数据机构课程设计报告书

姓名：杨家玺　学号：U201717007 班级：软工 1703

一、概要

图是一种重要且功能极其强大的数据结构，我们利用图上的节点与边来呈现对象与对象间的关系。通常来说，现实世界中的问题大都可以转化成图上的问题，搜索问题或是匹配问题。图的同构是一类模式匹配算法，现已被广泛应用于图像处理、蛋白质结构分析、信息网络的构建、化学键研究、社交领域，等等。图的同构算法在各种领域体现出了其具有的独特魅力与无与伦比的价值。

关键字：图的同构、子图同构、 NP 、Ullman、VF2、Nauty

二、文章结构

本文将粗略描述几种领先且广泛使用的图同构算法，并给出其中几个算法的大致实现，然后在代码基础上进行测试。其次，我会引援现有的评测论文中的结果来呈现各种不同算法在不同类型的图上的表现。最后，我将简要分析图的同构与子图同构算法在现实生活中的应用。

三、问题描述：图的同构

对于两个图 $G=<V_1,E_1>$ 与 $H=<V_2,E_2>$，如果节点集合 V_1 与 V_2 之间存在一个双射函数 f，使得对于 V_1 中的两个点 u 与 v，当且仅当 $\left(f(u),f(v)\right)\in E_2$ 时，有 $(u,v)\in E_1$，此时称双射函数 f 描绘了一个 $\mathrm{G}\leftrightarrow\mathrm{H}$ 的同构。

换句话说，同构描述了一种保属性序的映射关系，它将一个点集一对一地映射到另一个点集，并且维持了两个集合中点与点之间的关系，这种点与点之间的关系，暂且将其称之为点的属性。

所以，从“点对匹配”的角度，图同构的判定问题可以描述为，对于两图 $G=<V_1,E_1>$ 与 $H=<V_2,E_2>$，是否存在一个对 G 中节点 V_1 的一个排列 $G_{[n]}$，使得重排列后的图 G' 与图 H 完全一致。基于这种朴素的思想，我们给出以下第一种算法。

四、判断图同构的算法

1. 基于生成全排列序列的算法

从上文的分析中我们可以得知，两个图的同构判定可以简单地分析为恰当全排列的获取，因此，我们给出以下算法：

过程：基于全排列的图同构判定算法

输入：图$G=<V_1,E_1>$，$H=<V_2,E_2>$

开始：

对于图G的所有全排列$G_{[n]}$中的每一个元素G'：

判断G'是否与H完全相同：

是：返回（`yield`）当前排列

否：继续

如果没有满足条件的排列被返回，则认为不同构

结束

2. 两种基于深度优先搜索与根据局部匹配进行剪枝的算法

对于图的同构问题，我们除了想知道两个图是否是同构的，更想知道两图是怎样同构的，也就是说，我们希望得到f，那个标记$G \leftrightarrow H$的双射函数。

（1） An Algorithm for Subgraph Isomophism, J.R.Ullman[5]

该算法由 Jeffrey D. Ullman 于 1976 年提出，发表于期刊 JACM。论文提出的算法明确使用了“回溯”的概念，但限于当年计算机的限制，Ullman 先生在论文中给出的伪代码并没有很好地体现这样的过程，于是我将会先讲述该算法的普遍过程，然后给出适合于“现代”算法的伪代码例程。

该算法由三部分组成，矩阵M的初始化、针对每一个可用列的深度优先搜索以及每一个深度优先搜索步的剪枝，下面我将概括该算法的核心思想。

该算法主要针对的是子图同构问题，即在母图（大图）$M(Map)$中能否找到与给出$Q(Query)$图同构的子图，但考虑到一个图是自身的完全子图，便可以直接拿来使用。

虽然本文讨论的是精确图匹配问题，但我仍会先阐述原论文的思想，因为我认为具有一致普适性的算法更能体验其设计的巧妙所在。

对于在$M<V_1,E_1>$中寻找与小图$Q<V_2,E_2>$同构的子图，其结果可以构造成一个$\#V_2$行，$\#V_1$列的布尔矩阵，对于其中的(i,j)元素，当且仅当节点V_{2i}与节点V_{1j}构成同构映射时，值为 1。在此处我们可以看出，由于小图的节点数一定小于大图，故该布尔矩阵的行数一定小于列数，再根据该算法的匹配原则，最后得到的结果矩阵一定满足：每一行有且只有一个 1 ，且每一列最多有一个 1。

