

### 5.1.3 寿命周期形状的顺时针线性回归前牵引

本节介绍了研究中使用的两种片状线性趋势提取方法，以提取每个游戏的使用寿命周期形状。

#### 5.1.3.1 方法1--当件数未知时的逐件线性回归

本节介绍了一种片状线性趋势提取算法，在不事先了解生命周期阶段数的情况下提取游戏的使用寿命周期形状。

拟议算法的主要目标是通过使用最少的生命阶段来确定所代表的生命周期形状，同时也使生命阶段的线性拟合的总体误差最小。由于为此目的提出了分片线性回归方法，每个生命阶段将由通过最小二乘法确定的一片线性拟合来表示。此外，假设一个系列的最佳片状线性拟合是具有最小的片数和最小的整体误差。误差由公式5.2中描述的均方根误差（RMSE）来衡量，其中 $n$ 是数据系列的长度， $\hat{y}_i$ 是由分片线性拟合确定的数据点的值， $y_i$ 是数据点的实际值。然而，实现这样的最优解是不可能的，因为这两个目标可能是相互矛盾的。例如，分片拟合的最小RMSE可以通过拥有尽可能多的分片来实现，这可能会导致过度拟合。同样地，一个系列的最小件数可以通过牺牲RMSE来实现，从而导致可能的欠拟合。由于RMSE和分片线性拟合的片数涉及到权衡，所提出的算法不断地控制这种权衡以获得最佳的分片拟合。此外，算法中还使用了启发式方法，以使收敛速度更快，同时牺牲最终解决方案的最优性。此外，还引入了几个阈值来控制最终解决方案的最优性。

片状拟合。算法5.1介绍了拟议的方法。

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

算法5.1迭代识别片状线性拟合的片段。这里详细描述了识别第一个片段的过程。同样的过程从第一个片段的末端开始重复，以识别其余的片段。在识别第一片的初始步骤中，采用了基于增量窗口的方法。碎片的最小长度被选择为30天，因为我们认为一个生命阶段的长度至少应该是30天。该算法为该系列的前30天找到最佳线性拟合，并记录RMSE。最好的线性拟合是使残余平方和最小化的那个。这也被称为最小二乘法[138]。然后将窗口长度递增1，确定窗口内数据的最佳线性拟合。拟合的RMSE被记录下来。这个过程一直持续到窗口末端达到系列的终点。窗口的起点在系列的开始时保持不变，以使窗口逐渐增加。图5.3描述了窗口长度是如何以1个数据点递增的。然后按照公式5.6将记录的RMSE值归一化。

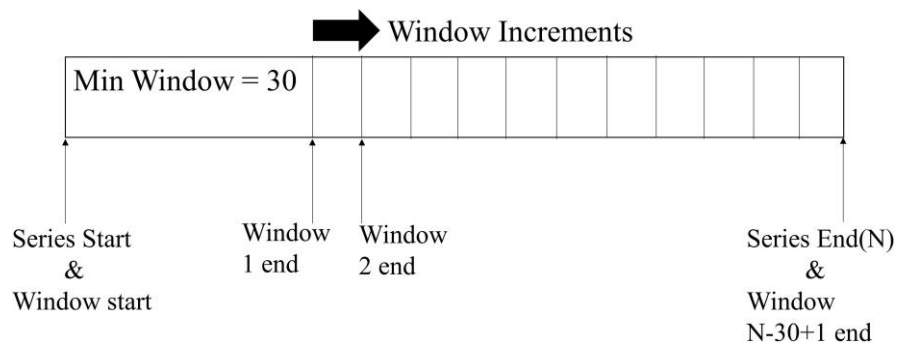


图5.3:递增系列的窗口

**算法5.1** 逐片线性趋势提取：方法1

输入。Xdata , Ydata , minPieceSize, threshold

seriesLen = length(Ydata)

件开始I =1

pieceEndI = -1

remainingLen = seriesLen - pieceStartI+1

**while** (remainingLen > minPieceSize) **do**

iterations = remainingLen-minPieceSize+1

**for** i = 1 : iterations **do**

找到数据子集的最佳线性拟合（pieceStartI : pieceStartI + minPieceSize-1 +i-1）。

将拟合的误差（RMSE）附加到errorArr中。

**结束**

将RMSE errorArr归一化为0-1范围

**for** i = 1 : length(errorArr) - 1 **do**

如果  $\frac{\text{errorArr}[i]-\text{errorArr}[1]}{\text{errorArr}[1]} * 100 \leq \text{errorThreshold}$  则  
将errorArr[i+1]追加到localOptimalErrArr中。

