Journal of Southeast University (English Edition)   Vol.19 No.3 Sept. 2003   ISSN 1003—7985

Matching spatial relation graphs using a constrained partial permutation strategy

Xu Xiaogang[1] Sun Zhengxing[1] Liu Wenyin[2]

([1] Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)
([2] Department of Computer Science, City University of Hong Kong, Hong Kong, China)

Abstract: A constrained partial permutation strategy is proposed for matching spatial relation graph (SRG), which is used in our sketch input and recognition system Smart Sketchpad for representing the spatial relationship among the components of a graphic object. Using two kinds of matching constraints dynamically generated in the matching process, the proposed approach can prune most improper mappings between SRGs during the matching process. According to our theoretical analysis in this paper, the time complexity of our approach is $O(n^2)$ in the best case, and $O(n!)$ in the worst case, which occurs infrequently. The spatial complexity is always $O(n)$ for all cases. Implemented in Smart Sketchpad, our proposed strategy is of good performance.

Key words: spatial relation graph; graph matching; constrained partial permutation; graphics recognition

Graphs are powerful data structures for structural relational descriptions in structural pattern recognition and computer vision. By assigning a suitable meaning to nodes and edges of graphs, it is possible to achieve complete and univocal representations of objects[1]. Typically, nodes represent the components of an object and edges represent relations between these components. When graphs are used for representing structured objects, the problem of measuring the similarity of objects turns into the problem of calculating the similarity of graphs, which is generally referred to as graph matching[2]. In the last decades, plenty of algorithms for graph matching have been proposed[3-8].

In this paper, a novel approach to graph matching is proposed. We use a so-called constrained partial permutation strategy to dynamically prune improper permutation states in which two graphs are determined unmatched without necessity to check their details. This matching strategy has been employed in our spatial relation graph (SRG) matching, which is proposed by X.G.Xu, et al.[9] and used for representing the spatial relationship among the components of a graphic object and matching between two graphic objects in our sketch input and recognition system Smart Sketchpad[10-12], and has yielded good results[12].

The rest of the paper is organized as follows. Section 1 presents the constrained partial permutation strategy. Discussion on the computational complexity of the constrained partial permutation algorithm is described in section 2. Finally, we give our conclusion in section 3.

1 Constrained Partial Permutation Strategy

In order to save the drawing efforts for the user, we try to predict what the user intends to draw before he completes it. This is done by matching the user's current sketchy input with the predefined graphic objects in the database and informing him of the list of those graphic objects that might be his intention so that he can simply select the real intended one from the list[10].During the matching process, we should check whether each graphic primitive type and each spatial relation between graphic primitives is preserved under a mapping of the nodes from the SRG of the incomplete sketchy object to that of the graphic object in the database. Obviously, the whole process depends on that mapping and enumerating all such mappings is a full permutation problem. However, since most mappings are illegal and can therefore be pruned directly during the permutation process according to the matching constraints. Hence, this permutation process is a constrained and partial one.

Definition 1 Constrained partial permutation: given two positive integers, m, n and $m \leqslant n$, select m different integers from $[1,\cdots,n]$, and then rank

Received 2003-01-27.
Foundation items: The National Natural Science Foundation of China (69903006) and grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (City U 1073/02E).
Biographies: Xu Xiaogang (1977—), male, master; Sun Zhengxing (corresponding author), male, doctor, associate professor, szx@nju.edu.cn.

them in a list, written as B_1, B_2, $\cdots$, B_k, $\cdots$, B_m, where 1, 2, $\cdots$, k, $\cdots$, m are the positions in the list and B_1, B_2, $\cdots$, B_k, $\cdots$, B_m are the values at the corresponding positions in the list, enumerate all possible such lists which satisfy one or both of the following two classes of constraints:

① Vertex matching constraint: i can't be put into position k, that is, $B_k \neq i$, denoted as $R_s(i, k)$.

② Edge matching constraint: If i has been put into position k, j cannot be put into position l, that is, if $B_k = i$, then $B_l \neq j$, where i, $j \in [1, \cdots, n]$, k, $l \in [1, \cdots, m]$ and $k < l$, denoted as $R_p(i, k) - (j, l)$. This is actually a conditional vertex matching constraint.

