Vlerick Leuven Gent Management School

the Autonomous Management School of
Ghent University and Katholieke Universiteit Leuven

Vlerick Leuven Gent Working Paper Series 2004/9

ON THE MORPHOLOGICAL STRUCTURE OF A NETWORK

MARIO VANHOUCKE

Mario.Vanhoucke@vlerick.be

JOSÉ COELHO

DIETER DEBELS

LUÍS V. TAVARES

D/2005/6482/9

ON THE MORPHOLOGICAL STRUCTURE OF A NETWORK

MARIO VANHOUCKE

Vlerick Leuven Gent Management School

JOSÉ COELHO

CESUR, Instituto Superior Tecnico, Technical University of Lisbon and

DCET, Universidade Aberta, Rua da Escola Politécnica

DIETER DEBELS

Faculty of Economics and Business Administration, Ghent University

LUÍS V. TAVARES

CESUR, Instituto Superior Tecnico, Technical University of Lisbon

Contact:

Mario Vanhoucke

Vlerick Leuven Gent Management School

Tel: +32 09 210 97 81

Fax: +32 09 210 97 00

Email: Mario.Vanhoucke@vlerick.be

ABSTRACT

In literature, both topological and resource-related measures are used to predict the difficulty of a project scheduling problem. Rapid progress regarding solution procedures has resulted in the development of a number of data generators in order to generate instances under a controlled design and in different standard sets with problem instances. These complexity measures need to serve as predictors for the complexity of the problem under study.

In this paper, we report on results for the topological structure of a network. The contribution of this paper is threefold. First, we review six topological network indicators in order to describe the structure of a network in a detailed way. These indicators were originally developed by [20] and have been modified or sometimes completely replaced by alternative indicators in order to give a better description of the topology of a network. Secondly, we generate a large amount of different networks with four network generators. This allows us to draw conclusions on both the performance of different network generators and to give a critical remark on well-known datasets from literature. Our general conclusions are that none of the network generators are able to capture the complete feasible domain of all networks. Moreover, each network generator covers its own network-specific domain and, consequently, contributes to the generation of instance data sets. Finally, we perform computational results on the well-known resource-constrained project scheduling problem to proof that our indicators are reliable and have significant predictive power to serve as complexity indicators.

Keywords: Networks; Topological structure; Graphs; Project Scheduling instances.

1 INTRODUCTION

Researchers in the area of Operations Research and project scheduling have used activity networks to visualize different kinds of projects. Due to the rapid progress regarding exact and heuristic procedures, they have paid a lot of attention to the description of the topological structure of a network. In doing so, they try to predict the difficulty of a particular problem for a particular solution procedure based on the structure of the network. Therefore, different sets with topological indicators that can discriminate between easy and hard instances and that can act as predictors of the computational effort of the procedures have been developed. The CPU-time that a solution procedure needs to solve a particular problem instance to optimality can typically be used to describe the hardness of this problem instance for the particular solution procedure. Hence, the comparison of procedures and good predictions of their required CPU-time allow the a priori selection of the fastest solution procedure, based on the simple calculation of the topological indicators.

Quite a number of complexity measures have been proposed in the literature. [9] call the attention to problem instances that span the full range of problem complexity. [10] have shown the occurrence of *phase transitions* in project scheduling problems and call the attention to the importance of measures with sufficient discriminatory power to allow for the observation of these dramatic changes in problem difficulty. These authors distinguish between measures capturing information about the size and the topological structure of the network and measures which are related to the different resources allocated to the project.

For an overview of previous research in complexity measures and their link with existing network generators, we refer to [7]. In their paper, the authors describe three important complexity measures that are used to describe the topological structure of a network, i.e. the coefficient of network complexity (*CNC*, see [3], [4], [11], [12] and [18]), the order strength (*OS*, see [2], [10], [13], [16], [19] and [21]) and the complexity index (*CI*, see [1]). During the development of efficient solution procedures, many researchers have relied on data instances with one of these complexity measures. [7] also provide a critical discussion of the different network generators that are available to the researchers in order to generate the data instances. In this paper, we will use *RanGen* [7], *ProGen* [15] and *RiskNet* [19] for our computational results section, as well as the well-known testset *PSPLIB* [14]. In the remainder of this paper, we refer to the original *RanGen* network generator as *RanGen*1. Moreover, we report on results of a modified version of *RanGen* by taking new topological indicators into account (referred to as *RanGen*2) that are based on improved definitions of the indicators presented in [20]. The description of these indicators is the subject of section 2.

In this paper, we focus on the topological structure of an activity-on-the-node network in order to compare well-known network generators from literature. The organization of the paper is as follows. In section 2 we present the six network-based indicators that describe the topological structure of any network in detail and are partly improved versions of the indicators presented in [20]. Section 3 describes the generation process of *RanGen*2 which takes these new indicators into account. Section 4 is reserved for a detailed description of the structure of a network based on a simulation with the four aforementioned network generators. We show that *RanGen*2 is able to generate networks with a wide variety of topological measures. Moreover, we test the influence of the topological indicators on the well-known resource-constrained project scheduling problem and show that the indicators can serve as complexity predictors. Section 5 provides a summary, overall conclusions and areas for future research topics.

2 THE TOPOLOGICAL INDICATORS

In this section we present six complexity measures that describe the topological structure of a network, based on the indicators proposed by [20]. More precisely, the I_1, I_2 and I_4 indicators are exact copies from the original paper, while indicators I_3 and I_5 are improved versions of their corresponding original versions. Indicator I_6 is completely new and does not appear in [20]. All these indicators (except the first one) have a value in the [0,1]-interval. Each indicator measures a specific characteristic of the topological structure of a network and will be discussed in following subsections.

In the remainder of this paper, a network consists of a set of nodes N and a set of arcs A. The nodes are numbered from the first node 1 to the last node n. The arcs are used to impose precedence relations between the nodes as follows: we use the set P_i (S_i) to denote the set of immediate predecessor nodes (successor nodes) of node i linked by means of an arc. The sets P'_i and S'_i are used to denote the set of *all* nodes that precede (succeed) node i linked by means of a path in the network. When we use an activity-on-the-node (AoN) network, the nodes represent activities and the arcs represent technological precedence constraints. In order to be in line with literature, a project network can be extended with a dummy start node 0 and a dummy end node $n + 1$.

Some indicators rely on the progressive level PL_i and regressive level RL_i for each activity $i \in N$. These concepts have been introduced by [8] and are defined as follows:

$$PL_i = \begin{cases} 1 & \text{if } P_i = \varnothing \\ \max\limits_{j \in P_i} PL_j + 1 & \text{if } P_i \neq \varnothing \end{cases}$$

and

$$RL_i = \begin{cases} m & \text{if } S_i = \varnothing \\ \min\limits_{j \in S_i} RL_j - 1 & \text{if } S_i \neq \varnothing \end{cases}$$

with m the maximal progressive level, i.e. $m = \max\limits_{i \in N} PL_i$.

2.1 Indicator I_1 – size of problem

The first indicator of [20] measures the size of the network as the number of activities and equals I_1 = number of activities = n.

2.2 Indicator I2 – serial or parallel indicator

Indicator $I_2 \in [0, 1]$ measures the closeness of a network to a serial or parallel graph for activity-on-the-node networks based on the number of progressive levels m. Indicator I_2 is defined as follows:

$$I_2 = \begin{cases} 1 & \text{if } n = 1 \\ \dfrac{m-1}{n-1} & \text{if } n > 1 \end{cases}$$

For a network with all activities in series, $m = n$ and $I_2 = 1$. When all activities are in parallel then $m = 1$ and, consequently, $I_2 = 0$.

2.3 Indicator I3 - activity distribution

[20] introduced the width w_a of each progressive level $a = 1, \ldots, m$ as the number of activities at that level. Based on w_a, they defined I_3 as a vector w that contains the width of each progressive level. In order to be in line with the other indicators presented in this paper, we define $I_3 \in [0, 1]$ as follows:

$$I_3 = \begin{cases} 0 & \text{if } m \in \{1, n\} \\ \dfrac{\alpha_w}{\alpha_{\max}} = \dfrac{\sum_{a=1}^{m} \left| w_a - \overline{w} \right|}{2(m-1)(\overline{w}-1)} & \text{if } m \notin \{1, n\} \end{cases}$$

This indicator measures the distribution of the activities over the progressive levels by calculating the total absolute deviations α_w and $\alpha_{\max}$. α_w measures the total absolute deviation of the activity distribution $w = (w_1, w_2, \ldots, w_m)$ from the average deviation $\overline{w} = n / m$ as follows:

$$\alpha_w = \sum_{a=1}^{m} \left| w_a - \overline{w} \right| \ .$$

$\alpha_{\max}$ determines the maximal value of α_w for a network with n activities and m progressive levels. $\alpha_{\max}$ corresponds to a network for which m - 1 progressive levels have a width w_a of 1, and one progressive level has a width w_a of n - (m - 1). The value of $\alpha_{\max}$ can be calculated as follows:

$$\alpha_{\max} = (m-1)(\overline{w}-1) \ + (n - m + 1 - \overline{w}) \, .$$

The first term calculates the absolute deviation between $w_a = 1$ and the average width $\overline{w}$ for m – 1 progressive levels. The second term calculates the difference between $w_a = n$ - (m - 1) and $\overline{w}$ for the remaining progressive level. The formula for $\text{m} \neq 1$ can be simplified to $\alpha_{\max} = 2(m-1)(\overline{w}-1)$, resulting in the I_3 indicator defined above. This indicator equals 1 when $\alpha_w = \alpha_{\max}$. At the other extreme, the indicator has a value of 0 when the activities are uniformly distributed over the progressive levels, i.e. $w_a = \overline{w} = n / m$ (for $a = 1, \ldots, m$).

2.4 Indicator I4 - short arcs

The length of an arc (i, j) can be defined as the difference between the progressive level of the end node j and the start node i, i.e. $PL_j – PL_i$. Consequently, the maximal length of an arc equals m – 1. We define the number of arcs in the network with length l as $n'_l = \#\{(i, j) \in A | \ PL_j – PL_i = l\}$, with $1 \leq l \leq m - 1$. Based on this parameter, the indicator $I_4 \in [0, 1]$ measures the presence of short arcs (i.e. with a length $l = 1$) and can be defined as follows:

$$I_4 = \begin{cases} 1 & \text{if } D = n - w_1 \\ \dfrac{n'_1 - n + w_1}{D - n + w_1} & \text{if } D > n - w_1 \end{cases}$$

where D represents the maximal number of short (l = 1) arcs in a network, given the width of each level, i.e. $D = \sum_{a=1}^{m-1} w_a * w_{a+1}$.

