1-1999

Phase transitions in project scheduling

Willy HERROELEN

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

Journal of the Operational Research Society (1999) 50, 148–156 0160-5682/99 $12.00

http://www.stockton-press.co.uk/jor

Phase transitions in project scheduling

W Herroelen[1] and B De Reyck[2]

[1]*Katholieke Universiteit Leuven, Belgium and* [2]*Erasmus University Rotterdam, The Netherlands*

Researchers in the area of artificial intelligence have recently shown that many NP-complete problems exhibit phase transitions. Often, problem instances change from being easy to being hard to solve to again being easy to solve when certain of their characteristics are modified. Most often the transitions are sharp, but sometimes they are rather continuous in the order parameters that are characteristic of the system as a whole. To the best of our knowledge, no evidence has been provided so far that similar phase transitions occur in NP-hard scheduling problems. In this paper we report on the existence of phase transitions in various resource-constrained project scheduling problems. We discuss the use of network complexity measures and resource parameters as potential order parameters. We show that while the network complexity measures seem to reveal continuous easy-hard or hard-easy phase transitions, the resource parameters exhibit a relatively sharp easy-hard-easy transition behaviour.

Keywords: phase transitions; project management; scheduling

Introduction

It is evidenced by practical experience that some computational problems are easier to solve than others. Complexity theory provides a mathematical framework which classifies computational problems as 'easy' or 'hard' (see for example, Karp[1] and Garey and Johnson[2]). A distinction is made between problems which are solvable in a polynomially bounded amount of time (classified in P) and problems which are not (classified in NP). The fact that a decision problem is shown to be NP-complete or the fact that an optimisation problem is shown to be NP-hard, implies that solving it is very hard. On the other hand, it is well-known that for many of these NP problems, many instances are easy to solve.[3] This is no surprise, however, since the classification of problems as P or NP (assuming that $P \neq NP$) is based on a *worst-case* analysis, which says nothing about the difficulty of typical instances. Clearly, the *average* case is also of interest. It may very well happen that if one generates thousands of NP-complete problems at random, simple algorithms quickly solve all but a few of them.

Looking at these results more closely, researchers in the area of *artificial intelligence* (AI) discovered that many NP-complete problems exhibit so-called *phase transitions,* resulting in a sudden and dramatic change in computational complexity. Often, problem instances change from being easy to being hard to solve to again being easy to solve when certain of their characteristics are modified.[4–6]. This easy-hard-easy phase transition can usually be described by one or more *order parameters* that are characteristic of the system as a whole. Hard to solve instances occur around a critical value of the order parameters. Moreover, the hard instances are often clustered around a small range of the order parameter values, which implies that most instances (when looking at the entire range of the order parameters) are easy to solve.

There are a number of open questions raised by these AI studies.[7] An important open issue concerns the range of problems and characteristics over which phase transition behaviour is exhibited. To date most of the research has been directed at studies of the *k*-satisfiability problem,[8–17] Hamiltonian paths,[4,18] graph colouring,[4] constraint satisfaction,[19] the travelling salesperson problem[20–22] and random tree search problems.[23–29] Another issue concerns the transition pattern itself. Most often the transition between easy and difficult regions is sharp, sometimes it is rather continuous. To the best of our knowledge, no evidence has been provided so far that similar phase transitions occur in NP-hard scheduling problems.

In this paper we study the existence of phase transitions in various project scheduling problems. In the next section we briefly review the objectives of phase transition research and offer a short review. The subsequent section reports on the phase transitions which have been observed in various resource-constrained project scheduling problems. We show that while the network complexity measures seem to reveal continuous easy-hard or hard-easy phase transitions, the resource parameters exhibit a relatively sharp easy-hard-easy transition behaviour. The last section is then reserved for overall conclusions and suggestions for future research.

Correspondence: Dr W Herroelen, Department of Applied Economics, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium.
E-mail: willy.herroelen@econ.kuleuven.ac.be

Phase transitions

A phase transition of a complex system is a dramatic change of some system property when an order or control parameter crosses a critical value. A simple example of a phase transition is water changing from a solid to a liquid when the temperature exceeds the freezing point.[7]

Phase transitions have also been observed in the field of AI. An intriguing problem in graph theory is to examine whether a given graph has a *Hamiltonian circuit* (HC) or not. A HC is a cyclic ordering of a set of nodes such that there is an edge which connects every pair of nodes in the graph in order. The cyclic condition ensures that the HC is closed. In addition, all the nodes have to be included with no repeats, which ensures that the HC does not cross over itself and passes through every node. Studies[4,18] have revealed that the existence of a HC in a random graph varies with the average connectivity of the graph. A fully connected graph always contains a HC. An almost fully connected graph has a very high probability of containing a HC. A random graph with an average connectivity of 2 is unlikely to even be connected, and so is unlikely to contain a HC. The probability of a HC changes steeply from almost 0 to almost 1 at an average connectivity of $\ln(N) + \ln(\ln(N))$.[18] Moreover, it has been shown empirically by Cheeseman *et al*[4] that the computational cost of finding a HC (if one exists) also exhibits a phase transition at the same point at which the probability that a random graph contains a HC changes dramatically.

