To be or not to be Yutsis: algorithms for the decision problem

D. Van Dyck, G. Brinkmann, V. Fack

Ghent University, Department of Applied Mathematics & Computer Science, Research Group Combinatorial Algorithms and Algorithmic Graph Theory, Krijgslaan 281-S9, 9000 Ghent, Belgium

B.D. McKay

Department of Computer Science, Australian National University, ACT 0200, Australia

Abstract

Generalized recoupling coefficients or $3nj$-coefficients can be expressed as multiple sums over products of Racah or $6j$-coefficients [1]. The problem of finding an optimal summation formula (i.e. with a minimal number of Racah coefficients) for a given $3nj$-coefficient is equivalent to finding an optimal reduction of a so-called Yutsis graph [2].

In terms of graph theory Yutsis graphs are connected simple cubic graphs which can be partitioned into two vertex induced trees. The two parts are necessarily of the same size. In this area Yutsis graphs are also studied under the name of cubic dual Hamiltonian graphs [3]. We present algorithms for determining whether a cubic graph is a Yutsis graph. This is interesting for generating large test cases for programs (as in [4], [5] or [6]) that determine a summation formula for a $3nj$-coefficient.

Moreover, we give the numbers of Yutsis and non-Yutsis cubic graphs with up to 30 vertices and cubic polyhedra with up to 38 vertices. All these numbers have been computed by two independent programs in order to reduce the probability of error. Since the decision problem whether a given cubic graph is Yutsis or not is NP-complete, we couldn't hope for a polynomial time worst case performance of our programs. Nevertheless the programs described in this article are very fast on average.

Key words: angular momentum; generalized recoupling coefficient; Yutsis graph; dual Hamiltonian graph; NP-complete; decision problem; heuristic.
PACS: 02.10.Eb, 02.20, 02.70, 03.65Fd, 31.15

1 Introduction

In various fields of theoretical physics the quantum mechanical description of many-particle processes often requires an explicit transformation of the angular momenta of the subsystems among different coupling schemes. Such transformations are described by *general recoupling coefficients* and arise mostly in atomic and nuclear structure and scattering calculations [1]. Several algorithms have been described to generate a summation formula expressing the recoupling coefficient as a multiple sum over products of Wigner $6j$ symbols multiplied by phase factors and square root factors [4–10]. It is desirable to find an optimal summation formula, i.e. with a minimum number of summation variables and Wigner $6j$ symbols.

The best algorithms at present are based on techniques developed by Yutsis, Levinson and Vanagas [2] and manipulate a graphical representation of the recoupling coefficient called a Yutsis graph. Reduction rules are defined for these graphs, which allow a stepwise transformation of the graph by reduction and removal of cycles. Each reduction step contributes part of the final summation formula. Section 2 summarizes some notions from the quantum theory of angular momenta, showing how a Yutsis graph is constructed for a given recoupling coefficient and which reduction rules can be used. For the general theory of Yutsis graphs we refer to [1] and [2].

For our purposes a Yutsis graph can be defined as follows. A binary coupling tree on $n + 1$ leaves is an unordered binary tree in which each leaf has a distinct label. By taking two binary coupling trees on $n+1$ leaves in which the unique leaf vertices with the same label are identified and then removed and where the root nodes are connected by an additional edge, we obtain a cubic multigraph with $2n$ nodes and $3n$ edges. In this multigraph the internal vertices of the coupling trees define two vertex induced trees and the former leaf and root edges form an edge-cut on $n + 2$ edges. Figure 1 shows an example. A multigraph that can be constructed this way is called a *Yutsis graph* or simply *Yutsis*. Two vertex induced trees coming from such a construction are called the *defining trees* and the edge-cut is called the defining cut. Note that a given Yutsis graph can in general be obtained from more than one pair of trees, so the defining trees and the defining edge-cut are in general not uniquely determined. Since the two endpoints of multiple edges in a Yutsis graph must obviously belong to different trees, they are trivial from the viewpoint of the decision problem and we will restrict ourselves to simple graphs when discussing the decision problem.

Email addresses: Dries.VanDyck@pandora.be (D. Van Dyck,), Gunnar.Brinkmann@UGent.be (G. Brinkmann,), Veerle.Fack@UGent.be (V. Fack), bdm@cs.anu.edu.au (B.D. McKay).

