Algorithmic tools for data-oriented law enforcement

Cocx, T.K.

Chapter 6

Detection of Common and Defining Subcareers

The search for common sequences in a database of timed events is a relatively recent addition to the data mining research area. It usually aims to provide insights into customer behavior, based upon the sequential acquisition of certain products, but is also applicable to criminal careers. A method for the automatic categorization of criminal careers has been established in Chapter 5 and using this classification, a method was devised that can find subcareers that can be said to be specific for one class, based upon its occurrence rate in this class and the other classes. It builds upon a well-known approach to search for common subsequences in databases. In this chapter we propose a number of key-changes to this approach that make it applicable to the search of common criminal careers. We propose to expand this method with the search of defining subcareers and show some experiments that investigate the possibilities of sequence mining in this area of law enforcement.

6.1 Introduction

Now that a method has been established for the classification of criminal careers and a solid basis has been created upon which prediction can be performed, it might also be of interest to discover if there are any chains of criminal activity that occur often. Such a search for common subcareers might both be of interest to the field of crime analysis at criminology institutes or police headquarters, to a police officer, who can use the knowledge of commonly occurring patterns in crime to his advantage in the field and to social workers at judicial institutions, dealing with starting offenders, who can employ knowledge of commonly occurring patterns in their treatment of delinquents.

Besides the mere discovery of these possibly occurring common subcareers, it is of importance to the causes mentioned above to establish a relationship between some of these common subcareers and their respective existence in different criminal career

classes as described in Chapter 4 and Chapter 5. If such distinguishing or defining subcareers can be found with a certain threshold, a reliability can be established that couples the occurrence of these subcareers to a certain class, enabling law enforcement personnel to quickly recognize and impede these behavioral patterns.

For this purpose we are given the criminal record database described in Appendix B, containing a list of crimes per offender organized per time frame (year or month). Obviously the number of occurrences of each single crime type within one time frame is important in this matter, especially when the time frames are rather large.

We introduce the problem of mining sequential patterns over this data. An example of such a pattern is that offenders typically first come in contact with law enforcement for possession of drugs, then minor theft, then getting involved in drug dealing felonies. Naturally, elements of such a sequence need not be single crimes, they can be sets of crimes or even sets of the same crime. Furthermore, the patterns should be consecutive, meaning that for an offender to satisfy the inclusion demands for a pattern, each of the included crimes must be committed right after one another or within the same time frame. Compliance with this demand guarantees that a sequence "Bicycle theft"→"Drug abuse" is treated differently than "Bicycle theft"→"Murder"→"Drug abuse", which is correct in the area of crime analysis, since both criminal careers are treated very differently.

Since sequential pattern mining has been researched for a while, a background to our work is provided in Section 6.2. The main contribution of this chapter is in Section 6.3 where the alterations to standard approaches are discussed that lead to successful subcareer mining. Section 6.4 shows some of the results reached by our approach when applied on the criminal record database.

6.2 Background

A lot of work has already been done in the area of sequential pattern mining, commonly motivated by decision support problems from the retail world. The authorative work on this subject is [2], where sequential consumer behavior was first automatically analyzed. This paper dealt with the analysis of sequentially bought *baskets*, sets of items bought in a single batch, and was a continuation of the work in [1], where the focus was more on intra-basket patterns, and the discovery of what items are commonly often bought together. The work contained a mixture of intelligent systems that predicted common continuation of a series [9, 36], instead of finding all common patterns, of systems that attempt to find text subsequences based upon regular expressions [3, 25, 34, 32] and on the discovery of similarities in a database of genetic sequences [33].

Although this work is archetypical for the field, its algorithms are still widely applicable and are very well suited as a basis for the problem under observation. There are however a number of critical issues that need to be dealt with:

1. The problem in [2] deals with itemsets, where each item is either present or not, while in the problem at hand, each crime can be present multiple times, therefore dealing with the multiset paradigm.

2. In the original problem, the items in the target pattern were not required to appear consecutively in a sequence. In contrast however, as stated above, within criminal career analysis, the items *must* be consecutive to fulfill the requirement. Within our approach we ignore possible empty time frames (gaps).

3. The "boundaries" between time frames have no implicit meaning for subcareer detection other than the fact that they separate groups of crimes of which the ordering is not known, which arises from the fact that crimes are only added to the database periodically, losing their original ordering within these periods or time frames. Consequently, the discovery of subcareers takes place on sequences of single crimes, where parts of these sequences can be ordered in all possible ways. This sharply contrasts with the original method where boundaries between the different itemsets were not to be broken during the pattern matching process.

These issues will be addressed in Section 6.3.

