

A Product Graph Based Method for Dual Subgraph Matching Applied to Symbol Spotting

Anjan Dutta¹(✉), Josep Lladós¹, Horst Bunke², and Umapada Pal³

¹ Computer Vision Center, Universitat Autònoma de Barcelona, Barcelona, Spain
`{adutta,josep}@cvc.uab.es`

² Institute of Computer Science and Applied Mathematics,
Universität Bern, Bern, Switzerland
`bunke@iam.unibe.ch`

³ CVPR Unit, Indian Statistical Institute, Kolkata, India
`umapada@isical.ac.in`

Abstract. Product graph has been shown as a way for matching subgraphs. This paper reports the extension of the product graph methodology for subgraph matching applied to symbol spotting in graphical documents. Here we focus on the two major limitations of the previous version of the algorithm: (1) spurious nodes and edges in the graph representation and (2) inefficient node and edge attributes. To deal with noisy information of vectorized graphical documents, we consider a *dual edge graph* representation on the original graph representing the graphical information and the product graph is computed between the dual edge graphs of the *pattern graph* and the *target graph*. The dual edge graph with redundant edges is helpful for efficient and tolerating encoding of the structural information of the graphical documents. The adjacency matrix of the product graph locates the pair of similar edges of two operand graphs and exponentiating the adjacency matrix finds similar random walks of greater lengths. Nodes joining similar random walks between two graphs are found by combining different weighted exponentials of adjacency matrices. An experimental investigation reveals that the recall obtained by this approach is quite encouraging.

Keywords: Product graph · Dual edge graph · Subgraph matching · Random walks · Graph kernel

1 Introduction

Product graph was introduced for computing the random walk graph kernel for measuring the similarity between two graphs. Recently it has been used for subgraph matching and applied for spotting symbols on graphical documents [3]. In this paper we propose an extension and improvement of the product graph methodology that was proposed in [3]. Particularly, this work mainly focuses on the two major limitations of the previous version of the method: (1) spurious nodes and edges that are generated during low level image processing and

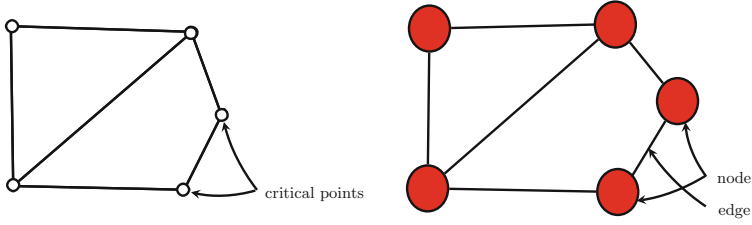


Fig. 1. Each critical point detected by the vectorization technique is considered as a node and the straight lines joining them are considered as the edges.

(2) inefficient node and edge attributes. Of course, these problems are application and representation dependent, but there is no doubt that the proposed solution is more robust to distortion and noise, which will be further useful for other applications and representations.

Graph representation of graphical documents involves some low level image processing such as binarization, skeletonization, vectorization etc. For example, we use Qgar¹ for vectorizing the given binary images. This particular vectorization generates critical points and connectivity information between them. For representing a document with a graph, we consider the critical points as the nodes and the lines joining them as the edges (Fig. 1). The main problem in this kind of low level image processing is the addition of noisy information. As an example, Fig. 2(b) shows the graph representation of the symbol in Fig. 2(a) under the previously mentioned representation scheme, whereas Fig. 2(c) shows an ideal graph representation of the symbol (considering the junctions as the nodes and their connections as the edges). Here we can see an example of the introduction of numerous spurious nodes near the junctions and corners. Note that such a vectorization can also generate spurious and discontinuous edges. This kind of structural noise always creates problems for matching or comparing (sub)graphs. It is true that with a different kind of graph representation it may be possible to solve the problem more efficiently, but dealing with this kind of distortions or noise at the graph level is interesting for other domain also as it gives more robustness in the matching method.

Product graph has been introduced for computing random walk graph kernels [6]. In our previous work, we introduced product graph for subgraph matching and applied it for spotting symbols in graphical documents [3]. Formally, a symbol spotting method can naturally be formulated as a subgraph matching problem where a *query symbol* can be represented as a *pattern graph* and the big *target document* can be represented as a *target graph*. For more information on symbol spotting methods based on graph representation we refer to [4, 8], where a literature review is given. Product graph provides an efficient comparison technique between a subgraph and a graph in terms of different substructures (in this case random walks) which allows the entire subgraph matching process to

