On the use of the complexity index as a measure of complexity in activity networks

Bert DE REYCK
Singapore Management University, bdreyck@smu.edu.sg

Willy HERROELEN

ELSEVIER

European Journal of Operational Research 91 (1996) 347–366

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

On the use of the complexity index as a measure of complexity in activity networks

Bert De Reyck, Willy Herroelen *

Department of Applied Economics, Katholieke Universiteit Leuven, Hogenheuvel College, Naamsestraat 69, B-3000 Leuven, Belgium,

Received 30 August 1994; revised 28 November 1994

Abstract

A large number of optimal and suboptimal procedures have been developed for solving combinatorial problems modeled as activity networks. The need to differentiate between easy and hard problem instances and the interest in isolating the fundamental factors that determine the computing effort required by these procedures, inspired a number of researchers to develop various complexity measures. In this paper we investigate the relation between the hardness of a problem instance and the topological structure of its underlying network, as measured by the complexity index. We demonstrate through a series of experiments that the complexity index, defined as the minimum number of node reductions necessary to transform a general activity network to a series–parallel network, plays an important role in predicting the computing effort needed to solve easy and hard instances of the multiple resource-constrained project scheduling problem and the discrete time/cost trade-off problem.

Keywords: Project planning; Network complexity measure; Complexity index; Network reduction

1. Introduction

A large number of optimal and suboptimal procedures have been described for solving combinatorial problems modeled as activity networks. Testing the accuracy and efficiency of these procedures requires the use of a set of benchmark instances. Ideally, such a set should span the full range of complexity, from very easy to very hard instances. The generation of easy and hard problem instances, however, appears to be a very difficult task which heavily depends on the possibility to isolate the factors that precisely determine the computing effort required by the solution procedure used to solve a problem, and the calibration of the scale that characterizes such effort. It is not surprising then that only a few commonly used benchmark instances are available in the field of activity networks. Patterson (1984) has assembled 110 test problems for the resource-constrained project scheduling problem under the objective of minimizing the makespan. These 110 test problems became a quasi standard in the field, and have been adapted to accommodate additional problem parameters and objectives (Demeulemeester et al., 1994; Baroum and Patterson, 1993). The problems, however, are a collection from different sources and have not

* Corresponding author. E-mail: Willy.Herroelen@econ.kuleuven.ac.be

been generated by using a controlled design of specified problem parameters. In addition, recent advances in the development of optimal procedures for several classes of resource-constrained project scheduling problems (Herroelen and Demeulemeester, 1995) have demonstrated that the Patterson set is solvable within very small average CPU times on a personal computer, which inspired some researchers to question the true benchmark nature of the set and to build an activity network generator (Kolisch et al., 1992). Kolisch et al. (1992) are correct in stating that the Patterson set has misled a number of researchers in believing that the resource-constrained project scheduling problem has become quite tractable. They have proven the opposite using 480 networks generated using their own network generator. The activity network generator originally developed by Caestecker and Herroelen (1979) and subsequently refined by Demeulemeester et al. (1993a), allows to generate dense and non-dense network structures at random from the space of all feasible networks. ProGen, the network generator developed by Kolisch et al. (1992), goes a step further in that it allows for the generation of activity network problem instances for a general class of resource-constrained project scheduling problems by using a controllable set of specified problem parameters.

Since the mid-sixties, parameters for the characterization of activity networks have been receiving attention from researchers who were interested in studying the effects of problem structure on algorithm performance (Davis, 1975; Patterson, 1976) and the development of a reliable set of measures of activity network 'complexity'. Evidently, a choice between algorithms or the determination of the efficiency of a particular algorithm, would be greatly facilitated if there exists a measure of network complexity. This would eliminate any possible bias in the conclusions regarding the efficiency of a particular algorithm relative to others by ensuring that the algorithm is evaluated at several points in the 'range of complexity' (Elmaghraby and Herroelen, 1980).

Quite a number of activity network 'complexity' measures have been proposed in the literature (Davis, 1975; Patterson, 1976). In this paper, our interest is limited to measures which aim to characterize the topological structure of an activity network, i.e., the network topology. The best known measure for the network topology is the *coefficient of network complexity* (CNC), introduced by Pascoe (1966) for activity-on-the-arc (AoA) networks, and simply defined as the ratio of the number of arcs over the number of nodes (different definitions have been used by Davies (1974) and Kaimann (1974, 1975)). The CNC has been adopted by Davis (1975) for the activity-on-the-node (AoN) representation and has been used in a number of studies since then (Kurtulus and Narula, 1985; Patterson, 1984; Talbot, 1982). As observed by Kolisch et al. (1992), in the AoN representation, 'complexity' has to be understood in the way that for a fixed number of activities (nodes), a higher complexity results in an increasing number of arcs and therefore in a greater connectedness of the network. A number of studies in the literature (Alvarez-Valdes and Tamarit, 1989; Kolisch et al., 1992) seem to confirm that problems become easier with increasing values of the CNC, which makes the term CNC somewhat confounding. Elmaghraby and Herroelen (1980) already questioned the use of the CNC as a measure of activity network complexity. The measure totally relies on the count of activities and nodes in the network. Since it is easy to construct networks of equal number of arcs and nodes but varying degrees of difficulty in analysis, they failed to see how the CNC can discriminate among them.

Recently Bein et al. (1992) introduced a new characterization of two-terminal acyclic networks which essentially measures how nearly series–parallel a network is. They define the *reduction complexity* as the minimum number of node reductions sufficient (along with series and parallel reductions) to reduce a two-terminal acyclic network to a single edge. We adopt the reduction complexity as our definition of the *complexity index* (CI) of an activity network. The objective of this paper is to investigate the potential use of the CI as a measure of activity network complexity. Elmaghraby and Herroelen (1980) argue that the measurement of network complexity – the measurement of the difficulty in analysis and synthesis of a given network – cannot be accomplished in a meaningful manner unless the use of the measure, i.e., the objective of analysis, is specified a priori. In addition, they argue that the measure of complexity may be confounded by the algorithm employed. We want to investigate the use of the CI as an explaining factor of the impact of network topology on algorithmic performance. Two problems are chosen for our analysis: the well-known resource-constrained project scheduling problem (RCPSP) and the discrete time/cost trade-off problem (DTCTP). The algorithm used for solving the RCPSP is the branch-and-bound procedure developed by Demeulemeester and Herroelen (1992). For the DTCTP we use the optimal procedure developed by Demeulemeester et al. (1993b), which is based on the procedure developed by Bein et al. (1992) for finding the minimum number of node reductions necessary to transform a general network to a series–parallel network.

The remainder of the paper is organized as follows. In Section 2, we give a formal definition of the complexity index (CI) and elaborate on its anticipated use as a measure of network complexity. In Section 3, we report on the computational experiment performed on the RCPSP, which forces us to conclude that the CNC is not a very good measure of network complexity, while the explanatory behaviour of the CI is more significant. The higher the value of the CI, the easier it is to solve the RCPSP. Section 4 is then reserved for the experiment conducted on the DTCTP. While the correlation between required CPU time and CI observed for the RCPSP was negative, this is not the case for the DTCTP: the higher the value of the CI, the harder the problem. Now that we have established that the CI can be used as an indicator of the hardness of RCPSP and DTCTP instances, Section 5 concentrates on the explanatory power of resource availability measures for given values of the CI. Using the RCPSP as our vehicle of analysis, we confirm the conjecture made by Elmaghraby and Herroelen (1980) that the relationship between the hardness of a problem (measured by the CPU time required for its solution) and resource scarcity (measured by the so-called resource strength and resource-constrainedness) varies according to a bell-shaped curve. Section 6 is then reserved for our overall conclusions.

2. The complexity index and its relation to problem hardness

2.1. The complexity index

Let $D=(N,A)$ be a two-terminal AoA network, where $N=1,2,\dots,n$ is the set of nodes representing the project events, and A is the set of arcs representing network activities. We assume, without loss of generality, that there is a single start node 1 and a single terminal node n, $n=|N|$. Since D is acyclic, we assume that its nodes are topologically numbered, i.e., $i<j$ whenever there exists an arc joining i to j. The resulting graph will be referred to as a *st-dag*.

Bein et al. (1992) define complexity in terms of a sequence of node reductions of a st-dag. There are three types of reductions: parallel, series and node reductions. These reductions, when applied consecutively in the right order, can reduce any dag to one single arc. A *parallel reduction* at i,j replaces two or more arcs a_1, a_2, …, a_k, all joining i to j, by a single arc $a=(i,j)$. A *series reduction* at i is possible when $a=(i,j)$ is the unique arc into j and $b=(j,k)$ is the unique arc out of j: a and b are replaced by a single arc $c=(i,k)$. Let $[D]$ denote the network obtained by applying to D all series–parallel arc reductions. If $D=[D]$ then D is said to be *irreducible*.