Usually a multiset of crimes is denoted by $\{i_1, i_2, \ldots, i_m\}$ where i_k is a crime ($1 \leq k \leq m$) such that elements can occur more than once; we will denote this multiset by ($i_1\ i_2\ \ldots\ i_m$). A criminal career is then denoted by $\langle s_1\ s_2\ \ldots\ s_n\rangle$, where s_k is a multiset of crimes representing a single time frame ($1 \leq k \leq n$). According to [2] a sequence $\langle a_1 a_2 \ldots a_n\rangle$ is *contained in* another sequence $\langle b_1 b_2 \ldots b_m\rangle$ if integers $1 \leq i_1 < i_2 < \ldots < i_n \leq m$ exist such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_n \subseteq b_{i_n}$. This means, for example, that $\langle$(1) (2 3) (4)$\rangle$ is contained in $\langle$(0) (1 11 12) (2 3 10 99) (55) (4)$\rangle$, but that $\langle$(1) (2)$\rangle$ is not contained in $\langle$(1 2)$\rangle$. Clearly, this last effect conflicts with our third requirement, mentioned above, hence we have to resort to another construct within our approach. A sequence is said to be *maximal* in a certain set of sequences, if it is not contained in another sequence from that set.

Using the definition above, a sequence from the database *supports* a sequence if it contains that sequence. The *support* of a certain (sub)sequence can then be defined as the fraction of total database sequences that support this sequence. Mining for common subsequences can thus be seen as the search for maximal subsequences that have a certain threshold support as defined by the analyst. These subsequences are called *frequent*.

As an example, we examine the data in Table 6.1, where a crime type is denoted as a number. If these were the only active periods of criminal activity of these offenders, their criminal careers would be described by Table 6.2, where the middle column contains the notation as laid out in [2]. In the third column we introduce a notation that is more suited to our approach, where the boundaries between time frames have been discarded and where a multiset has been overlined, meaning that its elements can be randomly ordered. Note that for clarity singletons are not overlined. The new notation is equivalent with the original notation of customer transactions Henceforward, we will use the new notation to *denote* that these series need to comply with the three requirements specified above.

Assuming a support threshold of 25%, meaning a minimum support of two offenders, the sequences $\langle$(1) (8)$\rangle$, $\langle$(2) (4)$\rangle$ and $\langle$(5) (6 7)$\rangle$ are the maximal frequent subsequences using the middle column. Note that $\langle$(5) (6)$\rangle$ and $\langle$(5) (7)$\rangle$ also satisfy the threshold, but since they are not maximal they are not considered as end-result.

If we consider the situation denoted in the third column, 18 is no longer frequent,

Table 6.1: Database of criminal activity sorted by Offender and Time frame

Offender	*Time frame*	*Crimes*
1	September	1 2
1	October	4
1	December	8
2	October	2 6 9
3	October	1 4 5
3	November	6 7 8
3	December	1 1
4	November	5
4	December	6 7
5	September	2 3
5	November	3 4

Table 6.2: Criminal career version of Table 6.1

Offender	*Career (cf.* [2]*)*	*Career (our approach)*
1	⟨(1 2) (4) (8)⟩	$\overline{12}48$
2	⟨(2 6 9)⟩	$\overline{269}$
3	⟨(1 4 5) (6 7 8) (1 1)⟩	$\overline{145}\,\overline{678}\,\overline{11}$
4	⟨(5) (6 7)⟩	$5\overline{67}$
5	⟨(2 3) (3 4)⟩	$\overline{23}\,\overline{34}$

since crimes 1 and 8 are separated by crime 4 in the career of offender 1, however, 24 is now frequent representing both ⟨(2) (4)⟩ and the "stronger" ⟨(2 4)⟩. Finally, using the same argument, ⟨(5) (6 7)⟩ or $5\overline{67}$, now also contains 567. However, the overlining of $\overline{67}$ can not be removed, since this would effectively omit the containment of 576, per the consecutiveness demand. Evidently, there are quite a few significant differences between the original, retail related, approach and the search for common subcareers.

In the new approach, containment is now defined as follows: a sequence $a = a_1 a_2 \ldots a_n$ is contained in sequence $b = b_1 b_2 \ldots b_m$ if there is a subsequence $b_{i+1} b_{i+2} \ldots b_{i+n}$ with $a_1 \subseteq b_{i+1}$, $a_n \subseteq b_{i+n}$ and $a_j = b_{i+j}$ for $1 < j < n$, where $i \geq 0$, $i + n \leq m$ and a_k and b_k denote time frames. Note that the ordering in a time frame is undefined, e.g., $\overline{1123} = \overline{3121}$, since it is a multiset.