¹ <http://www.qgar.org>

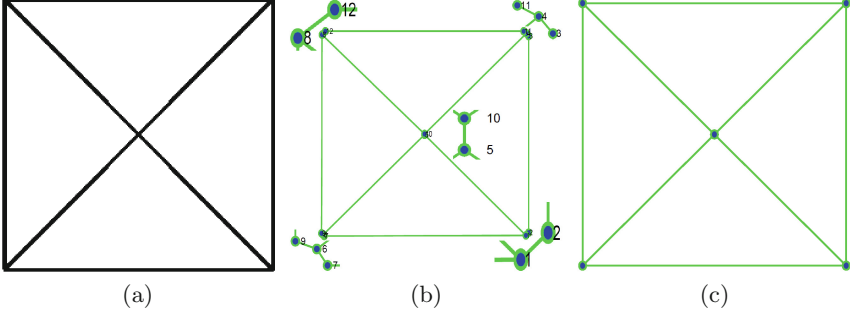


Fig. 2. Difference between a real graph representation and an ideal graph representation: (a) an architectural symbol, (b) the graph representation of the symbol in (a) after doing the vectorization (note the spurious nodes near the corners and junctions), (c) an ideal graph representation of the symbol in (a).

be computed online. The product graph finds the similar pair of edges in the operand graphs. This information can be obtained from the adjacency matrix of the product graph. Exponentiating the adjacency matrix provides the information about similar pair of nodes which are connected with random walks of greater lengths. This was the main motivation of the work in [3] but one of the problems was selecting efficient node and edge attributes, especially when the graphs contain noise or distortions like spurious nodes and edges, and the possibility of discontinuous edges. To solve this problem in this paper we introduce the *dual edge graph* DG with redundant edges of the original graph G and consider the product graph of the dual edge graphs. Here the dual edge graph DG of the original graph G is a graph that has a vertex corresponding to each edge of G and an edge joining two neighbouring edges in G . In this work we use a dual edge graph with redundant edges to cope with the distortions and noise as explained in Sect. 2. The variation of dual graph provides us robust node and edge labels and with this information the product graph is better suited for subgraph matching applied to symbol spotting.

The rest of the paper is organized into three sections. In Sect. 2, we present the methodology to represent the graphical documents with dual graph with redundant edges, computing the node and edge labels and computing the product graphs to spot the symbol on documents. Section 3 contains a description of our current experimental results. After that, in Sect. 4, we conclude the paper and discuss future directions of work.

2 Methodology

Let $G_M = (V_M, E_M)$ be the basic graph representing the model or query symbol and $G_I = (V_I, E_I)$ be the same representing the input or target document, where V_M and V_I are the set of vertices, in this case the critical points, and E_M and E_I

are the set of edges connecting the critical points. Now the subgraph matching is intended to find the different instances or occurrences of G_M in G_I . Below we describe the procedure of obtaining the dual edge graph from a basic graph and then we describe the product graph based subgraph matching methodology using the dual edge graph representation.

2.1 Dual Edge Graph

In general, the dual graph of a plane graph is a graph that has vertex corresponding to each face (or plane) and an edge joining two neighbouring faces sharing a common edge in plane graph. We bring the same analogy to our problem and assign a node to each edge of the original graph and an edge joining two neighbouring nodes. We can call this graph as dual edge graph. In this article, as we are not dealing with any other type of dual graph, we can alternatively call it dual graph. Furthermore, we will call the nodes, edges of the dual edge graph respectively as dual nodes, dual edges. We denote $DG = (DV, DE)$ as the dual edge graph of the edge graph $G = (V, E)$.

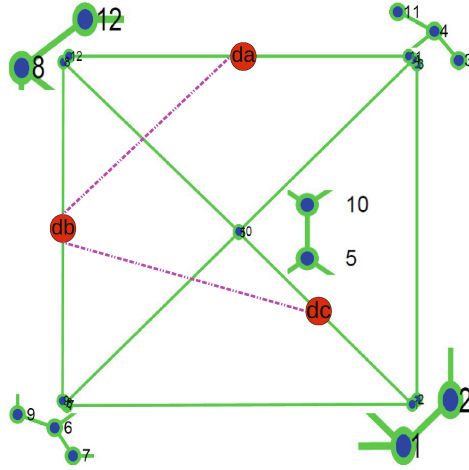


Fig. 3. Details of adding redundant edges in the dual graph considering only three nodes da , db and dc and $n = 3$: The $\text{gdist}(da, db) = 2$ and $\text{gdist}(db, dc) = 3$, that is why there exist edges (da, db) and (db, dc) but since $\text{gdist}(da, dc) = 4$, there is no edge (da, dc) (Note the spurious nodes, edges near the corners and junctions in the original graph that are plotted in green continuous line) (Color figure online).