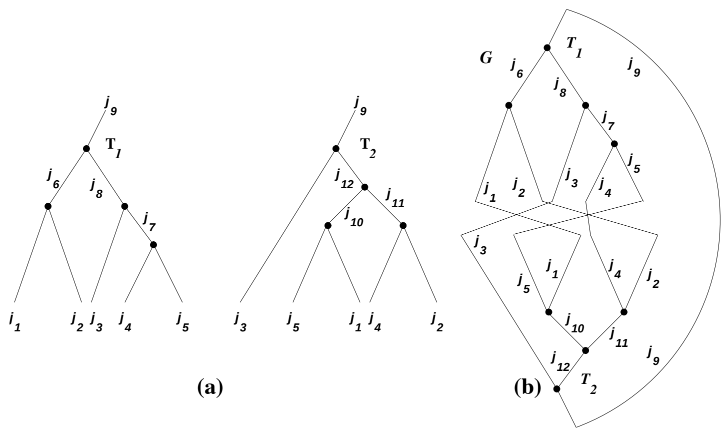

Fig. 1. (a) Two binary coupling schemes, and (b) the corresponding Yutsis graph

In mathematics, Yutsis graphs are also known as *dual Hamiltonian cubic graphs* [3].

Up to now no better method is known to determine whether a cubic graph is Yutsis than to search for a defining tree (or cut). For the quantum theory of angular momenta, we are interested in obtaining large test cases by generating large cubic graphs at random and filter out those graphs which are not Yutsis. In addition we would like to identify the non-Yutsis graphs and study their structure.

All graphs in this article are assumed to be connected.

2 Graphical representation of recoupling coefficients

In [1, Topic 12], recoupling theory is considered from the point of view of *binary coupling schemes.* A binary coupling scheme is the rooted binary tree representing the order of coupling of a state vector in the tensor product of $n+1$ angular momentum multiplets, labelled respectively by the angular momenta $j_1, j_2, \ldots, j_{n+1}$. The leaves of the binary tree are labelled by these angular momenta $j_1, j_2, \ldots, j_{n+1}$, and the remaining vertices of the tree can be labelled by the intermediate angular momenta. For example, the following vector can be considered :

$$|\, ((j_1, j_2)j_6, (j_3, (j_4, j_5)j_7)j_8)j_9 \,\rangle \,,$$

which corresponds to the left-side tree in Figure 1(a).

There are obviously several ways in which $n+1$ angular momenta can be coupled, and the quantities that typically appear in atomic and nuclear structure computations are the related general recoupling coefficients or $3nj$-coefficients.

A general recoupling coefficient (or a generalized $3nj$-coefficient) is defined to be the transformation coefficient between any such two coupling schemes, e.g.

$$\langle\, (j_1, j_2)j_6, ((j_3, (j_4, j_5)j_7)j_8)j_9 \mid (j_3, ((j_5, j_1)j_{10}, (j_4, j_2)j_{11}))j_9 \,\rangle \,. \qquad (1)$$

It is a fundamental theorem of recoupling theory [1, p. 455] that each such transformation coefficient (i.e. every generalized $3nj$-coefficient) can be expressed in terms of sums over products of Racah coefficients ($6j$-coefficients).

A famous program of Burke [9], `NJSYM`, already deals with this problem. Burke's approach is equivalent to finding a certain path between the two binary coupling schemes (representing the bra- and ket-part of the general recoupling coefficient) by successive elementary transformations on the trees. The shorter this path, the better the resulting formula. The path found by `NJSYM` is generally rather long, thus yielding expressions which are far from optimal. In order to improve `NJSYM`, other algorithms have been developed which implement graphical methods due to Yutsis, Levinson and Vanagas [2].

Consider a general recoupling coefficient of $n+1$ integer and half integer angular momenta. With each label in the recoupling coefficient an edge in the graph is associated and with each coupling a node is associated, resulting in a cubic graph with $2n$ nodes and $3n$ edges. The nodes representing the coupling of the left-hand side of the recoupling coefficient get a '$-$'-sign, those on the right-hand side get a '$+$'-sign. The edges corresponding to the compounded angular momenta on the left-hand side are directed away from the node while the edges representing the resultant are directed towards the node. The direction of the edges corresponding to the left-hand side are the reverse of those corresponding to the right-hand side. The Yutsis graph shown in Figure 1(b) corresponds to the recoupling coefficient in equation (1) (where signs and directions are omitted).