对于匹配矩阵M的初始化，论文给出了以下公式：

$$M^0=[m^0_{i,j}]$$

$$m^0_{i,j}\begin{cases}=1 & if\ \deg(i_Q)\le\deg(j_M)\\ =0 & otherwise\end{cases},$$

这一步本质上就是对搜索空间的剪枝，语义化描述为：“既然Q图的节点少边也少，那么能够成映射的点对，其在Q中的度数一定小于其像在M中的度数”。

在深度优先搜索步，“深度”是矩阵的行，也就是图Q的待匹配节点。在深度为d的层，算法对于所有尚未使用的列进行判断，在该行该列的M矩阵值为 1，则进入这个点，进行下一层的深度优先搜索，并在这一搜索子树遍历结束后，将该值还原为 0。

在每次遍历可用列向量之前，算法都会进行一次剪枝，剪枝的条件很简单：对于M矩阵中所有为 1 的元素对应的节点对(V_x^Q, V_y^M)，对于V_x^Q的每一个邻居$V_{N(x)}^Q$，如果不能在点V_y^M的所有邻居$V_{N(y)}^M$中找到一点使得$M[V_{N(x)}^Q, V_{N(y)}^M]=1$，则将$M[x,y]$置为 0，意为“将该枝剪掉以免未来进行无意义的搜索”。该过程的语义化描述也很显然：“如果算法想匹配两个节点，那么两个节点在各自图中的邻居也要满足匹配关系，否则这两个节点不可能被匹配。”

算法的基本数据结构叙述完了，算法的核心过程也呼之欲出。

过程：深度优先搜索步 *DFS*

输入：向量$\vec{C}$，图$M=<V_1,E_1>$，$Q=<V_2,E_2>$，搜索深度d，矩阵M

开始：

如果到达最后一层：

如果M呈现了一个同构关系：

返回结果

否则：

$M' \leftarrow$ 剪枝M

对于每一个未被使用的列C：

$\vec{C}' \leftarrow$ 标记C为已用

将M中该深度下其他位置置为 0

$CALL \quad DFS(\vec{C}', M, Q, d+1, M')$

恢复这一深度

如果始终没有找到满足条件的矩阵M，则认为不同构

结束

过程：Ullman 算法的初始化

输入：图 $M=<V_1,E_1>$，$Q=<V_2,E_2>$

开始：

$M_0 \leftarrow$ 根据规则初始化

$\vec{C} \leftarrow$ 标记列是否已用，布尔向量

结束

以上便是 Ullman 算法的大略表述，虽然该算法年代很久远，但由于其代码的低复杂度与实际表现的优异程度，它到现在仍被广泛使用。但是，我们必须要看到，由于其“剪枝步”所做的事情太少，能够切去的搜索空间并不能算大刀阔斧，所以我们应当寻求更好地解决方案，也就是更有效的状态空间表示与搜索空间压缩，于是便产生了 VF 算法[6]。

下面我将讲解 VF2 算法，是 VF 算法的改进版。

（2） **An Improved Algorithm for Matching Large Graphs, L. P. Cordella, P. Foggia, C. Sansone, M. Vento[7]**

VF2 算法本质上与 Ullman 算法并无太大差异，都是深度优先搜索配合剪枝以压缩搜索空间，但 VF2 算法的优胜之处便在于其严格且高效的剪枝条件。

论文提出了一种“状态空间表示”的技术，并把这种状态记做 s。状态 s 标记了一个部分的映射 $M(s)$，它只包含了完整映射函数 M 的一部分。

论文中给出的算法核心步骤如下：

```
PROCEDURE Match(s)
    INPUT:   an intermediate state s; the initial state s0 has M(s0)=∅
    OUTPUT: the mappings between the two graphs

    IF M(s) covers all the nodes of G2 THEN
      OUTPUT M(s)
    ELSE
      Compute the set P(s) of the pairs candidate for inclusion in M(s)
      FOREACH (n, m)∈P(s)
        IF F(s, n, m) THEN
          Compute the state s′ obtained by adding (n, m) to M(s)
          CALL Match(s′)
        END IF
      END FOREACH
       Restore data structures
    END IF
END PROCEDURE
```