If we consider the problem of strict borders between time frames, as described in Chapter 4 and Chapter 5, where criminal activity at the end of one time frame and at the beginning of the next time frame are strictly separated even though the time between these crimes is very short, another definition of containment might be better. When investigating a subsequence in a certain time frame, we could instead consider its appearance in the merger of this time frame and both its temporal neighbors, assuming that the borders between those time frames are somewhat fuzzy. This would result in the following definition of containment: a sequence $a = a_1 a_2 \dots a_n$ is contained in sequence $b = b_1 b_2 \dots b_m$ if there is a subsequence $b_{i+1} b_{i+2} \dots b_{i+n}$ with $a_j \subseteq b_{i+j-1} \cup b_{i+j} \cup b_{i+j+1}$ $(j = 1, 2, \dots, n)$ and $b_{i+j} \subseteq a_{j-1} \cup a_j \cup a_{j+1}$, $(j = 2, 3, \dots, n-1)$, $i \geq 0$, $i + n \leq m$ and a_k and b_k are time frames and $b_{-1} = b_{m+1} = \emptyset$.

There are however two related problems when dealing with the fuzzy border situation:

1. **Empty set problem** In contrast with the first definition of containment, the existence of empty sets, which in this case represent empty time frames (a time frame where a certain individual commits no crimes), poses a problem in this case. In the first situation no consecutiveness is violated when omitting periods without activity from a career. However, if we consider borders between periods to be fuzzy, and the empty time frames are omitted from a career, we can invalidly have fuzzy borders between two time frames that were no neighbors in the original sequence. For example if we consider the career $1x2$, where x is a period without activity, there is clearly a strict separation between activity 1 and 2 (at least a time frame). However, if we denote this career as 12 and consider the border between the two to be fuzzy, a sequence 21 would invalidly be contained.

2. **Overlapping neighbors problem** A related problem is the case where an overlap between the predecessor and successor time frames of the time frame under consideration occurs, even though they are strictly separated by that time frame. For example, in the career $\overline{12}\overline{345}$, the subcareer 52 would invalidly be contained. Even though the intention is to let $\overline{123}$ and $\overline{345}$ be considered, the combination thereof $(\overline{12345})$ should not.

These issues can be dealt with at the cost of a much more complicated containment relation, therefore, we will choose the first definition of containment within the scope of this paper.

The standard approach for the detection of frequent subsequences consists of five steps:

1. **Sort Phase.** The database is sorted using offender as major key and time frame as minor key. The database is now implicitly a database of criminal careers.

2. **Large Itemset Phase.** The set of single itemsets that are frequent (large itemsets) is retrieved. Standard frequent itemset methods are suitable for this approach, except for the fact that they use a slightly different notion of support. In these algorithms, the support of an itemset is defined as the number of transactions (or in our case

time frames) a certain itemset appears in, while in this case support should be seen of the appearances of an itemset in *any* of the transactions of a single individual. This deficit can be overcome easily if we only count the occurrence of an itemset once for each individual, even if it appears more.

3. **Transformation Phase.** The database will be transformed in such a way that the sequences or careers are replaced by sequences of frequent itemsets, discovered in the previous phase. Individuals without any frequent itemsets in their sequence are discarded, however they still contribute to the total amount of individuals. This phase is necessary to reduce the computation time when searching for frequent sequences.

4. **Sequence Phase.** The actual frequent sequences are found in this phase. Just like the search of the sequence building blocks, the large itemsets, the search for large sequences is based upon the Apriori-property that a certain sequence can only be frequent if all its subsequences are also frequent. Therefore, this phase is a multi-pass algorithm that generates candidate large sequences based upon sequences that are already large. These candidates are then compared to the transformed database from Step 3 and only the large sequences are kept. The cycle then starts over, with the generation of candidate large sequences.

5. **Maximal Phase.** All sequences that are frequent but not maximal are discarded. This can be accomplished easily by deleting all sequences that are contained in another sequence from the set discovered in Step 4. All sequences of length 1 are also left out.

This process is visualized in Figure 6.1.

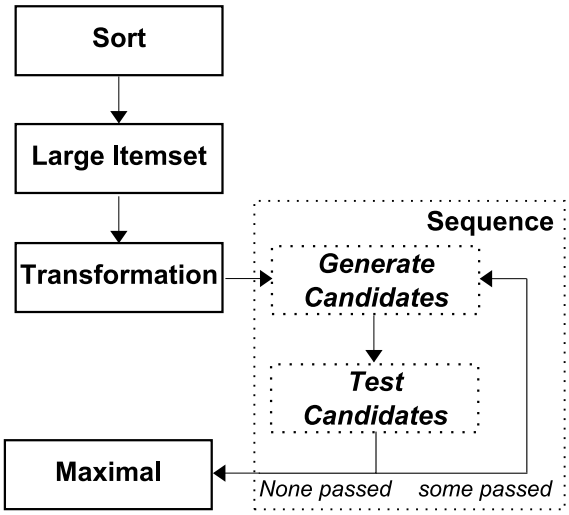

Figure 6.1: A common approach for frequent sequence mining

Within this approach, candidate selection from a large sequence of size k is done as follows:

1. **Join** If two sequences only differ in the last itemset, add that last itemset to the other set.
2. **Prune** All subsets of size k of all candidate itemsets of size $k+1$ should be present in the large sequence set of size k. All candidates failing on this requirement are pruned.