2.5 Indicator I5 - long arcs

Indicator $I_5 \in [0,1]$ is based on the same parameter n'_l , but takes - in contrast to I_4 - also the long arcs (i.e. $l > 1$) into account. The indicator is defined as follows:

$$I_5 = \begin{cases} 1 & \text{if } |A| = n - w_1 \\ \dfrac{\left(\sum_{l=2}^{m-1} n'_l \dfrac{m-l-1}{m-2}\right) + n'_1 - n + w_1}{|A| - n + w_1} & \text{if } |A| > n - w_1 \end{cases}$$

$|A|$ represents the total number of arcs, and equals $\sum_{l=1}^{m-1} n'_l$. The value of this I_5 can only be 1 if all the arcs have a length l = 1. Dependent on the number of long arcs and the exact length of these arcs, I_5 will have a value closer to 0.

2.6 Indicator I6 - topological float

This indicator takes the topological float of each activity into account and has not been incorporated in [20]. The topological float of activity i is defined as the difference between its regressive and progressive level, i.e. RL_i - PL_i. Based on this parameter, indicator $I_6 \in [0, 1]$ is defined as follows:

$$I_6 = \begin{cases} 0 & \text{if } m \in \{1, n\} \\ \dfrac{\sum_{i=1}^{n} (RL_i - PL_i)}{(m-1)(n-m)} & \text{if } m \notin \{1, n\} \end{cases}$$

The value of I_6 will be zero if none of the activities has a topological float > 0. At the other extreme, I_6 will be equal to 1 for a network with m serial activities with zero topological float and the remaining n - m activities without precedence relations (i.e. with a topological float of m - 1).

An illustrative Example

In figure 1, we displayed a network with 10 activities and 2 dummy activities. The values for the different indicators are calculated below.

Insert Figure 1 About Here

In table 1 we report the values for the progressive and regressive level for each activity. Based on this information, we can calculate the values for all indicators. In this example, we have n = 10, m = 4, w_1 = 3, w_2 = 5, w_3 = 1, w_4 = 1, n'_1 = 8, n'_2 = 4, n'_3 = 1 and consequently D = 3 * 5 + 5 * 1 + 1 * 1 = 21.

Insert Table 1 About Here

The values for the indicators of sections 2 can now be calculated by filling in all the needed parameters. In doing so, we obtain

$$I_1 = 10, \quad I_2 = \frac{4-1}{10-1} = 0.33, \quad I_3 = \frac{\left|3-\frac{10}{4}\right| + \left|5-\frac{10}{4}\right| + \left|1-\frac{10}{4}\right| + \left|1-\frac{10}{4}\right|}{2*(4-1)*(\frac{10}{4}-1)} = 0.67,$$

$$I_4 = \frac{8-10+3}{21-10+3} = 0.07, \quad I_5 = \frac{\frac{2-3}{2-4}*4 + \frac{3-3}{2-4}*1 + 8 - 10 + 3}{8+4+1-10+3} = 0.50, \quad I_6 = \frac{8}{3*6} = 0.44.$$

3 NETWORK GENERATION PROCESS: *RANGEN*1 AND *RANGEN*2

In section 3.1 we briefly review the generation process of *RanGen*1. In section 3.2, we present the logic behind *RanGen*2 which is an enhanced version of *RanGen*1 and takes the six new indicators of section 2 into account.

3.1 The generation scheme of *RanGen*1

The networks in *RanGen*1 are represented by a *strictly upper triangular precedence relations matrix*. This binary precedence matrix *PM* denotes whether or not a precedence relation exists between two nodes. The matrix can be defined as follows: $PM_{ij} = \begin{cases} i \in P'_j \Rightarrow 1 \\ i \notin P'_j \Rightarrow 0 \end{cases}$, and is illustrated by the network as given in figure 2. Notice that we add a dummy start activity *s* and a dummy end activity *t* to visualize the network but we do not incorporate these in the precedence matrix *PM*.

Insert Figure 2 About Here

The generation process of a network in *RanGen*1 with pre-specified *OS*-value boils down to the deletion of arcs. Indeed, the generator starts for each generation with a completely connected network for which *OS* = 1 and removes arcs until it obtains a network with the pre-specified order strength. The set of arcs is updated each time an arc has been removed. Due to this simple logic, *RanGen*1 guarantees to find a network with a given value for the order strength *OS*.

Since the removal of arcs can lead to networks with a different precedence matrix *PM* but with a similar topological structure, each network must be checked for uniqueness by the recursive enumeration procedure described in [7]. If the network has not yet been generated, it is added to the set of generated networks *GN* (initially *GN* = ∅). *RanGen*1 continues this way until a pre-defined number of networks has been generated or until a certain time limit has been reached.

3.2 The generation scheme of *RanGen*2

The generation process of *RanGen*1 is completely determined by pre-specified values for I_1 and *OS*. *RanGen*2 relies on the same basic logic of removing arcs, but aims at generating networks with pre-specified values for I_1 and I_2. Consequently, the generator removes arcs until it obtains a network with a pre-specified I_2-value. However, *RanGen*2 distinguishes from *RanGen*1 in two basic characteristics: the start network at each generation run and the number of networks per run.

A first fundamental difference is the start network at each generation run. While *RanGen*1 starts with a unique serial network with *OS* = 1, *RanGen*2 starts from a larger pool of possible networks. More precisely, *RanGen*2 starts with a network for which the I_2 is randomly chosen from the interval [$I_2^{'}$, 1], with $I_2^{'}$ the pre-specified value for I_2 and an I_3-value randomly chosen from the interval [0, 1]. The start values of all other indicators equal $I_4 = I_5 = 1$ and $I_6 = 0$. In doing so, we start from a larger pool of possible start networks, and hence more different network with pre-defined indicator values can be generated.

A second difference lies in the number of networks generated per run. *RanGen*1 removes arcs from a network until a pre-specified *OS*-value is obtained. *RanGen*2 removes, in a similar way, arcs from a network until a pre-specified I_2-value is reached. From this point onwards, *RanGen*2 continues with the removal of arcs. Indeed, further removals might be possible without decreasing the value of I_2. This process continues until no arc can be deleted without decreasing I_2. Thus, while for *RanGen*1 exactly one network is generated per run, a set of networks are generated in each run with *RanGen*2 with pre-specified values for I_1 and I_2 but with a different topological structure (i.e. other values for I_3, I_4, I_5, and I_6). Consider, as an example, the network of figure 2 for which *OS* = 0.5 and $I_2 = 0.5$. *RanGen*1 stops with removing arcs since otherwise the *OS*-value would decrease. *RanGen*2, however, can continue since removing certain arcs can lead to networks with the same value $I_2 = 0.5$. In figure 3, we have removed arc (2, 3) from the precedence matrix such that *OS* = 0.4 and $I_2 = 0.5$. These two networks (figures 2 and 3) are saved by *RanGen*2, since they both meet the condition $I_2 = 0.5$. At that point, *RanGen*2 stops with the removal of arcs, since this would lead to *OS* = 0.3 and $I_2 = 0.25$.

Insert Figure 3 About Here

After the generation of each network, *RanGen*2 checks whether the network has a different topological structure than previous generated networks by means of the recursive search procedure of [7]. The pseudo-code to generate a set of activity-on-the-node networks *GN* that meet pre-specified values $I_1^{'}$ and $I_2^{'}$ can be described as given below. Note that for each generation run, more than one activity network *AN* can be generated as explained previously in this section.

```
procedure generate($I_1^{'}$, $I_2^{'}$);
GN = ∅;
AN = activity network with $n = I_1^{'}$ and $I_2 \in [I_2^{'}, 1]$, $I_3 \in [0, 1]$, $I_4 = I_5 = 1$
and $I_6 = 0$;
REPEAT
    REPEAT
        arc = select_arc(AN);
        AN = remove_arc(AN, arc);
        Unique_representation(AN);
        If (AN ∉ GN) then save the network: GN = GN ∪ AN;
    Until $I_2 = I_2^{'}$;
    Repeat
        arc = find_removable_arc_without_changing_$I_2$(AN);
        AN = remove_arc(AN, arc);
        Unique_representation(AN);
        If (AN ∉ GN) then save the network: GN = GN ∪ AN;
    Until arc = 0;
Until #GN ≥ pre-specified number of generated networks
Return;
```

This generation process used for *RanGen*2 has been programmed in Borland C++, version 4.0, and can be downloaded at http://www.projectmanagement.UGent.be. Since the generator generates many networks in a small time limit and it calculates and stores the I_3, I_4, I_5 and I_6 values each time a new network has been generated, the user is able to generate networks with a pre-specified value for all indicators I_1 to I_6.

4 EXPERIMENTAL DESIGN - RELATIONS BETWEEN INDICATORS

In this section, we present results for the relations between the indicators as well as results for network generators. We also use the well-known *PSPLIB* dataset for comparison purposes. The generation of networks has been divided in three main parts. In subsection 4.1, we display the results of an exhaustive generation of all 10-activity networks. In doing so, we are able to present computational results that do not depend on the performance of network generators. In subsection 4.2, we discuss the settings and results of our computational tests and generate networks of 30 activities with four different network generators. Although we are not able to generate all possible networks of that size, the tests allow us to compare the performance of these network generators by means of the indicators presented in this paper. The results of this test are presented in section 4.3. In section 4.4, we report computational results for the well-known resource-constrained project scheduling problem and show that our indicators can be used as complexity indicators. For all our tests, we used a Toshiba personal computer with a Pentium IV 2 GHz processor and 512 MB Ram under a Windows XP operating system.

4.1 The exhaustive generation of all 10-activity networks

A network generator can be called *strongly random* if it is able to generate networks at random from the space of all feasible networks with a specified number of nodes and arcs [5]. [7] argued that the generation of strongly random networks could be possible by enumerating all possible network structures which satisfy preset values of the complexity parameters and by randomly selecting a subset of networks afterwards.. Due to CPU-time and memory restrictions, however, this method is only applicable for networks with a small amount of activities. In this subsection, we generate all networks with 10 activities and display some interesting results in table 2.

Insert Table 2 About Here

The table reveals that there exist 2,567,284 networks with 10 activities and with a different topological structure. Rows 2 to 7 display the number of settings for different indicators. Since we generate all 10-activity networks, the maximum number of precedence relations equals (10 * 9) / 2 = 45. Consequently, we have 46 settings for the *OS*, varying from 0 to 1 in steps of 1 / 45. This means that, on the average, 2,567,284 / 46 = 55,810.52 networks exist per setting. As shown in [7], there are many more networks with an *OS*-value of 0.5 than, for example, an *OS*-value of 0.20. Since I_2 has been defined as $\frac{m-1}{n-1}$ (with $n = I_1 = 10$), we have 10 settings for the I_2 indicator, varying from 0 to 1 in steps of 1 / 9. The number of settings for indicators I_3, I_4, I_5 and I_6 is somewhat more confounding. To that purpose, we have calculated the values for these indicators with two decimal places. In doing so, we have 101 settings, varying from 0 to 1 in steps of 0.01.