The sign of a node where angular momenta j_1, j_2 and j_3 meet can be inverted by multiplying the value the graph represents by $(-1)^{j_1+j_2+j_3}$. A change of direction of an edge with label j results in a multiplication by $(-1)^{2j}$. The transformation coefficient then equals the j coefficient represented by the diagram multiplied by (see [2], equations (22.1) and (22.2)):

$$(-1)^{2(J+\sum_{i=1}^{n-1} b_i+S)}[\prod_{i=1}^{n-1}(2a_i+1)(2b_i+1)]^{1/2} \,,$$

with S the sum of all 'first' coupled angular momenta, $n+1$ the number of angular momenta, a_i the intermediate angular momenta on the left side, b_i the intermediate angular momenta on the right side, and J the total angular momentum.

Once the graph is generated, it can be simplified with the help of the reduction rules developed by Yutsis, Levinson and Vanagas [2]. Using these rules the reduction algorithms iteratively eliminate cycles from the Yutsis graph, until the graph is simplified to a so-called “triangular delta”, i.e. a graph consisting of two nodes connected by three parallel edges. Several algorithms based on this approach have been developed [4–8,10,11].

From an algorithmic point of view one is interested in the complexity of the formula. Since the difference between the number of summations and the number of $6j$ coefficients is constant, the number of $6j$ coefficients suffices as measure for the complexity of the formula. With this idea in mind the signs of the nodes and the direction of the edges can be neglected, since they only contribute in phase and weight factors, not influencing the complexity of the generated summation formula.

3 The decision problem is NP-Complete

In this section we will prove that the problem of deciding whether a given graph is Yutsis or not is a very hard problem in the worst case. To be exact: The problem is NP-complete and it is even NP-complete when restricted to the subclass of cubic polyhedra, i.e. 3-connected planar cubic graphs.

Theorem 1 *The decision problem whether a given cubic polyhedron is Yutsis or not is NP-complete.*

Proof

Since it is easy to see that this problem is in NP (take e.g. n random vertices and check whether they and their complement each induce a tree), the result is a direct consequence of results of Jaeger, Chvátal and Wigderson:

In [3] Jaeger defines a graph G to be dual hamiltonian if it has an elementary cut on $|E| - |V| + 2$ edges. He mentions that the planar dual of a dual hamiltonian graph is hamiltonian, but also the reverse is well known and easy to prove. He also proves that a graph G is dual hamiltonian if and only if it has a partitioning of V as $T_1 \cup T_2$ such that T_1 and T_2 induce a tree. This means that a cubic polyhedron is Yutsis if and only if its dual triangulation is hamiltonian.

Chvátal and Wigderson proved, independently of each other, that determining the hamiltonicity of a graph is NP-complete, even when restricted to triangulations [12,13].

Since the computation of the dual can easily be done in polynomial time, combining these results proves the theorem.

4 Preliminaries

In this section we will give and prove some lemmas and remarks that we will use in the algorithm. $G = (V, E)$ always denotes a simple cubic graph with $2n$ nodes and $3n$ edges.

Lemma 2 *Let G be a cubic graph with $2n$ nodes, T an induced subgraph on n vertices that is a tree and S the complement of T.*

Then S, T are defining trees for a Yutsis decomposition if and only if S is connected.

Proof

If T is an induced tree on n vertices in a cubic graph, then there are $3n - 2(n-1) = n+2$ edges between T and its complement S. Therefore, there are $(3n - (n+2))/2 = n-1$ internal edges in S which gives that S is a tree if and only if it is connected.

Lemma 3 *Let G be a cubic graph with $2n$ nodes, T' an induced connected subgraph and S' the complement of T'.*

If S' is not connected, then there are no defining trees T, S of a Yutsis decomposition so that $T' \subseteq T$.

Proof

Suppose that there is such a decomposition. Since T is a tree and T' is a connected subgraph of T, each component of $T \setminus T'$ is a tree and contains a vertex which is an endvertex of T. Each such endvertex is adjacent to S, which implies that S' is connected contrary to hypothesis.

The following remark is trivial to prove, but we want to mention it nevertheless in order to be able to refer to it later on.