具体来说，该算法在两个地方添加了约束，一是候选对 $P(s)$ 的选取，二是可行性函数 $F(s,n,m)$ 对候选对的判别。

对于候选对选取函数 $P(s)$，论文给出了对于有向图的选取方法，但考虑到我这次只实现了无向图的同构匹配，我将重新描述论文中的公式。

首先定义几个符号，$M(s)$ 是映射函数，考虑到这是双射与计算上的简洁性，这里使用了 $M_1(s)$ 与 $M_2(s)$ 分别表示 $M(s)$ 在两图的节点集 V_1 与 V_2 上的映射。$T(s)$ 在文中比作了节点的“候车室/车站（Terminal）”，比如 $T_1(s)$ 中的节点都不会出现在 $M_1(s)$ 中，但他们一定是 $M_1(s)$ 中某个节点的邻居。从语义上来理解，因为搜索是按照深度递增的顺序搜索下去的，节点的扩展具有连续性，从一个状态延伸出去的可行节点一定与该状态的节点相邻，这便是“候车室/车站”的意思。

将要被使用的符号都定义完全了，下面先讨论“候选节点对选择函数 $P(s)$ ”。

情况一：如果 $T_1(s)$ 与 $T_2(s)$ 都非空，则构造 $P(s)=T_1(s)\times\min\{T_2(s)\}$

情况二：如果 $T_1(s)$ 与 $T_2(s)$ 都为空，则构造 $P(s)=(N_1-M_1(s))\times\min\{N_2-M_2(s)\}$

情况三：如果 $T_1(s)$ 与 $T_2(s)$ 情况相反，则不存在可行的候选对，也同时说明当前的状态 s 不可能成为一个匹配，算法也就没有必要在这条路径上再搜索下去了。

现在让我们讨论一下可行性函数，本算法最精彩的部分。

可行性函数 $F(s,n,m)$ 评判了在当前的状态 s 下，加入一对由 $P(s)$ 产生的点对 (n,m) 会不会导致搜索无法继续向下进行。

其检查分为三个主要的部分，论文中指出，可行性函数会检查所有与节点 n 或 m 邻接的节点。尽早发现不匹配，尽早拒绝，如果所有条件被满足，则接受节点对，并加入当前状态。

第一部分，检查邻居的匹配，对于所有连接到节点 n 的节点 p ，如果 p 已经存在于部分映射 $M_1(s)$ 中，那么就去检查节点 m 有没有类似的邻居 q ，如果没有就拒绝这两个节点。然后反过来检查 m 到 n 。

第二部分，检查节点 n 在 $T_1(s)$ 中的邻居数是否等于节点 m 在 $T_2(s)$ 中的邻居数，不相等就拒绝这两个节点。

第三部分，检查节点 n 的所有邻居中既不在 $T_1(s)$ 也不在 $M_1(s)$ 的数量是否等于节点 m 的所有邻居中既不在 $T_2(s)$ 也不在 $M_2(s)$ 的数量，如果不相等就拒绝这两个节点。

如果运行到这里，节点对 (n,m) 仍没有被可行性函数拒绝，则认为这两个节

点的加入不会使搜索无法进行，于是接受这对节点，纳入状态 s 变为 s'。

VF2 算法除了严苛的剪枝条件外，促使其成功的重要原因还在于作者使用了高效的数据结构与精心优化的集合运算过程，使得计算几乎没有冗余。但考虑到这部分不作为本文的重点，便不再细谈。

至此，我们可以看出，VF2 算法要求所有被候选的节点对拥有完全对称的同构条件，正是因为这些条件的相互作用，VF2 算法才能快速地削减搜索空间，大幅度地提高匹配的效率。

3. 基于 canonical labelling（CL）算法的图同构匹配问题

对于 canonical labelling 的翻译，我并没有找到比较合适的措辞，但考虑到 canonical labelling 的本质也是一张图，同时也被叫做 canonical form，就姑且翻译为“图的规范形式”吧。

CL 算法最早由 McKay 在上世纪 80 年代提出，其中最重要的思想就是任何同构的两张图 $G=<V_1,E_1>$ 与 $H=<V_2,E_2>$，其规范形式 $C(G)$ 与 $C(H)$ 一定是两张完全一致的图。用数学语言表达便是 $G \simeq H \Leftrightarrow C(G)=C(H)$。所以图的同构判定问题就可以转化为两个规范形式的图的比较问题。