The last row displays the number of networks with different values for the I_2 to I_6 indicators. To that purpose, we have created a matrix of size 10 * 101^4 according to the different settings as described before (10 settings for I_2 and 101 settings for I_3 to I_6). During the complete enumeration, we check whether a network with different values for the I_2, I_3, I_4, I_5 and I_6 indicators has been found. The total number of networks with a unique I_2 to I_6 combination equals 48,982.

In table 3, we display the correlation matrix for the *OS* and the indicators I_2, I_3, I_4, I_5 and I_6. As an example, the correlation coefficient r_{OS,I_2} = 0.70 is intuitively clear. The more levels (*m*) a network has, the larger the I_2-value and the more precedence relations a network can have (which results in a larger *OS*-value). The correlation r_{I_4,I_5} = 0.67 is also straightforward, since both I_4 and I_5 are constructed to measure the ‘length’ of the arcs in a network. All the other correlation coefficients are lower than 0.5, as displayed in the table.

Insert Table 3 About Here

Note that the results of this subsection are based upon the complete enumeration of all existing networks with 10 activities. The number of networks with different topological structure with 11 activities is equal to 46,749,566. We were not able to determine the number of networks with 12 activities, due to CPU-time and memory restrictions. In the next section, we generate networks with 30 activities by means of different network generators. Since it is impossible to generate all networks, we use two stop criteria as described in the following section.

4.2 Comparison of network generators

In this section we generate a large number of networks with different values for the indicators. This experiment allows the comparison of different network generators from literature. Moreover, it serves as an illustration of the relations and dependencies of the different indicators. In all experiments, we have chosen to set I_1 = 30 in order to compare it with the *PSPLIB* instances ([14]) with 30 activities. Similar results have been found with the 60, 90 and 120 instances. Our tests are based on the use of four network generators, i.e. *RanGen*1, *RanGen*2, *ProGen* and *RiskNet*. In order to have a fair comparison between network generators, we try to cover the whole domain of feasible networks as much as possible by exploiting specific information about the input parameters of each generator.

*RanGen*1 generates network instances with given values for both I_1 and *OS*. Since, in our experiments, $n = I_1 = 30$, the maximal number of arcs equals (30 * 29) / 2 = 435. In order to cover the whole domain specified by the *OS*, we vary this measure in steps of 1 / 435. In doing so, we have 436 settings, starting from a parallel network (*OS* = 0) and ending with a serial network (*OS* = 1), by allowing one extra arc at the time per setting. Note that, because of the simple logic of the generation process, each network can be generated in a very small amount of time.

*RanGen*2 generates network instances with given values for both I_1 and I_2. Since I_2 has been defined as $\frac{m-1}{n-1}$, $n = I_1 = 30$ and since we want to follow a similar reasoning as for the *RanGen*1 instances, we need to use 30 settings. Therefore, we fix the values for I_2 from 0 to 1 in steps of 1 / 29 in order to generate networks with all possible values of I_2.

For the generation of networks with *ProGen*, we relied on the source code to generate a large number of networks. The input values for this generator are the coefficient of network complexity, the maximal number of successor activities and the maximal number of starting and ending activities of the project. The *CNC* is defined as the number of arcs divided by the number of nodes in the network. The minimal *CNC*-value equals 0 and corresponds to a parallel network. The maximal *CNC*-value for a network with $I_1 = 30$ is 7.5. This corresponds to a two-level ($m = 2$) network with 15 activities on each level for which all activities of level 1 are predecessors of all the activities of the second level. In order to guarantee that every possible network can be generated, we use – in similarity to the previous network generators – 226 settings for the *CNC*, ranging from 0 to 7.5, in steps of 1 / 30. All other input parameters (such as the number of successor and predecessors per activity and the maximal number of start and end activities) have been chosen randomly such that they do not restrict the feasible domain of networks.

In order to generate networks with *RiskNet* we need to control 3 input parameters. The value for the first indicator, I_1, equals 30. The values for the second indicator, I_2, are similar to the settings of the *RanGen*2 test instances (30 settings). The third input parameter is equal to I_4 as described in this paper. Since this indicator measures the number of ‘short’ arcs, and it equals 0 when there is no such relations and it equals 1 when there are D such relations. D equals $\sum_{a=1}^{m-1} w_a * w_{a+1}$ and depends on the value of I_3. However, the maximal value for D, regardless of the I_3-value, equals 225 as described previously. Consequently, we use 226 settings for the I_4 indicator, ranging from 0 to 1, in steps of 1 / 225.

In the next section, we discuss the results of the computational tests and display them in a graphical way (see the scatter plots in the appendix). In order to have a fair comparison between network generators, it is necessary to generate as many networks as possible. To that purpose, we apply a stop criterion for each setting (and consequently, continue then with the next setting) if the generation run of networks exceed one of the following criteria:

(*i*) Maximum allowable CPU-time of 1 network of 100 seconds. Exceeding this time limit indicates that no network with the given input parameters can be found by the network generator under study. Note that this will only occur for the *RiskNet* and *ProGen* generator. *RanGen*1 and *RanGen*2 always assure the existence of a network with the given input parameter in a very small amount of CPU-time.

(*ii*) Maximum 1,000 consecutive generations without finding a new network. This means that the network generator is not able to find a new network which has not been generated previously. Due to memory restrictions, we define a new network as a network for which no similar I_2 to I_6 combination has been found. Note that we could also have used the recursive search procedure of [7], but this would lead to too many networks with a different topological structure but identical I_2 to I_6 combinations.

For each setting, we generate the network instances according to the input parameters described in this section and, afterwards, we calculate the values for all the other indicators.

4.3 Experimental results

As mentioned in previous section, we note that we have defined a ‘new’ network as a network with a different I_2 to I_6 combination than a previously found network. To that purpose, we have created a matrix of size 30 * 101^4 and indicated whether a network has been found for each field of this matrix. This is exactly the same approach that we have taken in generating all 10-activity networks (48,982 ‘new’ networks versus the 2,567,284 found networks). Consequently, if we refer to the number of new networks found, we refer to networks with different I_2 to I_6 combinations.

Table 4 presents the results of our computational tests for the four different network generators. The total number of new networks found equals 19,105,294. The different rows contain the number of new networks found by a combination of network generators. As an example, 450,593 networks were found by *RanGen*1 which were not found by any other network generator. 1,272,039 networks were found by *RanGen*1 and *RanGen*2 which were not found by *RiskNet* or *ProGen*. The row "Total" contains the total number of new networks found per network generator, and the row "% Total" expresses this number as a percentage of the total number of new networks found by all generators, i.e. % Total = Total / 19,105,294. The next two rows give an indication of the amount of new networks that were found by only one network generator. The row "% New – Generator" expresses this number as a percentage of the total amount of new networks found by the generator. The row "% New – Total" expresses this number as a percentage of the total amount of new networks found.

Insert Table 4 About Here

The table reveals that *RanGen*2 outperforms all other network generators in the total amount of networks generated. Indeed, somewhat more than 60 % of the networks were found by *RanGen*2. *ProGen* is also able to generate a large set of the total amount of networks (48.98 %). *RanGen*1 and *RiskNet* show a very poor performance since they can both generate only a small subset of all generated networks (22.06 % and 11.50 %, respectively). The total running time to generate all these networks amounted to about 10 hours for *RanGen*1, 18 hours for *RanGen*2, 2 days for *RiskNet* and about 2 weeks for *ProGen*.

Although an evaluation of different network generators may lead to interesting conclusions, our ultimate purpose is not to compare the network generators as such, but rather to show that improvements can be made by combining the use of different generators. This information can be found in the rows "% New – Generator" and "% New – Total". One the one hand, these rows reveal that, despite its poor performance on the total amount of new networks, *RiskNet* is able to generate a large set of networks that were not found by any other generator. Indeed, almost 78 % of the networks found by *RiskNet* were not found by any other network generator. However, this fraction counts only for 8.97 % of the total amount of networks found. *RanGen*1, on the other hand, is only able to provide 2.36 % new networks that were not found by any other network generator, and thus does not contribute much to the total set of 19,105,294 networks. *RanGen*2 and *ProGen* both have a high contribution to the total amount of networks generated. Approximately 50% of the networks found by both generators were completely new. Almost 32 % of all the networks have only been generated by *RanGen*2 and almost 25 % of all the networks have only been generated by *ProGen*. This information has been summarized in figure 4 for *RanGen*2, *ProGen* and *RiskNet*. Due to the low contribution of *RanGen*1, we have deleted this network generator in this figure.

Insert Figure 4 About Here

This picture clearly illustrates the advantage of the combination of different network generators. All network generators were able to generate a large amount of networks that were not found by any other. The intersection of different network generators contains a relatively small amount of networks, and only 40,382 similar new networks were found by the three network generators.

In the appendix, all pairs of indicators have been displayed by two-dimensional scatterplots. These graphical pictures allow a quick comparison of network generators and provide insight into the relations – albeit in only two dimensions – between the indicators. In order to have a complete overview, the results for the *PSPLIB* instances have also been displayed. These graphs must be interpreted from a ‘strongly randomness’ point of view. If the generated networks only cover a small portion of this whole domain, only a very small amount of possible networks can be generated and the network generator fails to be ‘as strongly random as possible’. Therefore, the better the scatterplots cover the domain, the better the network generator is.

The plots indicate that *RanGen*2 clearly outperforms *RanGen*1 since all areas generated by the last generator have also been generated by the first one. The *RanGen*1 procedure starts with a serial network and removes arcs until it obtains a network with the pre-specified order strength. It has been mentioned by [7] that some networks have only one representation in the PM while others have many. Although each possible network can be theoretically generated, the probability of a network to be generated is heavily related to its corresponding number of representations in the PM. Some networks have many different representations, while others have only one and thus are very unlikely to be generated by *Rangen*1. *Rangen*2 tries to overcome this problem by starting with a generated network with a randomly selected I_3 value. In doing so, *RanGen*2 starts from a larger pool of possible networks, and hence, more different networks can be generated.

The scatterplots reveal that *RiskNet* performs reasonably well and generates networks from almost the same space than *RanGen*2. This network generator uses I_2 and I_4 as input values for the generation process, and hence shows the good performance from a two-dimensional point. However, in the previous section, we illustrated that the generator is not able to generate a lot of different networks.