An example can be seen in Table 6.3

Table 6.3: An example of the candidate selection process

Large Sequences	*Candidates (After Join)*	*Candidates (After Prune)*
⟨5 6⟩	⟨5 6 7⟩	⟨6 8 9⟩
⟨5 7⟩	⟨5 7 6⟩	
⟨6 8⟩	⟨6 8 9⟩	
⟨6 9⟩	⟨6 9 8⟩	
⟨8 9⟩		

It is clear that a number of steps within this approach are different for the criminal career situation when looking at the different requirements we set forth. The notion of when an itemset is large differs in both situations, occurring either completely in a single transaction as in [2] or occurring in overlapping time frames, per requirement 3. Also, per requirement 3, since the time frame boundaries have no implicit meaning, the notion of time frames is now completely lost, as can be seen in Figure 6.2.

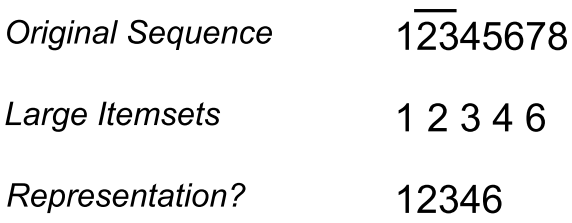

Figure 6.2: A bad representation of a criminal career in phase 3

Through this loss, Figure 6.2 clearly shows that we have unjustly lost sequences 13 and 24 and gained sequence 46, per requirement 2. Therefore, care must be taken in Step 3 to choose a representation that denotes all possible sequences consisting of the frequent itemsets in Phase 2. Depending on the choices made for the transformation phase, we can either keep or change the sequence phase. The options we chose for our

approach are discussed in Section 6.3. The first and fifth phase can be kept, regardless of the requirements or choices made for the other steps.

6.3 Approach

As the largest part of the approach described above can still be employed in the search for common subcareers, only a few changes need to be made to the different steps of the approach. They are described below. The main goal of our changes is to retain a straightforward, ad-hoc, human understandable and consecutive approach to the problem, that works reasonably fast, without a very complicated algorithmic foundation.

6.3.1 Changes to the Large Itemset Phase

In the original approach, Large *itemsets* were retrieved in this phase, however, since consecutiveness is required for our cause, the search for crimesets that contain more than one crime could already be seen as a search for subcareers. For example, if the crimeset 12 is Large, this implies that crime 2 immediately follows crime 1. Since the search for sequentiality only starts in phase 4, we should limit our approach to the search for singular items in this case.

An exception to this rule is the situation that both 12 and 21 are Large, indicating that the crimeset $\overline{12}$ is Large as well. However, the inclusion of this set provides no additional gain over the inclusion of separate crimesets 1 and 2; since the inclusion of $\overline{12}$ implies that all its subsets are also frequent per the Apriori-property, we cannot delete extra individuals in the next phase based upon the discovery of a frequent $\overline{12}$. We can therefore safely limit our search for Large itemsets to the search for frequent crimes in the database. Another advantage to this situation is that there is no need to do any Apriori search within the database since a simple count of each crime will yield the required results, greatly reducing the computational complexity of our approach.

Another difficulty that is overcome by the search for single Large crimes alone, is the confusion about the meaning of the statement "$\overline{12}$ is Large". This could either indicate that the summation of occurrences of 12 and 21 is above the threshold, or that both of their occurrences are above the threshold, the first representing a more loose statement and the latter describing a more strict situation. We therefore restrict our search to the retrieval of singular crimes in this phase or simple consecutive subcareers (without overlining) in the next phase.

6.3.2 Changes to the Transformation Phase

Since the crimes within a single time frame can be put in any order, we need to make sure the representation we choose within the transformation phase must retain the ability of searching for all of these possible subcareers. This class of problems is usually known as *trace monoids* [22, 14]. The most natural way of transforming a certain criminal career into a good representation would probably be a directed acyclic graph, that denotes every possible way a certain career can be traversed. Note that, since a time frame is a multiset

and crimes can therefore appear more than once in such a time frame, the subgraph that represents this time frame can not be deterministic.

Also, if two crimes appear next to one another after the discarding of another crime (that was not frequent in Step 2), the edge between those two crimes should be tagged to indicate that this edge can not be traversed within the matching process of a single subcareer.

The process of transforming a career is now as follows:

1. **Discard infrequent crimes** Single crimes that do need have a support above the threshold can not be part of any frequent subcareer (Apriori-property). They should therefore be discarded from the graph. A symbol ($) is inserted if this would result in the consecutive paring of crimes that were not consecutive in the original career.

