independent way.

If X and Y represent the CFV and error-prediction respectively, the correlation is:

\[
Corr = \frac{\bar{xy} - \bar{x} \bar{y}}{\sqrt{\bar{x}^2 - \bar{x}^2} \sqrt{\bar{y}^2 - \bar{y}^2}}, \text{ where } \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \text{ is the mean of the data } x
\]

The correlation is 1 in the case of an increasing linear relationship, -1 in the case of a decreasing linear relationship, and some value in between in all other cases, indicating the degree of linear dependence between the error and CFV. For these cases of study, we obtain a correlation value equal to 0.97 (corr=0.97) indicating the strong dependence between the variables. Figure 5 presents the results obtained by this method. We see that SAPIFE3rt has the best error level when the CFV is lower. The error tends to increase when the CFV increases. In addition, if CFV is lower than 1.5, prediction error of SAPIFE3rt is below 30%. We use this data to configure the CFV threshold in (1.5). In other words, if \( CVF > CVF_{threshold} \), then SAPIFE3rt should insert data at execution time.

4.2. CFV estimations

The results obtained in the previous section demonstrated that the working hypothesis for the two stages method is correct. We have observed that if the conditions in the Calibration stage are quite similar to the conditions in the Predictions stagei+1, the prediction error tends to be lower. However, if we focus on the problem in the context of real-time, we cannot calculate the CFV with all wind samples for prediction stagei+1. This means SAPIFE3rt can get all wind samples from calibration stagei, but only some wind samples from prediction stagei+1 (see figure 3). Therefore, \( \bar{w}_{i+1} \) and \( \sigma_{i+1} \) are calculated based on a number of samples n that depends on GA execution time.

Figure 6 shows the process and data insertion with time execution. We see that the first real map to be injected in the system by SAPIFE3rt is Mapini (used as initial map). The second real map Maperror is used by the GA process in the error function (see equation 1). When GA is running, the system is able to assimilate data. This data is the wind data coming from the weather station. The data gathering when GA is running is used by the system to calculate the CFV. If the CFV exceeds the CFV\(_{threshold}\), the final population scenarios from GA is modified between 25% and...
30%. We can see from this example that the SAPIFE$_{rt}^3$ can read three wind samples to estimate $\overline{w}_{i+1}$ and $\sigma_{i+1}$ before finishing the GA. Therefore, scenarios that have a high error are selected and their wind genes are modified. This new population is called Best Scenarios Modified and is then sent to StatisticalIntegration SI. The SI integrates all these maps, and according to the real map, it generates a probability map that represents the prediction for minute $t_60$.

The SAPIFE$_{rt}^3$ flexibility allows us to set the system for map insertion every one hour and data wind to be injected every 5 minutes. Therefore, we can configure the calibration and prediction stage in every one-hour steps. In this way, SAPIFE$_{rt}^3$ has more time for execution with a large number of scenarios and it can also gather more wind samples to estimate $\overline{w}_{i+1}$ and $\sigma_{i+1}$.

In figure 7, we see the CFV based on three samples for estimate $\overline{w}_{i+1}$ and $\sigma_{i+1}$. The CFV$_{real}$ based on the total samples of wind gathering on calibration stage, and prediction stage$_{i+1}$. We noticed that real and estimated CFV do not follow the same tendency. Therefore, we must take into account this factor to calculate the threshold. In this case, we see a penalty caused by the small number of samples. However, in figure 8, we show that the values of real and estimate CFV values do follow the same tendency. The estimation for $\overline{w}_{i+1}$ and $\sigma_{i+1}$ is better than the previous test because, in this example, we have more samples to calculate CFV. Consequently, if we decrease the frequency of reading the map and keep reading frequency of wind, the system has more samples to improve the estimation of $\overline{w}_{i+1}$ and $\sigma_{i+1}$. It is important to notice that the number of samples is always 50% of the total in these two cases. These results show that the wind pattern tends to have a defined behavior through time. However, we see how over time, the wind pattern tends to have a defined behavior. Consequently, we could identify those patterns if we have enough samples.
5. Experiments Results

In November 2008, southern California was hit by devastating fires. The extreme conditions of Santa Ana’s winds [2] combined with the environment’s low humidity, created the ideal conditions for fires. In order to test our DDDAS forest-fire propagation prediction system, we performed a set of postmortem experiments based on the conditions of the Freeway Complex Fire. The main objective of these experiments is to demonstrate the benefits of DDDAS for forest fire prediction, especially when environmental conditions are quite dynamic showing suddenly changes in wind speed and wind direction. Figure 9 shows the fire perimeter in November 16, 2008 where it is possible to visualize the KCAYORBA4 weather station at the bottom of the image. We used KML language to visualize the fire into Google Earth, so it is possible to verify the situation of the fires.