Let $G_M = (V_M, E_M)$ be an unattributed edge graph representing a model or query symbol and $G_I = (V_I, E_I)$ be an unattributed edge graph representing an input or target document. Then we can get the dual edge graphs $DG_M = (DV_M, DE_M)$ and $DG_I = (DV_I, DE_I)$ of G_M and G_I respectively, where $DV_M = E_M$ and $DV_I = E_I$. Here it should be mentioned that we will

denote a dual node which joins two nodes, say, v_i, v_j in the edge graph as dv_{ij} (see Fig. 3). Now the dual edge sets of DG_M and DG_I are respectively defined as follows:

$$DE_M = \{(du_{ij}, du_{kl}) : \text{gdist}(du_{ij}, du_{kl}) \leq dn \in \mathbb{N} \text{ and } du_{ij}, du_{kl} \in DV_M\} \quad (1)$$

$$DE_I = \{(dv_{ij}, dv_{kl}) : \text{gdist}(dv_{ij}, dv_{kl}) \leq dn \in \mathbb{N} \text{ and } dv_{ij}, dv_{kl} \in DV_I\} \quad (2)$$

Here $\text{gdist}(du, dv)$ stands for the minimum number of edges (in the edge graph) one has to traverse for going to dv from du (or to du from dv for undirected graph, which is our case). dn is connectivity parameter regulating the connections between any two dual nodes. The dual graphs with dual edge sets (in magenta) can be seen in Fig. 4 which shows some stability in the representation as visually we can observe the existence of common graph structures.

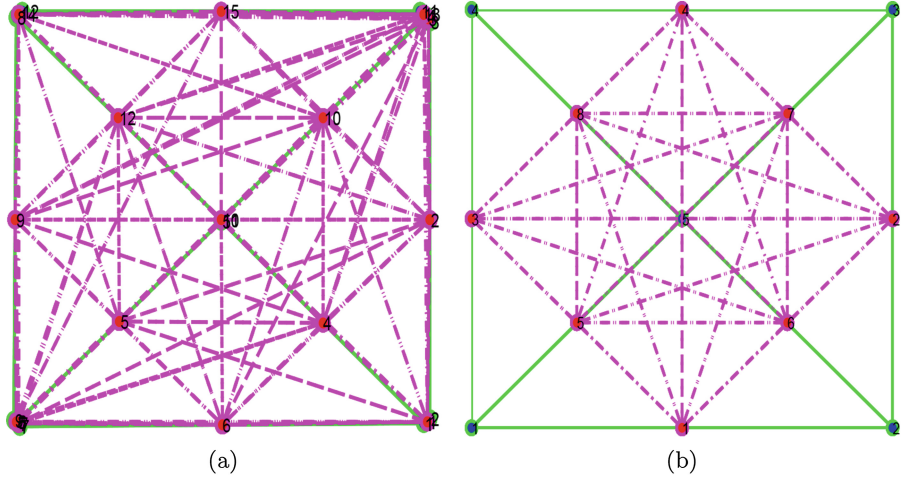


Fig. 4. Original graph and redundant dual graph representation (with $dn = 3$) of (a) a real symbol and (b) an ideal symbol. The original graph is plotted with green continuous line and its corresponding dual graph with redundant edges is plotted with magenta discontinuous line. (Best viewed in colour.) (Color figure online)

Given this dual edge graph representation we assign node and edge labels to capture the local structural information and create attributed dual graphs $DG_M = (DV_M, DE_M, \alpha_M, \beta_M^1, \beta_M^2)$ and $DG_I = (DV_I, DE_I, \alpha_I, \beta_I^1, \beta_I^2)$. Here $\alpha_I : DV_I \rightarrow \mathbb{R}^7$ is a node labelling function and is defined as $\alpha_I(dv_{ij}) = \text{Hu}$ moments invariants [7] of the acyclic graph paths between v_i and v_j of length less than or equal to $m \in \mathbb{N}$, $dv_{ij} \in DV_I$. Similarly, $\alpha_M(du_{ij})$ is the same node labelling function but defined in DG_M . It is to be mentioned that for each path, we get a vector of dimension seven and hence for a node dv in a dual graph DG ,

we get a set of Hu moments invariants, let us denote the set as $Hu(dv)$. The distance between two dual nodes du, dv is computed as the minimum cost of assigning each path of $Hu(du)$ to a path in $Hu(dv)$.