Remark 4 *A vertex of degree 2 is a cutvertex in a graph if and only if it is not contained in a cycle.*

In planar graphs we will use the embedding of the graph to speed up the algorithm. Note that in a two-connected plane graph every edge is in the boundary of two different faces. This gives the following remark and definition:

Remark 5 *In a cubic 2-connected graph $G = (V, E)$ embedded in the plane, for every pair e, e' of edges sharing a vertex, there is exactly one face $f_G(e, e')$ so that both e and e' are in its boundary.*

This remark enables us to formulate the following lemma we use in the algorithm:

Lemma 6 *Given a cubic 2-connected graph $G = (V, E)$ embedded in the plane, a subgraph $T = (V_T, E_T)$ that is a tree and a vertex $v \notin V_T$ that has one neighbour in T and 2 neighbours v', v'' in $V \setminus V_T$. Let $e = \{v, v'\}, e' = \{v, v''\}$.*

The vertex v is a cutvertex in the graph T^c induced by $V \setminus V_T$, if and only if the boundary of $f_G(e, e')$ contains a tree vertex.

Proof

First suppose that $f_G(e, e')$ does not contain a tree vertex. Then the boundary is a cycle in $V \setminus V_T$, so v is contained in a cycle and therefore no cutvertex due to Remark 4.

Now suppose that $f_G(e, e')$ does contain a tree vertex t and let t' denote the tree neighbour of v. Then there is a path from t to t' in T and adding the edge $\{t', v\}$ to it we have a path between two vertices of the boundary of $f_G(e, e')$. Connecting the endvertices through the interior of $f_G(e, e')$ we get a Jordan curve with the two non-tree neighbours of v in different components. So they are also in different components of $T^c - \{v\}$ while they are in the same component of T^c. So v is a cutvertex of T^c.

We also use the following easy criterion to determine that some graphs are in fact not Yutsis graphs. As the tables show, this criterion doesn't eliminate too many graphs, but all computations necessary to apply it are also used to determine a good starting vertex for the exhaustive search for a tree decomposition, so applying it is practically for free:

Lemma 7 *Let t denote the number of triangles in a cubic graph and f the number of vertices not contained in a triangle. If $t > f + 4$ then the graph is not a Yutsis graph.*

Proof

Note that in a Yutsis-decomposition in every triangle there is exactly one edge that belongs to one of the trees. This implies that all vertices in triangles have degree 1 or 2 in one of the trees and in every triangle there is at least one vertex with degree 1 in one of the trees. So the number of triangles is a lower bound for the numbers of leaves in two trees that form a Yutsis decomposition. If for a given decomposition tree T we let v_3 denote the number of vertices of T with degree 3 and b the number of leaves, then we have $v_3 = b - 2$ so for both trees together we have that there must be at least $t - 4$ vertices of degree 3 in the two trees which can – as noticed before – not belong to a triangle.

5 Fast heuristics – a greedy approach

In spite of the fact that this decision problem is NP-complete, in most cases a set of defining trees can be found very quickly by a heuristic we will now describe. Both of our filters work by first applying a heuristic a couple of times. Only in cases where the heuristics do not find a tree decomposition do we apply exhaustive search methods.

Given a connected cubic graph G, we start with a random vertex forming a one-vertex tree T_1 and a list L_1 of all its neighbours. We increase the tree vertex by vertex forming trees $T_2, T_3, \ldots, T_k$ and corresponding lists $L_2, L_3, \ldots, L_k$. In each step i, L_i is a list of all vertices in $V \setminus T_i$ which neighbour vertices in T_i. If we manage to build a tree T_n this way and the subgraph S induced by the remaining n vertices is connected, then we have proved that G is a Yutsis graph.

The tree T_{i+1} is formed by adding a vertex from L_i to T_i. We never add a vertex to the tree T_i that has two neighbours in T_i, as this would lead to a cycle in T_i and therefore also in T_n. Nor do we add a cutvertex of the graph induced by $V \setminus T_i$, because Lemma 3 applied to $T_i \cup \{v\}$ gives in that case that T_n cannot be a tree of a Yutsis decomposition. If we attempt to add a vertex but it is rejected by one of these conditions, we remove it from the list and choose a new vertex.

As long as there are vertices in the list, this process can continue – it must only stop when the list is empty. This is where the greedy aspect comes in: when choosing the next vertex to be added to T_i we always choose one for which the list grows the most – or in other words: a vertex v so that the number of *unplaced* neighbours of v – that is: neighbours that are not yet contained in L_i or the tree – is as large as possible.