The NP-complete *graph colouring problem*[30] consists of a graph, a specified number of colours, and the requirement to find a colour for each vertex in the graph such that adjacent vertices (i.e. nodes linked by an edge in the graph) have distinct colours. Graph colouring is a fundamental constraint satisfaction problem which essentially deals with partitioning a set of objects into classes according to certain rules. The objects form the set of vertices $V(G)$ of a graph G, two vertices being joined by an edge in G whenever they are not allowed in the same class. In order to distinguish between the classes, a set C of colours is used and the division is given by a colouring $\varphi: V(G) \rightarrow C$, where $\varphi(x) \neq \varphi(y)$ for all (x, y) belonging to the set of edges $E(G)$ of G. If C has cardinality k, then φ is a k-colouring. Therefore each colour class forms an independent set of vertices, that is, no two of them are joined by an edge. The minimum cardinal k for which G has a k-colouring is the chromatic number of G. Turner[3] showed that almost all instances of a k-colouring problem are easy to solve. Cheeseman *et al*[4] empirically investigated the probability of a solution for k-colourability problems for different values of k and N (number of nodes). They observed an abrupt change in the solution probability at higher values of the connectivity for larger k. Moreover, they observed a phase transition in the computational cost of solving k-colourability problems, which occurs at the critical average connectivity where the probability of a solution changes dramatically. Because Turner[3] in his experiments failed to generate instances with that specific value for the connectivity, he concluded that almost all instances are easy to colour.

The satisfiability problem is the first problem ever to be classified as NP-complete. Given a set of boolean variables and a collection of clauses (a set of literals—variables in either affirmative or negative form—or true/false conditions over the variables of which at least one should be satisfied), the *satisfiability problem* (SAT) concerns the search for a solution (an assignment of boolean values to each of the variables; also referred to as a *truth assignment*) that simultaneously satisfies all the clauses (referred to as a *satisfying truth assignment*). Looking at the computational results from solving thousands of SAT problems, a phase transition was discovered when computational cost is plotted against the ratio of clauses to variables. The cost reaches a peak where the instances change from probably satisfiable to probably unsatisfiable. Formulas with only a few clauses and many variables can almost always be satisfied, since most of the variables appear only once or twice, and a conflict among them is unlikely (the formulas are said to be *underconstrained*). A feasible solution can be found very easily. At the other end, where there are many clauses and only a few variables, each variable can be expected to appear in many clauses, such that conflicts are frequent and a feasible solution is unlikely (the formulas are *overconstrained*). Proving that no such feasible solution exists is very easy. However, when the ratio between the number of clauses and variables reaches an intermediate value, determining whether a feasible solution exists becomes very difficult. Selman *et al*[17] have shown that random instances of SAT can be generated in such a way that easy and hard sets of instances (for a particular SAT procedure) can be predicted in advance. They confirmed previous observations that many instances are quite easy and showed that for random 3-SAT the hardest area for satisfiability is near the point where 50% of the formulas are satisfiable.

A number of real-word problems, including numerous scheduling problems (for instance with sequence-dependent set-up times), can be formulated and solved as *travelling salesperson problems* (TSP). In a TSP, the goal is to find a Hamiltonian circuit among a set of nodes (namely, the cities) such that the total cost of the circuit is minimised. The costs of the edges in the graph are represented by an integer-valued cost matrix. When the distance matrix is symmetric, that is the distance from city i to city j is the same as that from j to i, the problem is referred to as a *symmetric* TSP. When the distance from city i to j is not necessarily equal to that from j to i, the *asymmetric* TSP (ATSP) results. Cheeseman *et al*[4] randomly generated intercity distances for the symmetric TSP from a log-normal distribution and used the branch-and-bound procedure of Little *et al*[31] for solving the resulting problem instances. They found that when the standard deviation of the intercity distance distribution (or the square root of its variance) is either very small or very large, the symmetric TSP is easy to solve. However, when the standard deviation has an intermediate value, the problem is very difficult. Stated otherwise, the complexity transition appears as an easy-hard-easy pattern as the standard deviation of the intercity distances increases. The magnitude and sharpness of the phase transition increases with city size. In their study of the ATSP, Zhang and Korf[22] found that when the discrete intercity distances are chosen uniformly from $\{0, 1, 2, \ldots, r\}$, the complexity exhibits an easy-hard transition as r increases. When the intercity distances are drawn from a discretized log-normal distribution, the complexity displays easy-hard-easy transitions as the standard deviation of the distribution grows. The authors also show that the control parameter that determines the two different transition patterns is the total number of distinct intercity distances. The complexity transition follows an easy-hard transition as the number of distinct intercity distances increases. However, the transition between easy and difficult regions is not as sharp as expected.

The reviewed studies inspired Cheeseman *et al*[4] to conjecture that all NP-complete problems have at least one order parameter for which it can be shown that the hard instances of that problem occur around a critical value of this parameter. This critical value (phase transition) separates the problem space in separate regions, such as overconstrained and underconstrained regions. Phase transitions are not merely a common feature of NP-complete problems, but are conjectured to be a *defining* characteristic of all such problems.

By now, it seems well established that phase transitions are not an artifact of any particular algorithm, but are intrinsic to the problem itself.[5] Yet, the connection between phase transitions and NP-completeness remains complex. Since all NP-complete problems exhibit phase transition behaviour one might think that, when a particular problem reveals a phase transition, it must belong to NP. However, this is not the case. There are problems, such as 2-SAT, which are in P and nevertheless show an easy-hard-easy pattern. Conversely, there are problems in NP, such as the TSP, whose hard instances are not clustered at a strict phase boundary. Some phase transitions are continuous (for example 2-SAT, while others are discontinuous (for example the freezing and boiling of water and 3-SAT).