$\beta_I^1 : DE_I \rightarrow \mathbb{R}$ is an edge labelling function and is defined as:

$$\beta_I^1(dv_{ij}, dv_{kl}) = \min \angle(dv_{ij}, dv_{kl}), (dv_{ij}, dv_{kl}) \in DE_I.$$

$\beta_I^2 : DE_I \rightarrow \mathbb{R}$ is another edge labelling function and is defined as:

$$\beta_I^2(dv_{ij}, dv_{kl}) = \frac{\text{mdist}(dv_{ij}, dv_{kl})}{\max(\text{length}(dv_{ij}), \text{length}(dv_{kl}))}, (dv_{ij}, dv_{kl}) \in DE_I.$$

here $\text{mdist}(dv_{ij}, dv_{kl})$ is the length of the line joining the midpoint of the edges dv_{ij} and dv_{kl} . β_M^1 and β_M^2 are defined respectively as β_I^1 and β_I^2 but in DE_M .

The main limitation of this representation that one could face is when the extremities of a certain edge of the edge graph do not have any other connection. In that case the corresponding dual node will lose local discrimination as shown in Fig. 5a.

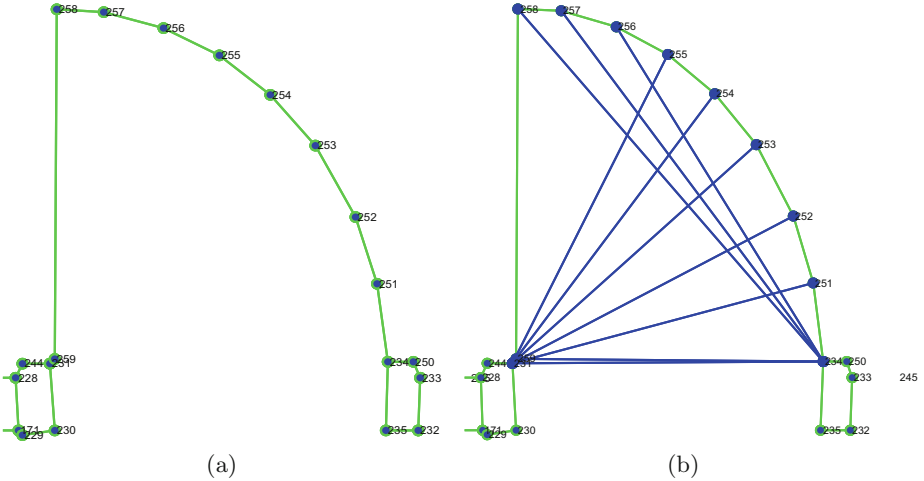


Fig. 5. An example of transitive closure edges: (a) original graph representation of *door1*, (b) edges (shown in blue) connect the pair of nodes that were not connected by more than one graph paths in the original graph representation (Color figure online).

We resolve this difficulty by relating those nodes in the edge graph by a relation inspired by the *transitive closure* of a graph. For that we select the pair of nodes u, v such that $(u, v) \in E$ but there is no other graph path between u and v . Let V^* be the set of all such nodes. V^* may contain two different kind of nodes. Ones having an adjacent node from $V - V^*$ i.e. $u \in V^*$ and $v \in V - V^*$ such that $\exists (u, v) \in E$, let us call all such nodes as V_1^* (for example nodes like 251

and 254 in Fig. 5). And others having adjacent nodes only from V^* i.e. $u, v \in V^*$ so that $(u, v) \in E$, let us call such nodes as V_2^* (for example nodes like 271, 272, 273 and so on in Fig. 5). We update the edge set E by adding edge (u, v) where $u \in V_1^*$ and $v \in V_2^*$ and there exist at least one graph path between them. We can call this new kind of edge as *transitive closure edge*. We update both the edge graphs G_I , G_M and assign a dual node to each transitive closure edge and connect them with dual edges accordingly. The Hu moments of the set of paths joining the extremities of the transitive closure edges serve as the attributes of the new nodes. The attributes of the new dual edges are computed as explained previously. In this way we update DG_I and DG_M , respectively.

This dual graph gives us the attributed graphs and because of having redundant edges the representation is stable to distortions such as spurious edges, nodes. Moreover, the idea of transitive closure gives us the possibility to group several similar patterns.

So given the dual graph representation of the graphical documents, the symbol spotting problem can be formulated as a subgraph matching problem. Let $DG_M = (DV_M, DE_M, \alpha_M, \beta_M^1, \beta_M^2)$ and $DG_I = (DV_I, DE_I, \alpha_I, \beta_I^1, \beta_I^2)$ be the dual graphs of the edge graphs G_M and G_I respectively.