Following Bein et al. (1992), we say that node j of an irreducible network is eligible for a *node reduction* when j has unit in-degree or out-degree, and $j\neq 1,n$. Let $a=(i,j)$ be the unique arc into j and $b_1=(j,k_1)$, …, $b_s=(j,k_s)$ be the arcs out of j. Then the reduction of node j replaces a, b_1, …, b_s by the arcs $c_1=(i,k_1)$, …, $c_s=(i,k_s)$. The case where j has unit out-degree is symmetric. Note that in an irreducible network any node whose only predecessor is 1 or whose only successor is n is eligible for reduction. Therefore, every acyclic network can be reduced to the single arc $(1,n)$ by a sequence of node reductions interleaved with series and parallel reductions. The number of node reductions in such a sequence may differ. Bein et al. (1992) define the reduction complexity of D as the minimum number of node reductions sufficient (along with series and parallel reductions) to reduce D to a single arc. More formally, let Doj denote the network obtained from reduction of node j in D. Then the reduction complexity is the smallest q for which there exists a sequence of nodes $(j_1,j_2,\dots,j_q)$ such that $[\dots[[[D]oj_1]oj_2]\dots oj_q] = (1,n)$. Such a sequence is called a *reduction sequence*. The length of a reduction sequence, i.e., the reduction complexity, is taken as our definition of the *complexity index*, CI, of D. Since all series-parallel networks have a CI value equal to zero, the complexity index CI seems to be a good measure of how close the network is to being series-parallel. Bein et al. (1992) have developed an algorithm for calculating the CI of a dag in polynomial time. For ease of reference, Appendix A summarizes the fundamental concepts underlying the complexity index and illustrates these concepts using a small problem example.

2.2. The complexity index and the RCPSP

Essentially the CI measures how nearly series-parallel a network is. One of the objectives of this paper is to investigate its potential to measure the impact of network topology on the computational effort required by an algorithm to solve resource allocation problems in activity networks. The first resource allocation problem addressed in this paper is the multiple resource-constrained single-project scheduling problem (RCPSP), in which it is assumed that an activity is subject to technological precedence constraints (an activity can only be started if all its predecessor activities have been finished) and cannot be interrupted once begun (no job preemption allowed). *Renewable resources* are assumed to be available per period in constant amounts and are also demanded by an activity in constant amounts throughout the duration of the activity. The objective is to schedule the activities subject to precedence and resource constraints in order to minimize the total project duration.

Conceptually, the RCPSP can be formulated as follows:

$$\min t_n \tag{1}$$

subject to

$$t_j - t_i \geq p_j, \quad (i,j) \in H, \tag{2}$$

$$\sum_{i \in S_t} r_{ik} \leq a_k, \quad t = 1,2,\dots,t_n, \quad k = 1,2,\dots,K \tag{3}$$

where t_i is the finish time of activity i, $i = 1,2,\dots,n$; H is the set of pairs of activities indicating precedence constraints; p_i is the fixed processing time of activity i; r_{ik} is the amount of resource type k required by activity i; S_t is the set of activities in process in time interval $(t-1,t\,] = \{i \mid t_i - p_i < t \leq t_i\}$; and a_k is the total availability of renewable resource type k.

The precedence constraints given by Eq. (2) indicate that an activity j can only be started if all predecessor activities i are completed. The resource constraints given in Eq. (3) indicate that for each time period $(t-1,t]$ and for each resource type k, the resource amounts required by the activities in progress cannot exceed the resource availability. The objective function is given as Eq. (1). The project duration is minimized by minimizing the finish time of the unique dummy ending activity n.

Demeulemeester and Herroelen (1992) developed a branch-and-bound procedure for solving the RCPSP which currently seems to be the most advanced exact procedure for solving makespan minimization problems. The procedure (subsequently referred to as DH) has been programmed in Turbo C for use on a personal computer. DH is a depth-first branch-and-bound procedure based on an AoN representation of a project network, which makes use of a critical-path lower bound and some very powerful dominance rules (left-shift rule, cutset rule, minimal delaying alternatives rule and two other rules based on activities which cannot be scheduled simultaneously with other activities). In Section 3, we investigate whether the complexity index (CI) can be used as a measure of network complexity for the RCPSP, using the DH procedure as our vehicle of analysis. It will be shown that the CI is negatively correlated with the computational effort needed to solve the RCPSP, i.e., the higher the value of CI, the easier it is to solve the corresponding RCPSP. This result may seem to be counterintuitive. It implies that the more complex the network topology, the smaller the number of feasible parallel paths, which renders the RCPSP more tractable.

2.3. The complexity index and the DTCTP

The second resource allocation problem setting used in this paper is the discrete time/cost trade-off problem (DTCTP) in activity networks of the CPM type, using a single *nonrenewable resource*. Note that this problem, contrary to the RCPSP, is based on an AoA representation of a project network. We assume that the duration y_a of activity $a \in A$ is a discrete, nonincreasing function $p_a(x_a)$ of the amount of a single resource allocated to it, i.e., $y_a = p_a(x_a)$. The pair y_a, x_a shall be referred to as a '*mode*', and shall be written as: $y_a(x_a)$. Thus an activity that assumes four different durations according to four possible resource allocations to it shall be said to possess four modes. Demeulemeester et al. (1993b) consider three possible objectives for the DTCTP. These objectives are to be used separately, not simultaneously (not a multicriteria problem). For the first objective function they specify a limit R on the total availability of a single nonrenewable resource type. The problem is then to decide on the vector of activity durations $(y_1,\dots,y_m)$, $m = |A|$, that completes the project as early as possible under the limited availability of the single nonrenewable resource type. If we denote an activity a by its end nodes i and j and if we let t_i denote the (earliest) realization time of node i, then the problem can be formulated as follows:

$$\min t_n \tag{4}$$

subject to

$$t_1 = 0, \tag{5}$$

$$t_i + p_{ij}(x_{ij}) \le t_j \quad \text{for all } (ij) \in A, \tag{6}$$

$$\sum_{(ij)\in A} x_{ij} \le R. \tag{7}$$

In the objective function (4) we minimize the realization time of the single terminal node n, where Eq. (5) indicates that the project is started at time 0. Constraint set (6) is used to satisfy the precedence constraints, while constraint set (7) indicates that the limit on the resource availability cannot be violated.

A second objective function reverses this problem formulation: now we specify a limit T on the project length and we try to minimize the resource usage. Using the same notation as for the previous formulation, this problem may be stated as follows:

$$\min \sum_{(ij)\in A} x_{ij} \tag{8}$$

subject to

$$t_1 = 0, \tag{9}$$

$$t_i + p_{ij}(x_{ij}) \le t_j \quad \text{for all } (ij) \in A, \tag{10}$$

$$t_n \le T. \tag{11}$$

In this formulation constraints (9) and (10) are identical to constraints (5) and (6), but now constraint (11) specifies that the maximal project length T cannot be violated. The objective function (8) minimizes the sum of the resource usage over all the activities.

For the third objective function we have to compute the complete time/cost trade-off function for the total project, i.e., in the case of the DTCTP all the efficient points (T, R) such that with a resource limit R a project length T can be obtained and such that no other point (T', R') exists for which both T' and R' are smaller than or equal to T and R. It is this objective function which we consider in Section 4 in order to investigate the potential use of the CI as a measure of complexity for the DTCTP.

The optimal procedure (*Reduction Plan 1*) developed by Demeulemeester et al. (1993b), will be used as our vehicle of analysis. The procedure is based on the method by Bein et al. (1992) for determining the minimum number of node reductions necessary to transform a general acyclic network to a series–parallel network. The procedure (subsequently referred to as DEH) has been coded in Turbo C for use on a personal computer. DEH is a depth-first branch-and-bound procedure based on an AoA representation of a project network, in which in each node of the branch-and-bound tree the duration of a certain activity which is eligible for reduction is fixed. When the duration of all activities which have to be reduced are fixed, the resulting project cost curve can be calculated using dynamic programming logic. Some lower bounds and dominance rules are added to improve efficiency.

In Section 4, we investigate whether the CI can be used as a measure of network complexity for the DTCTP using the DEH procedure on a set of test problems. It will be shown that the CI has a positive correlation with the computational effort required to solve the DTCTP: the higher the value of the CI, the harder the corresponding DTCTP.