Basically, the empirical AI studies all plot some average or median performance measure against simple structural parameters. Although the plots reveal easy-hard-easy patterns, they are still associated with extreme variances. Problem instances situated in the supposedly 'hard' region may sometimes not be that hard to solve. The current parameters used to specify the problem structure may well be too crude. The discovery of the characteristic easy-hard-easy pattern which is centered at a fixed transition point makes the phase transition phenomenon interesting. Exploring the differences between the (anomalous) hard instances in the easy region and the hard instances in the hard region is of similar interest. To date, most of the AI research has been concentrated on NP-complete decision problems. It would be utmost interesting to learn whether similar phase transitions manifest themselves in NP-hard optimisation problems. In the next section, we discuss the phase transitions which have recently been observed in resource-constrained project scheduling.

Phase transitions in project scheduling

The characterisation of activity networks has attracted attention since the mid-sixties. Researchers were interested in studying the effects of problem structure on algorithmic performance[32,33] 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[34] already recognised that the isolation of the unique and unambiguous factors that determine the required computing effort for solving an activity network problem proves to be a formidable task. Firstly, the measurement of activity network complexity cannot be accomplished in a meaningful manner unless the use of the measure is specified a priori. The apparently 'one and the same' activity network problem may be complex on one scale and easy on another. Moreover, complexity measures may be confounded by the procedure of analysis. The algorithm used to solve an activity network problem may indeed be inextricably entwined with whichever properties one wants to isolate and characterise as the determining factors of complexity. Finally, it is very unlikely that the complexity of a problem instance can be captured by one single measured quantity. Rather one would expect that a combination of different factors would determine the required computational effort for solving an activity network problem.

It is not a surprise then, that quite a number of activity network 'complexity' measures have been proposed in the literature.[32,33] Most measures try to capture information about the *size of the project network,* the *topological structure (morphology) of the project network* and the *availability of the different resource types* in relation to the resource requirements. Naturally, some measures may capture information about several of these classes simultaneously. Recent extensive computational experience[35] provides additional insight in the potential of the measures as an explaining factor for the computational complexity experienced by solution procedures for solving several types of resource-constrained project scheduling problems. Detailed examination of the results in the next section reveals the existence of easy-hard and hard-easy phase transitions which, in contrast to what has been experienced in AI, are not abrupt but continuous in the parameters used to describe the topological structure of a network. It will be shown in a subsequent section, however, that the resource availability measures exhibit a continuous bell-shaped easy-hard-easy complexity pattern with a relatively sharp easy-hard-easy phase transition around a critical value of the resource-constrainedness.

Topological network structure and the complexity of resource-constrained project scheduling

Network-based parameters. Various parameters for describing the topology of a project network have been presented in the literature. The best known is the *coefficient of network complexity* (CNC), introduced by Pascoe[36] for activity-on-the-arc (AoA) networks. CNC is simply defined as the ratio of the number of arcs over the number of nodes (different definitions have been used by Davies[37] and Kaimann[38,39]). CNC has been adopted by Davis[32] for the activity-on-the-node (AoN) representation and has been used in a number of studies since then.[40–42] As observed by Kolisch *et al,*[43] 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[43,44] seem to confirm that problems become easier with increasing values of CNC. Elmaghraby and Herroelen[34] already questioned the use of 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 CNC can discriminate among them.

Another well-known measure of the topological structure of an activity network is the *order strength* (OS), which is defined as the number of precedence relations, including the transitive ones, divided by the theoretical maximum of such precedence relations, namely $n(n-1)/2$, where n denotes the number of activities.[45] It is sometimes referred to as the *density*[46] and, as has been observed by Elmaghraby and Herroelen,[34] is equal to 1 minus the *flexibility ratio,* defined by Dar-El[47] as the number of zero entries in the precedence matrix divided by the total number of matrix entries. De Reyck[48] has shown that OS is identical to *RT,* an estimator for the restrictiveness (P) of an activity network.[49] If F_{seq} denotes the number of feasible sequences, that is the number of possible permutations of the activities of a project such that each activity does not precede one of its predecessors, the restrictiveness is defined as $P = 1 - [\log(F_{seq})/\log(n!)]$. P varies between 0 and 1, and assumes the value 0 for a parallel digraph and 1 for a series digraph.[49] However, F_{seq} (and, consequently, P) are very hard to calculate. Therefore, Thesen[49] has tested several estimators for P, best of which seemed to be (with the lowest mean relative error with respect to P):

$$RT = \frac{2\sum_{i,j \in V} r_{ij} - 6(n+1)}{n(n-1)}$$

with

$$r_{ij} = \begin{cases} 1, & \text{if there exists a directed path from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

which is shown to be identical to OS. Therefore, we can conclude that the order strength, the density, the flexibility ratio and the restrictiveness estimator *RT* actually constitute one and the same complexity measure.