2.2 Product Graph

Product graph $G_P = (V_P, E_P)$ of DG_I and DG_M relates the target graph DG_I and the pattern graph DG_M with the node and edge sets. The properties or the conditions are included in the set definitions as follows:

$$V_P = \{(du_{ij}, dv_{ij}) : du_{ij} \in DV_M, dv_{ij} \in DV_I, \alpha_M(du_{ij}) \simeq \alpha_I(dv_{ij}), \\ \beta_M^1(du_{ij}, du_{kl}) \simeq \beta_I^1(dv_{ij}, dv_{kl}) \text{ and } \beta_M^2(du_{ij}, du_{kl}) \simeq \beta_I^2(dv_{ij}, dv_{kl})\}$$

and given the above set of nodes, the edge set E_P will be:

$$E_P = \{((du_{ij}, dv_{ij}), (du_{kl}, dv_{kl})) : (du_{ij}, du_{kl}) \in DE_M, (dv_{ij}, dv_{kl}) \in DE_I\}$$

We use the parameters t_α , t_{β_1} and t_{β_2} for measuring the node and edge similarities as follows:

$$\alpha_M(du_{ij}) \simeq \alpha_I(dv_{ij}) \Leftrightarrow |\alpha_M(du_{ij}) - \alpha_I(dv_{ij})| \leq t_\alpha$$

Here

$$|\alpha_M(du_{ij}) - \alpha_I(dv_{ij})| = \sum_{p_1 \in Hu(du_{ij})} \min_{p_2 \in Hu(dv_{ij})} d(p_1, p_2) + \sum_{p_2 \in Hu(dv_{ij})} \min_{p_1 \in Hu(du_{ij})} d(p_1, p_2)$$

is a modified Hausdorff distance as investigated in [5] and $d(.,.)$ denotes the Euclidean distance.

$$\beta_M^1(du_{ij}, du_{kl}) \simeq \beta_I^1(dv_{ij}, dv_{kl}) \Leftrightarrow |\beta_M^1(du_{ij}, du_{kl}) - \beta_I^1(dv_{ij}, dv_{kl})| \leq t_{\beta_1}$$

$$\beta_M^2(du_{ij}, du_{kl}) \simeq \beta_I^2(dv_{ij}, dv_{kl}) \Leftrightarrow |\beta_M^2(du_{ij}, du_{kl}) - \beta_I^2(dv_{ij}, dv_{kl})| \leq t_{\beta_2}$$

where nl is the maximum number of times the matrix E_P is being exponentiated and λ is sufficiently small to converge the above summation.

These weights to different walk lengths are inspired by the traditional random walk graph kernel [6]. This way of weightings reduces the influence of walks of greater lengths because they often contains redundant or repeated information. We have observed that for sufficiently smaller value of λ such as $\lambda \in [0.1, 0.2]$ the above sum converges rapidly. However, this fact was also observed by Gärtner *et al.* [6].

Now let us take an example of the matrix A as follows:

$$A = \begin{matrix} & \begin{matrix} (du_1, dv_1) & \cdots & (du_k, dv_l) & (du_{k+1}, dv_{l+1}) & \cdots & (du_m, dv_n) \end{matrix} \\ \begin{matrix} (du_1, dv_1) \\ \vdots \\ (du_{l-1}, dv_{k-1}) \\ (du_l, dv_k) \\ (du_{l+1}, dv_{k+1}) \\ \vdots \\ (du_m, dv_n) \end{matrix} & \begin{bmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & x_1 & \cdots & 0 \\ 0 & \cdots & x_2 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & x_3 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} \end{matrix}$$

where $m < n$ and $1 < k, l \leq m, n$.

Let us further assume for simplicity that only x_1, x_2 and x_3 s are the real numbers greater than zero and all the other values in A are zero. As, for example, $x_1 \neq 0$, this particularly signifies that $du_{l-1} \simeq dv_{k-1}$ and $du_{l+1} \simeq dv_{k+1}$ and also $(du_{l-1}, du_{l+1}) \simeq (dv_{k-1}, dv_{k+1})$ according to the node and edge similarity defined before. Now it can be noted that if two nodes u, v in a graph $G = (V, E)$ are connected with more than one walks, they supposed to have more and more random walks of different lengths which should be reflected in the matrix A . Following this explanation it is to be mentioned that similar nodes in DG_M and DG_I should be connected with different walks in the product graph G_P . So the non zero entry in the combined weighted matrix A identifies the similar nodes in DG_I with DG_M and with this the occurrences of the graph DG_M can be found in DG_I . The similar pair of nodes in the redundant dual graphs should be connected in the matrix A as above and the dissimilar pair of nodes should get the zero entries, as in the exponentiation of the adjacency matrix E_P , the existence of solitary graph edge must be diminished. The connected sub-component in the matrix A (in this case component with non zero entries viz. x_1, x_2 and x_3) can be found by searching maximal subgroup of entries that are mutually reachable (`graphconncomp` function in the `matlab`). Each component is then regarded as a single instance of the pattern graph in the target graph and can be found according to the position of the second nodes of the vertices of A .