将i+1附加到localOptimalIndexArr中。

**end if**

**end for**

如果length(localOptimalErrArr)>0, 那么

optimalErr = localOptimalErrArr的最小值

optimalPieceEndI = optimalErr值的索引

如果length(localOptimalErrArr)- optimalPieceEndI ≥ 1, 那么

**for** i = length(localOptimalErrArr) :-1 : optimalPieceEndI +1 **do**

tempOptimalErr = localOptimalErrArr(i)

errDiff = tempOptimalErr - optimalErr

lenGain =  $\frac{(\text{localOptimalIndexArr}(i)-1) - (\text{localOptimalIndexArr}(\text{optimalPieceEndI})-1)}{(\text{length}(\text{errArr})-1)}$

如果 errDiff + (1 - lenGain) < 阈值, 那么

optimalErr = tempOptimalErr

optimalPieceEndI = i

突破

**end if**

**end for**

**end if**

**end if**

片尾曲 = 件始点+ 最小件数-1 + localOptimalIndexArr(optimalPieceEndI) -1

保存 pieceStartI 和 pieceEndI

pieceStartI = pieceEndI

remainingLen = seriesLen -pieceStartI +1 ;

pieceEndI =-1;

**结束时**

如果剩余长度 ≤ minPieceSize, 那么

将剩余的Len合并到最后一块

---

结束 如果

系列的第一片可以是任何一个记录的窗口。然而，应该选择第一片，以使分片拟合的总体误差最小，并使片数最小。尽量减少片断的数量可以通过增加数据点的数量或片断的长度来实现。考虑到误差和件数之间的权衡，选择几个局部最优解作为第一个件的可能候选。为了选择局部最优解，该算法通过每个窗口的RMSE值记录。

如果窗口 $i+1$ 的RMSE小于或等于窗口 $i$ 的RMSE，窗口 $i+1$ 可以被认为是一个局部最优解。原因是窗口 $i+1$ 比窗口 $i$ 覆盖了更多的系列长度，同时具有较低的误差值。因此，通过选择窗口 $i+1$ ，作为局部最优解，系列的整体误差和系列的件数可以最小化。

然而，可能有一些数据系列，随着窗口长度的增加，窗口的误差不断增加。如果使用之前的选择标准，这样的系列不会有任何局部最优解。因此，局部最优解的选择标准被放宽了，它可以由公式5.3中给出的阈值控制。当公式5.3中的误差百分比小于或等于用户确定的误差阈值时，一个窗口被选为局部最优解。这

放宽的方程还包括之前的窗口误差标准 $i+1$ 。

当误差阈值被选为零时，小于或等于窗口 $i$ 的误差。局部最优解空间可以随着误差阈值的增加而增长。特别是，对于误差随着窗口的增加而不断增加的系列，该算法将能够识别基于阈值的误差增加率最小的局部最优解决方案。本研究的误差阈值被选为1%，以避免过度增加局部最优解空间，这反过来又会增加时间的复杂性，同时也不会增加件数。

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

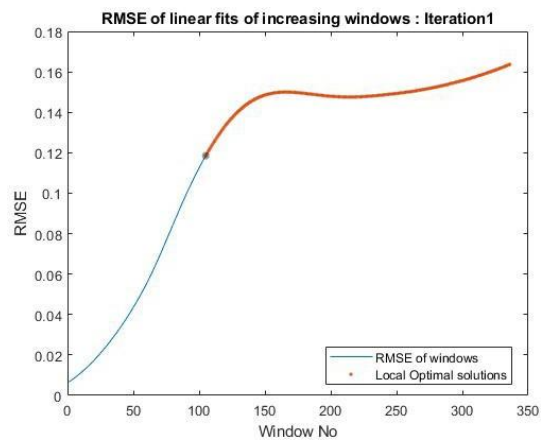
图5.4a描述了已确定的局部最佳窗口的RMSE值，包括

图5.5a描述了《怪物猎人:世界》游戏的第一件作品，图5.5a描述了《SCP：秘密实验室》游戏的情况。一旦确定了局部最优解，就需要从这些解中选择一个作为最优的第一片。最初，优先考虑最小化RMSE，选择误差最小的窗口。这在图5.4a和5.5a中用黑点表示。然而，可能还有其他的局部最优方案，可以在牺牲RMSE的情况下帮助最小化拟合的总件数。例如，在图5.5a中，所有位于黑点所代表的所选方案右侧的局部最优方案都可以使总体件数最小化，因为它们的窗口长度比所选方案要高。然而，这些解决方案可能会增加最终拟合的整体RMSE。因此，采取的措施是将提供长度增益的局部最优方案与所选方案进行比较，以确定最优片。为此，从最右边开始计算所选方案与每个局部最优窗口之间的误差差和长度增益。同时，根据误差和长度差，使用公式5.4中的阈值来确定是否有比所选方案更好的最优方案。