ProGen outperforms *RiskNet* and *RanGen*1 in the number of different networks, but performs less in generating networks from the feasible domain, as illustrated in a number of graphs (see e.g. I_2 vs. I_4, I_2 vs. I_5 or I_2 vs. I_6). As an example, *ProGen* fails in producing networks with high I_2-values (see the scatterplots of the appendix). These long serial chains of activities are hard to generate, due to the specific generation process of *ProGen* and the extra input parameters, such as the maximal number of successor activities. The process assigns predecessors to each activity (see step 2 of [15]) which results in a tree structure when the maximal number of successors exceed the value of 1. Hence, a final network structure with a high I_2-value is quite unlikely. Only when the maximal number of successor activities is set to one, a limited number of networks with a high I_2-value will be generated.

Remark that the results are sometimes misleading due to the settings of our experiment. The 'I_2 vs I_5' plot, for example, shows vertical lines for almost all values for I_2. The reason is that we only have generated networks with I_2-values in 30 discrete steps. This graph must therefore be seen as a black picture, covering the domain of feasible networks, since intermediate areas do not exist. Note that the *J*30 set only covers a very limited space, while *RanGen*1, *RanGen*2, *ProGen* and *RiskNet* are able to generate networks from a wider part of all the feasible networks. Of course, *PSPLIB* contains only a small fraction of the networks generated by the network generators, and therefore, the comparison serves only for illustrative purposes.

4.4 The predictive power of the indicators

Although the aim of our paper is to investigate network generators and detect possible improvements, it is interesting to give a first impression of the predictive power of the indicators presented in section 2. More precisely, we give preliminary results that the proposed indicators have discriminatory power to predict between easy and hard instances for a particular algorithm and hence, allow the a-priori selection of the fastest solution procedure for a problem instance. To that purpose, we have generated 10 classes of network instances, each containing 100 instances, with indicator values as given in table 5. We extended each network instance with 4 renewable resources with a resource-constrainedness (see [17]) of 0.4 and a resource use [7] of 0.75.

Insert Table 5 About Here

We have used these networks as resource-constrained project scheduling problem instances (RCPSP), that can be solved to optimality by the procedure of [6]. The number of created nodes in this branch-and-bound procedure used to solve these instances to optimality is given per class in figure 5. The number of created nodes can be used as a measure to predict the problem instance complexity for the branch-and-bound procedure used.

Insert Figure 5 About Here

This figure reveals that instances within a class are rather homogeneous with respect to the number of created nodes, while the number varies drastically between different classes. This gives a first indication of the predictive power of our indicators, and hence they might serve as a predictor for problem instance complexity and the a-priori selection of the fastest procedure. Furthermore, it indicates that networks within one class are rather similar while network instances between classes show significant difference. Consequently, a combination of I_1 to I_6 values can be used to predict the behaviour of a solution procedure.

5 CONCLUSIONS

In this paper we discussed six indicators that describe the topological structure of a network into detail. Some of these indicators are improved versions of the [20] indicators. These network-based indicators have been used to modify and enhance the generation process of a network generator *RanGen*1 [7].

The experimental section consists of three main parts. In a first section, we displayed the results of the generation of all 10-activity networks and reported some interesting results. In a second section, we have generated a large set of networks in order to have an overview of the relation between the indicators and to compare the performance of different network generators. To that purpose, we rely on the generation technique of the original network generator (*RanGen*1) and on the newly presented technique in this paper (*RanGen*2). We also included networks generated by *ProGen* and *RiskNet*, and we compared our nets and associated indicators with the *PSPLIB* library. In a last section, we show that our indicators are reliable to predict problem instance complexity by reporting results on the resource-constrained project scheduling problem. We show that the variance between networks within one [I_2 to I_6] vector is rather low, while networks between classes show a significant variance with respect to the number of created nodes to solve the problem instances to optimality.

Our future intentions are twofold. First, we want to create a benchmark dataset to use for further research in the project scheduling community. Therefore, we can rely on the experimental research of this paper, that illustrates that a combination of different network generators is necessary during the generation process. Moreover, the *PSPLIB* results show that these network instances only represent a small portion of the feasible domain and, consequently, more networks are needed. A second topic of future research lies in a benchmark comparison of different existing procedures by using networks generated by our new procedure. In doing so, we can detect whether the topological indicators reveal some phase transitions for certain problem types. We are convinced that this benchmark comparison opens new insights into the understanding and predictive power of the complexity of existing and new algorithms.

ACKNOWLEDGEMENTS

The network generator *ProGen* can be downloaded at
http://www.bwl.uni-kiel.de/Prod/psplib/progen/progen-sfx.exe

The network generator *RiskNet* can be downloaded at
http://jcoelho.m6.net/freeware/risknet.msi

The network generators *RanGen*1 and *RanGen*2 can be downloaded at
http://www.projectmanagement.UGent.be

We would like to thank Prof. Dr. Erik Demeulemeester for his useful comments at the *EURO/INFORMS joint international meeting*, Istanbul, Turkey, 6-10 July, 2003.

REFERENCES

1. W.W. Bein, J. Kamburowski and M.F.M. Stallmann, Optimal reduction of two-terminal directed acyclic graphs", SIAM Journal on Computing 21 (1992), 1112-1129.

2. E.M. Dar-El, MALB - A heuristic technique for balancing large single-model assembly lines, IIE Transactions 5 (1973), 343-356.

3. E.M. Davies, An experimental investigation of resource allocation in multiactivity projects, Operational Research Quarterly 24 (1974), 587-591.

4. E.W. Davis, Project network summary measures and constrained resource scheduling, IIE Transactions 7 (1975), 132-142.

5. E. Demeulemeester, B. Dodin and W. Herroelen, A random activity network generator, Operations Research 41 (1993), 972-980.

6. E. Demeulemeester and W. Herroelen, New benchmark results for the resource-constrained project scheduling problem, Management Science 43 (1997), 1485–1492.

7. E. Demeulemeester, M. Vanhoucke and W. Herroelen, A random network generator for activity-on-the-node networks, Journal of Scheduling 6 (2003), 13-34.

8. S.E. Elmaghraby, Activity Networks: Project Planning and Control by Network Models, New York: John Wiley and Sons, Inc, 1977.

9. S.E. Elmaghraby and W. Herroelen, On the measurement of complexity in activity networks, European Journal of Operational Research 5 (1980), 223-234.

10. W. Herroelen and B. De Reyck, Phase transitions in project scheduling, Journal of Operational Research Society 50 (1999), 148-156.

11. R.A. Kaimann, Coefficient of network complexity, Management Science 21 (1974), 172-177.

12. R.A. Kaimann, Coefficient of network complexity: Erratum, Management Science 21 (1975), 1211-1212.

13. E.P.C. Kao. and M. Queranne, On dynamic programming methods for assembly line balancing, Operations Research 30 (1982), 375-390.

14. R. Kolisch and A. Sprecher, PSPLIB - A project scheduling library, European Journal of Operational Research 96 (1996), 205-216.

15. R. Kolisch, A. Sprecher and A. Drexl, Characterization and generation of a general class of resource-constrained project scheduling problems, Management Science 41 (1995), 1693-1703.

16. A.A. Mastor, An experimental and comparative evaluation of production line balancing techniques, Management Science 16 (1970), 728-746.

17. J.H. Patterson, Project scheduling: the effects of problem structure on heuristic scheduling, Naval Research Logistics 23 (1976), 95-123.

