CHAPTER VII

CONCLUSIONS AND FUTURE WORK

The entire work has addressed three problems in interval data mining. The conclusions that are drawn from the work done on these problems are presented below. Future lines of research have also been suggested.

In the context of the first problem addressed in the present work, viz. the task of mining maximal k-frequent intervals, it was shown that the introduction of a total order among the input intervals can lead to a substantial reduction of the computational time of an earlier method [Lin03] proposed by Lin for mining maximal k-frequent intervals. Experimental results showed that the modified Lin's method proposed in the present work, consistently outperforms the original Lin's method [Lin03] by a factor of at least 4. In most of the cases, the improvement is at least by a factor of 7. However, both Lin's method [Lin03] and the modified Lin's method use a data-structure called I-Tree to identify the maximal k-frequent intervals and this results in a $\theta(n^2)$ worst-case time complexity for both the methods. In the present work, it was shown that by abandoning the use of the I-Tree data-structure and viewing the input intervals as a sorted sequence of endpoints, an asymptotically faster method for mining maximal k-frequent intervals can be developed. To this end, the O(n log n) MIntMiner algorithm, which was proposed for mining maximal k-frequent intervals, consistently outperforms even the modified Lin's method by a factor of at least 5. The correctness of the MIntMiner algorithm can be inferred from the detailed proof that has been provided. Similarly, the worst-case behavior

of the algorithm can be understood from a detailed discussion on the same. Experimental results obtained after implementing and testing all the fore-mentioned methods/algorithms – viz. Lin's method [Lin03], the modified Lin's method and MIntMiner – revealed variations in the performances of the algorithms with respect to different input parameters. These observations have been noted in detail in Section 4.8.1. Further, in the present work, it has been shown that the set of maximal k-frequent intervals can be efficiently used to determine whether any given non-empty interval is k-frequent. The correctness of the O(log n) Chk_Freq algorithm developed for this task can be inferred from the corresponding proof that has been provided. The theoretical expectations of the algorithm's time-efficiency has been confirmed by experimental results.

In the context of the second problem addressed in the present work, viz. the task of mining closed k-frequent intervals and their respective absolute support values, the two proposed algorithms, viz. Basic_CMiner and CMiner, employ the same strategy to identify the set of closed k-frequent intervals and their respective absolute support values. However, the use of the IS-Tree data-structure in the $O(n^2 \log n)$ CMiner algorithm lead to an improvement in the worst-case time-complexity of the CMiner algorithm over the $O(n^4)$ Basic_CMiner algorithm. Experimental results showed that CMiner consistently outperforms Basic_CMiner by a factor of at least 45. In most of the cases, the improvement is at least by a factor of 100. Experimental results also revealed variations in the performances of the two algorithms with respect to different input parameters. These observations have been noted in detail in Section 5.6.1. The correctness and time-efficiency of the two proposed algorithms can be inferred from the respective proofs and time-complexity discussions that have been provided. Further, in the present work, it

has been shown that the set of closed intervals (viz. closed 1-frequent intervals) and their respective absolute support values can be efficiently used to determine the absolute support value of any given non-empty interval. The correctness of the O(log n) CloSup algorithm developed for this task can be inferred from the detailed proof that has been provided. Experimental results validated the theoretical expectations of the algorithm's time-efficiency.

The problem of detection of periodicity of a pattern/event which occurs in certain time-intervals was earlier studied in [MMB08], where a notion of *superimposition* of the time-intervals was used. In the present work, it has been shown that the same problem can be tackled in a substantially more efficient manner by processing the endpoints of the stripped time-intervals. The proposed algorithms viz. CapChange, FindCert and LocMax work because of a result proved in the chapter, which shows that a change in the certainty (which indicates the periodicity) of such a pattern/event can occur only at an endpoint of a stripped time-interval. The efficiency of the proposed algorithms can be inferred from the time-complexity discussions that have been provided. The experimental results presented in the chapter show the usefulness of the proposed algorithms on real-life data.

It is considered worthwhile that further work on the lines mentioned below may be pursued for additional advancement –

(i) The variations in the performances of Lin's method [Lin03], the modified Lin's method, MIntMiner, Basic_CMiner and CMiner with respect to different input parameters, are not comprehensively explained by the respective time-complexity discussions that

have been provided in the present work. In future, more theoretical work can be done for this purpose.

(ii) In the CMiner algorithm, when a closed k-frequent interval is detected, it is immediately inserted in an IS-Tree data-structure. The same closed k-frequent interval may be detected many times by the algorithm. However, it need not be re-inserted in the IS-Tree each time it is detected. If an algorithm can be developed which detects a closed k-frequent interval only once, then it could potentially lead to a substantial improvement over CMiner. In future, attempts can be made to develop such an algorithm.

(iii) In the present work, a method has been proposed to determine the periodicity of a pattern/event which occurs in certain time-intervals with respect to a period given a priori. In future, algorithms can be developed to determine periodicities of such a pattern/event with respect to periods that are not given beforehand. These algorithms could be designed to find the most appropriate period for the given data.