Recently, Bein *et al*[50] introduced a new characterisation of two-terminal acyclic networks which essentially measures how nearly series-parallel a network is. They define the *reduction complexity* on an activity network in AoA format 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. De Reyck and Herroelen[51] adopted the reduction complexity as the definition of the *complexity index* (CI) of an activity network. For a more detailed description of the CI and an algorithm to compute it, we refer the reader to De Reyck and Herroelen.[51]

Topology measures and the complexity of the resource-constrained project scheduling problem. Recent computational experience has provided useful insight in the potential explanatory power of the topological network parameters on the hardness of resource-constrained project scheduling instances. The *resource-constrained project scheduling problem* (RCPSP) involves the deterministic scheduling of project network activities, subject to finish-start precedence constraints and renewable resource constraints, in order to minimise the project duration. The problem is referred to as problem $m, 1|cpm|C_{max}$ in the classification scheme of Herroelen *et al.*[52] The problem is strongly NP-hard. For a recent review see Herroelen *et al.*[53]

De Reyck and Herroelen[51] investigated the potential use of CNC and CI as a measure of activity network complexity for the RCPSP. They generated five sets of 1000 RCPSP instances using ProGen,[43] each with 25 activities. In each of the five sets, CNC is set at a different value, varying from 1.5 in the first set to 2.5 in the fifth. Each RCPSP instance was then solved using the branch-and-bound procedure of Demeulemeester and Herroelen.[54] Both Alvarez-Valdés and Tamarit[44] and Kolisch *et al*[43] observed a negative correlation between CNC and the required solution time for solving an RCPSP instance. De Reyck and Herroelen,[51] however, reached the conclusion that it is very ambiguous to attach all explanatory power of problem complexity to CNC. A positive correlation can be observed between CNC and the *complexity index,* CI. The CI-values for the instances used in the experiment range from 9–21. They found that CI plays an important role in predicting the required computing effort for solving an RCPSP instance. The generated plots of the required CPU-time against CI revealed a *rather continuous hard-easy complexity pattern: the higher CI, the easier the RCPSP instance.*

In a subsequent experiment, De Reyck[48] again used ProGen[43] to generate 4200 RCPSP instances with 25 activities, CNC ranging from 1.2–2.5 and CI ranging from 1–17. Each instance was then solved using the enhanced procedure for the RCPSP developed by Demeulemeester and Herroelen.[55] Again CI was found to have a strong impact on the required processing time. In addition, OS was found to be a good network complexity measure. Using values of OS ranging from 0.15–0.70, a *plot of the logarithm of the average CPU-time versus OS reveals a linear hard-easy complexity transition* (Figure 1).

Schwindt[56] has chosen to use *RT* (OS) as a network complexity measure while developing the problem generator ProGen/max, which is capable of generating instances with so-called *generalised precedence relations* (start-start, start-finish, finish-start and finish-finish relations with minimal and maximal time lags). The resource-constrained project scheduling problem with generalised precedence relations (problem $m, 1|gpr|C_{\max}$ in the classification scheme of Herroelen *et al*[52]) is NP-hard. Even the problem of determining whether an arbitrary feasible solution exists is NP-complete. De Reyck[35] used ProGen/max to generate a set of 7200 instances and found the order strength, OS, to be the most powerful measure in explaining the variations in the CPU-time required by his branch-and-bound procedure (see also De Reyck and Herroelen[57]). Again the complexity transition follows a *continuous hard-easy pattern: the higher OS, the easier the instance.*

Topology measures and the complexity of trade-off problems in project scheduling. The *discrete time/cost trade-off problem* (DTCTP) assumes a single nonrenewable resource. The duration of an activity is a discrete, nonincreasing function of the amount of a single resource allocated to it. An activity assumes different execution modes according to the possible resource allocations. Demeulemeester *et al*[58] developed exact procedures for generating the complete time/cost trade-off curve (problem $1, T|cpm, disc, mu|curve$ in the classification scheme of Herroelen *et al*[52]). Computational experience on a total of 250 instances[51] indicates that both the number of modes and CI have a strong effect on the required processing time. The results exhibit a *continuous easy-hard complexity pattern: the higher CI, the harder the problem.* Recently, Demeulemeester *et al*[59] have developed a new exact horizon-varying procedure based on the iterative optimal solution of the problem of minimising the sum of the resource use over all activities subject to a project deadline. Computational results obtained on 1800 test instances confirm the easy-hard complexity pattern.

The *discrete time/resource trade-off problem* (DTRTP) assumes that the duration of an activity is a discrete, nonincreasing function of the amount of a single renewable resource committed to it. Given a specified work content for an activity, all its efficient execution modes are determined based on time/resource trade-offs. An activity when performed in a specific mode has a duration and a resource requirement during each period it is in progress, such that the resource-duration product is at least equal to the specified work content. The single resource has a constant availability. The objective is to schedule each activity in one of its modes, subject to the precedence and the renewable resource constraints, under the objective of minimising the project duration (problem $1, 1|cpm, disc, mu|C_{\max}$ in the