3 Experimental Results

Our experiments were conducted on the SESYD dataset² [2]. This dataset contains 10 different subsets and 16 query symbols (Fig. 8). Each of the subsets

² <http://mathieu.delalandre.free.fr/projects/sesyd/symbols/floorplans.html>

contains 100 synthetically generated floorplans. All the floorplans in a subset are created from the same floorplan template by putting different model symbols in different places in random orientation and scale. In this experiment we have only considered a subset of 300 images (floorplans16-01, floorplans16-05 and floorplans16-06) and all the query symbols. For each retrieved instance of a given symbol if there exist an overlapping with the ground truth of the same symbol, we compute the overlapping ratio as follows:

$$\text{overlapping ratio} = \frac{\text{area}(A) \cap \text{area}(B)}{\text{area}(A) \cup \text{area}(B)}$$

where A is the area of the retrieved symbol and B is the area of the corresponding symbol in the ground truth. A retrieval is classified as true positive if the overlapping ratio is greater than 0.5.

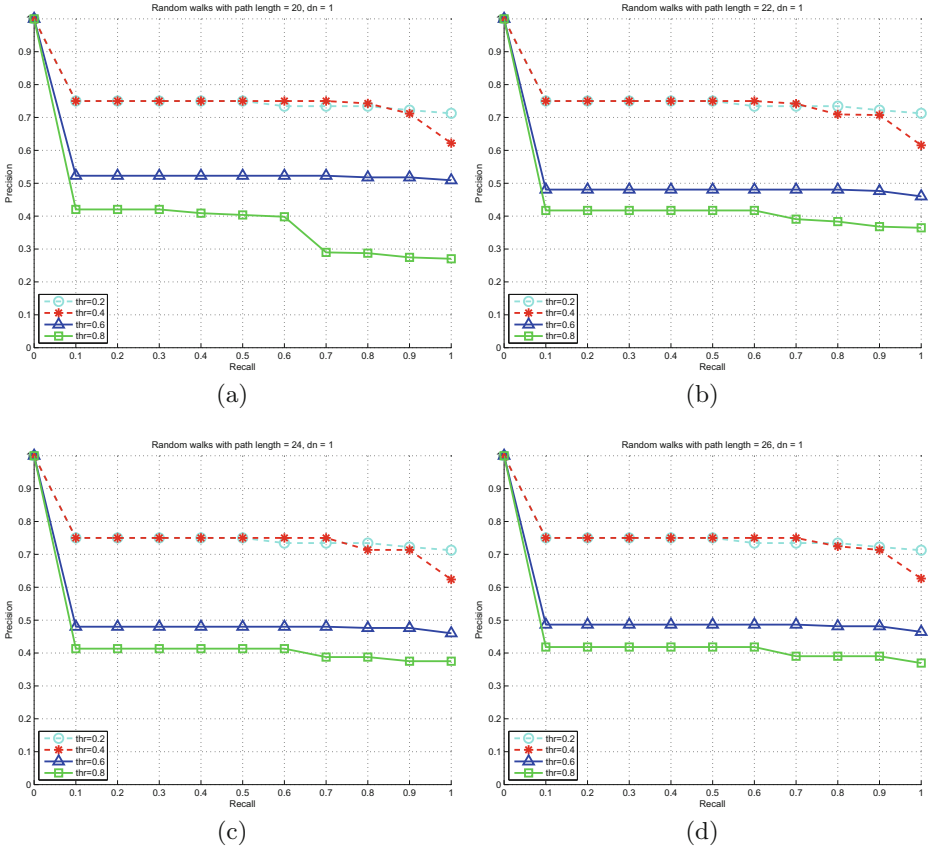


Fig. 7. Precision Recall curves for path length (a) 20, (b) 22, (c) 24, (d) 26.

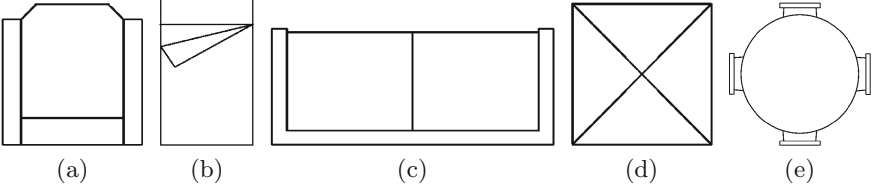


Fig. 8. Examples of model symbols: (a) *armchair*, (b) *bed*, (c) *sofa2*, (d) *table1*, (e) *table3*.