In order to make this approach run quickly, it is necessary to be able to find the next vertex to add efficiently. That is, we must find those vertices in the list that have the largest number of unplaced neighbours very quickly and be able to check whether they are cutvertices of the complement or have two neighbours in the tree.

To this end we keep three lists L_i^j for every step i with $0 \leq j \leq 2$ and append a new vertex that has to be added to one of the lists to list L^j if it has j unplaced neighbours at step i. When there are two vertices with the same number j of unplaced neighbours, they are added to L^j in a random order.

When choosing the next vertex to add, we always choose the list L^j with j as large as possible so that L^j is nonempty and take the last vertex added to this list (so it is a kind of a depth-first or LIFO approach). If $j > 0$, this vertex is then tested for the number k of unplaced neighbours, which may have decreased since the vertex was added to the list. If $j \neq k$, so $k < j$, the vertex is placed in the list L_i^k, becoming the last vertex added to the list, and we choose again. Since the computation of the number of unplaced neighbours can obviously be done in constant time and every vertex is tested and moved at most 3 times, we have:

Remark 8 *The total number of steps necessary for choosing vertices in one run of the heuristic in a graph with $2n$ vertices is at most $O(n)$.*

The most time consuming part in this heuristic is the computation whether a vertex is a cutvertex in the complement of T_i. This computation takes time $O(n)$ and must be performed $O(n)$ times. Therefore we get a total running time of $O(n^2)$. Some techniques sped up the cutvertex testing routine considerably though not enough to guarantee a total of $O(n)$ steps in total per application of the heuristic in the general case:

Note that we only have to test vertices v that have degree 2 in the complement for being cutvertices in the complement, so due to Remark 4 we just have to find out whether v lies on a cycle. We do a simultaneous breadth-first search started at both neighbours of a vertex to be tested and stop as soon as a vertex is reached from both neighbours. If g is the size of a smallest cycle containing v, the routine will only visit vertices at distance at most $\lceil g/2 \rceil$ from v, so in case of small g this gives a sub-linear performance, but in case of v being a cutvertex, the whole complement still has to be searched. In addition we mark vertices that are known to not lie on a cycle (e.g. discovered in an earlier test) and perform our breadth-first searches on the graph with those vertices removed. Though speeding up the search considerably, this still doesn't lead to a sub-quadratic worst case performance.

A similar technique with three simultaneous breadth-first searches is also performed for the starting vertex. If it is found to be a cutvertex of G, then G is not Yutsis.

In case of plane graphs that come with an embedding, we keep a list of all $6n$ pairs of edges sharing a vertex. When the first vertex is added to the tree, all pairs of edges belonging to faces containing this vertex are marked. This can easily be done in time proportional to the number of pairs to be marked. Testing a vertex for being a cutvertex can now – due to Lemma 6 – be done in constant time: we just have to check whether the pair of edges that do not lead to tree neighbours is marked or not. Adding a new vertex to the tree we just have to mark all pairs of edges belonging to the face just checked, which again can be done in time proportional to the number of pairs marked. Since there is a linear number of pairs of edges and every pair is marked at most once we get a time consumption of $O(n)$ for all connectivity tests and markings done in one application of the heuristic. So for planar graphs we have a total running time of $O(n)$.

5.2 Discussion

Though the heuristic is extremely simple and can easily be implemented to run in time $O(n^2)$ per trial, $O(n)$ for plane graphs, the results are astonishingly good. We tested various variants of this approach by modifying the ways of choosing the next vertex to add – e.g. choosing vertices from the list completely at random or in a depth-first or breadth-first manner. The greedy approach always turned out to give the lowest average number of attempts necessary to find a decomposition.

In [14] a much more elaborate approach via the local search method requiring time $O(n^4)$ per trial is described. Nevertheless in the local search approach the number of graphs where a decomposition is found in the first one or two attempts decreases rapidly from 87% (one trial), 97% (two trials), for $n = 10$ (1000 random cubic graphs tested) to about 30%, resp. 50%, for $n = 200$ (2000 random cubic graphs tested), which is the largest case for which the program was run, while the greedy method described above finds a decomposition for about 96% ($n = 10$), 89% ($n = 200$), of the graphs in the first trial and 99.5%, resp. 98.5%, in the first two trials. For the greedy approach the average number of trials needed to find a set of defining trees grows very slowly – from 1.04 for $n = 10$ to 1.16 for $n = 150000$.