3. The RCPSP and network complexity

In this section we investigate the potential use of both the coefficient of network complexity (CNC) and the complexity index (CI) as a measure of network complexity for the RCPSP. A full factorial experiment would require the generation of networks with prespecified values of CNC and CI. A procedure for creating networks which satisfy preset values of the CI, however, is not yet available. In addition, ProGen, which is the most powerful project generator available, generates AoN networks, while the CI has been defined for AoA networks. As a result, a full factorial experiment is out of order. When we generate networks (in AoN format), we can prespecify the CNC, but not the CI, which makes it cumbersome to set up an experiment with a full factorial design, in which all levels of the independent variables are crossed. Instead, we proceeded as follows. Using ProGen, the project network generator developed by Kolisch et al. (1992), we generated five sets of 1000 RCPSP instances (AoN networks) each. Each network has a single start and end node. For each network, the number of activities is set to 25. The maximum number of predecessors, resp. successors is set to 25. Three resource types are assigned a constant availability of 6 units. The activity durations are drawn from the uniform distribution in the range [1, 10], while the resource requirements for each of the three resource types are drawn from the uniform distri-

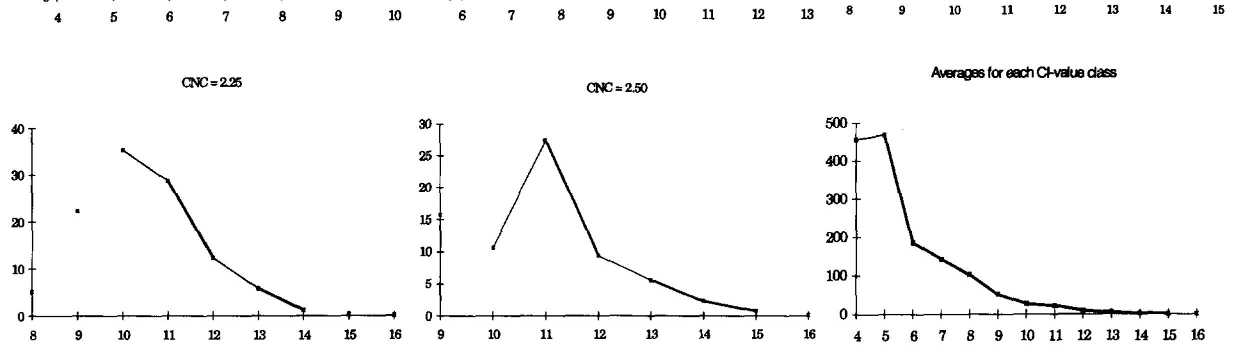


Fig. 1. Average processing time as a function of CI for constant CNC values.

bution in the range [1,5]. Both the activity durations and the resource requirements are kept constant over the 5000 instances. In each of the five sets, the CNC (arcs over nodes) is set at a different value, varying from 1.5 in the first set to 2.5 in the fifth.

Each AoN network is transformed to a corresponding AoA network with minimum CI value using the efficient *polynomial* procedure developed by Kamburowski et al. (1992). For ease of reference, this algorithm is illustrated on a small problem example in Appendix B. In each of the five problem sets, the AoA networks obtained are assigned to different classes, depending on the CI values. We obtain five data sets, with the following characteristics:

Set No.	CNC	CI
1	1.50	4–10
2	1.75	6–13
3	2.00	8–15
4	2.25	8–16
5	2.50	9–16

Each RCPSP instance is then solved using the DH procedure on a personal computer IBM PS/2 Model 70 A21. The plots of the CPU time required to solve the RCPSP instances to optimality, versus the CI values, are given in Fig. 1. Bold lines indicate significant differences using the Wilcoxon test statistic (a non-parametric test statistic based on order statistics). Only samples with more than 20 networks are connected by a line, because a smaller sample size does not allow for statistical inferencing.

Clearly, the CI has an effect on the computational effort required to solve the corresponding RCPSP. Moreover, the correlation is negative: the higher the CI of the network, the easier it is to solve the RCPSP. A loglinear regression performed on every set yielded the following equation:

$$\text{CPU time} = \exp(a + b \cdot \text{CI})$$

or

$$\ln(\text{CPU time}) = a + b \cdot \text{CI}$$

which resulted in the following:

Set No.	a	b	F	R^2
1	6.891	−0.738	94.88	8.69%
2	6.874	−0.618	80.99	7.52%
3	4.982	−0.441	53.08	5.06%
4	4.696	−0.438	58.52	5.56%
5	3.990	−0.371	39.32	3.88%

This result confirms our conjecture that the complexity is negatively correlated to the effort needed to solve the RCPSP ($p < 0.0001$). However, R^2 is only 6.1% on average, which means that only a small portion of the total variability can be explained by the CI. The variability in each set is quite high, especially for the problem set with CNC = 2.50 (which contains, on average, the easiest RCPSP instances). This is the main explaining factor for the irregularity observed in the plot of Fig. 1(e) for the fifth data set. The slight irregularities in Fig. 1(a) and Fig. 1(f) observed at low CI values are due to the fact that we imposed a maximum CPU time limit of 3600 s for the DH procedure. As a result, the highest CPU time values are somewhat deflated, because some of the RCPSP instances could not be solved to optimality within that time limit.

The small R^2 values obtained indicate that it may not be very realistic to use the CI for making individual predictions of the computational effort required by an algorithm for solving RCPSP instances. Comparisons of alternative solution procedures over a sufficient number of random networks, however, are very well possible, since there is a clear trend in the average processing times.

Fig. 2 shows the relationship between the average CPU time required for solving the 5000 RCPSP instances and the CNC. Bold lines indicate significant differences using the Wilcoxon test statistic. On first sight, the apparent negative correlation between the CNC and the required computational effort seems to confirm the results obtained by Kolisch et al. (1992) using the same algorithm. Similar results have been found by Alvarez-Valdes and Tamarit (1989) in their vali-

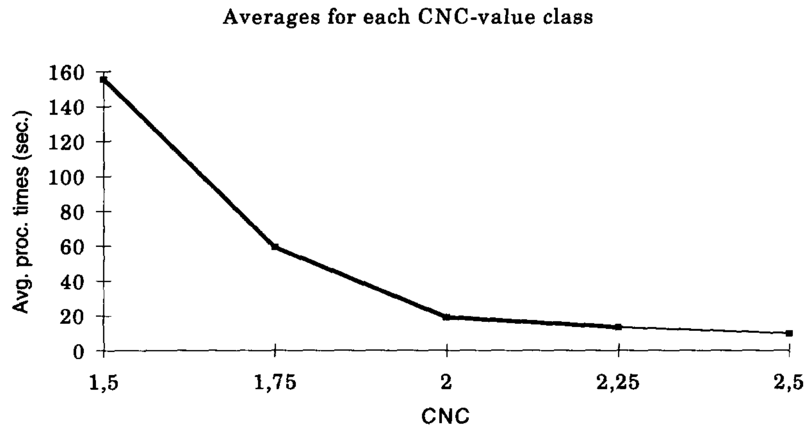


Fig. 2. Average processing time as a function of CNC for all 5000 RCPSP instances.

dation experiment of heuristics for solving the RCPSP.

It should be observed, however, that there is a positive correlation between the CNC and the CI. Fig. 3 again shows the results obtained on the same 5000 networks. The problems are now classified in 13 different sets, however, where the CI is the same for each set. The only varying parameter in each set is the CNC. Fig. 3 shows some of the corresponding plots of the relationship between the CPU time and the CNC for constant CI values. Again, only samples with more than 20 networks are connected by a line and bold lines indicate significant differences using the Wilcoxon test statistic. The negative correlation no longer shows. Apparently, the negative correlation effect

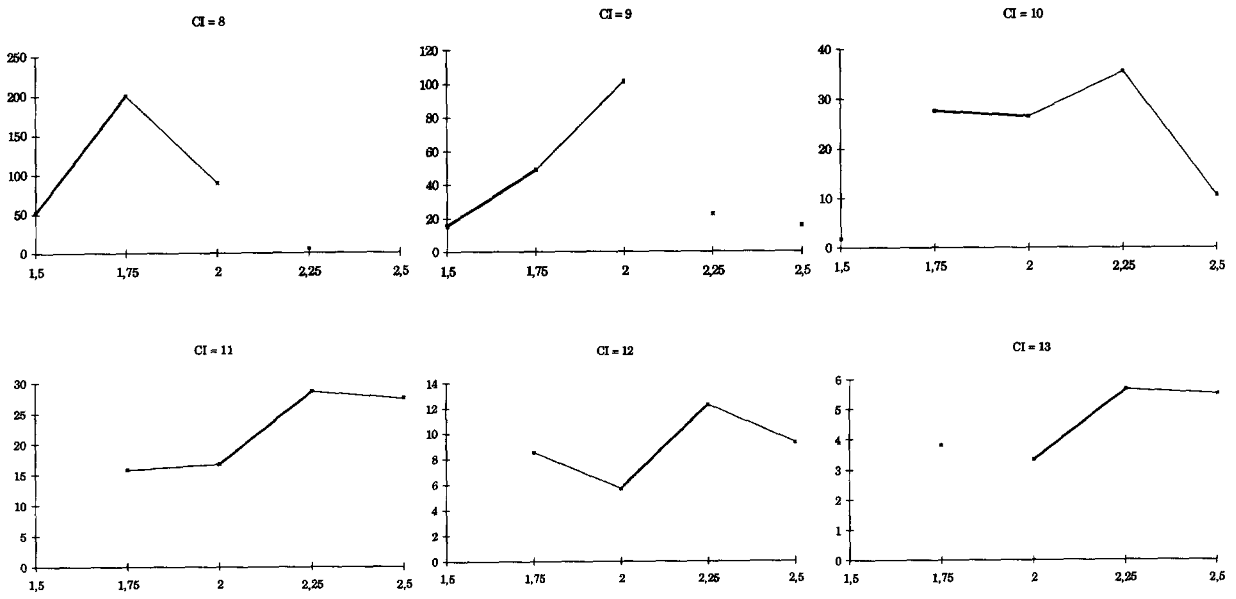


Fig. 3. Average processing time as a function of CNC for constant CI values.

measured in Fig. 2 over all 5000 networks is caused by the variation in the CI and the positive correlation between the CI and the CNC.