不得不承认，非常遗憾，由于临近期末，考试压力增大，我实在没有时间将 McKay 的 Nauty[1][2][9]算法与 Adolfo 的 Traces[3]算法完全掌握，尤其是搜索树的建立与利用同构子图进行剪枝的过程，不同的论文甚至给出了不同的实现方法。但主要是因为个人数学能力过于薄弱，对于 McKay 亲著论文 Practical graph isomorphism, II 无力接受，因为其中充满了看似熟悉的离散数学知识，不能进行深究实属遗憾。

五、算法实现

我实现了本文中所提到的前三种算法：全排列、Ullman，VF2。在实现 VF2 算法时，我从 networkx 库汲取了大量编程经验与技巧，在此感谢一切向开源事业做出无私奉献的开发者。

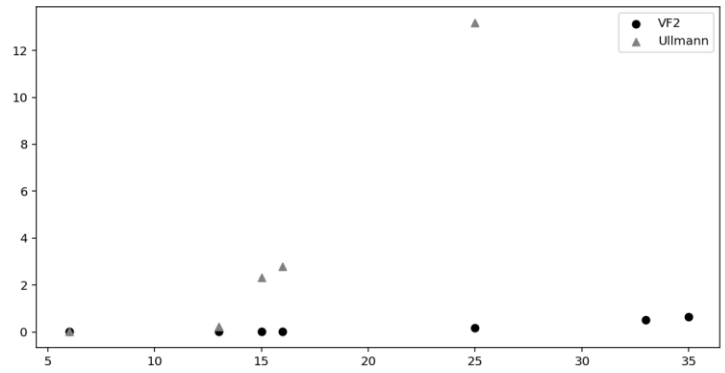
实现代码我将不会贴在文中，因为这只关乎实现而非算法本身。本次实验我使用了 Python 语言，主要是因为其简洁与灵活。值得一提的是，Python 的内置集合数据结构（set）为 VF2 算法的实现提供了极大地便利，由于并集、差集等集合常用运算都被重载为了具有语义性的数学运算符，代码的可读性与简洁性都有了质的飞跃。

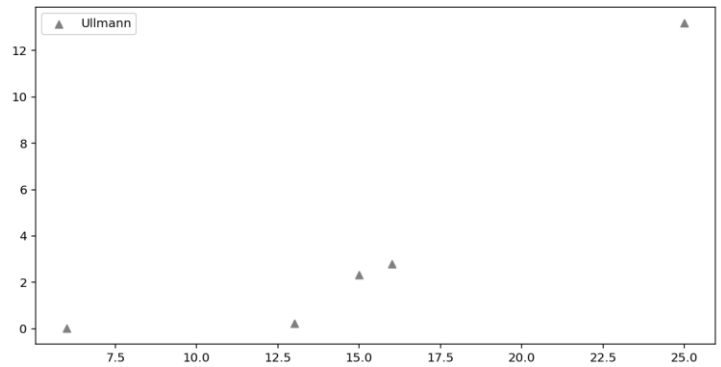
六、算法测试

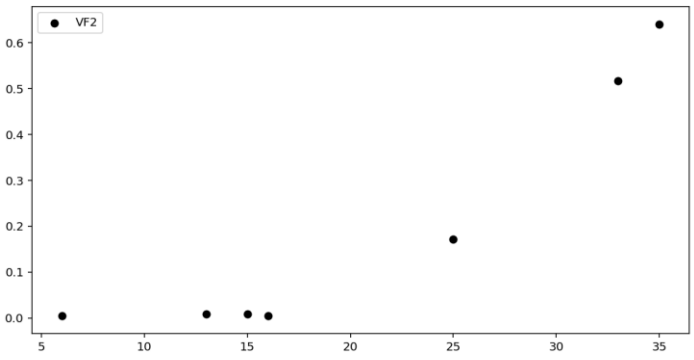
我使用的测试图来自 pynauty 库中的 test 文件，其中包含了若干张节点数小于 40 的无向图。对于同构的测试，我采用以下方法：对于一张图 G，生成一个其节点的排列 $[n]$，使用这个新排列对原图进行重新打标生成新图 $G_{[n]}$，记做 H，此时对图 G 与图 H 就可进行算法上的测试。

由于全排列算法实际意义不大，其执行效率也是低得惊人，在节点数大于 10 时就已然放弃了挣扎。所以本部分我将对 Ullman 算法与 VF2 算法进行运行时间上的比较。但考虑到本人的编码水平问题，并不能保证两个算法的差异真正是由