In spite of testing more than 350 000 large graphs we only once came across a graph where the heuristic did not find a set of defining trees. This graph turned out to have a bridge. In the other more than 350 000 cases the maximum number of times the heuristic had to be applied to find a decomposition was 8. In figure 2 you will find the development of the number of trials needed when choosing the vertex to add randomly instead of greedily, but still applying the same rules to remove vertices from the list.

Taking into account that it is an NP-hard problem, even the random approach works astonishingly well, though choosing a completely random tree (that is without the deletion of cutvertices from the list) only leads to the finding of defining trees for very small vertex numbers (see Figure 3 where this approach is named *Plain Random*). The key is in fact the application of the simple Lemma 3.

There are graphs for which the heuristic can't find a tree decomposition in spite of the fact that one exists. The smallest examples are for 22 vertices. The random approach does find a decomposition, but as the graphics show, for by far most of the graphs the performance of the greedy heuristic is considerably better.

6 An exhaustive search method

Our testing method works in 4 phases: first the greedy heuristic is applied to the graphs, then the remaining graphs are tested for having bridges (such graphs are trivially non-Yutsis), then the heuristic is applied again to the graphs still remaining, and finally an exhaustive search is applied to the graphs.

The number of applications of the heuristic is determined by the input graphs. We found that $\lfloor (1/5)|V| \rfloor$ trials for the first series and $\lfloor (1/10)|V| \rfloor$ trials for the second series are suitable values for (small) graphs and two times $(1/2)|V|$ trials are good for cubic polyhedra, which are listed in table 2. As an example: in the case of all cubic graphs on 24 vertices, 98.25% of the input graphs were determined to be Yutsis after the first 4 runs of the heuristic. That were 99.86% of the graphs that turned out to be Yutsis in the end. In the following step 76.92% of the remaining graphs were detected to have a bridge and after the last two applications of the heuristic 99.95% of the Yutsis graphs had been detected and only 0.31% of all graphs remained to be examined by the exhaustive test.

Our exhaustive search is simply a branch-and-bound version of the construction also used for the heuristic, without the greediness. That is: We start with some vertex and recursively add vertices from the list of vertices neighbouring the tree. In each iteration, vertices that became cutvertices in the complement or have two neighbours in the tree are removed from the list. All vertices remaining in the list are first added to the tree and in case no Yutsis decomposition is found in the following recursion steps then removed from the list and forbidden for later addition.

We tried several more elaborate search methods, but though they decreased the number of iterations, all of them in fact slowed down the computations for small graphs.

One thing that did in fact pay off was the determination of a good starting vertex based on the number of triangles in the neighbourhood:

Define $t(v) = \#\{w \in V | \{v, w\} \in E \text{ and } w \text{ is contained in a triangle}\}$ and the quality $q(v)$ of a vertex as $\sum_{\{w|\{v,w\}\in E\}} t(w)$. We choose a random vertex among those with maximal quality as the starting vertex. For 24 vertices this choice of the starting vertex decreases the number of iterations needed in the exhaustive test by a factor of 4.4 (2.9 for graphs that turned out to be Yutsis and 4.5 for non-Yutsis graphs).

Since in order to apply this criterion we have to search for triangles anyway, we can also use the result to apply Lemma 7. The fraction of graphs that can be proven not to be Yutsis this way is fairly small (e.g. 1660 for 24 vertices) but since applying the lemma causes no extra cost, it is of course worth doing it.

7 Results

The following numbers of Yutsis and non-Yutsis graphs were computed independently by the program described here and another somewhat slower approach that we have not described. The number of graphs with bridges and graphs with too many triangles were only computed by the first program. The programs used to generate the graphs were *minibaum* for all cubic graphs (see [15]), *plantri* for all cubic polyhedra (see [16]) and *genrang* for the tests of large cubic random graphs (see [17]). Some of the larger numbers were computed by a program using a preliminary version of the greedy heuristics in which the next vertex to add was always chosen among the last two vertices added to the list. Since the number of Yutsis and non-Yutsis graphs or graphs that can be determined to be non-Yutsis due to Lemma 7 are independent of this, we did not repeat the computation. Times where given always refer to the method described here implemented in C and run on a 2.6 GHz Pentium 4 Linux computer. For large vertex numbers the computation was done on clusters with various machine types in Bielefeld, Canberra and Ghent.