In definition 1, position k in the vertex matching constraint and position l in the edge matching constraints are referred to as the increasing point for calculating the subsequent proper permutations. In order to acquire all possible permutations, we regard the string $B_1 B_2 \cdots B_m$ as an m-digit n-cardinality integer (i.e., B_k can be an integer between 1 and n), in which all digits are different, and then enumerate all such m-digit integers in the lexicographic order, i.e., from the beginning i.e., $123 \cdots m$, to the end, i.e., $n\ (n-1)\ (n-2)\ \cdots\ (n-m+1)$. Each time we obtain its next permutation by finding the smallest legal m-digit number that is larger than the current one.

1.1 How to find next permutation

To record a permutation state, we use an n-digit n-cardinality integer B, whose first m-digit integer, $B_1 B_2 \cdots B_m$, is the permutation we need, which is referred to as the working area and the latter $(n - m)$-digit integer, $B_{m+1}\ B_{m+2}\ \cdots\ B_n$, is used to record extra information of current permutation, which is referred to as the backup area, as illustrated in Fig.1. In algorithm 1, we describe how to compute the next possible legal permutation from the current permutation according to a given d, where d is the increasing point.

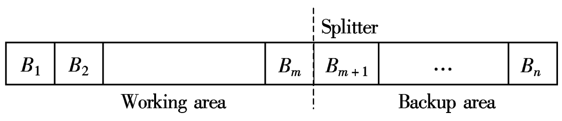

Fig.1 The data structure used to record the permutation state

Algorithm 1 Next-permutation (B, d)

Objective: to calculate the next permutation of the current permutation according to a given d.

Input: d, B, where B is the current permutation state and position d is the increasing point.

Output: the next proper permutation or null, where null means there is no proper next permutation of the current permutation according to the given d.

```
Begin
sort ($B_{d+1}$, $\cdots$, $B_n$) in ascending order;
if ($B_d \geqslant B_n$)
{   do {$d = d - 1$;} while (($d \neq 0$) && ($B_d > B_{d+1}$));
    if ($d == 0$)
    {   output NULL;
        End;
    }
    sort ($B_{d+1}$, $\cdots$, $B_n$)in ascending order;
}
find the smallest number $B_t$ in the backup area satisfying $B_t > B_d$;
exchange $B_d$ with $B_t$;
output $B$;
End
```

1.2 Enumerating with state pruning

Enumerating all the permutations is both uneconomical and unnecessary. As we mentioned before, many permutations can be pruned during the enumerating process. If there is no matching constraint being satisfied (i.e., all components are matched), we regard m as the increasing point, which means the next immediate permutation will be found and no permutation is necessary to be pruned. If a certain matching constraint is satisfied, we specify the increasing point according to this constraint. For instance, $m = 3$ and $n = 5$, the first state is 123 | 45. If there is a vertex matching constraint $R_s(1,1)$(i.e., 1 cannot be put in position 1), we do not compute the next permutation by nextpermutation (123 | 45, 3) with increasing point 3, since all permutations like 1 ** | (* is digit between 1 and 5.) are illegal according to this matching constraint. We have to obtain the next permutation directly by next-permutation (123 | 45, 1) so that B_1 is increased. Hence, we can directly jump to the permutation of 213 | 45 and eleven states (124 | 35, 125 | 34, 132 | 45, 134 | 25, 135 | 24, 142 | 35, 143 | 25, 145 | 23, 152 | 34, 153 | 24, 154 | 23) are skipped over. The whole process is described in algorithm 2.

Algorithm 2 Enumerating with state pruning

Objective: to enumerate all the proper permutations with state pruning.

Input: n, m, and the matching constraints including some vertex matching constraints and some edge matching constraints.

Output: all the proper permutations according to the matching constraints.

```
Begin
for( $i = 1$; $i <= n$; i++) $B_i = i$;
$S = B$;
while ($S != $ NULL)
{  if ($S$ satisfies some matching constraint $r$)
   {  set $d = k$ (if $r \in R_s$) or set $d = l$ (if $r \in R_p$);
      $S$ = Next$_d$-permutation ($S$);
   }
Else
{  {  output $S$;
   }
   $S$ = Next$_m$-permutation ($S$);
 }
}
End
```

2 Complexity Analysis and Performance Evaluation

From the permutation process discussed in section 1, we can find that some improper permutation states must be pruned according to the matching constraints. In this section, we are interested in how many permutations can be excluded using our constrained partial permutation approach for efficient SRG matching.

