

### 5.1.3 Piecewise Linear Regression for Life Cycle shape extraction

This section introduces the two piecewise linear trend extraction approaches used in the study to extract life cycle shapes of each game.

#### 5.1.3.1 Method 1 - Piecewise linear regression when the number of pieces is unknown

This section introduces a piecewise linear trend extraction algorithm to extract the life cycle shape of a game without prior knowledge about the number of life cycle stages.

The main objective of the proposed algorithm is to identify the life cycle shape represented through the use of a minimum number of life stages while the overall error of the linear fits of life stages is also minimized. Since a piecewise linear regression approach is proposed for this purpose, each life stage will be represented by a linear fit of a piece identified through the least-squares method. Moreover, it is hypothesized that the best piecewise linear fit for a series is the one that has the minimum number of pieces and minimum overall error. The error is measured by Root Mean Squared Error (RMSE) depicted in Equation 5.2 where  $n$  is the length of a data series,  $\hat{y}_i$  is the value of a data point identified by the piecewise linear fit and  $y_i$  is the actual value of the data point. However, achieving such an optimal solution would not be possible as these two objectives could be contradictory. For instance, minimum RMSE for the piecewise fits could be achieved by having as many pieces as possible leading to a possible overfit. In the same way, the minimum number of pieces for a series could be achieved by sacrificing RMSE leading to a possible underfit. Since RMSE and the number of pieces of a piecewise linear fit involves a trade-off, the proposed algorithm continually controls this trade-off to obtain the best possible piecewise fit. Furthermore, heuristics are used in the algorithm to make the convergence faster while sacrificing the optimality of the final solution. Moreover, several thresholds are introduced to control the optimality of the final

piecewise fits. Algorithm 5.1 presents the proposed approach.

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}} \quad (5.2)$$

Algorithm 5.1 iteratively identifies the pieces for the piecewise linear fit. The process of identifying the first piece is described here in detail. The same process is repeated from the end of the first piece to identify the rest of the pieces. An incremental window based approach is conducted as the initial step in identifying the first piece. The minimum length of the piece is chosen to be 30 days, as it is assumed a life stage length should be at least 30 days. The algorithm finds the best linear fit for the first 30 days of the series and records the RMSE. The best linear fit is the one that minimizes the residual sum of squares. This is also known as least-squares approach [138]. The window length is then incrementally increased by 1 and the best linear fit for the data within the window is identified. RMSE of the fit is recorded. This process is continued until the window end reaches the end of the series. The starting point of the window remains constant at the beginning of the series to allow the window to incrementally increase. Figure 5.3 depicts how the window length incrementally increases by 1 data point. Then the recorded RMSE values are normalized as per Equation 5.6.