2. **Transform to graph** The career in the form of a string of crime numbers is now transformed into a directed, acyclic graph. The symbol $ is transformed into an "untraversable" edge within a single subcareer. Each time frame with multiple elements is transformed in such a way that a path exists in the graph for every possible subcareer in this time frame. If this time frame is of size n, the number of possible paths in the resulting subgraph for this time frame will be $n!$, possibly greatly increasing the computation time. However, the severity of this effect largely depends of the size of these time frames in the actual database, which is investigated in Section 6.4. The directed graph for a time frame can be seen as a tree, since it branches out. Especially when such a tree is very large, it is possible to reduce memory load by specifying multiply occurring subtrees only once and connecting to them from all relevant points in the tree.

3. **Add escape edges** An unfortunate effect that occurs within this approach is the following. Suppose we consider the criminal career $\overline{123456789}$ and would try to match the subcareer 789. The consecutiveness demand requires that 7, 8 and 9 are consecutive so matching should be done at the leaves of the tree, effectively requiring the tree to be "searched" for leaves with the subcareer 78, even though the same subcareer is also present close to the root. As a solution we propose to add escape edges that connect every none leaf directly to the first node after the tree, redirecting the traversion to the so-called "main-path" as quickly as possible. this way a tree search can be prevented, linking the first occurrence of 78 from the example directly to the 9 of the next time frame. This process is illustrated in Figure 6.3, where only the escape edges for the first level have been added as dotted lines. Of course, the second level has escape edges to node y as well. The addition of escape edges obviously increases memory load. The effects of this addition are investigated in Section 6.4. Note that an escape edge can only be traversed when the matching of a certain subcareer started at (one of) the root(s) of that subtree, otherwise the consecutiveness requirement could be violated. An example of this violation would be the successful matching of subcareer $x3y$ (x and y are only connected through all elements of the tree (1, 2 *and* 3 in this case)).

The process of transforming a career according to this paradigm can be viewed in Figure 6.4. More efficient methods exist for matching a substring in a trace monoid, but

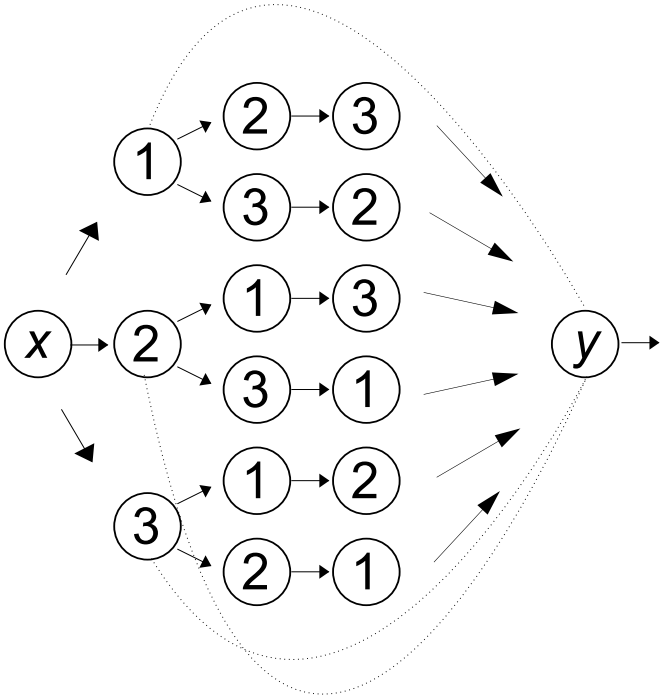

Figure 6.3: The addition of escape edges (dotted) for the first level

they require a reduction of the target string. Since we are matching a large number of completely different substrings, an equal amount of reductions would have to be done. Therefore, it is more efficient, for our approach, to do one transformation that can be easily traversed throughout the entire sequence phase, instead of multiple reductions of the target string, even though the initial calculation takes longer and more memory is required to store the representation.

Using the directed graph representation, there are no major changes needed in the sequence detection phase of Step 4. Although the implementation of graph traversion compared to transaction overlaying can be quite different, the underlying ideas of candidate creation and testing remain the same and the underlying search method unaltered.

6.3.3 Finding Defining Subcareers

Using the mechanism described above, we are now able to detect common subcareers within criminal careers, fulfilling one of our initial goals. The second goal, the detection of subcareers that are only frequent in one specific class, can now be reached easily as well. For this purpose we set two different thresholds, one representing the least amount of occurrences (as a fraction $\mathcal{T}_{\min}$) a subcareer must have to be frequent in a certain class and one representing the maximum number of occurrences (as a fraction $\mathcal{T}_{\max}$) this same