This result is quite interesting since it clearly shows that it is very ambiguous to attach all explanatory power to the CNC. Kolisch et al. (1992) attribute the observed negative correlation between the CNC and the solution times to the fact that adding more precedence relations to the network lowers the number of feasible schedules for a given bound on the project makespan. This reduces the enumeration tree and makes the problems more easy. In their experiment on 480 randomly generated networks, they set the CNC at 1.5, 1.8 and 2.1 respectively. We computed the CI values for each of the networks and obtained the following:

Set No.	CNC	CI
1	1.5	9–11
2	1.8	14–16
3	2.1	19–21

This clearly indicates that the explanatory power of the CNC is strongly confounded by its strong correlation with the CI.

In order to verify our conjecture that the CI is a better measure of network complexity than the CNC, we performed the following loglinear regression on all 5000 RCPSP instances:

$$\text{CPU time} = \exp(a + b \cdot \text{CI} + c \cdot \text{CNC})$$

or

$$\ln(\text{CPU time}) = a + b \cdot \text{CI} + c \cdot \text{CNC}.$$

This led to the following:

$$\text{CPU time} = \exp(5.222 - 0.470 \cdot \text{CI} + 0.001 \cdot \text{CNC})$$

with $F = 389.53$, $R^2 = 13.57\%$, standard error of first coefficient (CI) $= 0.025$ ($p < 0.0001$), and standard error of second coefficient (CNC) $=$ 0.168 ($p > 0.99$) $\Rightarrow$ insignificant

The coefficient related to the CI is negative, which indicates a negative correlation between the CI and the hardness of the RCPSP. Not only is the coefficient relating to CNC insignificant, it is almost equal to zero. We can conclude that the CNC explains nothing extra above what is already explained by the CI. The reason for the strong explanatory power attributed to the CNC in previous experiments performed in the literature is probably due to the fact that when the CNC was varied, other parameters (such as the CI) were varied also, which led to problems with significant differences in 'complexity'.

We also performed an analysis of variance (using the ANOVA procedure of SAS; without prespecifying the exponential form of the relationship), which resulted in CI being significant ($p < 0.0001$) and CNC and CI $\cdot$ CNC (the interaction effect between CI and CNC) being insignificant ($p > 0.1$ and $p > 0.5$ respectively).

Our results should be interpreted with sufficient care and should not be understood as a credo for CI and a requiem for CNC. First of all, it should be realized that for problem instances generated by ProGen, the CNC determines the range of the CI. Moreover, as can be seen from Fig. 1, the impact of CI on required computer time seems to become smaller with ascending values of CNC. However, when we include the interaction variable CI $\cdot$ CNC in our regression analysis, the interaction effect is insignificant ($p > 0.25$), which implies that we can ignore the correlation between the CI and the CNC in interpreting the results. In this analysis, the coefficient of CI was still significant ($p < 0.025$), whereas the coefficient of CNC was not ($p > 0.5$).

In addition, our experiments did not allow us to study the impact of both the CNC and the CI for extremely small CI values. CI will assume a value of zero for a series–parallel graph. It is to be expected that starting with a series graph and gradually altering it into a parallel graph, ceteris paribus, renders the problem to be more and more difficult. Our experiments did not allow to investigate this assertion. As already mentioned, the network generator used does not allow for the generation of networks with prespecified CI values. The probability that networks are generated with CI $= 0$ and CNC varying from 1.5 to 2.5 is extremely small.

The question remains whether the better explanatory power of CI is worth the effort of transferring a (for known reasons more favourite) AoN network into an AoA network in order to compute the CI. Fortunately, both the algorithm to transform an AoN network into an AoA network with minimum CI, as the algorithm for computing the CI, are polynomial in time. This makes it fairly easy to compute the CI even if the original network is in AoN format. The issue remains, however, that a procedure for generating networks with preset values of CI is not yet available.

4. The DTCTP and network complexity

In this section we investigate the potential use of the CI as a measure of complexity for the DTCTP. As mentioned above, we expect a positive correlation to exist between the CI and the computational effort needed to solve the DTCTP.

A total of 250 AoA networks are generated using the random network generator developed by Demeulemeester et al. (1993a). The number of activities is set between 5 and 20, the number of precedence relationships is defined randomly and the number of activity modes is varied from 2 to 4. Each problem is solved using the DEH procedure (Demeulemeester et al., 1993b).

The following equation is estimated through loglinear regression. (m being the number of nodes):

$$\text{CPU time} = a \cdot (m)^{(b \cdot \text{CI} + c)}$$

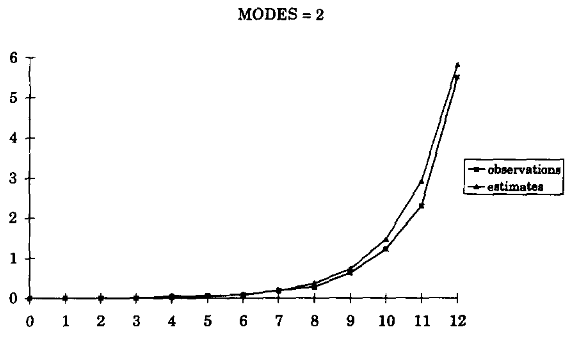
or

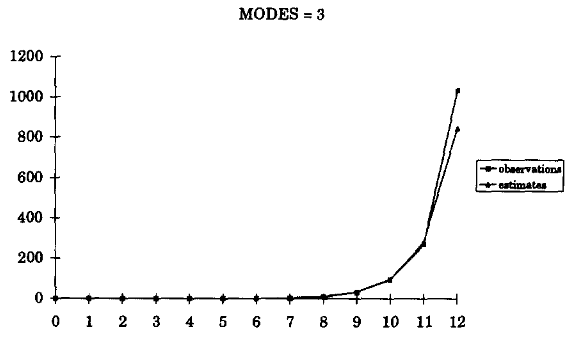
$$\ln(\text{CPU time}) = \ln a + (b \cdot \text{CI} + c) \cdot \ln m,$$

which results in

$$\text{CPU time} = 0.00117 \cdot (m)^{(0.995 \cdot \text{CI} + 0.3366)}$$

with $R^2 = 95\%$





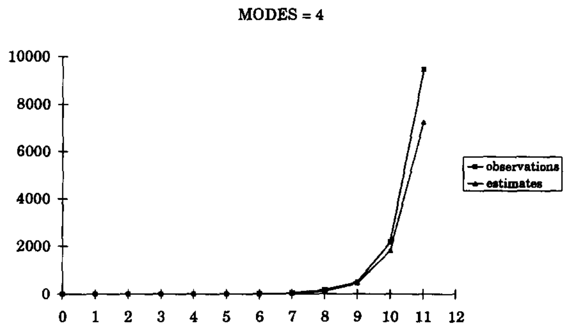


Fig. 4. Average processing time as a function of CI for the DTCTP instances.

The graphs shown in Fig. 4 represent the averages obtained over the different settings and the estimates based on the equation above. The results are shown in three different graphs with the number of activity modes equal to 2, 3 and 4.

It is clear from the results above that both the number of modes for each activity and the CI have a strong effect on the processing time needed to solve instances of the DTCTP. The portion of the variability which can be explained by both factors is very high (95%). At this juncture, we should realize that this high explanatory power is partially due to the fact that the DEH solution procedure is essentially based on the CI concept. This illustrates – as observed by Elmaghraby and Herroelen (1980) – that a measure of network complexity is indeed dependent on the objective of analysis and is 'necessarily' confounded by the procedure of analysis. The solution algorithm used to solve a problem is inextricably entwined with whichever properties we isolate and incorporate in a measure of network complexity under the 'current state of technology'. We doubt, however, that the explanatory power of the CI is ephemeral. We are confident that the notion of complexity expressed by the CI lies at the very heart of the DTCTP itself.

5. The RCPSP and resource availability

Our objective in the previous two sections was to study the potential use of the CI as a measure of network complexity for both the RCPSP and the DTCTP. In this section we keep the CI constant and try to isolate the impact of resource availability (or resource constrainedness) on the required solution effort for solving the RCPSP. Elmaghraby and Herroelen (1980) have made the conjecture that the relationship between the hardness of a problem (as measured by the CPU time required for its solution) and resource availability (scarcity) varies according to a bell-shaped curve similar to the one depicted in Fig. 5.