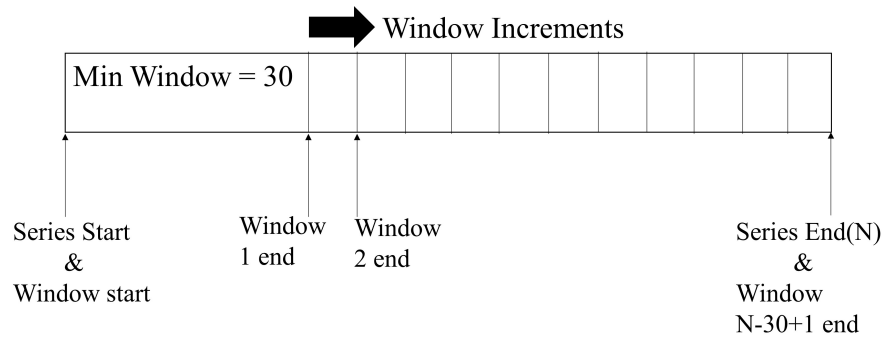


Figure 5.3: Incrementing Windows for series

---

**Algorithm 5.1** Piecewise Linear Trend Extraction : Method 1

---

**Inputs:** Xdata , Ydata ,minPieceSize, threshold  
seriesLen = length(Ydata)  
pieceStartI =1  
pieceEndI = -1  
remainingLen = seriesLen - pieceStartI+1  
**while** (remainingLen > minPieceSize) **do**  
    iterations = remainingLen-minPieceSize+1  
    **for** i = 1 : iterations **do**  
        find the best linear fit for the data subset (pieceStartI : pieceStartI +  
            minPieceSize-1 +i-1)  
        append error (RMSE) of the fit to *errorArr*  
    **end for**  
    normalize RMSE errorArr to 0-1 range  
    **for** i = 1 : length(errorArr) - 1 **do**  
        **if**  $\frac{errorArr[i+1]-errorArr[i]}{errorArr[i]} * 100 \leq errorThreshold$  **then**  
            Append errorArr[i+1] to *localOptimalErrArr*  
            Append i+1 to *localOptimalIndexArr*  
        **end if**  
    **end for**  
    **if** length(localOptimalErrArr) > 0 **then**  
        optimalErr = minimum value of localOptimalErrArr  
        optimalPieceEndI = index of the optimalErr value  
        **if** length(localOptimalErrArr) - optimalPieceEndI  $\geq 1$  **then**  
            **for** i = length(localOptimalErrArr) : -1 : optimalPieceEndI +1 **do**  
                tempOptimalErr = localOptimalErrArr(i)  
                errDiff = tempOptimalErr - optimalErr  
                 $lenGain = \frac{(localOptimalIndexArr(i)-1)}{(length(errArr)-1)} - \frac{(localOptimalIndexArr(optimalPieceEndI)-1)}{(length(errArr)-1)}$   
                **if** errDiff + (1 - lenGain) < threshold **then**  
                    optimalErr = tempOptimalErr  
                    optimalPieceEndI = i  
                    break  
                **end if**  
            **end for**  
        **end if**  
    **end if**  
    pieceEndI = pieceStartI+ minPieceSize-1 + localOptimalIn-  
    dexArr(optimalPieceEndI) -1  
    save pieceStartI and pieceEndI  
    pieceStartI = pieceEndI  
    remainingLen = seriesLen -pieceStartI +1 ;  
    pieceEndI =-1;  
**end while**  
**if** remainingLen  $\leq$  minPieceSize **then**  
    merge the remainingLen to the last piece  
**end if**

---

The first piece of the series could be any of the recorded windows. However, the first piece should be chosen so as to minimize the overall error of the piecewise fits and to minimize the number of pieces. Minimizing the number of pieces could be achieved by increasing the number of data points or the length of a piece. Considering the trade off between error and the number of pieces, several local optimal solutions are chosen as possible candidates for the first piece. In order to choose the local optimal solutions, the algorithm goes through the recorded RMSE values of each window. If the RMSE of  $window_{i+1}$  is less than or equal to RMSE of  $window_i$ ,  $window_{i+1}$  could be regarded as a local optimal solution. The reason is that  $window_{i+1}$  covers more length of the series than  $window_i$  while having a low error value. Thus, by choosing  $window_{i+1}$  as a local optimal solution the overall error of the series and the number of pieces of the series could be minimized.

However, there could be data series where the error for windows keeps on increasing as the window length increases. Such series would not have any local optimal solutions if the previous selection criterion is used. Hence, the criterion of local optimal solution selection is relaxed and it can be controlled by a threshold as given in Equation 5.3. A window is chosen as a local optimal piece when the error percentage in Equation 5.3 is less than or equal to the user-determined error threshold. This relaxed equation also includes the previous criterion of the error of  $window_{i+1}$  is less than or equal to the error of  $window_i$  when the error threshold is chosen to be zero. The local optimal solution space could grow as the error threshold is increased. Especially, for series where the error keeps on increasing as windows increases, the algorithm would be able to identify local optimal solutions where the rate of error increase is minimal based on the threshold. The *errorThreshold* value for the study is chosen to be 1% so as not to over increase the local optimal solution space which could in turn increase time complexity, and to not increase the number of pieces.