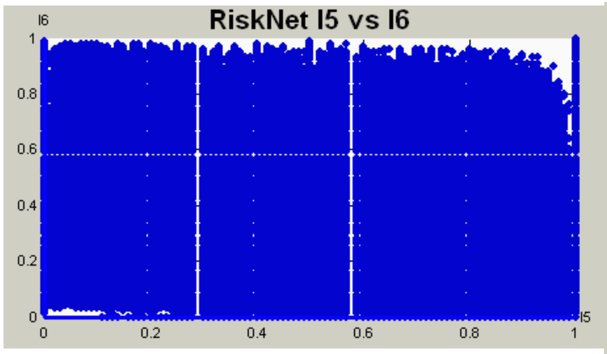
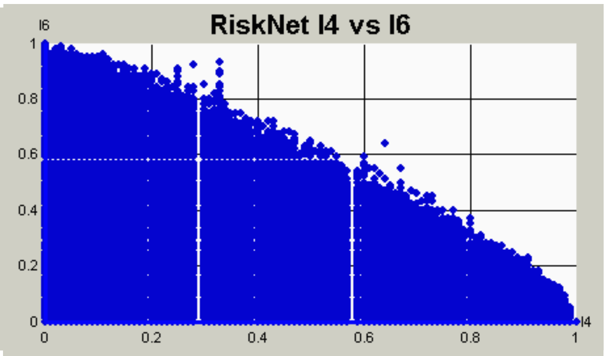
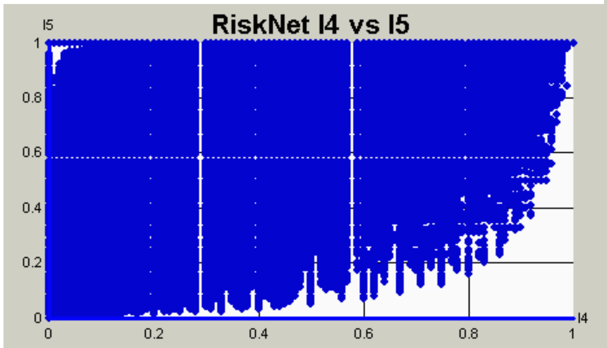
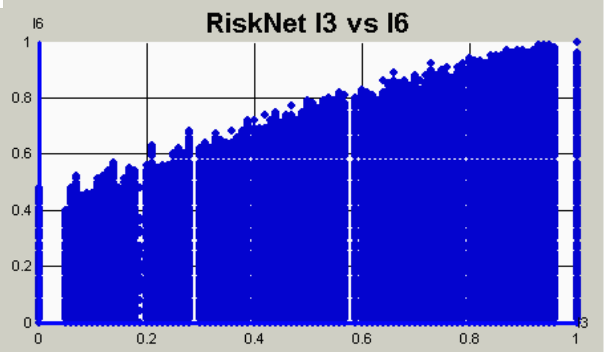
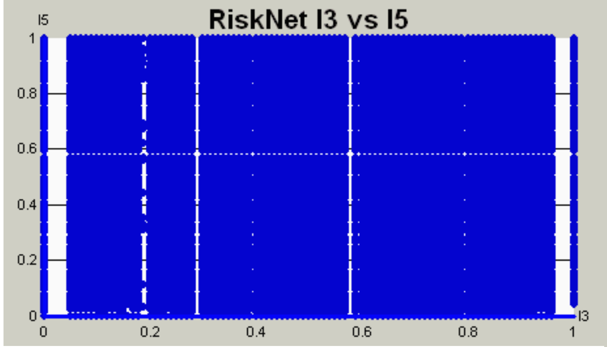
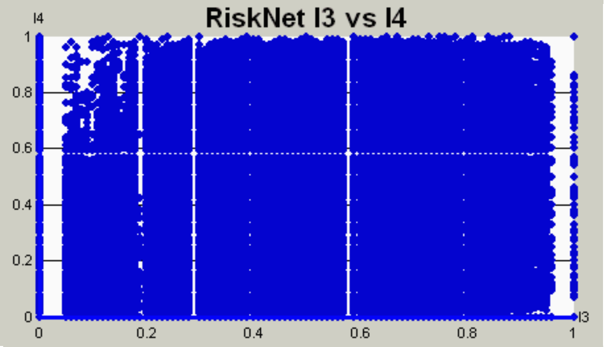
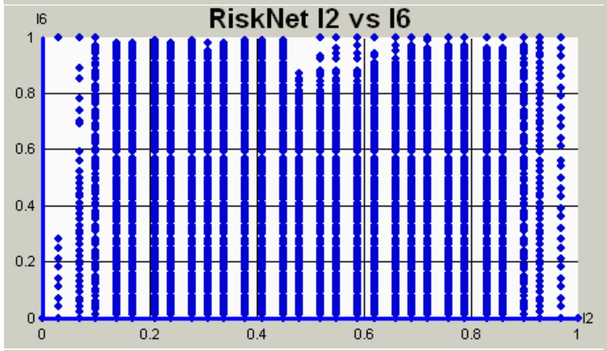
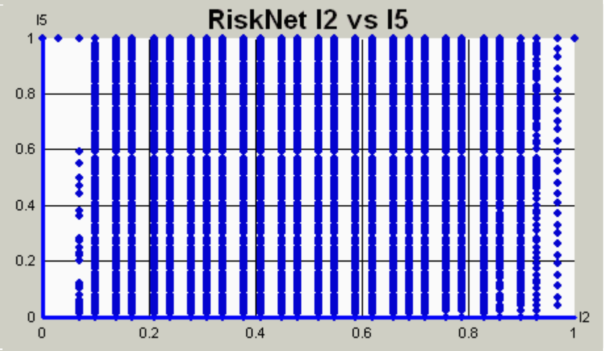
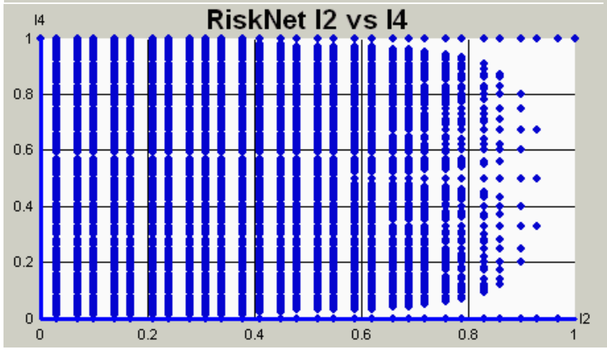
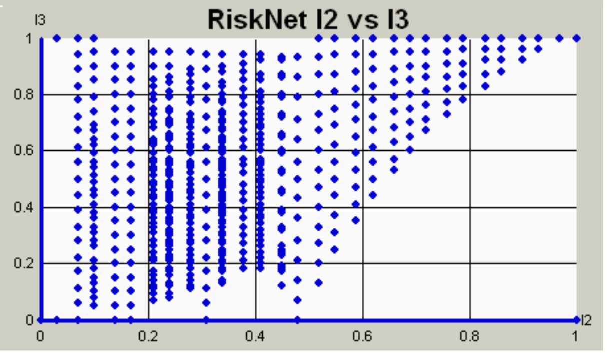
18. T.L. Pascoe, Allocation of resources - CPM, Revue Française de Recherche Opérationelle 38 (1966), 31-38.

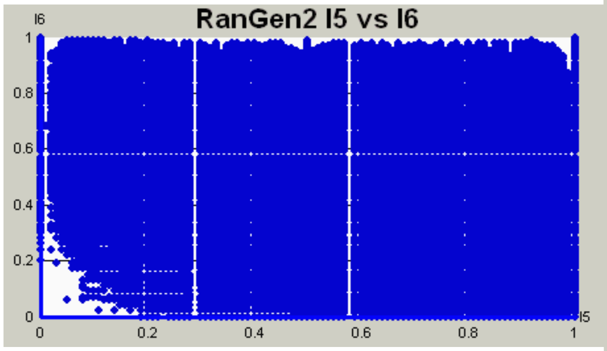
19. L.V. Tavares, Advanced Models for Project Management, Kluwer Academic Publishers, Dordrecht, 1999.

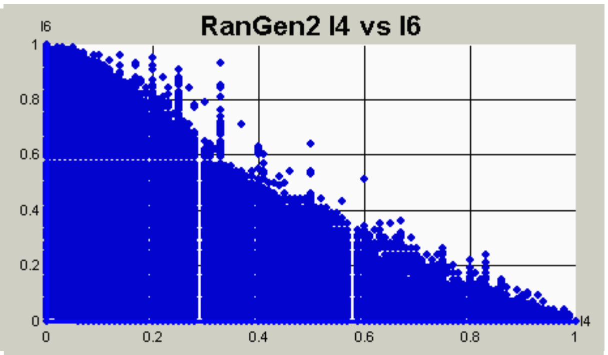
20. L.V. Tavares, J.A. Ferreira and J.S. Coelho, The risk of delay of a project in terms of the morphology of its network, European Journal of Operational Research 119 (1999), 510-537.

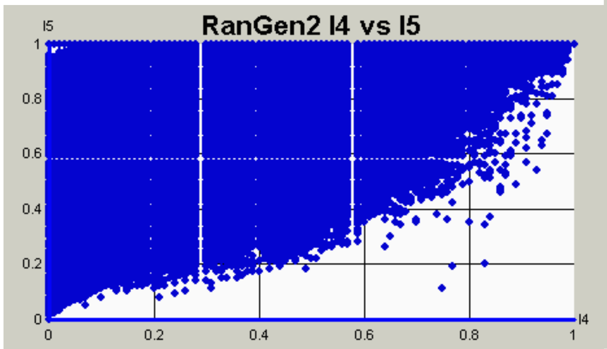
21. A. Thesen, Heuristic scheduling of activities under resource and precedence restrictions, Management Science 23 (1976), 412-422.