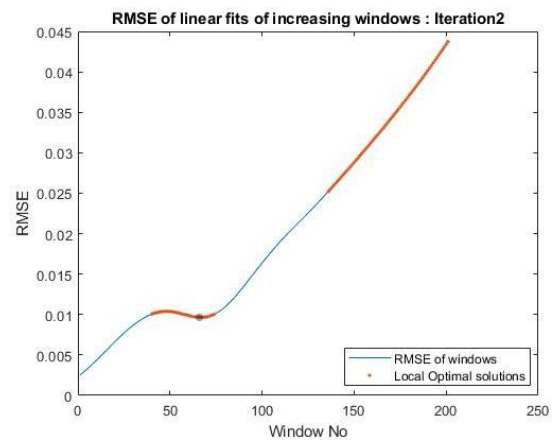
$$\text{误差差} + (1 - \text{长度增益}) < \text{阈值} \quad (0 < \text{阈值} < 2) \quad (5.4) \quad \text{其中}$$

$$\begin{aligned} \text{误差差} &= \text{归一化RMSE}_{\text{current}} - \text{归一化RMSE}_{\text{chosen}} \\ \text{长度增益} &= \text{归一化Length}_{\text{current}} - \text{归一化Length}_{\text{chosen}} \end{aligned}$$

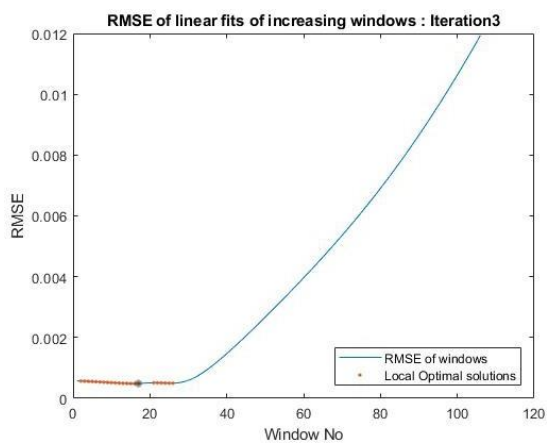
每个窗口的RMSE值和每个窗口的长度都被归一化，以使这些值在0-1的范围内。这个归一化步骤对阈值的确定至关重要，因为它为RMSE和长度提供了同等重要性。此外，归一化有助于确定一个可以普遍用于任何数据系列的阈值，而不是单一系列的唯一阈值。数据的长度