Now we want to test if the policy for data insertion can improve the prediction results. For this purpose, we have set up two types of experiments.

5.1. Experiment 1

We set SAPIFE$^3$ to read wind data every 5 minutes and real maps every 30 minutes. Therefore, in figure 10 we see the results obtained by SAPIFE$^3$ using a policy for data insertion and without using any insertion. In last case, if the system does not use a policy to insert data, the data is always injected at execution time.
Figure 10 represents in the $y$ axis the error between the simulated and the real map. In the $x$ axis we have the simulation steps for each time period. The obtained results show that the errors are lower in steps 1 and 5 when SAPIFE$_3rt$ uses a policy for data insertion. It also improves the prediction error about 15% about the SAPIFE$_3$ without policy. That means that, the $CFV$ is lower than $CFV_{threshold}$, so the system has decided not to inject data. Obviously, inserting data in these two cases has a negative effect in the prediction results because the working hypothesis, based on the idea that the environmental conditions keep quite similar during both the Calibration stage and the Prediction stage, for these steps is not broken. Therefore, the GA is the most efficient scheme under those conditions and injecting data disturbs the GA.

In figure 11, we see in the $y$ axis the arithmetic mean of wind speed throughout each stage. The $x$ axis shows the standard deviation for each stage. Every step is composed of two stages, stage, and stage$i+1$, respectively. That is the correspondence between Calibration and Prediction stages in our method. Such as, step 1 (in the figure 10) is composed of stages 0 and 1. Step 2 is composed of stages 1 and 2. Finally, the last step 5 is composed of stages 4 and 5. Therefore, we see the distances between points is the $CFV$ (shown in figure 5). Consequently, we can establish the relationship between the prediction error level and the wind behavior. These results show that the system made the correct decision to inject data. We see that the distance between stages 0 and 1 is short because the environmental conditions in these stages are quite similar. That means that in this point, it is not necessary to inject data, according with the working hypothesis of the method. In the same way, we observe a similar behavior in stages 4 and 5. Therefore, if we look for the steps 1 and 5 in figure 10, we see the consequence for injecting data when it is not necessary.

5.2. Experiment 2

For the second experiment, we configure the SAPIFE$_3rt$ inputs to read maps every one hour and to have the wind insertion every 5 minutes. The main difference in this experiment is that the number of samples of wind to estimate is bigger than in the previous test. It is important to notice that if the timeframe is long, we have a higher probability of sudden changes.

![Figure 12: Experiment 2, sudden changes impact on the data insertion policy](image)

![Figure 13: Experiment 2, ratio between mean win speed and standard desviation](image)

In figure 12, we observe that SAPIFE$_3rt$ (with policy for data insertion) improves the prediction results on steps 4 and 5. Furthermore, in figure 13, we note that the distance between stages 3 and 4 or 4 and 5 are very short. This means, if steps 4 and 5 are composed by stages 3 with 4 and 4 with 5, respectively, we have a direct relationship between the error and distance. In other words, when the distance is short, data insertion is not necessary. For this case, SAPIFE$_3rt$ makes a correct decision when it uses a policy for data insertion. It is important to notice that the distance is the $CFV$ we showed in equation 2 where the main difference is that $\bar{w}_{i+1}$ and $\sigma_{i+1}$ are estimated by six samples of wind. Finally, in the other steps, the error level is quite uniform when the data is injected.

6. Conclusion and ongoing work

In this work, we have demonstrated that to improve the results of a DDDAS, a policy for data injection is required in order not to disturb the simulation outputs. As a study case, we have used a forest fire spread prediction system...
called SAPIFE$^3_{rt}$. Although the underlying fire spread simulator has a wide set of input variables, we focus our study on the wind parameter and, in particular, on the wind speed value. We have proposed an injection policy based on parameter variability through time (Change Factor of a given Variable, CFV) to decide whether it is profitable to inject data or not. The proposed policy can be improved if we add more variables to the equation. We have validate CFV using data from the Free Complex Fire occurred at California in November 2008. We also conclude that the acquisition data frequency directly affects the prediction results, as well as the precision on the detection of sudden changes. Since data insertion is useful in the presence of sudden changes, as much accurate the detection of this event is, better prediction results will be obtained.

This work also encourage us in the way to achieve the implementation of policies for data insertion on DDDAS for forest fire behavior prediction that could provide the rules for design a policy of data insertion for any DDDAS.

Acknowledgment

This work is supported by the MEC-Spain under contract TIN 2007-64974.

References