$$\frac{error_{i+1} - error_i}{error_i} * 100 \leq errorThreshold \quad (5.3)$$

Figure 5.4a depicts the RMSE values of the identified local optimal windows for

the first piece of *Monster Hunter :World* game and Figure 5.5a depicts the same for *SCP: Secret Laboratory* game. Once the local optimal solutions are identified, one solution out of these needs to be chosen as the optimal first piece. Initially, giving priority to minimizing RMSE, the window with the smallest error is chosen. This is represented by the black dot in the Figures 5.4a and 5.5a. However, there could be other local optimal solutions that could aid in minimizing the overall number of pieces of the fit while sacrificing the RMSE. For instance, in Figure 5.5a all the local optimal solutions positioned on the right hand side of the chosen solution represented by the black point could minimize the overall number of pieces as their window length is higher than the chosen solution. However, those solutions might increase the overall RMSE of the final fit. Hence, measures are taken to compare the local optimal solutions that provide length gain with the chosen solution in order to identify the optimal piece. For this purpose, the error difference and the length gain is calculated between the chosen solution and each local optimal window beginning from the rightmost. Also, a threshold as in Equation 5.4 is used to determine if there are any better optimal solutions than the chosen one based on the error and length difference.

$$error\ difference + (1 - length\ gain) < threshold \quad (0 < threshold < 2) \quad (5.4)$$

where

$$error\ difference = normalizedRMSE_{current} - normalizedRMSE_{chosen}$$

$$length\ gain = normalizedLength_{current} - normalizedLength_{chosen}$$

The RMSE values of each window and the length of each window is normalized to bring the values into the 0-1 range. This normalization step is vital for the threshold determination as it provides equal importance to the RMSE and length. Moreover, normalization helps in determining a threshold value that can be universally used for any data series rather than being unique to a single series. The length of data