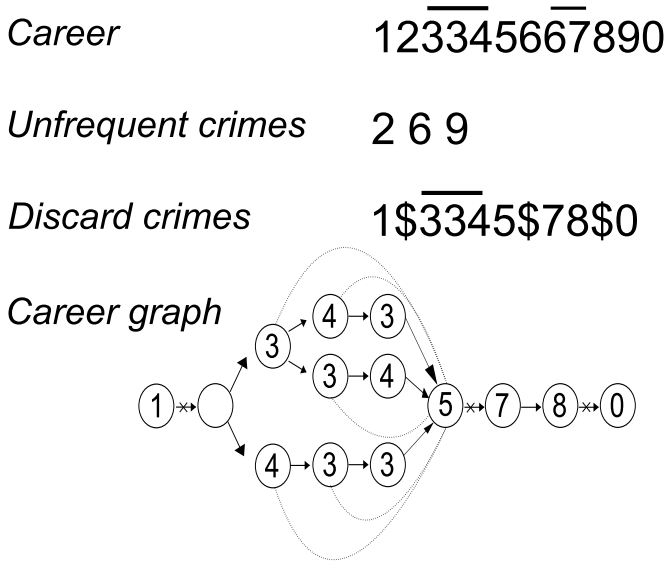

Figure 6.4: The process of transforming a criminal career in Step 3

subcareer can have in all other classes. Note that, counterintuitively, $\mathcal{T}_{\min} > \mathcal{T}_{\max}$. As a rule of thumb, the assumption that both numbers should be the same, the fraction strictly denoting the boundary between a frequent and infrequent status, could be maintained, however, a "grey area" may exist between these thresholds where subcareers are neither frequent nor infrequent. Section 6.4 investigates this relation more closely.

For the detection of these so-called *defining subcareers*, a separate search should be started for all of the 11 classes found in Chapter 5. Consequently there will be 11 different investigations. However, as both frequent and infrequent occurrence should be tested, one would assume all 11 classes would have to be investigated in each of these searches, leading to 11 times the work of the regular search for common subcareers in the entire database.

However, if we employ a three-step approach to this problem, most of this work can be omitted, leading to a very small increase compared to the regular search.

1. **Transformation Phase** All criminal careers in the database are transformed according to the method described above, performing steps 1, 2 and 3 for each class separately. Note that there are possibly crimes that do occur in one class after transformation and do not appear in another. Each career gets assigned its corresponding class number.

2. **Separate Sequence Phase** Each class of criminal careers is searched for common subcareers separately, performing steps 4 and 5 for every class. The results are stored in a Subcareer × Class matrix, where each entry is:

   - 0 Not investigated for this class.

- 1 Frequent in this investigation ($\geq \mathcal{T}_{\min}$).
- 3 Infrequent in this investigation($< \mathcal{T}_{\max}$).

Note that a row is only added when a subcareer is frequent in at least one class. After this phase, the same amount of computer work has been done as in the regular search.

3. **Discarding and Separation Phase** All the rows in the table that have 2 or more entries set to 1 are discarded. All the rows that have exactly one 1 and ten 2's as entries are set aside in a separate table. These represent the defining subcareers we want tot find.

4. **Testing Phase** All remaining rows are tested in all classes where the entry was set to 0. As soon as testing in one of the classes yields a 1 for a specific row, this row is discarded. Within this process, the maximum number of test that are performed is the number of candidate subcareers times 11, if every candidate in the matrix actually yields 2's for every class. Since a very large number of single subcareer tests was already performed in the sequencing phase, only a very small amount of extra computation time is needed for this phase. After this phase, all subcareers that remain are defining subcareers for one of the classes.

6.4 Experimental Results

As in the previous chapters, we tested our approach on the criminal record database (cf. Appendix B), that contains approximately one million offenders and as many criminal careers. Since a large number of the offenders in the list are one-time offenders, these individuals are left out of most of the calculation, greatly reducing the computation time. Furthermore, this class is also left out of the search for common subcareers, since only careers of length 1 are present in this class (except for a small percentage (<0.01%) that was erroneously classified as one-time offender).

As a first test a simple search for common subcareers was performed that yielded the following results:

Table 6.4: Amount of common subcareers discovered per threshold

	50%	40%	30%	20%	10%	5%
Subcareers Found	37	98	243	1,017	5,869	20,017

It appears that 30% is a reasonable threshold, yielding a manageable amount of careers. Figure 6.5 shows clearly how the amount of discovered common subcareers rises in the graph even with an exponential vertical axis. The longest common subcareer we discovered when using a threshold of 30% was of length 7:

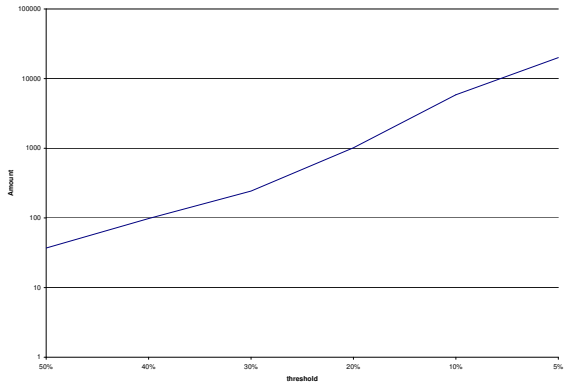

Figure 6.5: The relation between threshold and number of discovered subcareers

Minor Theft → Minor Theft → Minor Theft → Minor Theft → Major Theft → Major Theft → Major Theft

Figures 6.6 and 6.7 show how the discovered common subcareers are distributed over length and how the amount changes with length.