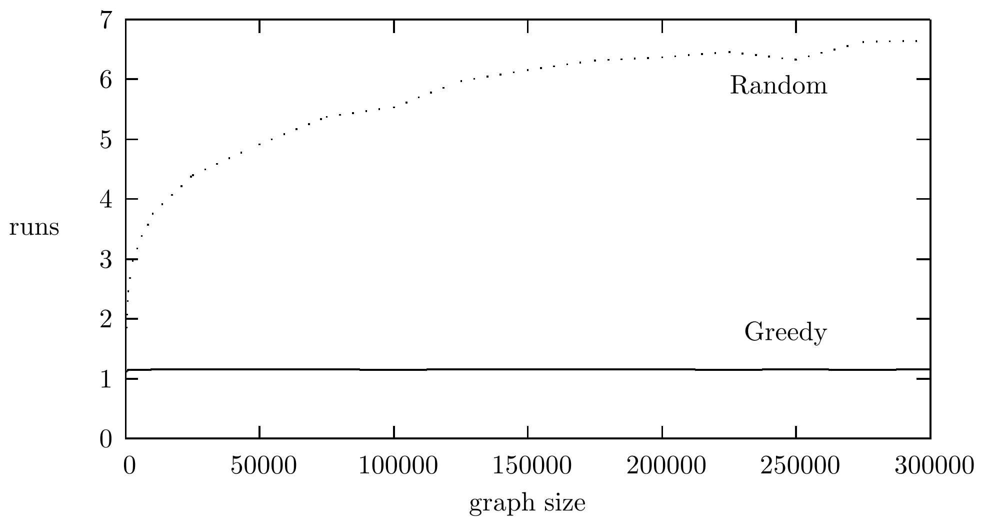

Fig. 2. The average number of runs on large random cubic graphs. For every size at least 25 000 graphs have been tested by the greedy heuristic and 1000 by the random heuristic.

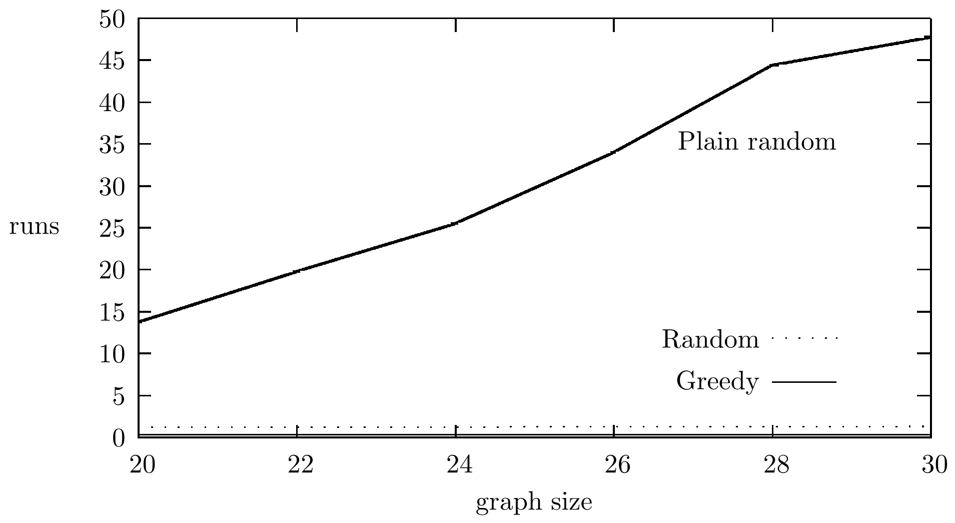

Fig. 3. The average number of runs on small random cubic graphs. For every size at least 9000 graphs have been generated and tested.

References

[1] L.C. Biedenharn and J.D. Louck, “Coupling of n angular momenta: recoupling theory”, in: *The Racah-Wigner Algebra in Quantum Theory, Encyclopedia of Mathematics and its Applications*, Vol. 9, pp. 435–481 (Addison-Wesley, 1981).