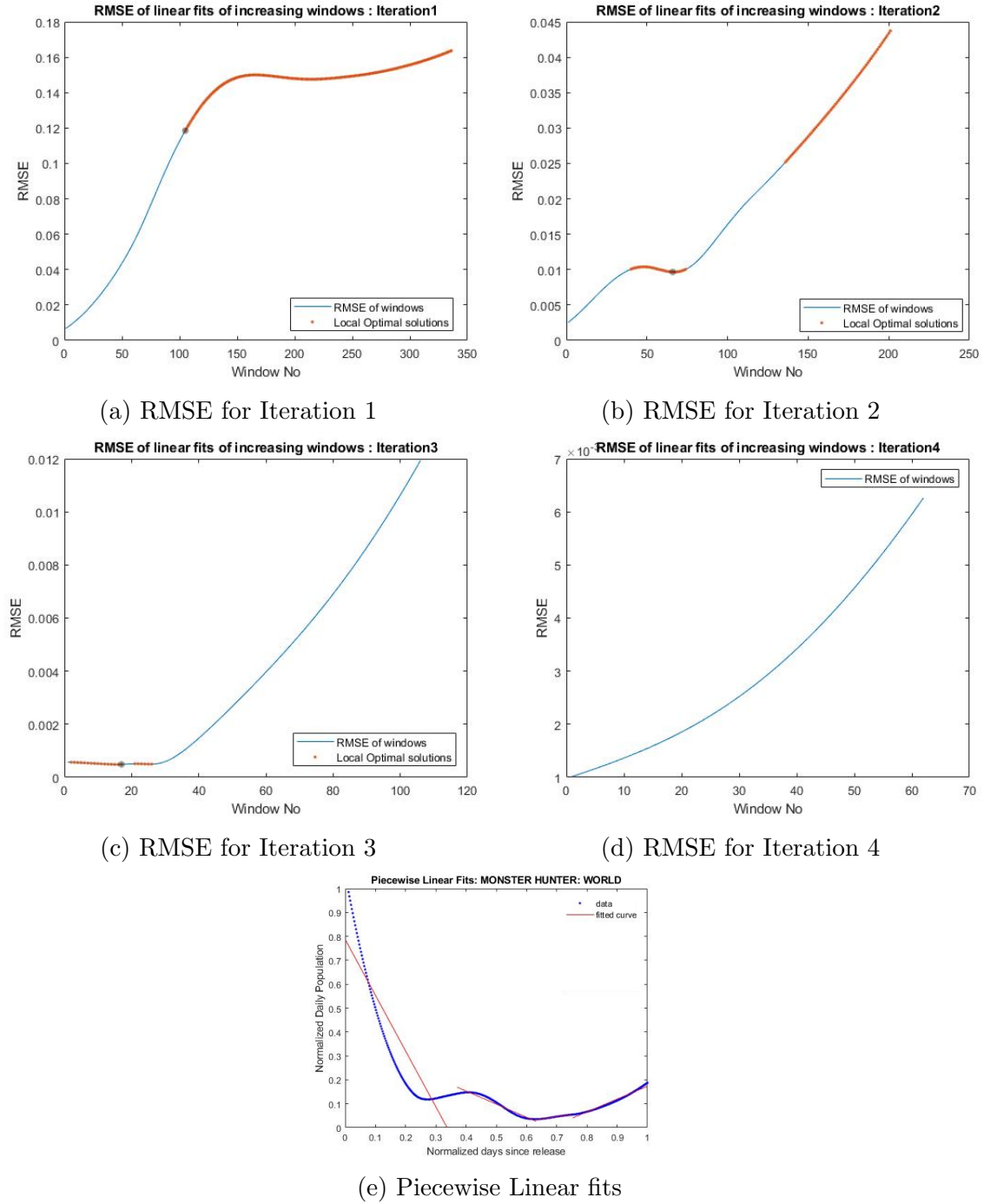


Figure 5.4: RMSE of iterations and final piecewise linear fit for *Monster Hunter:World* game