Indeed, if resources are only available in extremely small amounts, there will be relatively little freedom in scheduling the activities (for instance, the activities may have to be placed in series and the resulting project duration will equal the sum of activity durations). Hence, the corresponding RCPSP instance should be quite easy to solve (point A in Fig. 5). If, on the other hand, resources are amply available, the activities can simply be scheduled in parallel and the resulting project duration will be equal to the critical path length. Hence, the required computational effort

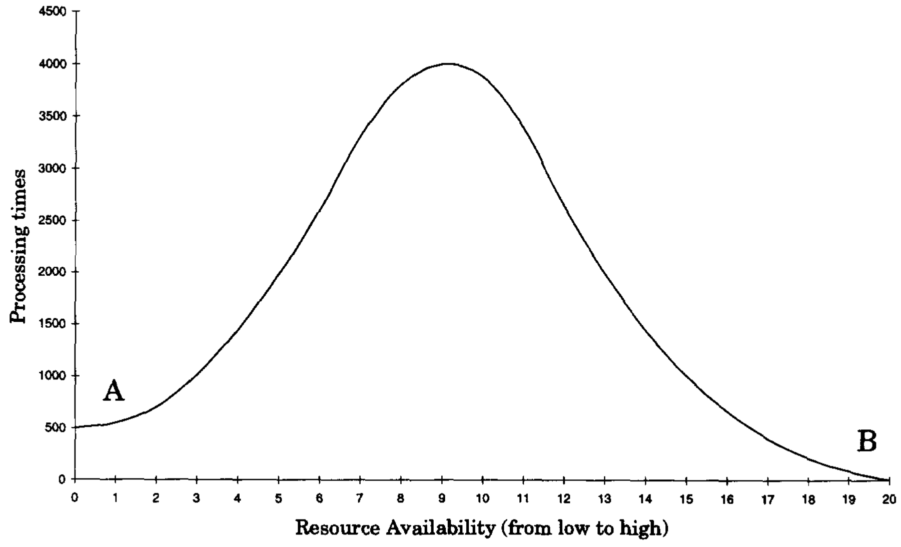


Fig. 5. Presumed effect of resource availability on processing time for the RCPSP.

should be very small again (point B in Fig. 5). Elmaghraby and Herroelen (1980) argued that the main problem is to obtain the exact shape of the complexity curve in the region between these two extremes, and questioned the availability of a measure that is able to resolve this problem.

Two of the best known parameters for describing resource availability (scarcity) that have been proposed in the literature are the resource factor and the resource strength. The *resource factor* (RF) has been introduced by Pascoe (1966), and has been utilized later in studies by Cooper (1976), Alvarez-Valdes and Tamarit (1989) and Kolisch et al. (1992). The RF can be calculated as follows:

$$\mathrm{RF}_k = \frac{1}{nK} \sum_{i=1}^{n} \sum_{k=1}^{K} \begin{cases} 1 & \text{if } r_{ik} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where, according to our previous notation, n is the number of activities, K is the number of resource types, and r_{ik} is the amount of resource type k required by activity i.

The RF reflects the average portion of resources requested per activity. It is a measure of the density of the array r_{ik}. If we have $\mathrm{RF} = 1$, then each activity requests all resources. $\mathrm{RF} = 0$ indicates that no activity requests any resource.

Kolisch et al. (1992) conclude that an increase of the RF results in an increase of the solution times required to solve the RCPSP. This result contradicts the results of Alvarez-Valdes and Tamarit (1989) who observed that problems with $\mathrm{RF} = 1$ were easier to solve than problems with $\mathrm{RF} = 0.5$.

The *resource strength* (RS) was introduced by Cooper (1976) to express the relationship between the resource requirements and the resource availability, and used later by Alvarez-Valdes and Tamarit (1989). Kolisch et al. (1992) criticize its use and propose the following new definition:

$$\mathrm{RS}_k = \frac{a_k - r_k^{\min}}{r_k^{\max} - r_k^{\min}} \tag{13}$$

where a_k is the total availability of renewable resource type k, $r_k^{\min} = \max_{i=1,\dots,n} r_{ik}$, and $r_k^{\max}$ is the peak demand of resource type k in the precedence-preserving earliest start schedule

With respect to one resource, the smallest resource availability is obtained for $\mathrm{RS} = 0$. For $\mathrm{RS} = 1$, the problem is no longer resource-constrained. Given the definition of $r_k^{\max}$, the RS as defined by Eq. (13) already incorporates information about the precedence structure of the network.

In their experiments, Kolisch et al. (1992) conclude that the RS has the strongest impact on solution times: the average solution time continuously increases with decreasing RS. The hardest problems seem to be the ones where the minimal resource availability is provided. This is in contradiction with the conclusion obtained by Alvarez-Valdes and Tamarit (1989). In addition, the effect of RS on computational effort required is monotone decreasing, which deviates from the bell-shaped curve proposed by Elmaghraby and Herroelen (1980). The Elmaghraby and Herroelen conjecture, on the other hand, has been confirmed to exist for the DTCTP case by Demeulemeester et al. (1993b).

Inspired by these contradictory findings we set up the following experiment. Using ProGen, nine sets of 500 RCPSP instances are generated. For each network, the number of activities is set to 25 and one resource type is defined. The activity durations are drawn from the uniform distribution in the range [1, 10]. The minimum and maximum resource requirements are set to 1 and 10 respectively. The CNC is set to 2, while the RF is set to 1. Using increments of 0.125, the RS is set to 0 for the first set of 500 networks, 0.125 for the second, up to 1 for the last set. Each of the 4500 networks is then transformed to a corresponding AoA network with minimum CI value using the procedure of Kamburowski et al. (1992). The CI values obtained varied from 7 to 17. The networks are then grouped per CI value and solved using the DH procedure (Demeulemeester and Herroelen, 1992).

For the nine groups of networks, the required CPU time varies in function of the RS according to a bell-shaped curve similar to the one projected in Fig. 5. As an example, Fig. 6 plots the results for the group of networks with $\mathrm{CI} = 12$ (similar results are obtained for the other CI values). The values on the abcissa correspond to the various classes of RCPSP instances with RS values ranging from 0 to 1. The required CPU times are the averages for each class of networks. When the difference between two averages is significantly different from zero (using the Wilcoxon test statistic), a ' < ' or ' > ' indication is drawn between two RS values. In contradiction with the findings of Kolisch et al. (1992), the results do not show a continuous increase of the required solution time with decreasing RS. We assume that the fact that Kolisch et al. (1992) did not find a bell-shaped curve relationship is due to the fact that the CI was not held constant in their experiment.

Still another measure of resource availability has been introduced by Patterson (1976). He defines the so-called *resource-constrainedness* (RC) for each resource k, as follows:

$$\mathrm{RC}_k = d_k / a_k \tag{14}$$

where a_k is the availability of resource k and d_k is the average quantity of resource k demanded when required by an activity, $d_k = (\Sigma_i r_{ik})/\Sigma_i\{1 \text{ if } r_{ik} > 0;\ 0 \text{ otherwise}\}$.

The arguments for using RC and not RS as a measure of resource availability can be summarized as follows. First, RC does not yet incorporate information about the precedence structure of a network, and as such can be considered as a 'pure' measure of resource availability. Second, there are occasions where RS can no longer distinguish between easy and hard problem instances while RC continues to do so. A small example can be used to illustrate this point. For a network for which the resource requirement of a particular activity equals the maximum availability of a single resource, while the resource requirements for the other activities are smaller than the resource availability, the RS = 0, no matter what the latter resource requirements are. Depending on precisely these requirements, however, the hardness of the resulting RCPSP may vary considerably. This variation in problem hardness can be captured by the RC. For an easy to solve problem with 50 activities, $a = 20$ and $r_i = 20$ while $r_j = 1$ ($j = 1,\ldots,50$; $j \neq i$), RS = 0 while RC = 0.069. When the $r_j = 20$ ($j = 1,\ldots,50$; $j \neq i$), RS = 0 while RC = 1, and the problem is still easy to solve. When the $r_j = 10$ ($j = 1,\ldots,50$; $j \neq i$), RS = 0. However, RC = 0.51 and the problem may be very hard to solve.

The experiment on the 4500 networks is re-

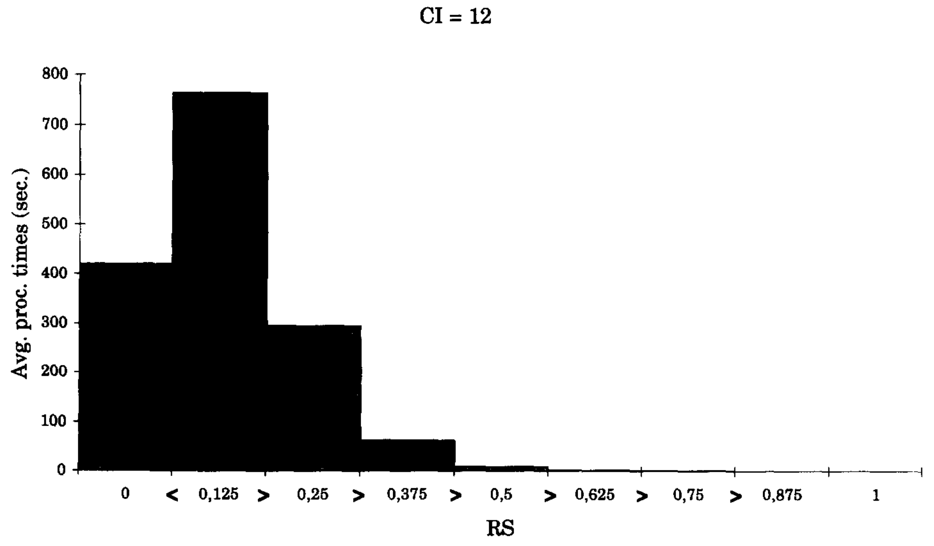


Fig. 6. Average RCPSP processing time as a function of RS for CI = 12.