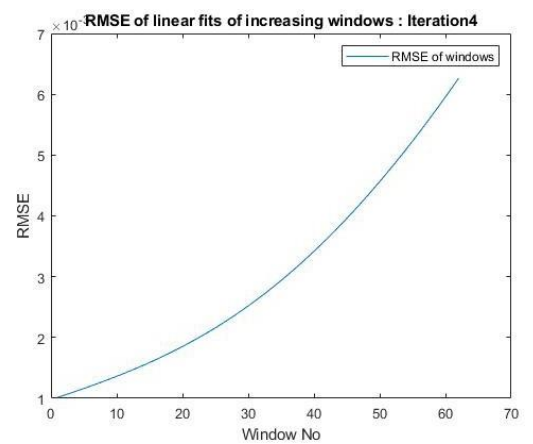
(a) 迭代1的RMSE



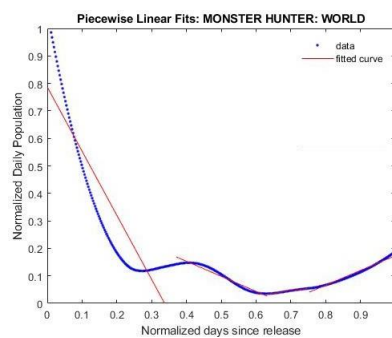
(b) 迭代2的RMSE



(c) 迭代3的RMSE

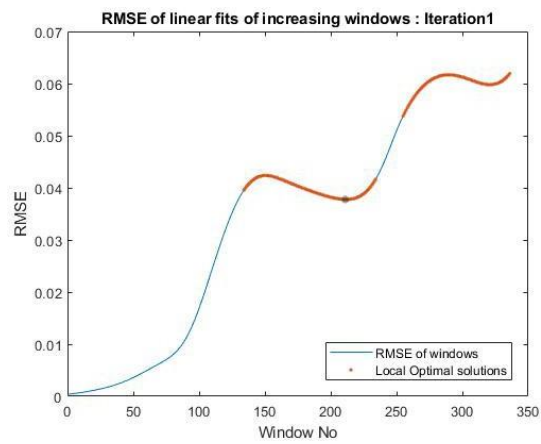


(d) 迭代4的RMSE

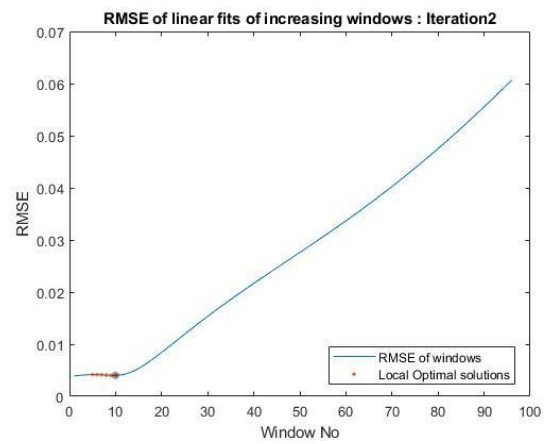


(e) 平行线性拟合

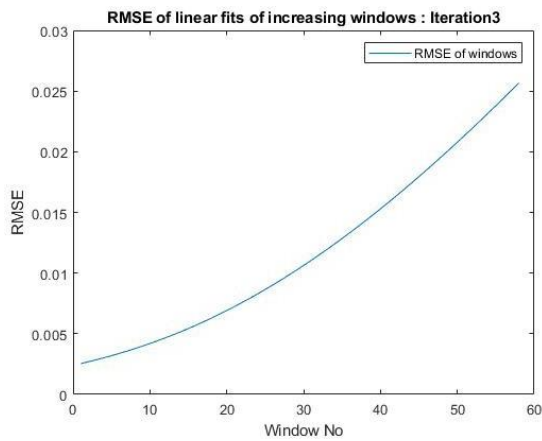
图5.4: 《怪物猎人：世界》游戏的迭代和最终的片状线性拟合的RMSE



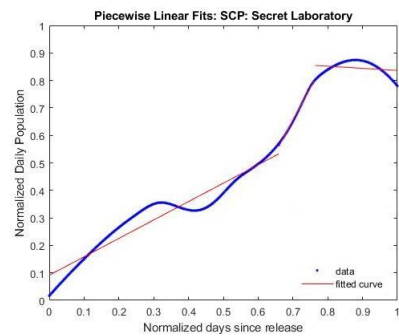
(a) 迭代1的RMSE



(b) 迭代2的RMSE



(c) 迭代3的RMSE



(d) 分片线性拟合

图5.5: 迭代的RMSE和SCP：秘密实验室游戏的最终片状线性拟合



每个窗口中的点是  $WindowNo + (30-1)$ 。因此，窗口号被用于在公式5.5中，对窗口的长度进行归一化。另外，由于RMSE值是与标度有关的，所以如公式5.6所示，提前进行了归一化处理。从图5.5a中可以看出，定位在所选方案右侧的局部最优方案比所选方案具有更高的RMSE和更长的长度。理想情况下，如果有可能找出一个窗口，提供比所选方案更多的长度增益，同时RMSE只有小幅增加，就应该把它作为最佳片，而不是所选方案。然而，需要一个数值来代表RMSE和长度的可容忍的组合。为此，如果在所选方案的右侧有任何局部最优方案，则按照公式5.4计算这些局部最优方案的误差差和长度增益的倒数，从最右侧的方案开始。计算长度增益的逆值有助于用一个阈值来控制RMSE和长度。如果遇到任何与阈值一致的局部最优解，它将被选为最优解。从最右边开始浏览局部最优解，对于快速识别是否有更好的解决方案非常重要，因为长度增益在右边最高。如果有比基于阈值的所选方案更好的局部最优方案，其中一个将被选为最优件。如果没有更好的解决方案，最初选择的具有最小RMSE的最优解决方案将被保留为最优件。一旦确定了第一个最优解，整个过程将重复进行，以确定从第一个最优解的末端开始的其他解。

$$\text{归一化长度}_i = \frac{\text{窗口号码}_i - \text{窗口号码}_{\min}}{\text{窗口号码}_{\max} - \text{窗口号码}_{\text{阈}}} \quad (5.5)$$

这里  $windowNo_i$  代表系列的  $i^{th}$  窗口的窗口号。

$$\text{归一化RMSE}_i = \frac{RMSE_i - RMSE_{\min}}{RMSE_{\max} - RMSE_{\text{阈}}} \quad (5.6)$$

这里  $RMSE_i$  代表系列的  $i^{th}$  窗口的RMSE。