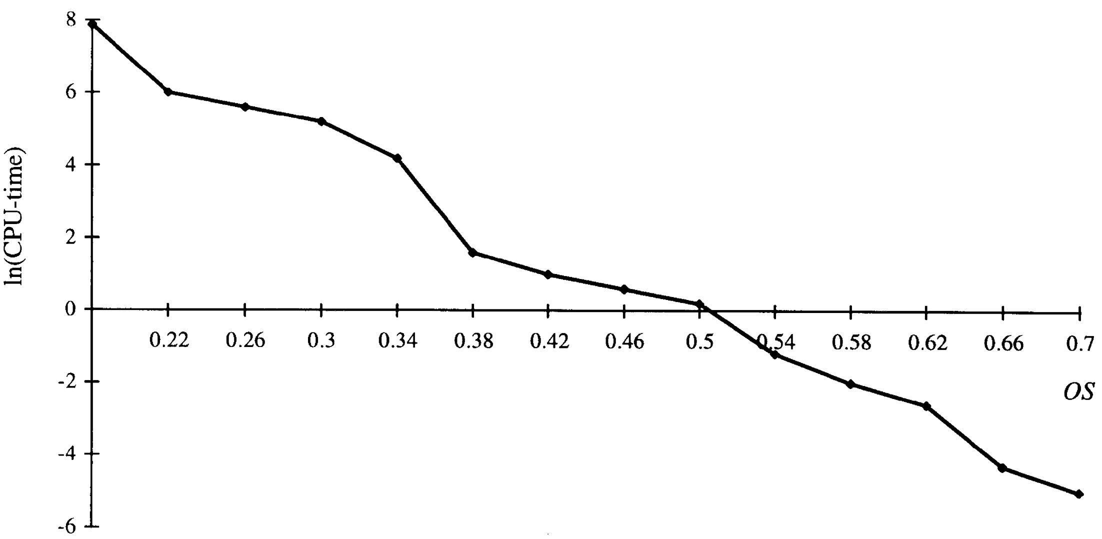

Figure 1 Logarithm of CPU time *vs* OS.

classification of Herroelen *et al*[52]). Exact[60] and heuristic solution procedures[61] have been recently developed. *OS again exhibits an hard-easy complexity pattern* (the higher OS, the easier the corresponding DTRTP instance).

Topology measures and maximising the net present value of a project. Interesting project scheduling problems result if the regular minimum makespan objective is replaced with the non-regular performance measure of maximising the net present value (*npv*) of a project. Herroelen *et al*[62] have developed an exact recursive procedure for solving the unconstrained *max-npv* problem, that is the problem of maximising the *npv* of a project subject to finish-start zero-lag precedence constraints in the absence of resource constraints (problem $cpm, c_j|npv$ in the classification of Herroelen *et al*[52]). De Reyck and Herroelen[63] have extended the algorithm to the case of generalised precedence relations with minimal and maximal time lags (problem $gpr, c_j|npv$). The procedure has been tested on a set of 7200 randomly generated problem instances using the number of activities as a problem size-based measure and the order strength (OS) as a network-based measure. The cash flows for each of the activities are generated randomly in the interval [−500, 500]. Despite the fact that the problem is in *P*, the results reveal a *continuous easy-hard phase transition for the order strength OS:* the higher OS, the more dense the network becomes, and the more recursion steps are needed. The percentage of activities with a negative cash flow has a *bell-shaped easy-hard-easy* impact on the computational complexity of the problem. If no activities with negative cash flows are present, the optimal solution reduces to the early-start schedule, that is, no forward shifts and no recursion steps are necessary. If all activities carry negative cash flows, all activities can be shifted forward till one of them hits the deadline, which requires limited computational effort. If, however, activities with positive and negative cash flows are mixed, the problem becomes harder.

Resource availability parameters and the complexity of resource-constrained project scheduling

Resource-based parameters. Elmaghraby and Herroelen[34] were the first to conjecture that the relationship between the complexity of a resource-constrained project scheduling problem (as measured by the CPU-time required for its solution) and resource scarcity (availability) varies according to a bell-shaped curve. If resources are only available in extremely small amounts, there will be relatively little freedom in scheduling the activities. Hence, the corresponding RCPSP instance should be relatively easy to solve. If, on the other hand, resources are amply available, the activities can be simply scheduled in parallel and the resulting project duration will be equal to the critical path length, leading again to a small computational effort ($O(n^2)$).

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)[36] reflects the average portion of resources requested per activity. If RF = 1, then each activity requests all resources. RF = 0 indicates that no activity requests any resource:

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

The *resource strength* RS_k (Cooper[64]) is redefined by Kolisch *et al*[43] as $\left(a_k - r_k^{\min}\right)/(r_k^{\max} - r_k^{\min})$, where a_k is the total availability of renewable resource type k, $r_k^{\min} = \max_{i=1,\ldots,n} r_{ik}$ (the maximum resource requirement for each resource type), and $r_k^{\max}$ is the peak demand for resource type k in the precedence-based early start schedule. Hence, with respect to one resource the smallest feasible resource availability is obtained for $\mathrm{RS}_k = 0$. For $\mathrm{RS}_k = 1$, the problem is no longer resource-constrained. In their experiments, Kolisch *et al*[43] conclude (in contradiction with Alvarez-Valdés and Tamarit[44]) that RS has the strongest impact on solution times: the average solution time continuously increases with decreasing RS.

Patterson[33] defined the *resource-constrainedness,* RC_k, for each resource k as p_k/a_k, where a_k is the availability of resource type k and p_k is the average quantity of resource k demanded when required by an activity. The arguments for using RC and not RS as a resource-based parameter are that (a) RC is a 'pure' measure of resource availability in that it does not incorporate information about the precedence structure of a network, and that (b) there are occasions where RS can no longer distinguish between easy and hard instances while RC continues to do so.