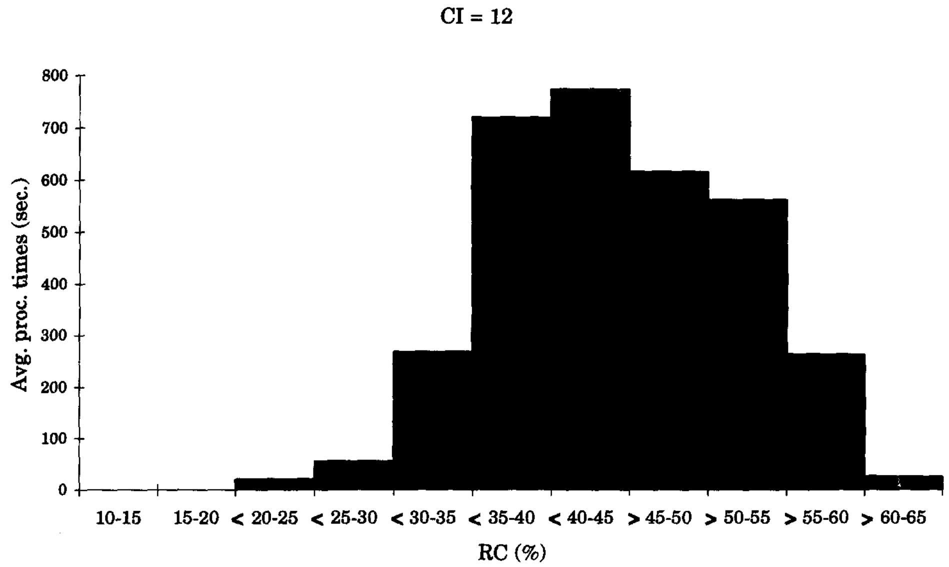


Fig. 7. Average RCPSP processing time as a function of RC for CI = 12.

peated, computing for each network the RC according to Eq. 14. For the nine groups of networks, the required CPU time varies in function of the RC according to a bell-shaped curve similar to the one projected in Fig. 5. As an example, Fig. 7 again plots the results for the group of networks with the midrange CI value, i.e. CI = 12. The values on the abcissa now correspond to the various classes of RCPSP instances with RC values ranging from RC = 10% to RC = 65%. The required CPU times are the averages for each class of networks. Again, when the difference between two averages is significantly different from zero (using the Wilcoxon test statistic), a '<' or '>' indication is drawn between two RC values. Evidently, the results do not show a continuous increase of solution time with increasing resource constrainedness, as would be expected from the experimental results obtained by Kolisch et al. (1992). On the contrary, beyond a certain value of RC, the average CPU time required starts to decrease.

It should be noted that a network for which the RS is small, will have a high value for the RC. This can be observed in Figs. 6 and 7. Fig. 6 shows positive CPU times for the networks with RS = 0, indicating that problem instances with RS = 0 are not necessarily very easy to solve. On the other hand, the required CPU times for the networks with RS values approaching 1 are negligible. Fig. 7 shows negligible CPU times for the networks with low RC value, while the CPU times are positive for the networks with the RC in the range 60–65%.

We also performed an analysis of variance (ANOVA) in order to assess the relative importance of RS and RC in explaining the variance in processing times. We also included the CI in this analysis as a network complexity measure. The analysis revealed that the CI was significant ($p < 0.0001$), whereas RS and RC were only moderately significant ($p < 0.1$). These results suggest that the network complexity measures explain more of the variance in processing times than the resource availability measures.

6. Conclusions

The need to differentiate between easy and hard instances of combinatorial problems modeled as activity networks and the interest in isolating the factors that determine the computing effort required by solution procedures, inspired a number of researchers to develop so-called measures of network complexity. The objective of this paper was to investigate the relation between the hardness of a problem instance and the logic of the underlying network. A series of experiments are performed on instances of the resource-constrained project scheduling problem (RCPSP) solved by the branch-and-bound procedure developed by Demeulemeester and Herroelen (1992). The results demonstrate that the previously proposed coefficient of network complexity (CNC), defined as the ratio arcs over nodes, is not a very good measure of complexity. The negative correlation which was presumed in the literature to exist between the CNC and problem complexity cannot be confirmed. In addition our experiments reveal a positive correlation between the CNC and the so-called complexity index (CI), essentially the minimum number of node reductions necessary to transform a general activity network to a series–parallel network. The results of a loglinear regression performed on five sets of 1000 networks, indicate a negative correlation between problem hardness and the CI. The higher the value of CI, the easier it is to solve the corresponding RCPSP. This result, counterintuitive at first sight, implies that the more complex the network topology, the smaller the number of feasible parallel paths, which seems to render the RCPSP more tractable. The small R^2 values obtained, however, indicate that it may not be wise to rely completely on the CI for making individual predictions of the computational effort required by an algorithm to solve RCPSP instances. It seems evident that the structure of the network, in whichever way it is measured, will not be sufficient to reflect the difficulty encountered in the resolution of such problems.

A second set of experiments involved the solution of a number of instances of the discrete time/cost trade-off problem (DTCTP) using the optimal procedure by Demeulemeester et al. (1993b) which is based on the method by Bein et al. (1992) for determining the minimum number of node reductions necessary to transform a general acyclic network to a series–parallel network. The results indicate that both the number of execution modes of each activity and the CI do have a strong effect on the processing time needed to solve instances of the DTCTP. The variability portion which can be explained by both factors is very high (95%). This was to be expected. The solution algorithm itself is built on the basic concepts which lie at the very heart of the CI. As such, our results confirm the conjecture previously made by Elmaghraby and Herroelen (1980) that a measure of network complexity depends on the objective of analysis and may be confounded by the solution algorithm.

A last set of experiments aimed at the investigation of the impact of resource availability on the solution effort required to solve instances of the RCPSP with constant CI values. Our results confirm the conjecture made by Elmaghraby and Herroelen (1980), that there is a bell-shaped relationship between the CPU time required to solve RCPSP instances and the resource availability, measured in terms of both the resource strength RS and the resource constrainedness RC.

The experiments performed in this paper suffered from the fact that a procedure for the generation of activity network instances which satisfy preset CI values, is not yet available. This makes it very cumbersome to use the CI as the controlling parameter in a full factorial experiment. The development of an algorithm for generating activity networks which satisfy preset CI values constitutes a viable area of future research.

Acknowledgements

We would like to thank Andreas Drexl, Rainer Kolisch and Arno Sprecher, Institut für Betriebswirtschaftlehre, Christian-Albrechts-Universität zu Kiel, F.R.G., for providing us with the project generator ProGen. We also thank Erik Demeulemeester, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium, for providing us with the codes for solving the RCPSP and the DTCTP.

Appendix A. The complexity index

Bein et al. (1992) show that the complexity index of a directed acyclic graph (dag) $D = (N,A)$

is equal to the number of nodes in a minimum node cover of its complexity graph $C(D)$. The *complexity graph*, $C(D)$, of a network $D=(N,A)$ is defined as follows: $(i,j) \in C(D)$, i.e., (i,j) is an arc of $C(D)$, if there exists paths $\pi(1,j)$, $\pi(i,n)$, $\pi_1(i,j)$ and $\pi_2(i,j)$ such that $\pi(1,j) \cap \pi_1(i,j)=\{j\}$ and $\pi(i,n) \cap \pi_2(i,j)=\{i\}$. Note that the paths $\pi_1(i,j)$ and $\pi_2(i,j)$ may be the same. The definition implies that neither 1 nor n appears as a node in $C(D)$. The same authors have developed an algorithm for constructing the complexity graph in time $O(n^2+M(n))$, where $M(n)$ is the time required for computing the transitive closure of a graph of n vertices (a recent upper bound for $M(n)$ is $O(n^{2.37})$ (Bein et al., 1992)). For ease of understanding, we revise the fundamental concepts underlying the computation of the complexity graph and the complexity index. In doing so, our main concern is on clarity of exposition, rather than on issues of computational efficiency.

A.1. Constructing the dominator trees

In constructing the complexity graph, the first step is to construct the *dominator tree* $T_1(D)$ and the *reverse dominator tree* $T_2(D)$. Consider a dag $D=(N,A)$ with root (initial) node 1, i.e., there exists a path from 1 to every node in N. Node v is a *dominator* of node w if every path from node 1 to w contains v. Every node is a dominator of itself, and node 1 dominates every node (Aho et al., 1975). Node v is a reverse dominator of node w if every path from w to the end node contains v. The set of dominators of a node w can be linearly ordered by their order of occurrence on a shortest path from the root to w. The dominator of w closest to w (other than w itself) is called the *immediate dominator* of w. Since the dominators of each node are linearly ordered, the relation 'v dominates w' can be represented by a tree with root 1. This tree is called the *dominator tree*, $T_1(D)$, for D. The *reverse dominator tree*, $T_2(D)$, is obtained by reversing all the arcs in D and constructing the dominator tree of the resulting graph. A dag is *series-parallel reducible* (s-p) if and only if for every arc (i,j) either i dominates j or j reverse-dominates i.