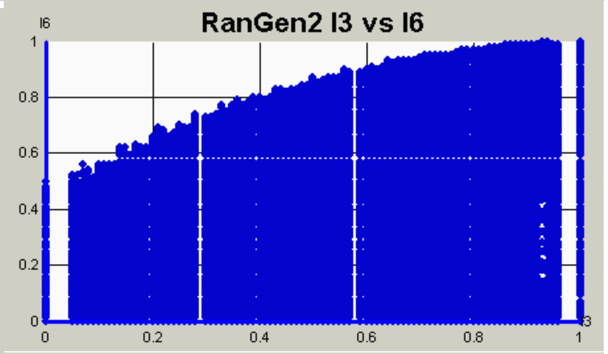
APPENDIX

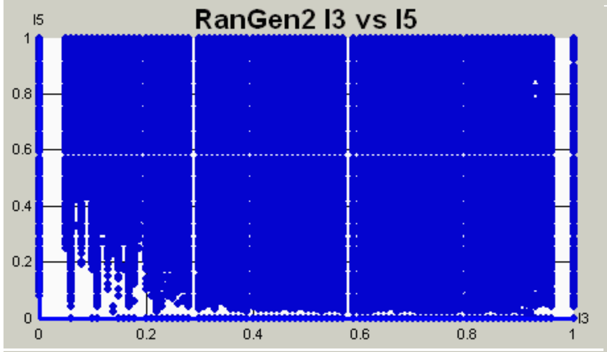

RiskNet I5 vs I6
I6
I5

RiskNet I4 vs I6
I6
I4

RiskNet I4 vs I5
I5
I4

RiskNet I3 vs I6
I6
I3

RiskNet I3 vs I5
I5
I3

RiskNet I3 vs I4
I4
I3

RiskNet I2 vs I6
I6
I2

RiskNet I2 vs I5
I5
I2

RiskNet I2 vs I4
I4
I2

RiskNet I2 vs I3
I3
I2

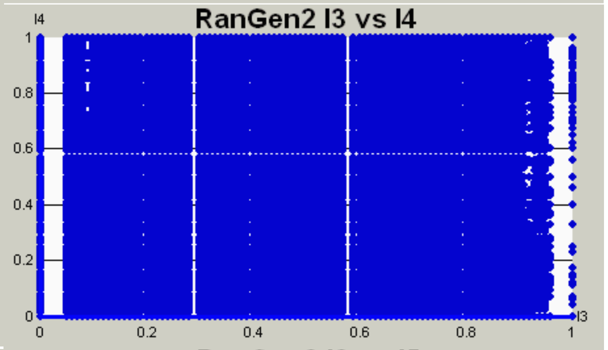

RanGen2 I5 vs I6
I6
I5

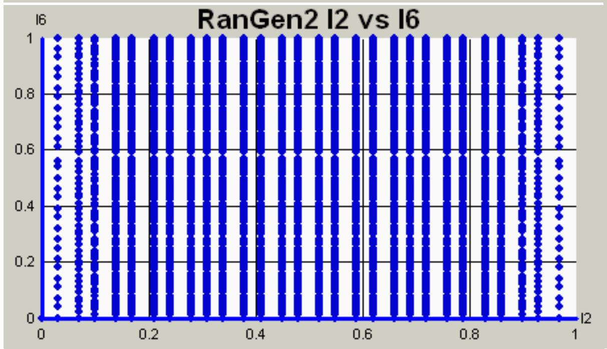

RanGen2 I4 vs I6
I6
I4

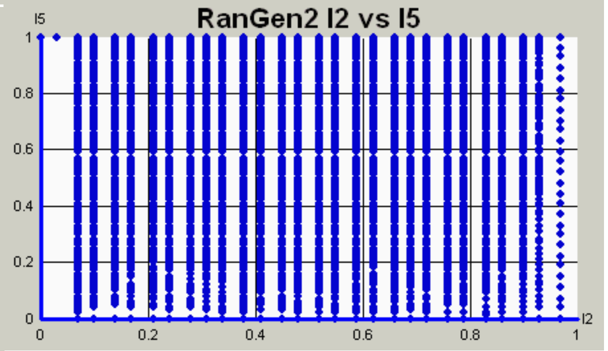

RanGen2 I4 vs I5
I5
I4

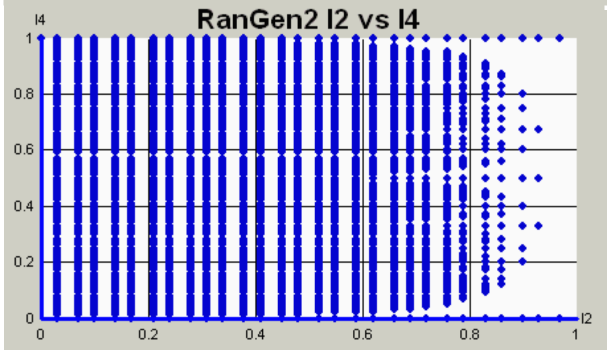

RanGen2 I3 vs I6
I6
I3

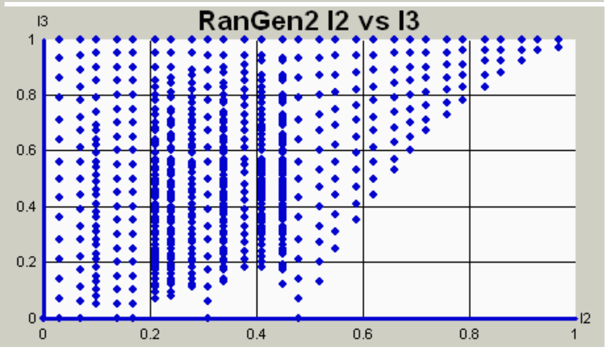

RanGen2 I3 vs I5
I5
I3

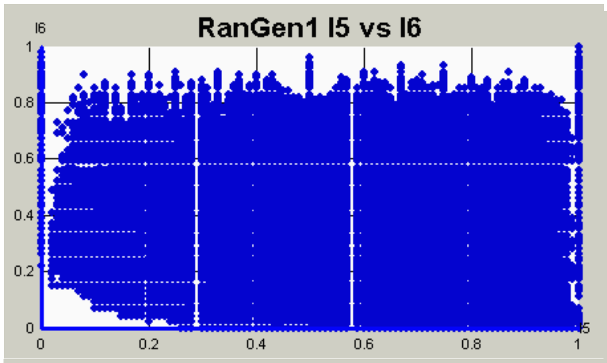
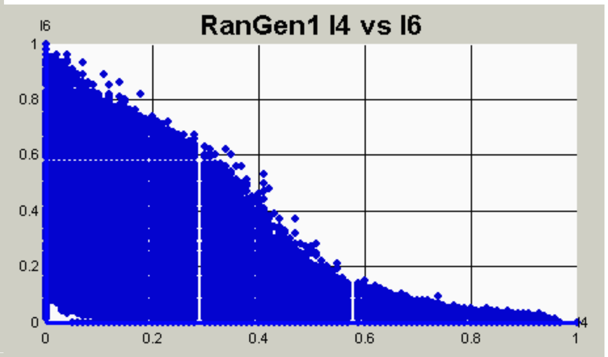
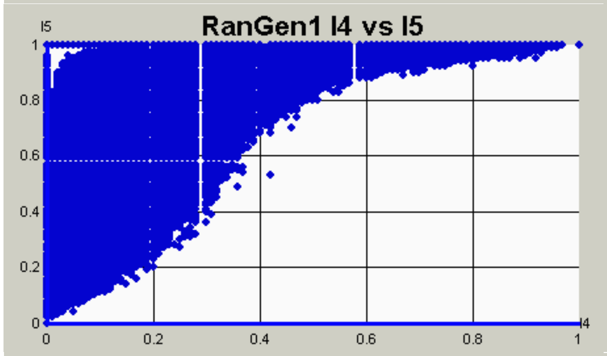
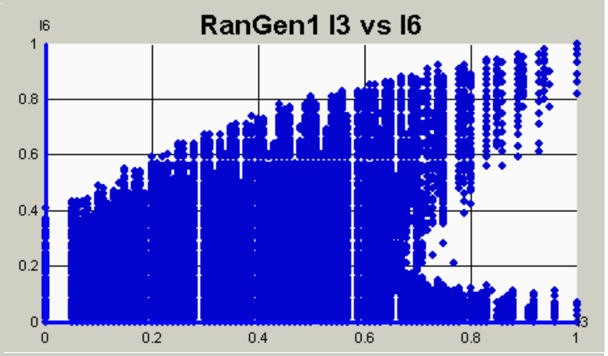
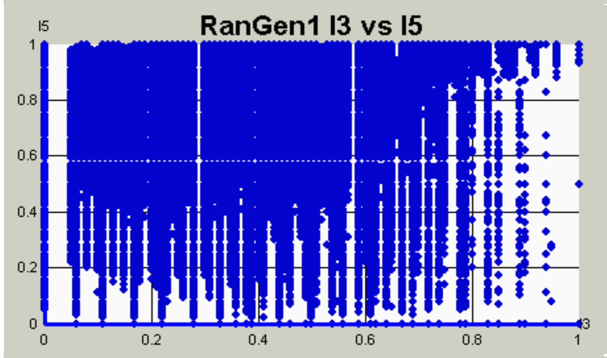
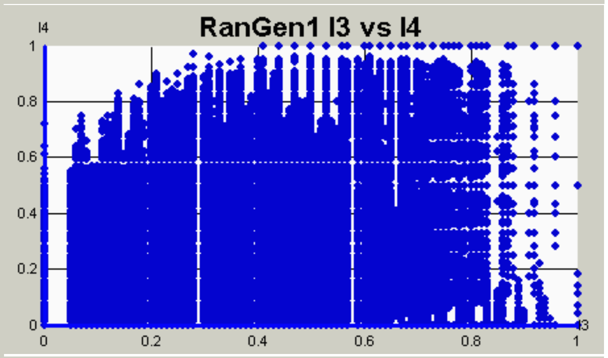
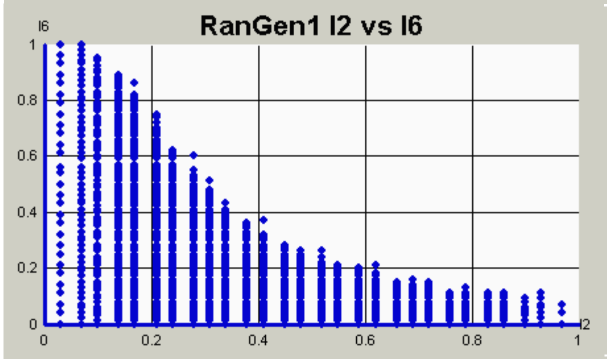
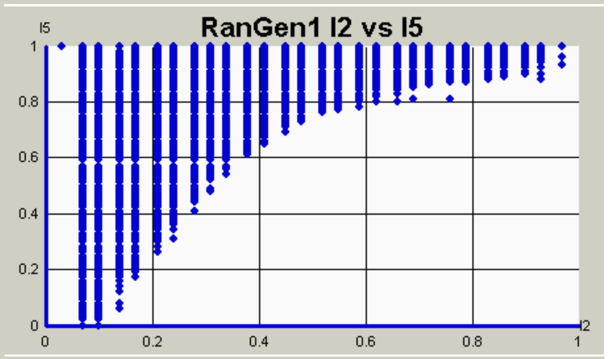
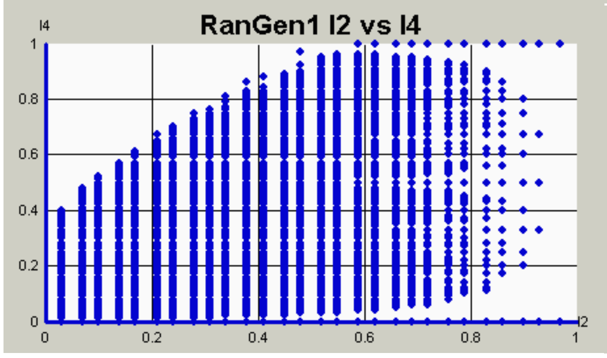
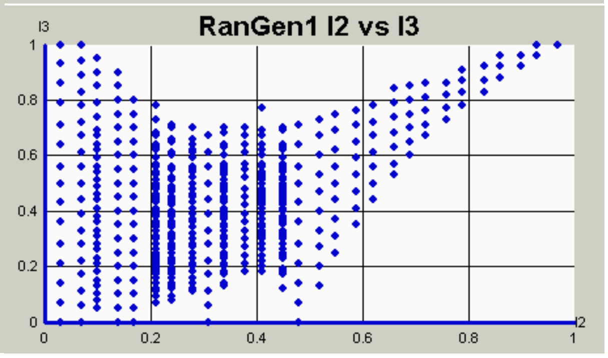