算法本身引起的，所以本部分主要观察算法趋势。

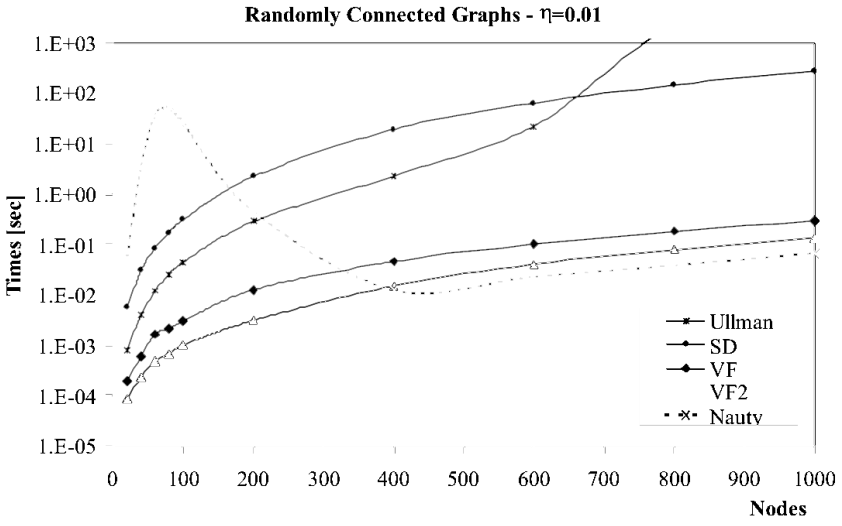


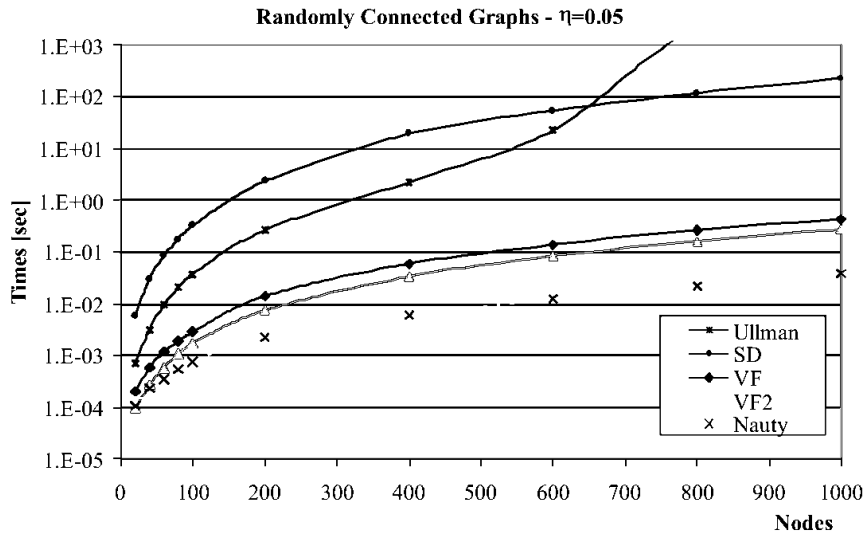




从图中可以明显看出，两种算法在节点数增大时，所消耗时间的增长幅度更为剧烈，我将在下一部分讨论图同构问题的时间复杂度与其 NP 性质。

以下来自论文 A Performance Comparison of Five Algorithms for Graph Isomorphism[8]的一些测试结果





从上述结果可看出 Nauty 算法在一定程度上是同构问题判定的最佳算法。

七、时间复杂度与 NP

McKay 在论文中提到，$GI \in NP$，但不能判断是否$GI \in co-NP$。这意味着我们目前还不能找到一个多项式时间复杂度的算法，但我们也不知道GI是否是NP完备性问题。目前理论证明中最低的时间复杂度界是由 László Babai 于 2015 年提出的[4]，指出了$e^{O(\sqrt{n\log n})}$的时间界。Harald Andrés Helfgott 很快便指出 Babai 论文中的一处错误，Babai 也很快修正了该错误并更新了论文。

八、图同构算法的实际应用

1. 计算机视觉与模式识别

图的匹配技术已经在模式识别领域取得了长足的进展，具体而言其渗透到以下几个方面：图像分析与处理、文本一致性的鉴定、图像的匹配与提取等。对于

计算机视觉领域，现阶段，许多机器人的目标检测部分都会包含图的匹配，从深海探测到太空遥感识别，图的匹配技术在计算机视觉领域有着十分成功的应用，其还渗透到了医学中有害细胞的识别，例如癌细胞的识别，与场景分析领域等。