Aho et al. (1975) describe a polynomial procedure to compute the dominator tree for a rooted dag with m edges. Harel (1985) presents a procedure for computing the dominator tree $T(D)$ in time $O(m+n)$, where m is the number of edges in D. For our purposes, we rely on the procedure described below, which, though less efficient, can be easily incorporated in a computer code. In the description of the procedure we assume that each node $w=2,\dots,n$ is considered in sequence and that on considering node w, the immediate dominator for any node $v<w$ has already been determined. The following observations (given here without proof) form the underlying basis of our computer code for transforming D into its dominator tree $T(D)$.

Observation 1. *Let $D=(N,A)$ be a dag. Let node $w \in N$ have only one incoming arc (v,w). Then v is an immediate dominator of node w.*

Observation 2. *Let $D=(N,A)$ be a dag. Let node $w \in N$ have more than one incoming arc and let v be the highest numbered immediate predecessor of w. Then replacing (v,w) by arc (a,w), where a is the immediate dominator of node v, does not change the dominators of any node in D.*

Observation 3. *Let $D=(N,A)$ be a dag. Let there be an arc $(1,w)$. Then node 1 is the immediate dominator of node w.*

The repeated use of these observations will transform D into its dominator tree $T(D)$. The data structure which allows us to find efficiently the arcs to which to apply the observations uses the upper diagonal part of the incidence matrix for the dag D. Rows 1 and n, as well as columns 1 and n must not be considered (node 1 and n will not appear in the complexity graph). We check each node $w=3,\dots,n-1$ for the number of incoming arcs. If node w has only one incoming arc (v,w), then v is its immediate dominator (Observation 1), and the (v,w) entry is left unchanged in the incidence matrix. If node w has more than one incoming arc, we consecutively replace (v,w), where v is the highest numbered predecessor node, by arc (a,w), where a is the

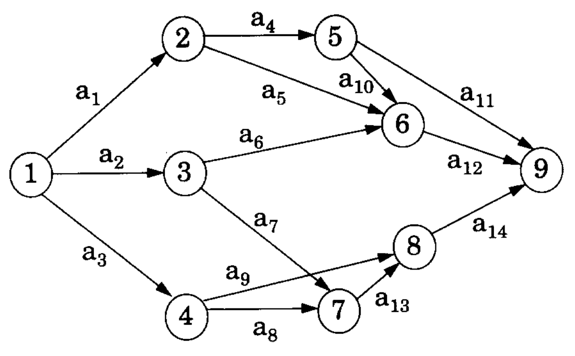


Fig. 8. Problem example.

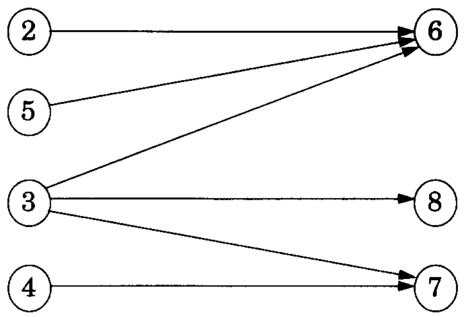


Fig. 10. Complexity graph for the problem example.

immediate dominator of node v (Observation 2). The corresponding entries in the incidence matrix are updated. The construction of the reverse-dominator tree is symmetric.

Fig. 8 represents a 9-node, 14-arc network. The dominator tree and the reverse-dominator tree are represented in Fig. 9.

A.2. Constructing the complexity graph.

The data structure for the algorithm is the incidence matrix $[a_{ij}]$ of the dag D with $a_{ij} = 1$ if node i immediately or transitively precedes node j. So, first the incidence matrix has to be updated such that it also contains the transitive arcs. For every node w with immediate dominator b, set $a_{vw} = 0$, for all arcs (v, w), $v \leq b$. For every node

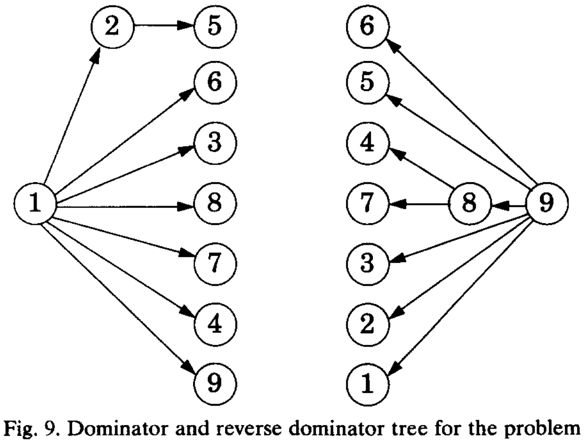


Fig. 9. Dominator and reverse dominator tree for the problem example.

v with immediate reverse-dominator c, set all $a_{vw} = 0$ for all arcs (v, w), $w \geq c$. As already mentioned above, it is shown by Bein et al. (1992) that the complexity graph can be computed in time $O(n^{2.37})$. The complexity graph for our problem example is depicted in Fig. 10.

A.3. The minimum node cover of $C(D)$

Having constructed the complexity graph $C(D)$, the next step is to determine its minimum node cover N^*. When $C(D)$ is empty, the original dag is series-parallel. Because $C(D)$ is a transitive dag, N^* can be computed by reducing the problem to finding the maximum matching in a bipartite graph, since the complement of a minimum node cover is a maximum independent set, which in a transitive dag corresponds to a Dilworth chain decomposition (Ford and Fulkerson, 1962). Hopcroft and Karp (1973) offer an $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In the sequel, we follow the original arguments of Ford and Fulkerson (1962) who offer a simple procedure that runs in time $O(n^2)$.

Following Ford and Fulkerson's notation, let P be a finite partially ordered set with elements $1, 2, \ldots, n$ and order relation '$>$'. A *chain* in P is a set of one or more elements $i_1, i_2, \ldots, i_k$ with $i_1 > i_2 > \cdots > i_k$. If we associate a directed graph with P by taking nodes $1, 2, \ldots, n$ and arcs (i, j) corresponding to $i > j$, this notion of a chain coincides with the notion of a chain in the graph, except that now we allow a single node to be a chain. A *decomposition* of P is a partition of P

into chains. Thus P always has the trivial decomposition into n 1-element chains. A decomposition with the smallest number of chains is *minimal.*

Two distinct members i and j of P are unrelated if neither $i > j$ nor $j > i$. The maximal number of mutually unrelated elements of P is less than or equal to the number of chains in a minimal decomposition of P, since two members of a set of mutually unrelated elements cannot belong to the same chain. Ford and Fulkerson (1962) establish the proof that the problem of constructing a minimal decomposition can be solved by their labeling algorithm for constructing a maximal independent set of admissible cells in an array. In the context of the minimum node cover problem on hand, the array to be considered is the $(n \times n)$ incidence matrix for the complexity graph $C(D)$, in which a cell is *admissible* if the corresponding arc appears in $C(D)$.

For our problem example, a minimal node cover can be found consisting of unlabeled rows 3 and 4, and labeled column 6. The nodes to be reduced in the network of Fig. 8 are nodes 3, 4 and 6. Hence, CI = 3.

Appendix B. Transforming an AoN network into an AoA network with minimal CI

Kamburowski et al. (1992, 1993) present two algorithms for generating an AoA network from a given AoN network. Of interest here is the polynomial time algorithm which generates an AoA network with minimal CI value and the minimum number of nodes from a given AoN network (Kamburowski et al., 1992).

If $G = (V,E)$ is the AoN representation of an activity network (with V denoting the vertices or activities and E the edges or precedence relationships), then the problem consists of finding $D = (N,A)$ (with N denoting the nodes or events and A the arcs or activities), which is an equivalent AoA representation of the activity network with minimum CI. Each activity $v \in V$ has to be represented in AoA format by an arc (s_v, t_v).

In order for the two formats to be equivalent, the following conditions should hold:

$$P^*(s_v) = P^*(v) \text{ and } S^*(s_v) = \bigcap_{u \in P(v)} S^*(u), \tag{15}$$

$$S^*(t_v) = S^*(v) \text{ and } P^*(t_v) = \bigcap_{w \in S(v)} P^*(w), \tag{16}$$

where $P(v)$ is the set of immediate predecessors of activity v, $S(v)$ is the set of immediate successors of activity v, $P^*(v)$ is the set of predecessors of activity v, $S^*(v)$ is the set of successors of activity v, $P^*(s_v)$ is the set of predecessors of node (event) s_v, $S^*(s_v)$ is the set of successors of node (event) s_v, $P^*(t_v)$ is the set of predecessors of node (event) t_v, and $S^*(t_v)$ is the set of successors of node (event) t_v.

The set of distinct pairs $(P^*(s_v), S^*(s_v))$ and $(P^*(t_v), S^*(t_v))$ represents the minimum set of nodes N. We have $s_v = i$ if $P^*(i) = P^*(s_v)$, and $t_v = j$ if $S^*(j) = S^*(t_v)$, so that all activities (except the dummy activities) are defined. If there are precedence relationships in the AoN notation which are not yet represented in the AoA notation (that is if $t_u \neq s_v$ for $(u,v) \in E$), a dummy path is introduced connecting t_u and s_v.