[2] A.P. Yutsis, I.B. Levinson and V.V. Vanagas, *Mathematical Apparatus of the Theory of Angular Momentum*, (Israel Program for Scientific Translation,

$2n$	# Graphs	Yutsis	Not Yutsis	with bridge	too many triangles	time (sec)
4	1	1	0	0	0	
6	2	2	0	0	0	
8	5	5	0	0	0	
10	19	18	1	1	0	
12	85	80	5	4	1	
14	509	475	34	29	2	
16	4 060	3 836	224	186	6	
18	41 301	39 555	1 746	1 435	22	<0.1
20	510 489	495 045	15 444	12 671	77	1.3
22	7 319 447	7 159 696	159 751	131 820	351	25
24	117 940 535	116 040 456	1 900 079	1 590 900	1 660	480
26	2 094 480 864	2 068 782 009	25 698 855	21 940 512	8 875	8 400
28	40 497 138 011	40 107 422 184	389 715 827	339 723 835	49 978	
30	845 480 228 069	838 931 116 609	6 549 111 460	5 821 548 438	301 277	

Table 1
The number of Yutsis graphs with $2n$ nodes for $n \leq 15$. The numbers of graphs with too many triangles are only for bridgeless graphs. The times given are on a 2.6 GHz Pentium 4 running Linux.

Jerusalem, 1962).

[3] F. Jaeger, "On vertex-induced forests in cubic graphs", Proceedings 5th Southeastern Conference, Congressus Numerantium (1974) 501–512.

[4] P. M. Lima, *Comput. Phys. Commun.* **66** (1991) 89.

[5] S. Fritzsche, T. Inghoff, T. Bastug and M. Tomaselli, *Comput. Phys. Commun.* **139** (2001) 314.

[6] D. Van Dyck and V. Fack, "GYutsis: heuristic based calculation of general recoupling coefficients", *Computer Physics Communications* **154** (2003) 219–232.

[7] V. Fack, S. N. Pitre and J. Van der Jeugt, "Calculation of general recoupling coefficients using graphical methods", *Comput. Phys. Commun.* **101** (1997) 155–170.

[8] D. Van Dyck and V. Fack, "New heuristic approach to the calculation of general recoupling coefficients", *Computer Physics Communications* **151** (2003) 353–368.

2n	# Graphs	Yutsis	too many triangles	Not Yutsis	time (sec)
4	1	1	0	0	
6	1	1	0	0	
8	2	2	0	0	
10	5	5	0	0	
12	14	14	0	0	
14	50	50	0	0	
16	233	233	0	0	
18	1 249	1 248	1	1	
20	7 595	7 593	0	2	
22	49 566	49 536	4	30	0.2
24	339 722	339 483	2	239	1.1
26	2 406 841	2 404 472	67	2 369	9
28	17 490 241	17 468 202	16	22 039	76
30	129 664 753	129 459 090	1268	205 663	663
32	977 526 957	975 647 292	414	1 879 665	5 830
34	7 475 907 149	7 458 907 217	29 984	16 999 932	73 273
36	57 896 349 553	57 744 122 366	11 109	152 227 187	
38	453 382 272 049	452 028 275 567	744 000	1 353 996 482	

Table 2
The numbers of Yutsis and non-Yutsis cubic polyhedra up to 38 vertices. The times given are on a 2.6 GHz Pentium 4 running Linux.

[9] P.G. Burke, "A program to calculate a general recoupling coefficient", *Computer. Phys. Commun.* **1** (1970) 241–250.

[10] A. Bar-Shalom and M. Klapisch, "NJGRAF – An efficient program for calculation of general recoupling coefficients by graphical analysis, compatible with NJSYM", *Comput. Phys. Commun.* **50** (1988) 375–393.

[11] D. Van Dyck and V. Fack, "On the Reduction of Yutsis Graphs", accepted by *Discrete Mathematics* (2004).

[12] V. Chvátal, "Hamiltonian cycles", in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, *The Traveling Salesman Problem*, pp. 403–429 (John Wiley, 1985).

[13] A. Wigderson, "The complexity of the hamiltonian circuit problem for maximal planar graphs", *Technical Report Computer Science Department Princeton*

University **298** (1982).

[14] D. Van Dyck and V. Fack, “To be or not to be Yutsis”, Electronic Notes in Discrete Mathematics **17C** (2004) 275–279.

[15] G. Brinkmann, “Fast generation of cubic graphs”, *Journal of Graph Theory*, 23(2), pp. 139–149, (1996).

[16] G. Brinkmann and B.D. McKay, “Fast generation of non-isomorphic planar cubic graphs.”, in preparation, see http://cs.anu.edu.au/∼bdm/plantri/.

[17] B. D. McKay, *gtools* programs available at `http://cs.anu.edu.au/∼bdm/nauty`.