RanGen2 I3 vs I4
I4
I3

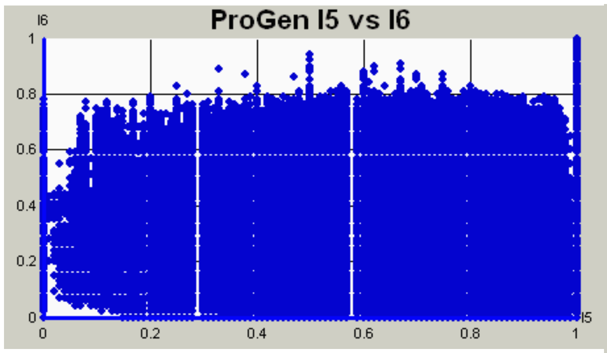

RanGen2 I2 vs I6
I6
I2

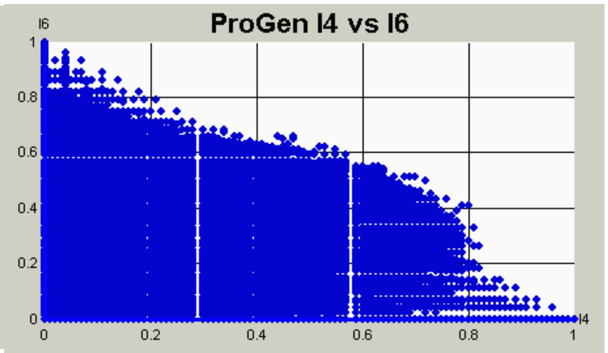

RanGen2 I2 vs I5
I5
I2

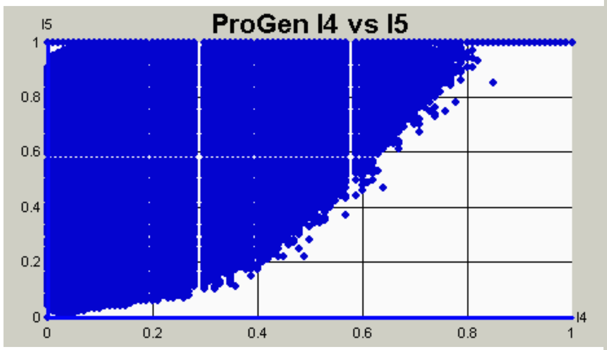

RanGen2 I2 vs I4
I4
I2

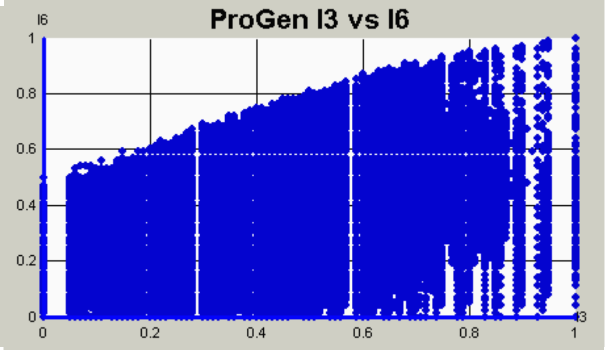

RanGen2 I2 vs I3
I3
I2

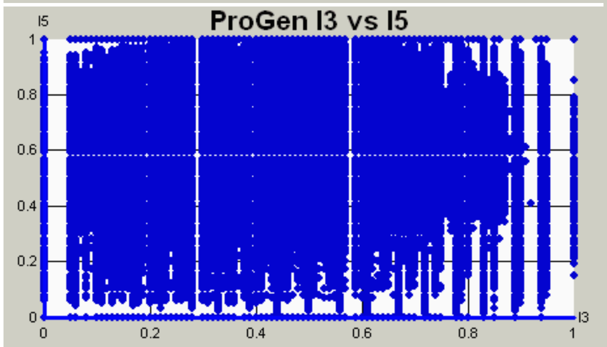

RanGen1 I5 vs I6
I6
I5

RanGen1 I4 vs I6
I6
I4

RanGen1 I4 vs I5
I5
I4

RanGen1 I3 vs I6
I6
I3

RanGen1 I3 vs I5
I5
I3

RanGen1 I3 vs I4
I4
I3

RanGen1 I2 vs I6
I6
I2

RanGen1 I2 vs I5
I5
I2

RanGen1 I2 vs I4
I4
I2

RanGen1 I2 vs I3
I3
I2

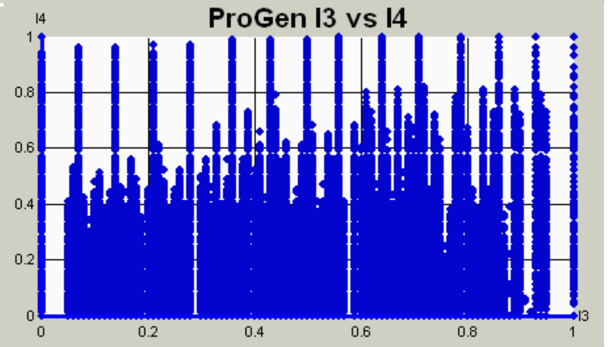

ProGen I5 vs I6
I6
I5

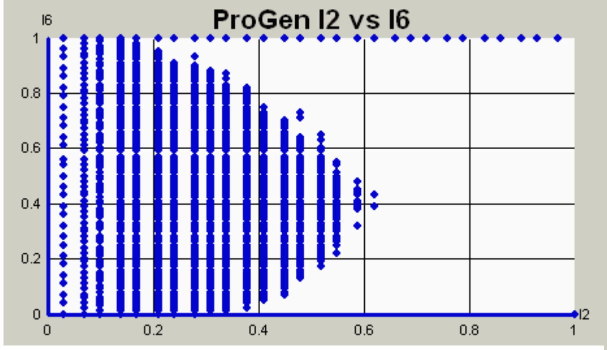

ProGen I4 vs I6
I6
I4

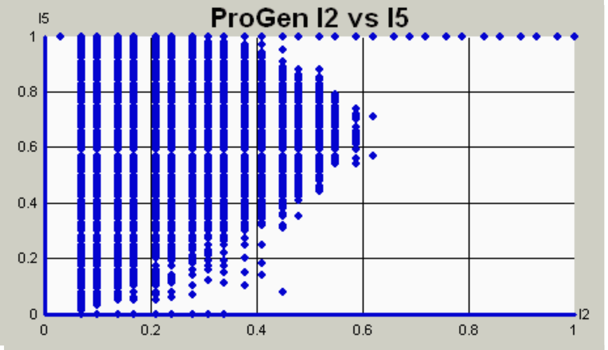

ProGen I4 vs I5
I5
I4

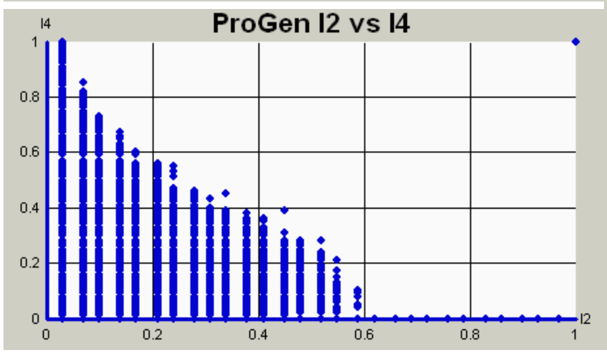

ProGen I3 vs I6
I6
I3

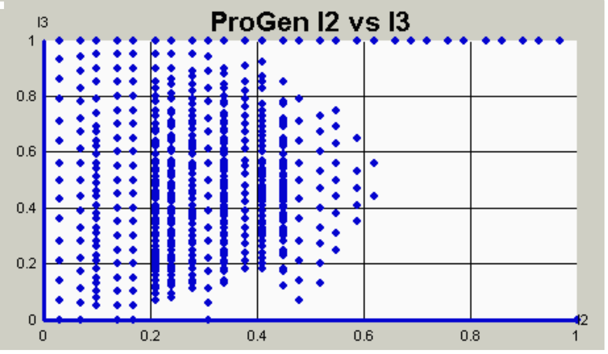

ProGen I3 vs I5
I5
I3

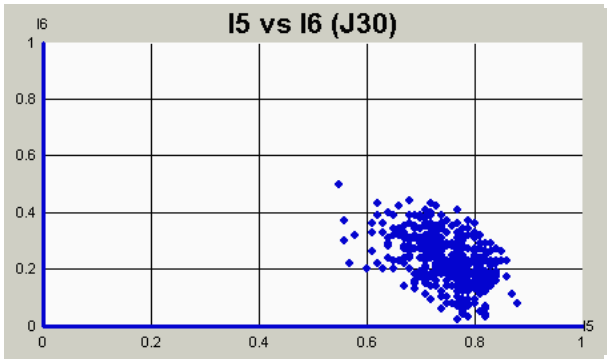

ProGen I3 vs I4
I4
I3

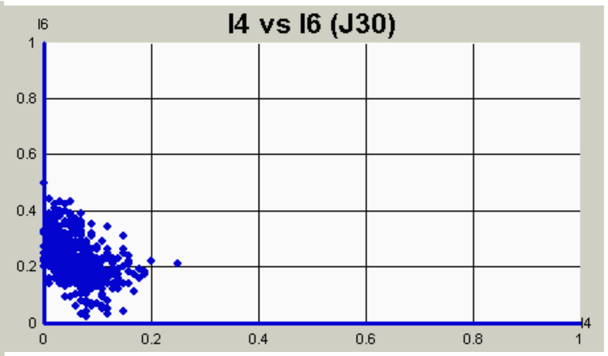

ProGen I2 vs I6
I6
I2

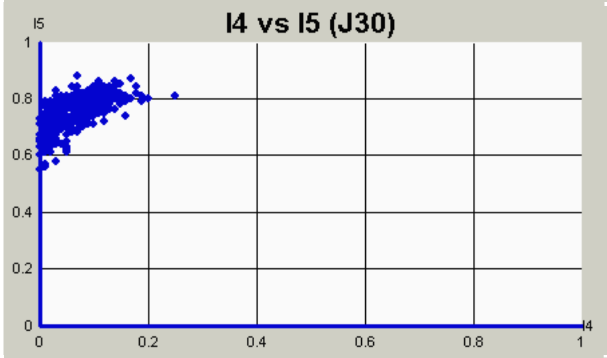

ProGen I2 vs I5
I5
I2

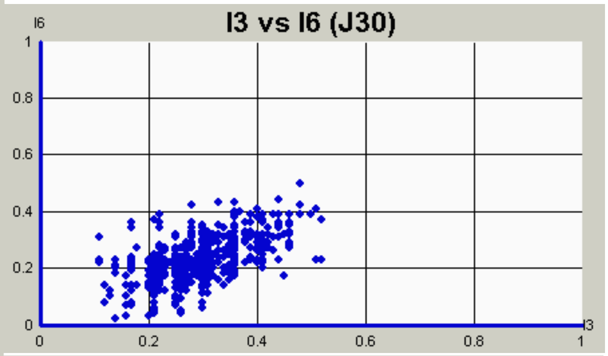

ProGen I2 vs I4
I4
I2

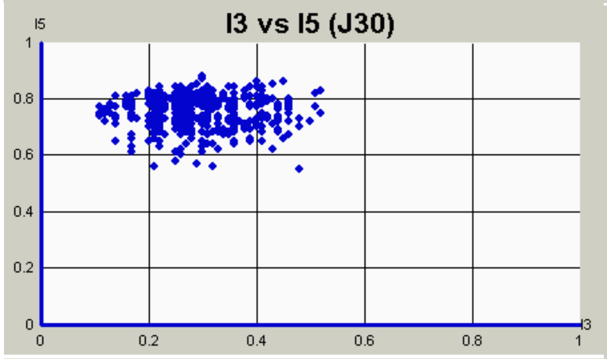

ProGen I2 vs I3
I3
I2

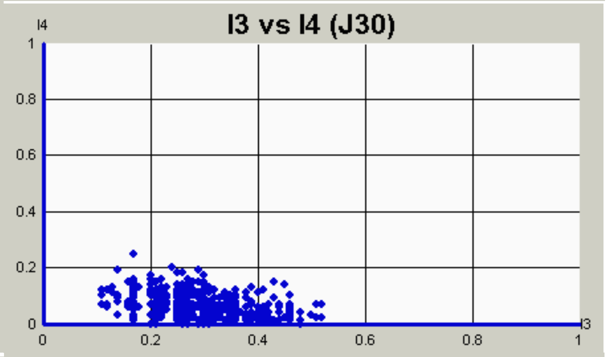

I5 vs I6 (J30)
I6
I5

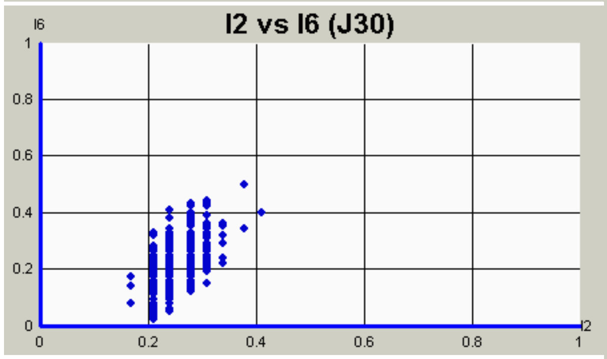

I4 vs I6 (J30)
I6
I4

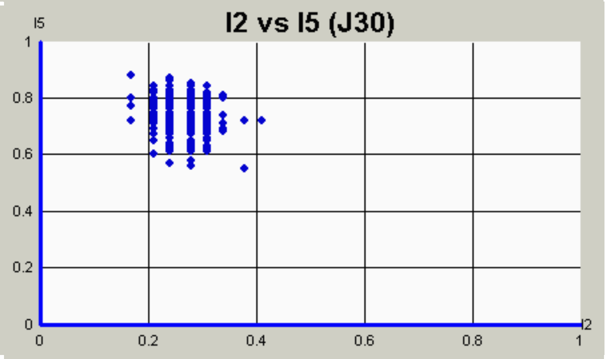

I4 vs I5 (J30)
I5
I4

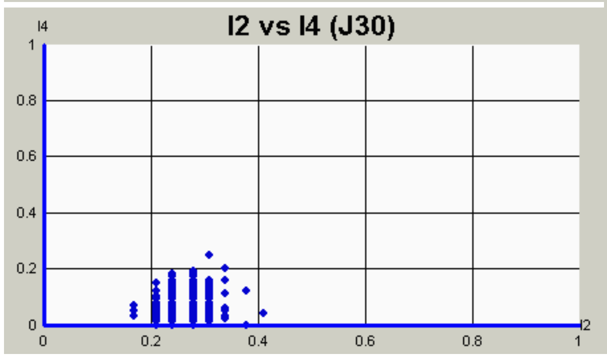

I3 vs I6 (J30)
I6
I3


I3 vs I5 (J30)
I5
I3


I3 vs I4 (J30)
I4
I3


I2 vs I6 (J30)
I6
I2


I2 vs I5 (J30)
I5
I2


I2 vs I4 (J30)
I4
I2

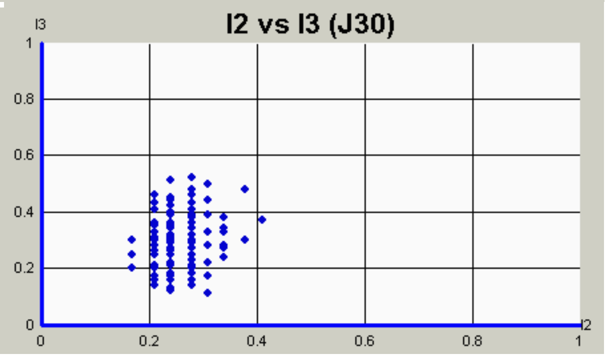

I2 vs I3 (J30)
I3
I2

FIGURE 1

An example project network with 10 activities

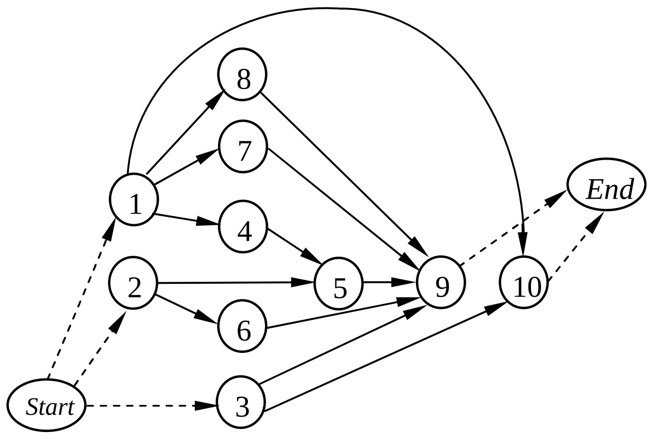


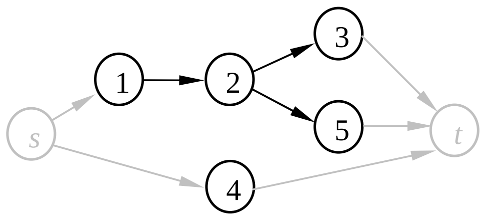
TABLE 1

Progressive and regressive level of each activity in figure 1

Activity i	PL_i	RL_i	*Activity i*	PL_i	RL_i
1	1	1	6	2	3
2	1	2	7	2	3
3	1	3	8	2	3
4	2	2	9	4	4
5	3	3	10	2	4

FIGURE 2

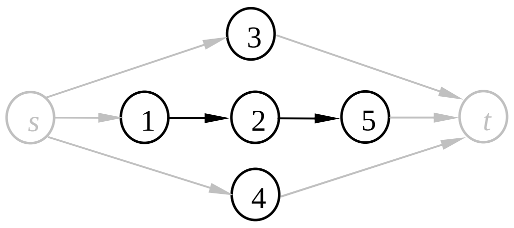
An example network with its Precedence Matrix representation PM



$$\begin{array}{c|cccc}
 & 2 & 3 & 4 & 5 \\
\hline
1 & 1 & 1 & - & 1 \\
2 & & 1 & - & 1 \\
3 & & & - & - \\
4 & & & &
\end{array}$$

FIGURE 3

A second example network with its Precedence Matrix representation PM