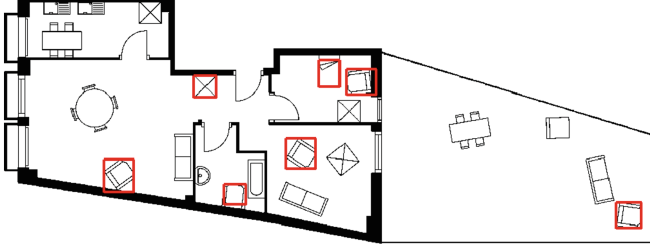


Fig. 9. Qualitative results of spotting *armchair* which shows correct detection of all the occurrences of *armchair*, at the same time it includes two false detection of *bed* and *table1* respectively.

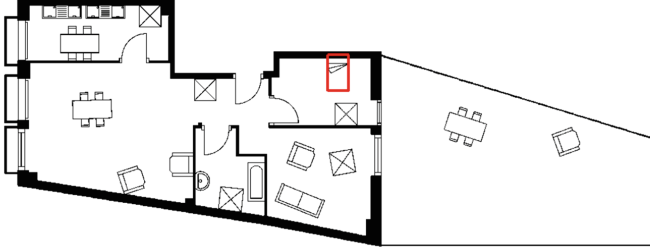


Fig. 10. Qualitative results of spotting *bed* which shows correct detection of the only one occurrence of *bed*, note there is no false detection.

All the experiments are done with the parameter values set as: $t_{\beta_1} = 6$, $t_{\beta_2} = 0.2$, we have observed that these two parameters are quite stable through all the images. However the other parameter t_{α} plays an important role selecting the compatible nodes while computing the product graph. For that reason we have performed a detailed experiments varying the parameter t_{α} from 0.2 to 0.8 with the step 0.2. Also we consider the length of the paths used as node labels as another parameter and for that we have performed experiments varying the path length from 20 to 26 with the step 2. All the experiments are performed by setting the connectivity parameter $dn = 2$.

For each of the pattern graphs we perform our product graph based subgraph matching method for spotting symbols on the documents and as an output we

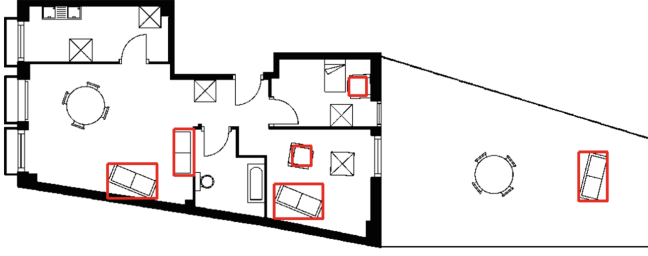


Fig. 11. Qualitative results of spotting *sofa2* which shows correct detection of all the occurrences of *sofa2* and also it false detection of two instances of *armchair*. This is because of the square regions in *armchair* resemble with the square region in *sofa2*.

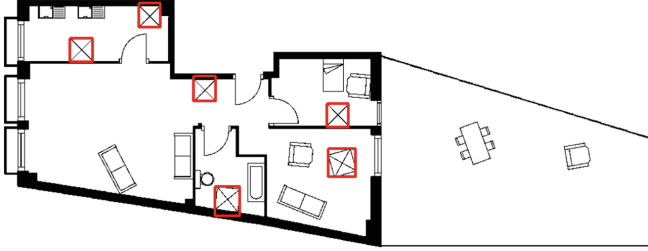


Fig. 12. Qualitative results of spotting *table1* which shows all the correct detection.

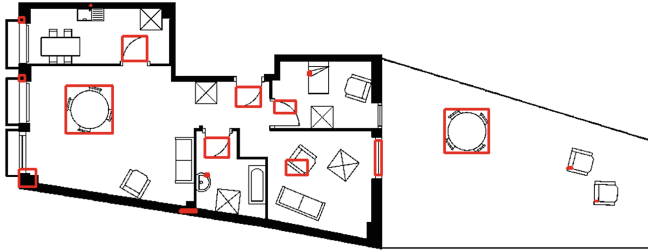


Fig. 13. Qualitative results of spotting *table3* which shows all the correct detection but a lot of false positives most of them resemble with the smaller subpart of the symbol *table3*.

obtain a ranked list of retrieved zones supposed to contain the queried graph. To evaluate the ranked list of retrievals we have drawn the precision recall curves for each set of parameters for each path length and they are shown in Fig. 7, where Fig. 7a shows the precision recall curves for path length 20, Fig. 7b for path length 22 and so on. From the plots, it is clear that the length of the paths used for node labels do not effect so much the overall performance of the system. It sometime can affect the bigger symbols where small paths can not connect the extremities of a dual node but in our case (or dataset) the range of path lengths considered were enough to handle all category of symbols. The parameter t_α has

a good influence on the performance of the system. It is clear from the plots that the performance of the system drops while increasing t_α , this is because larger values of t_α create lot of nodes in the product graph and hence increase the number of false positives. Moreover, increasing this parameter increase the computation time, this is also due to the increment of the number of nodes in the product graph.