It should be noted that both the resource strength RS_k and the resource-constrainedness RC_k are defined for each renewable resource type k. Hence, their unambiguous use is restricted to the case $k = 1$ or the case where RS_k and RC_k are constant over all k. When this is not the case and the RS- and RC-values would be averaged over all resource types, serious bias may be introduced in the results. A totally unambiguous resource availability measure does not yet exist and remains a valid topic for further research.

Resource availability and the complexity of the RCPSP (problem $m, 1|cpm|C_{\max}$). De Reyck and Herroelen[51] used ProGen to generate nine sets of 500 RCPSP instances with 25 activities and one resource type. 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. CNC is set to 2, while RF is set to 1. Using increments of 0.125, RS is set to 0 for the first set of 500 networks, to 0.125 for the second, up to 1 for the last set. The CI values varied from 7–17. The instances were solved using the branch-and-bound procedure of Demeulemeester and Herroelen.[54] For the nine groups of networks, the required CPU-time varies in function of RS according to a continuous bell-shaped *easy-hard-easy complexity pattern,* in accordance with the conjecture of Elmaghraby and Herroelen.[34] De Reyck and Herroelen[51] observed a similar easy-hard-easy bell-shaped complexity relationship between the CPU-time and RC.

An instance for which RS is small will have a high value for RC. Figure 2 gives a clarifying plot of the required CPU-time versus the resource strength RS (ranging from 1–0) and the resource-constrainedness RC (ranging from 0–100%). The precise correspondance between the RS- and RC-values is not fixed and is only shown for illustrative purposes. Instances with RS $\geqslant 1$ are no longer resource-constrained and can be solved using straightforward critical path analysis (time complexity $O(n^2)$). Instances with RS close to 0 are typically very difficult to solve. For instances with RS < 0, the problem boils down to checking whether the resource requirements exceed the availabilities, in which case the problem becomes infeasible (time complexity $O(nK)$). The plot exhibits a *relatively sharp easy-hard-easy phase transition.* The curve is skewed towards the end of the spectrum with low RS (high RC) values. It should be emphasised that although the average complexity is high at the phase transition boundary, so is the variance. Various problem instances situated in the 'hard' region are not that hard to solve. This again illustrates the fact that the currently used resource availability measures may well be too crude.

Conclusions and suggested research

The observations reported in this paper have revealed intriguing regularities in the structure of various resource-constrained project scheduling problems which confirm the existence of phase transitions in project scheduling. Extensive computational evidence could be obtained for the existence of a continuous hard-easy complexity pattern using the network topology measures order strength (OS) and complexity index (CI) as order parameters. This was found to be the case for the resource-constrained project scheduling problem (RCPSP) with finish-start zero-lag precedence relations (problem $m, 1|cpm|C_{max}$) as well as for the resource-constrained project scheduling problem with generalised precedence relations with both minimal and maximal time lags (problem $m, 1|gpr|C_{max}$), the discrete time/cost trade-off problem (problem $1, T|cpm,disc,mu|curve$) and time/resource trade-off problem (problem $1, 1|cpm,disc,mu|C_{max}$). A continuous easy-hard complexity pattern could also be observed for OS for the problem of maximising the net present value of a project in the absence of resource constraints (problem $cpm,c_j|npv$). The resource-based parameters resource strength (RS) and resource-constrainedness (RC), however, exhibit an easy-hard-easy complexity pattern for the

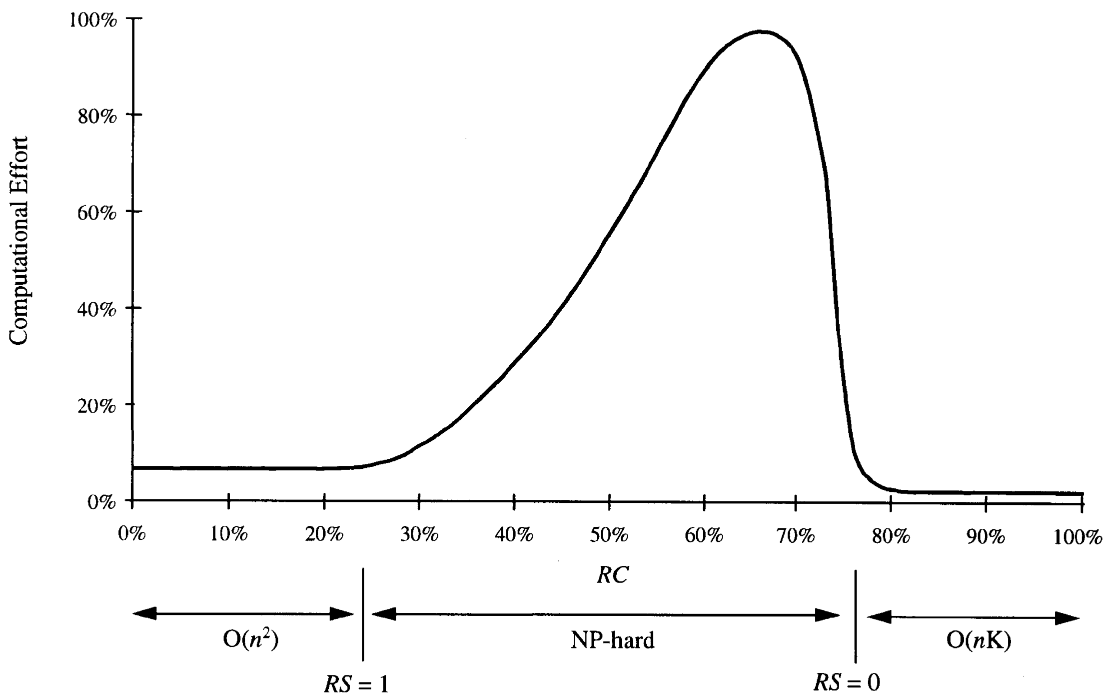

Figure 2 Computational complexity *vs* RC and RS.