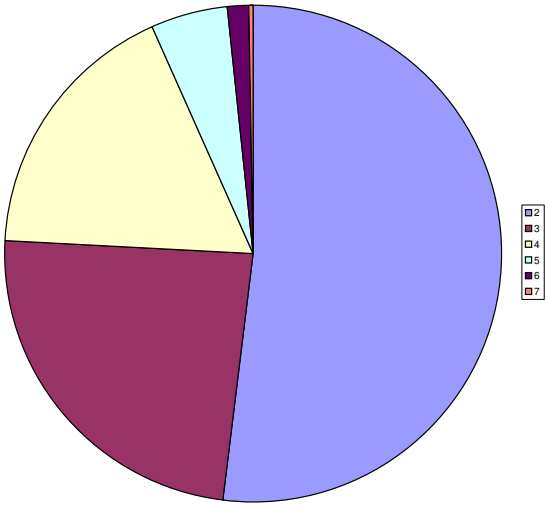

Figure 6.6: The distribution over length

It might be interesting to know how many of the discovered subcareers occur in the different classes. Figure 6.8 shows the division over these classes. It turns out that, when added, they together support 628 common subcareers, being approximately 2.5 times the total amount. This means that an average discovered subcareer appears in 2.6 different classes. The standard deviation in this matter is 1.1.

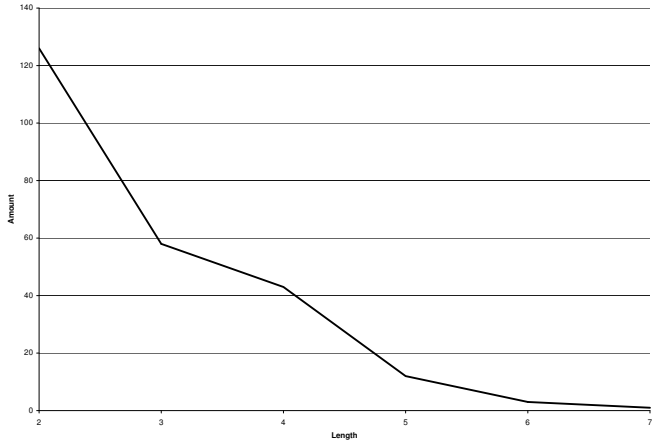

Figure 6.7: The relation between length and number of discovered subcareers

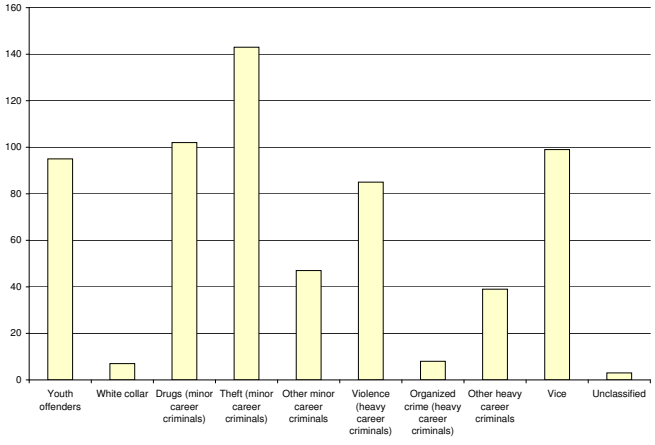

Figure 6.8: Distribution of the results over class

During the creation of the transformation the largest "subtree" (see Section 6.3.2) we encountered contained 6 levels and therefore 720 different paths (threshold: 30%), excluding escape edges, occuring in only one career. This is fortunate, since a slightly higher number would have had some serious effects on the computation time for our representation model. It is also surprising because the maximum amout of crimes in a single time frame in the database is 14. We can therefore state that at least 8 of those crimes were discarded. The maximum amount of paths in a single career, again excluding escape paths, was $6 \times 720 = 4{,}320$, occuring in the same career. Fortunately, both these trees were sufficiently distant from one another in the total career that they were probably not traversed simultaneously.

We investigated how the number of discovered defining subcareers depends on the different values for $\mathcal{T}_{\min}$ and $\mathcal{T}_{\max}$. As it makes no sense to set the threshold for infrequency higher than the threshold for frequency so these were omitted from our tests. Table 6.5 shows the results of batch testing with all possible combinations of $\mathcal{T}_{\min}$ and $\mathcal{T}_{\max}$.