So, $R = \{(i,j) \mid i \neq j \text{ and } \exists (u,v) \in E \text{ such that } t_u = i \text{ and } s_v = j\}$ defines the set of node pairs that still have to be connected by a dummy path. A dummy arc (i,j) is feasible if $i \neq j$ and $P^*(i) \subset P^*(j)$, or equivalently $S^*(j) \subset S^*(i)$. For each node pair $(i,j) \in R$ for which the following condition holds:

$$\{k \mid (k,j) \in R \text{ and } (i,k) \text{ is feasible}\}$$
$$\cap \{l \mid (i,l) \in R \text{ and } (l,j) \text{ is feasible}\} = \varnothing,$$

which is equivalent to:

$$\exists! \text{ node } k : (i,k) \in R \text{ and } (k,j) \in R,$$

a dummy arc (i,j) is introduced into the AoA network. In this way, an equivalent AoA network with minimal complexity is obtained. Note that the resulting AoA network is based on the minimum number of nodes necessary to represent the AoN network in AoA format. So, adding extra

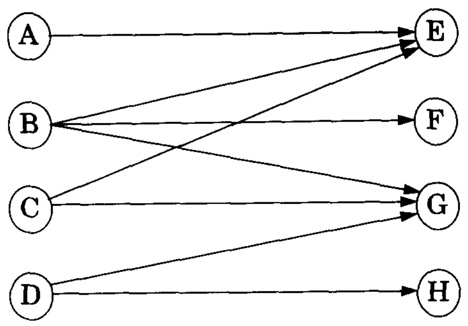


Fig. 11. Example AoN network.

nodes is not allowed, although this might lead to a lower CI.

The algorithm can now be illustrated on the small example given in Fig. 11.

Using the conditions (15) and (16), we determine for each activity v the values of $P^*(s_v)$, $S^*(s_v)$, $P^*(t_v)$ and $S^*(t_v)$, which represent the minimum number of nodes, where s_v and t_v denote the starting and ending node of activity v, see Table 1.

The set of distinct pairs $\{(P^*(j), S^*(j))\}_{j=1,\dots,n}$ of the pairs $(P^*(s_v), S^*(s_v))$ and $(P^*(t_v), S^*(t_v))$ is:

node 1:	$(\varnothing, \{A, B, C, D, E, F, G, H\})$
node 2:	$(\{A, B, C\}, \{E\})$
node 3:	$(\{B\}, \{E, F, G\})$
node 4:	$(\{B, C\}, \{E, G\})$
node 5:	$(\{D\}, \{G, H\})$
node 6:	$(\{B, C, D\}, \{G\})$
node 7:	$(\{A, B, C, D, E, F, G, H\}, \varnothing)$

Table 1

v	$P^*(S_v)$	$S^*(S^v)$	$P^*(t_v)$	$S^*(t_v)$
A	∅	A B C D E F G H	A B C	E
B	∅	A B C D E F G H	B	E F G
C	∅	A B C D E F G H	B C	E G
D	∅	A B C D E F G H	D	G H
E	A B C	E	A B C D E F G H	∅
F	B	E F G	A B C D E F G H	∅
G	B C D	G	A B C D E F G H	∅
H	D	G H	A B C D E F G H	∅

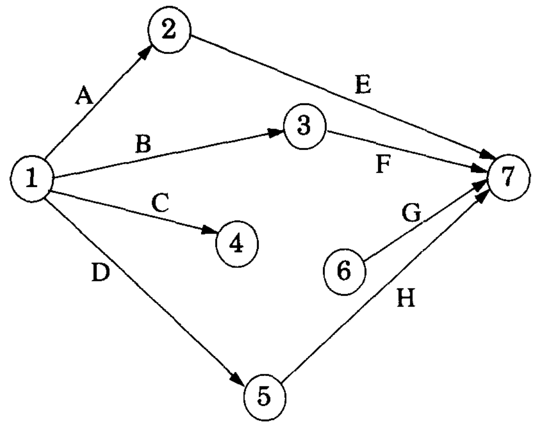


Fig. 12. Intermediate AoA network.

For each activity, we now determine the starting and ending node or event. If $P^*(i) = P^*(s_v)$, then $s_v = i$ and if $S^*(j) = S^*(t_v)$, then $t_v = j$:

activity A:	s_A = node 1	t_A = node 2
activity B:	s_B = node 1	t_B = node 3
activity C:	s_C = node 1	t_C = node 4
activity D:	s_D = node 1	t_D = node 5
activity E:	s_E = node 2	t_E = node 7
activity F:	s_F = node 3	t_F = node 7
activity G:	s_G = node 6	t_G = node 7
activity H:	s_H = node 5	t_H = node 7

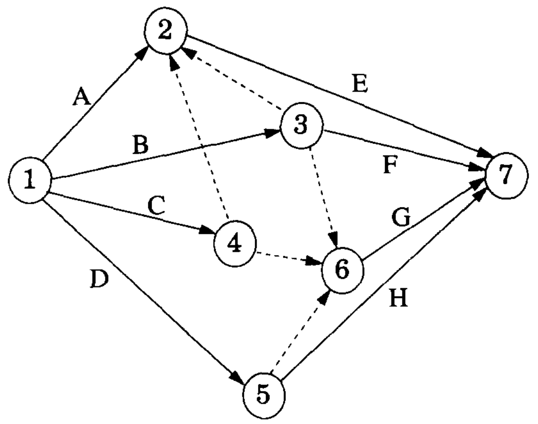


Fig. 13. AoA network with dummy arcs included.

This results in the AoA network given in Fig. 12.

The precedence relationships between activities (B, G), (B,E), (C, E), (C, G) and (D, G) are not yet represented. Therefore, $R = \{(3,6), (3,2), (4,2), (4,6), (5,6)\}$. There does not exist a node k for which one of the following conditions is true:

$$(3,k) \in R \text{ and } (k,6) \in R$$
$$(3,k) \in R \text{ and } (k,2) \in R$$
$$(4,k) \in R \text{ and } (k,2) \in R$$
$$(4,k) \in R \text{ and } (k,6) \in R$$
$$(5,k) \in R \text{ and } (k,6) \in R$$

Therefore, we should add all five dummy arcs. The resulting AoA network is given in Fig. 13.

References

Aho, A.V., Hopcroft, J.E., and Ullman, J.D. (1975), *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA.

Alvarez-Valdes, R., and Tamarit, J.M. (1989), "Heuristic algorithms for resource-constrained project scheduling: A review and empirical analysis", in: R. Slowinski and J. Weglarz (eds.), *Advances in Project Scheduling*, Elsevier, Amsterdam.

Baroum, S.M., and Patterson, J. (1993), "A comparative evaluation of cash flow weight heuristics for maximizing the net present value of a project", Research paper, Indiana University, Bloomington, IN.

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

Caestecker, G., and Herroelen, W. (1979), "The generation of random activity networks", Research Report No. 7906, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

Cooper, D.F. (1976), "Heuristics for scheduling resource-constrained projects: An experimental comparison", *Management Science* 22, 1186-1194.

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

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

Demeulemeester, E., and Herroelen, W. (1992), "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem", *Management Science* 38, 1803-1818.

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

Demeulemeester, E., Elmaghraby, S.E., and Herroelen, W. (1993b), "Optimal procedures for the discrete time/cost trade-off problem in project networks", Research report No. 9337, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

Demeulemeester, E., Herroelen, W., Simpson, W.P., Baroum, S., Patterson, J., and Yang, K. (1994), "On a paper by Christofides et al. for solving the multiple-resource constrained, single project scheduling problem", *European Journal of Operational Research* 76, 218-228.

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

Ford, L.R., and Fulkerson, D.R. (1962), *Flows in Networks*, Princeton University Press, Princeton, NJ.

Harel, D. (1985), "A linear time algorithm for finding dominators in flow graphs and related problems (extended abstract)", *Proceedings 17th Annual ACM Symposium on Theory of Computing*, Providence, RI, 185-194.

Herroelen, W.S., and Demeulemeester, E.L. (1995), "Recent advances in branch-and-bound procedures for resource-constrained project scheduling problems", in: P. Chrétienne et al. (eds.), *Scheduling Theory and Its Applications*, Wiley, New York.

Hopcroft, J.E., and Karp, M. (1973), "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs", *SIAM Journal on Computing* 2, 225-231.

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

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

Kamburowski, J., Michael, D.J., and Stallmann, M. (1992), "Optimal construction of project activity networks", *Proceedings of the 1992 Annual Meeting of the Decision Sciences Institute*, San Francisco, CA, 1424-1426.

Kamburowski, J., Michael, D.J., and Stallmann, M. (1993), "On the minimum dummy-arc problem", *Revue Française de Recherche Opérationelle* 27, 153-168.

Kolisch, R., Sprecher, A., and Drexl, A. (1992), "Characterization and generation of a general class of resource-constrained project scheduling problems: Easy and hard instances", Research report No. 301, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Germany.

Kurtulus, I.S., and Narula, S.C. (1985), "Multi-project scheduling: Analysis of project performance", *IIE Transactions* 17, 58-66.

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

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

Patterson, J.H. (1984), "A comparison of exact procedures for solving the multiple constrained resource project scheduling problem", *Management Science* 30, 854-867.

Talbot, F.B. (1982), "Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case", *Management Science* 28, 1197-1210.