RCPSP. These results confirm the Elmaghraby and Herroelen[34] conjecture made back in 1980. Especially the use of RC as an order parameter, reveals the existence of a sharp phase transition.

Phase transition research in AI has been mainly concentrated on NP-complete decision problems. The empirical results reported in this paper provide a confirmative answer to one of the most often cited open questions in AI research, that is the fundamental question whether phase transitions do exist for NP-hard problems.[4,5] Continuous hard-easy transitions for both polynomial and various NP-hard project scheduling problems have been observed for the order parameter OS (order strength), making a strong case for the inclusion of OS in popular problem generators such as ProGen,[43] as evidenced by the recently developed generator ProGen/max.[56] Relatively sharp easy-hard-easy complexity transitions have been observed for the NP-hard resource-constrained project scheduling problem when using resource-constrainedness (RC) as an order parameter. These results also provide additional insight in the intriguing phenomenon observed in AI research (see for example, Hogg *et al*[7]) that hard problems may actually occur in the 'non-critical' region while a random problem instance generated in the supposedly 'hard' region may not actually be that hard to solve.

Obviously, a number of other intriguing open issues and research prospects emerge from the confrontation of AI phase transition research and the validation of (exact) procedures for solving NP-hard scheduling problems. The derivation of network topology measures with sufficient discriminatory power to allow for the observation of sharp easy-hard-easy phase transitions besides the observed continuous hard-easy transitions must be stimulated. Moreover, additional research is needed to refine the location of the phase transitions for resource-constrained project scheduling problems as well as the examination of hard instances among generally easy underconstrained problems. Refining the location of phase transitions might provide a systematic basis for selecting the type of algorithm to use on a given project scheduling problem. Additional research is needed to include order parameters of sufficient discriminatory power in existing and future random problem generators. Random problem generators should generate problem ensembles which span the full range of problem complexity and which can be tuned to fit the unique characteristics of real-world scheduling problems. If the insights provided by the validation results of exact and suboptimal solution procedures for solving NP-hard scheduling problems are to be of practical use, the validation must be done on problem ensembles which distinguish between easy and hard instances and which span the full range of complexity. Even if the order parameters used for evaluating possible phase transitions are still imperfect, knowing where the really hard project scheduling problems are is extremely useful.

Acknowledgements—We are much indebted to Salah Elmaghraby (North Carolina State University at Raleigh, USA) for alerting us to the emerging phase transition literature in artificial intelligence. We also would like to thank the anonymous referees for their helpful suggestions.

References

1 Karp RM (1975). On the computational complexity of combinatorial problems. *Networks* **5**: 45–68.
2 Garey MR and Johnson DS (1979). *Computers and Intractability: A Guide to the Theory of NP Completeness*. Freeman: San Francisco.
3 Turner JS (1988). Almost all *k*-colourable graphs are easy to colour. *J Algorithms* **9**: 63–82.
4 Cheeseman P, Kanefsky B and Taylor WM (1991). Where the really hard problems are. *Proceedings of the International Joint Conference of Artificial Intelligence* **1**: 331–337.
5 Hayes B (1997). Can't get no satisfaction. *Am Sci* **85**: 108–112.
6 Huberman BA and Hogg T (1987). Phase transitions in artificial intelligence systems. *Artificial Intelligence* **33**: 155–171.
7 Hogg T, Hubermann BA and Williams CP (1996). Phase transitions and the search problem. *Artificial Intelligence* **81**: 1–15.
8 Crawford JM and Auton LD (1993). Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence* **81**: 31–57.
9 Freeman JW (1996). Hard random 3-SAT problems and the Davis-Putnam procedure. *Artificial Intelligence* **81**: 183–198.
10 Gent IP and Walsh T (1994). The SAT phase transition. *Proceedings ECAI* **94**: 105–109.
11 Gent IP and Walsh T (1996). The satisfiability constraint gap. *Artificial Intelligence* **81**: 59–80.
12 Kirkpatrick S and Selman B (1994). Critical behavior in the satisfiability of random boolean expressions. *Science* **264**: 1297–1301.
13 Mitchell DG and Levesque HJ (1996). Some pitfalls for experimenters with random SAT. *Artificial Intelligence* **81**: 11–125.
14 Mitchell DG, Selman B and Levesque HJ (1992). Hard and easy distributions of SAT problems. *Proceedings AAAI* **92**: 459–465.
15 Schrag R and Crawford JM (1996). Implicates and prime implicates in random 3-SAT. *Artificial Intelligence* **81**: 199–222.
16 Selman B and Kirkpatrick S (1996). Critical behavior in the computational cost of satisfiability testing. *Artificial Intelligence* **81**: 273–295.
17 Selman B, Mitchell DG and Levesque HJ (1996). Generating hard satisfiability problems. *Artificial Intelligence* **81**: 17–29.
18 Bollobas B (1985). *Random Graphs*. Academic Press: New York.
19 Hogg T (1996). Refining the phase transition in combinatorial search. *Artificial Intelligence* **81**: 127–154.
20 Gent IP and Walsh T (1994). Computational phase transitions in real problems. Research paper 724, Department of Artificial Intelligence, Edingburgh University.
21 Gent IP and Walsh T (1995). The TSP phase transition, Technical Report, Department of Computer Science, University of Strathclyde, Scotland.
22 Zhang W and Korf RE (1996). A study of complexity transitions on the asymmetric travelling salesman problem. *Artificial Intelligence* **81**: 223–239.
23 Karp RM and Pearl J (1983). Searching for an optimal path in a tree with random costs. *Artificial Intelligence* **21**: 99–117.