Table 6.5: Amount of common subcareers discovered per threshold

	$\mathcal{T}_{\max}$ = 50%	40%	30%	20%	10%	5%
$\mathcal{T}_{\min}$ = 50%	6	2	0	0	0	0
40%		4	4	3	1	0
30%			112	63	54	3
20%				121	66	4
10%					763	13
5%						812

It is clear that the most results were reached with $\mathcal{T}_{\min}$ at 5 or 10%, only these thresholds were shown to yield too many common subcareers to be called reliable. Again, the best value for $\mathcal{T}_{\min}$ seems to be 30% where even a very strict setting for $\mathcal{T}_{\max}$, 10%, yields a good amount of defining subcareers.

If we choose these settings, we can see how the different defining subcareers are distributed over the different classes. Figure 6.9 shows the results. Clearly, the "Theft" class is best defined by its subcareers, largely outperforming the other classes.

We can now also check the effects of the addition of escape edges in our representation. We redid the experiments both with and without escape edges and checked the change in memory usage and computation time. The results are all averaged over 5 experiments per category. Figure 6.10 shows that the difference in memory load are reasonably low compared to the gain in computation time. Consequently the addition of the escape edges is an asset to our approach.

6.5 Conclusion and Future Directions

In this chapter we described a method that discovers common subcareers from a criminal record database. For this purpose we adapted a well-known algorithm that was more suitable for the retail industry, modifying a number of different phases to suit the search for these frequently occurring subcareers. A representation of the database was chosen that was computationally hard to create but allowed fast searching in the next phase, which led to a better performance overall. Within this representation, we introduced the notion of escape edges that made it easy to find certain types of subcareers that would otherwise have been located very slowly. We also presented a number of results, investigating both

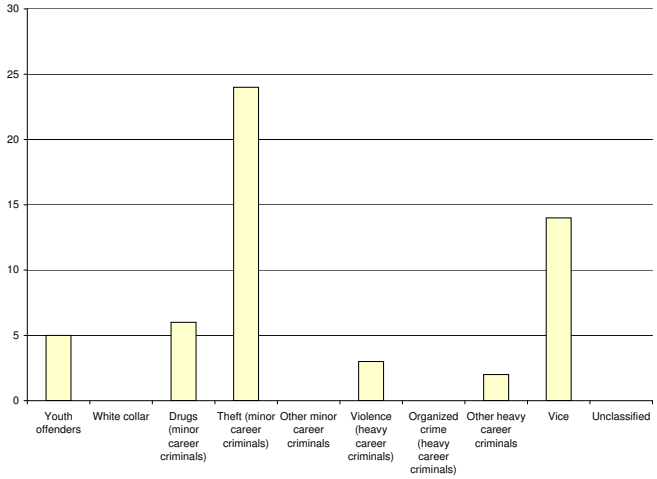

Figure 6.9: Distribution of the defining subcareers over class

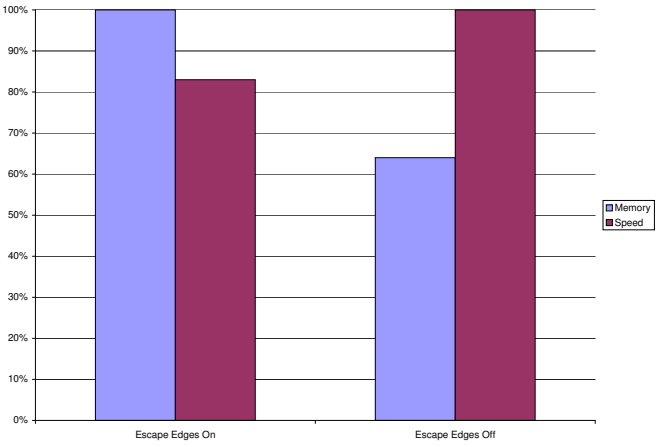

Figure 6.10: Comparison between escape edges off and on as an percentage of the maximum

the usability of our approach on actual data and investigated some of its parameters.

Next to the search for common subcareers, we discovered some subcareers that are frequent in only one class of criminal career (see Chapter 5) and infrequent in all others. These subcareers were discovered through two different thresholds, a minimal threshold to reach frequent status and a maximum threshold denoting the number occurrences this same sequence can have in other classes. Even though we maintained reasonably strict thresholds, a small number of these defining thresholds were discovered.

A possible goal for this research, other than the analysis of crime in general, was

to use it as a prediction tool. In this case, a defining subcareers detection for an offender could in theory predict the complete career this individual might develop. However, since only 54 different defining subcareers were discovered, the potential of this research for that purpose is limited. Future research in this area could therefore focus on using the outcome of this result in criminology research to better understand the mechanics of developing criminal careers.