According to algorithm 1, the information recorded in the n-digit n-cardinality integer is sufficient for the permutation enumeration process. The space complexity is therefore $O(n)$.

In algorithm 1, d plays a critical role in finding the next proper permutation that represents a good match. The number of pruned permutation states depends on the value of d, i.e., if d is very small and close to 1, the eliminated states are much more than those when d is bigger and close to m. When d is equal to m, that is, the increasing point is m, we will calculate the next immediate permutation using this increasing point and there is no permutation state that can be pruned. Different SRGs may have different matching constraint lists satisfied and may have different d values for the next-permutation (B, d) calculation. Consequently, there are diverse numbers of pruned states for different SRGs. In other words, the complexity of constrained partial permutation strategy is SRG-dependent. We refer to the existence of different matching constraints with different d values as a distribution of matching constraints.

Obviously, if there are no matching constraints generated and no states can be pruned, it is the worst case, in which the computational cost is P_n^m, where m is the number of components in the incomplete object and n is the number of components of the complete object in the matching process. Its computational complexity is $O(n!)$, which increases exponentially with n. In the best case, there are at least $m(n-1)$ different vertex matching constraints, and only $n + (n-1) + \cdots + (n-m+1) = m(2n-m+1)/2$ permutation states need to be checked. Its computational complexity is $O(n^2)$. In other cases, the number of the pruned permutations depends on the specific SRG and it relates to m, n, and the distribution of matching constraints.

In our Smart Sketchpad system, we have created 300 representative graphic objects, which have diverse numbers of components ranging from 2 to 14. We use each of the 300 graphic objects as a query and try to match it with the rest 299 graphic objects in the database. Obtaining the result, we mainly focus on the average time cost of each query. The time cost during the constrained partial permutation procedure is a much concerned factor for evaluating its performance. The time cost on a Pentium III 450 PC with 256 MB memory in ms of our approach is listed in Tab.1.

Tab.1 Time used in SRG matching ms

m	n												
	2	3	4	5	6	7	8	9	10	11	12	13	14
2	0.41	0	0.20	0.19	0.08	0.08	0.22	0.11	0.14	0.06	0	0.36	0
3	N/A	0.03	0.07	0.15	0.27	0.11	0.09	0.09	0.39	0.29	0.59	0	0.29
4	N/A	N/A	0.15	0.12	0.09	0.12	0.20	0.14	0.41	0.35	0.28	0.28	0.83
5	N/A	N/A	N/A	0.12	0.11	0.25	0.27	0.41	0.70	1.08	0.54	0.77	3.08
6	N/A	N/A	N/A	N/A	0.16	0.16	0.30	0.5	0.88	1.01	0.36	4.02	5.90
7	N/A	N/A	N/A	N/A	N/A	0.14	0.25	0.73	0.72	1.08	0.21	2.13	5.86
8	N/A	N/A	N/A	N/A	N/A	N/A	0.33	1.23	2.43	3.51	3.33	54.62	26.08
9	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.56	1.76	1.54	0	4.8	106.32
10	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	3.11	1.34	0.67	0.33	1
11	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.67	40	0	1.25
12	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	10	0	0
13	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	20	80.25
14	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	7.5

"N/A" means that we do not calculate it because the value of m is bigger than that of n and this conflicts with the definition of partial matching[12]. The zero values in the table are very small real numbers.

From Tab.1, we can find that the maximum time used is 106.32 ms and the minimum time used is less than 1 ms. The time used in the constrained partial permutation procedure waves with m and n. It has no mono-ascending or mono-descending trends while m and n increase. However, the general trend is that the time cost increases with m and n. To some extent this confirms the conclusion we made above that our strategy is SRG-dependent. Generally, the time cost is much smaller than 1/10 s. The performance is sufficiently fast for real-time interaction.