For an overall performance evaluation we have computed the precision (**P**), recall (**R**) and F-measure (**F**). The present system with the best set of parameters has obtained **P** = 78.20 %, **R** = 81.43 %, **F** = 76.49 %, which is quite better than the method in [3]. Some qualitative results are shown in Figs. 9, 10, 11, 12 and 13³.

4 Conclusions

In this paper we have extended the product graph methodology by using the dual graph rather than the original representation as a basis. It turns out that this dual graph representation with redundant edges provides a robust way to deal with noise and distortions in the structural information. The product graph exhibits similar walks and exponentiation of the product graph's adjacency matrix allows one to extract simultaneous similar walks of any desired length. So these walks similarities help to get similar set of nodes in the pattern and target graph. The precision recall values obtained by the method are very encouraging albeit we experienced false positives with larger values of t_α . A closer analysis reveals that this kind of false positives are generated due to the tottering between the nodes in the product graph [9]. So one direction of our future work will address the removal of totterings from exponentiated adjacency matrices [1, 10]. Another possible improvement can address the exact nature of matching the nodes while computing the product graph, which needs a threshold. Finding such a threshold is often difficult and heuristic. A possible improvement can come from working with similarities on the full product graph, which needs further research activities. One more direction of future work will concern detailed experimental study on various datasets to show the efficiency of the graph matching algorithm.

Acknowledgements. This work has been partially supported by the Spanish projects TIN2009-14633-C03-03, TIN2011-24631, TIN2012-37475-C02-02, 2010-CONE3-00029 and the PhD scholarship 2013FLB2 00074.

References

1. Aziz, F., Wilson, R.C., Hancock, E.R.: Backtrackless walks on a graph. IEEE Trans. Neural Networks Learn. Syst. (TNNLS) **24**(6), 977–989 (2013)

³ For all the qualitative results the interested readers are referred to http://www.cvc.uab.es/~adutta/Research/ProductGraph/res_rw.php.

2. Delalandre, M., Pridmore, T.P., Valveny, E., Locteau, H., Trupin, É.: Building synthetic graphical documents for performance evaluation. In: Liu, W., Lladós, J., Ogier, J.-M. (eds.) GREC 2007. LNCS, vol. 5046, pp. 288–298. Springer, Heidelberg (2008)
3. Dutta, A., Gibert, J., Lladós, J., Bunke, H., Pal, U.: Combination of product graph and random walk kernel for symbol spotting in graphical documents. In: Proceedings of the International Conference of Pattern Recognition (ICPR), pp. 1663–1666 (2012)
4. Dutta, A., Lladós, J., Pal, U.: A symbol spotting approach in graphical documents by hashing serialized graphs. *Pattern Recogn. (PR)* **46**(3), 752–768 (2013)
5. Fischer, A., Suen, C.Y., Frinken, V., Riesen, K., Bunke, H.: A fast matching algorithm for graph-based handwriting recognition. In: Kropatsch, W.G., Artner, N.M., Haxhimusa, Y., Jiang, X. (eds.) GbRPR 2013. LNCS, vol. 7877, pp. 194–203. Springer, Heidelberg (2013)
6. Gärtner, T., Flach, P.A., Wrobel, S.: On graph kernels: hardness results and efficient alternatives. In: Schölkopf, B., Warmuth, M.K. (eds.) COLT/Kernel 2003. LNCS (LNAI), vol. 2777, pp. 129–143. Springer, Heidelberg (2003)
7. Ming-Kuei, H.: Visual pattern recognition by moment invariants. *IRE Trans. Inf. Theor.* **8**(2), 179–187 (1962)
8. Lladós, J., Valveny, E., Sánchez, G., Martí, E.: Symbol recognition: current advances and perspectives. In: Blostein, D., Kwon, Y.-B. (eds.) GREC 2001. LNCS, vol. 2390, pp. 104–128. Springer, Heidelberg (2002)
9. Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., Vert, J.-P.: Graph kernels for molecular structure-activity relationship analysis with support vector machines. *J. Chem. Inf. Model. (JCIM)* **45**(4), 939–951 (2005)
10. Stark, H.M., Terras, A.A.: Zeta functions of finite graphs and coverings. *J. Adv. Math. (JAM)* **121**(1), 124–165 (1996)

Graphics Recognition. Current Trends and Challenges
10th International Workshop, GREC 2013, Bethlehem,
PA, USA, August 20-21, 2013, Revised Selected Papers
Lamiroy, B.; Ogier, J.-M. (Eds.)
2014, XII, 267 p. 162 illus., Softcover
ISBN: 978-3-662-44853-3