$$
\begin{array}{c|cccc}
 & 2 & 3 & 4 & 5 \\
\hline
1 & 1 & 1 & - & 1 \\
2 & & - & - & 1 \\
3 & & & - & - \\
4 & & & &
\end{array}
$$

TABLE 2

Results of the exhaustive generation of all 10-activity networks

Total number of networks found	2,567,284
Number of different *OS* values	46
Number of different I_2 values	10
Number of different I_3 values	23
Number of different I_4 values	81
Number of different I_5 values	91
Number of different I_6 values	49
Total number of networks with different I2 to I6 combinations	48,982

TABLE 3

The correlation matrix of the OS and the indicators I_2, I_3, I_4, I_5 and I_6

	OS	I_2	I_3	I_4	I_5	I_6
OS	1	0.70	0.09	0.30	0.24	-0.26
I_2		1	0.42	-0.21	-0.02	0.36
I_3			1	0.03	0.16	0.45
I_4				1	0.67	-0.44
I_5					1	-0.32
I_6						1

TABLE 4

Computational results for the four network generators

	***RanGen* 1**	***RanGen* 2**	***ProGen***	***RiskNet***
$RG\,1$	450,593			
$RG\,2$		6,052,338		
$RG\,1 \cup RG\,2$	1,272,039	1,272,039		
RN				1,713,331
$RG\,1 \cup RN$	9,512			9,512
$RG\,2 \cup RN$		207,536		207,536
$RG\,1 \cup \mathrm{RG2} \cup RN$	41,887	41,887		41,887
PG			4,772,146	
$RG\,1 \cup PG$	480,579		480,579	
$RG\,2 \cup PG$		1,986,598	1,986,598	
$RG\,1 \cup \mathrm{RG2} \cup PG$	1,893,001	1,893,001	1,893,001	
$PG \cup RN$			118,728	118,728
$RG\,1 \cup PG \cup RN$	15,243		15,243	15,243
$RG\,2 \cup PG \cup RN$		40,382	40,382	40,382
$RG\,1 \cup RN\,2 \cup PG \cup RN$	51,381	51,381	51,381	51,381
Total	4,214,235	11,545,162	9,358,058	2,198,000
% Total	22.06	60.43	48.98	11.50
% New - Generator	10.69	52.42	51.00	77.95
% New - Total	2.36	31.68	24.98	8.97

FIGURE 4

Total number of networks found by RanGen2, ProGen and RiskNet

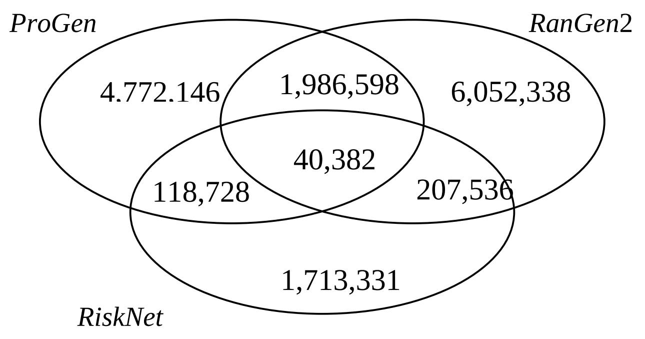


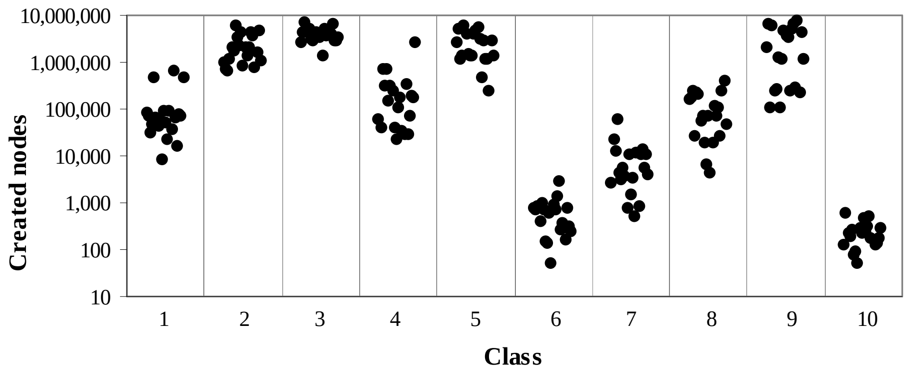
TABLE 5

10 classes of network instances with different indicator values

Class	I_1	I_2	I_3	I_4	I_5	I_6
1	30	0.17	0.20	0.14	0.75	0.25
2	30	0.17	0.45	0.10	0.76	0.60
3	30	0.17	0.60	0.03	0.77	0.75
4	30	0.17	0.30	0.12	0.71	0.25
5	30	0.17	0.50	0.06	0.70	0.54
6	30	0.38	0.30	0.16	0.86	0.16
7	30	0.38	0.42	0.13	0.75	0.32
8	30	0.38	0.45	0.02	0.66	0.50
9	30	0.38	0.58	0.00	0.66	0.80
10	30	0.38	0.55	0.02	0.50	0.72

FIGURE 5

The number of created nodes for each class for the RCPSP


Created nodes
Class