3 Conclusion

SRGs have been used to represent the primitive components and their relations within graphic objects in our sketch recognition system Smart Sketchpad. It has very strong representation ability in sketch processing. To compute the similarity of two graphic objects represented in spatial relation graphs, we proposed a new approach, which uses a constrained partial permutation strategy to prune most improper mapping states during the matching process. The constrained partial permutation strategy dynamically detects a set of matching constraints in the matching process to predict those improper permutations. Thus, we can save much computational cost. According to our theoretical analysis in this paper, the time complexity of our approach is $O(n^2)$ in the best case, and $O(n!)$ in the worst case, though the worst case does not occur frequently. From the experimental results we can see that our approach is sufficiently efficient for real-time sketch recognition.

References

[1] Foggia P, Sansone C, Vento M. A database of graphs for isomorphism and sub-graph isomorphism benchmarking [A]. In: *Proc of 3rd IAPR-TC15 Workshop on Graph-Based Representations in Pattern Recognition* [C]. Ischia, 2001.

[2] Bunke H. Recent developments in graph matching [A]. In: *Proc of 15th International Conference on Pattern Recognition* [C]. Barcelona, 2000, **2**: 117-124.

[3] Ullman J. An algorithm for sub-graph isomorphism [J]. *Journal of the Association for Computing Machinery*, 1976, **23**(1): 31-42.

[4] Corneil D G, Gotlieb C C. An efficient algorithm or graph isomorphism [J]. *Journal of the Association of Computing Machinery*, 1970, **17**(1): 51-64.

[5] Schmidt D C, Druffel L E. A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices [J]. *Journal of the Association of Computing Machinery*, 1976, **23**(3): 433-445.

[6] Mckay B D. Practical graph isomorphism [J]. *Congressus Numerantium*, 1981, **30**(1): 45-87.

[7] Cordella L P, Foggia P, Sansone C, et al. Evaluating performance of the VF graph matching algorithm [A]. In: *Proc of 10th International Conference on Image Analysis and Processing* [C]. 1999. 1172-1177.

[8] Cordella L P, Foggia P, Sansone C. An improved algorithm for matching large graphs [A]. In: *Proc of 3rd IAPR-TC15 Workshop on Graph-Based Representation in Pattern Recognition* [C]. Ischia, 2001.

[9] Xu X G, Sun Z X, Peng B B, et al. A SRG-based online composite graphic recognition strategy for sketch-based user interface [A]. In: *Proc of the 1st International Conference on Machine Learning and Cybernetics* [C]. Beijing, 2002. 723-728.

[10] Liu W Y, Qian W, Xiao R, et al. Smart sketchpad — an on-line graphics recognition system [A]. In: *Proc of the ICDAR2001 Conference* [C]. 2001. 1050-1054.

[11] Liu W Y, Jin X Y, Sun Z X. Sketch-based user interface for inputting graphic objects on small screen devices [J]. *Lecture Notes in Computer Science*, 2002, **2390**: 67-80.

[12] Xu X G, Liu W Y, Jin X Y, et al. Sketch-based user interface for creative tasks [A]. In: *Proc of 5th Asia Pacific Conference on Computer Human Interaction* [C]. Beijing, 2002. 560-570.

基于约束的部分枚举策略的空间关系图匹配算法研究

徐晓刚[1] 孙正兴[1] 刘文印[2]

([1] 南京大学计算机软件新技术国家重点实验室,南京 210093)
([2] 香港城市大学计算机科学系, 中国香港)

摘 要 本文提出了一种基于约束的部分枚举空间关系图匹配策略.该策略通过使用在匹配过程中动态生成的 2 类匹配约束条件智能预测当前匹配状态的后继有效的枚举状态以跳过无效的中间匹配状态,达到状态空间剪枝的目的,可以有效降低空间关系图匹配过程中状态搜索空间.根据理论分析,该策略在最好情况下的时间复杂度为 $O(n^2)$,在几乎很少发生的最坏情况下时间复杂度为 $O(n!)$;其空间复杂度都是 $O(n)$.所提出的方法已在笔者研发的手绘草图识别系统 Smart Sketchpad 中取得了很好的识别效果.

关键词 空间关系图; 图匹配; 约束的部分枚举; 图形识别

中图分类号 TP391.4