对于实际事物应用匹配算法时，通常会将事物利用图来进行表述。我们知道图主要在呈现一种具有组织性的关系，而这种关系体现在具体事物上，便是同类事物具有的方方面面的属性。这一理论不仅适合于图像的匹配，各种事物都可以用来匹配，因为联系是必然的、内在的、多样化的，只要能提取特征，就能进行匹配。

2. 蛋白质结构的研究

在蛋白质研究领域，人们往往好奇具有某种特定结构的蛋白质分子或蛋白质分子簇会不会始终如一地具有某些性质，在进行大量研究后人们找到了一些能体现特定功能的特殊基团。而在研究新的蛋白质分子时，就可以利用自图同构算法进行部分匹配，以利用先验经验对未知蛋白质分子进行学习和探索。

3. 化合物分子的识别

对于化合物的结构，必须有一种唯一且无歧义的标签对各种化合物进行标记。以前人们使用树结构来进行这方面的工作，随着数学尤其是群论的发展，一类专门研究化合物结构性质的学科诞生，并推动了分子结构表示从树形图发展到了平面图。对于分子结构的特殊性质，再得益于各种图同构匹配算法的蓬勃发展，化合物分子的匹配可以在有限的时间内高效地完成。

4. 社交网络

如何检测人与人之间的相似性，利用他们的社交网络可以轻松完成这一任务。人所进行的关系建立是有目的性、有目标性的，所以对于拥有不同属性的人来说，他们的社交网络图会呈现出不同的结构。理论上认为，在某些方面相似的两个人，他们的社交网络图会呈现出某些相似的特征。

九、小结

图的同构算法是一类古老且经典的问题，虽然其被认为是一类 *NP* 的问题，但这并不妨碍人们对高效匹配算法的追求。

本次课程设计使我初步了解了图的同构问题，看到了图同构算法的发展历程与各种侧重点不同的算法。使我印象最深刻的一篇论文便是 McKay 对于 Nauty 算法的原论文，我被其严谨的数学论证过程所震撼，也深刻意识到自己要学的东西还有很多。

软件并不是单纯的算法与数据结构的堆叠，简单的堆叠是没有灵魂的，它更应该是适应实际问题的最佳匹配。同理，算法也不是对于前人智慧结晶的简单背诵，更应该是对其内在缘由的深度把握。知道、会用一个新算法没什么了不起，真正厉害的，是通过研究算法与算法被研究出来的过程，将前人的思维方式纳入自己的技术栈。熟练掌握的算法与数据结构是一个程序员的硬实力，而对于体系与性能的精当把控就完全体现了一个人的技术软实力。从长远来看，软实力的积累是至关重要且应当长久不断的一环。

十、参考文献

[1] B.D. McKay, Practical Graph Isomorphism, Congressus Numerantium, 30, pp. 45-87,1981.

[2] B. D. McKay and A. Piperno. Practical graph isomorphism, II. J. Symb. Comput., 60:94–112, 2014.

[3] Piperno, A. Search Space Contraction in Canonical Labeling of Graphs. 26 Jan 2011.

[4] L. Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In STOC, pages 684–697. ACM, 2016.

[5] J.R. Ullmann, An Algorithm for Subgraph Isomorphism, Journal of the Association for Computing Machinery, vol. 23, pp. 31-42, 1976.

[6] L.P. Cordella, P. Foggia, C. Sansone, M. Vento, Evaluating Performance of the VF Graph Matching Algorithm, Proc. of the 10th International Conference on Image Analysis and Processing, IEEE Computer Society Press, pp. 1172-1177, 1999

[7] L. P. Cordella, P. Foggia, C. Sansone, M. Vento, An improved algorithm for matching large graphs, Proc. of the 3rd IAPR TC-15 Workshop on Graphbased Representations in Pattern Recognition (2001), pp. 149-159

[8] P. Foggia, C. Sansone, M. Vento, A performance comparison of five algorithms for graph isomorphism, Proc. of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition (2001), pp. 188-199

[9] B. D. McKay and A. Piperno. nautytraces software distribution web page. http://cs.anu.edu.au/~bdm/nauty/ and http://pallini.di.uniroma1.it