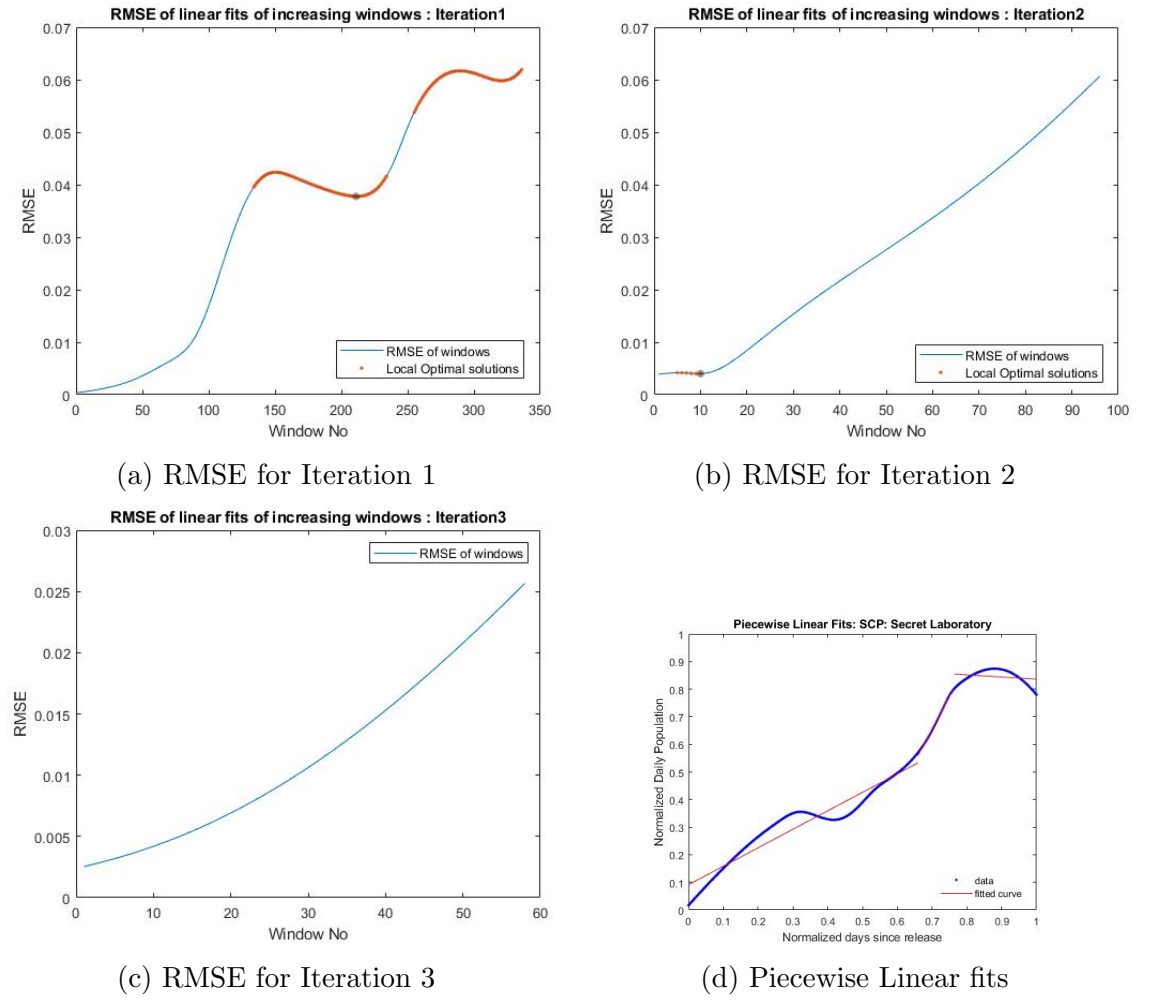


Figure 5.5: RMSE of iterations and final piecewise linear fit for *SCP: Secret Laboratory* game

points in each window is  $WindowNo + (30 - 1)$ . Thus, the window number is used in Equation 5.5 to normalize the length of windows. Also, since RMSE values are scale-dependent normalization was performed earlier as in Equation 5.6. It can be seen in Figure 5.5a that the locally optimal solutions positioned in the right hand side of the chosen solution have higher RMSE and higher length than the chosen solution. Ideally, if it is possible to identify a window that provides more length gain than the chosen solution, while having only a small increase in RMSE, it should be used as the optimal piece instead of the chosen solution. However, a numerical value is needed to represent the tolerable combination of RMSE and length. For this purpose, if there are any locally optimal solutions on the right hand side of the chosen solution, the error difference and inverse of length gain are calculated as per Equation 5.4 for those local optimal solutions starting from the rightmost one. Calculating the inverse of length gain aids in controlling RMSE and length together using a single threshold. If any locally optimal solution is encountered that agrees with the threshold, it would be chosen as the optimal solution. Going through the locally optimal solutions starting from the rightmost is important to quickly identify if there are better solutions, as length gain is highest towards the right. If there are better locally optimal solutions than the chosen one based on the threshold, one of them will be chosen as the optimal piece. If no better solution exists, the initially chosen optimal solution with the minimum RMSE will be kept as the optimal piece. Once the optimal first piece is identified, the whole process is repeated to identify the other pieces starting from the end of the first piece.

$$normalizedLength_i = \frac{windowNo_i - windowNo_{min}}{windowNo_{max} - windowNo_{min}} \quad (5.5)$$

Here  $windowNo_i$  represents the window number of the  $i^{th}$  window of the series.

$$normalizedRMSE_i = \frac{RMSE_i - RMSE_{min}}{RMSE_{max} - RMSE_{min}} \quad (5.6)$$

Here  $RMSE_i$  represents the RMSE of the  $i^{th}$  window of the series.