24 McDiarmid CJH (1990). Probabilistic analysis of tree search. In: Gummett GR and Welsh DJA (eds). *Disorder in Physical Systems.* Oxford Science: Oxford, pp 249–260.
25 McDiarmid CJH and Provan GMA (1991). An expected cost analysis of backtracking and non-backtracking algorithms. *Proceedings IJCAI* **91**: 173–177.
26 Pemberton JC and Zhang W (1996). Epsilon-transformation: exploiting phase transitions to solve combinatorial optimization problems. *Artificial Intelligence* **81**: 297–325.
27 Zhang W and Korf RE (1992). An average case analysis of branch-and-bound with applications: Summary of results. *Proceedings AAAI* **92**: 545–550.
28 Zhang W and Korf RE (1993). Depth-first vs. Best-first search: New results. *Proceedings AAAI* **93**: 769–775.
29 Zhang W and Korf RE (1995). Performance of linear-space search algorithms. *Artificial Intelligence* **79**: 241–292.
30 Jensen TR and Toft B (1995). *Graph Colouring Problems.* John Wiley and Sons: New York.
31 Little JDC, Murty KG, Sweeney DW and Karel C (1963). An algorithm for the travelling salesman problem. *Ops Res* **11**: 972–989.
32 Davis EW (1975). Project network summary measures and constrained resource scheduling. *IIE Trans* **7**: 132–142.
33 Patterson JH (1976). Project scheduling: The effects of problem structure on heuristic performance. *Nav Res Logist* **23**: 95–123.
34 Elmaghraby SE and Herroelen WS (1980). On the measurement of complexity in activity networks. *Eur J Opl Res* **5**: 223–234.
35 De Reyck B (1998). Scheduling projects with generalized precedence relations—Exact and heuristic procedures. Ph.D. dissertation, Department of Applied Economics, Katholieke Universiteit Leuven (Belgium).
36 Pascoe TL (1966). Allocation of resources—CPM. *Rev fr Rech opér* **38**: 31–38.
37 Davies EM (1974). An experimental investigation of resource allocation in multiactivity projects. *Opl Res Q* **24**: 587–591.
38 Kaimann RA (1974). Coefficient of network complexity. *Mgmt Sci* **21**: 172–177.
39 Kaimann RA (1975). Coefficient of network complexity: Erratum. *Mgmt Sci* **21**: 1211–1212.
40 Kurtulus IS and Narula SC (1985). Multi-project scheduling: Analysis of project performance. *IIE Trans* **17**: 58–66.
41 Patterson JH (1984). A comparison of exact procedures for solving the multiple resource-constrained project scheduling problem. *Mgmt Sci* **30**: 854–867.
42 Talbot FB (1982). Resource-constrained project scheduling problem with time-resource trade-offs: The nonpreemptive case. *Mgmt Sci* **28**: 1197–1210.
43 Kolisch R, Sprecher A and Drexl A (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Mgmt Sci* **41**: 1693–1703.
44 Alvarez-Valdés R and Tamarit JM (1989). Heuristic algorithms for resource-constrained project scheduling. In: Slowinski R and Weglarz J (eds). *Advances in Project Scheduling.* Elsevier: Amsterdam, pp 134–143.
45 Mastor AA (1970). An experimental and comparative evaluation of production line balancing techniques. *Mgmt Sci* **16**: 728–746.
46 Kao EPC and Queranne M (1982). On dynamic programming methods for assembly line balancing. *Ops Res* **30**: 375–390.
47 Dar-El EM (1973). MALB—A heuristic technique for balancing large single-model assembly lines. *AIIE Trans* **5**: 343–356.
48 De Reyck B (1995). On the use of the restrictiveness as a measure of complexity for resource-constrained project scheduling. Research Report 9535, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.
49 Thesen A (1977). Measures of the restrictiveness of project networks. *Networks* **7**: 193–208.
50 Bein WW, Kamburowski J and Stallmann MFM (1992). Optimal reduction of two-terminal directed acyclic graphs. *SIAM J on Comp* **21**: 1112–1129.
51 De Reyck B and Herroelen W (1996). On the use of the complexity index as a measure of complexity in activity networks. *Eur J Opl Res* **91**: 347–366.
52 Herroelen W, Demeulemeester E and De Reyck B (1998). A classification scheme for project scheduling problems. In: Weglarz J (ed). *Project scheduling: Recent models, algorithms and applications (Chapter 1).* Kluwer Academic Publishers, Boston, USA.
53 Herroelen W, De Reyck B and Demeulemeester E (1998). Resource-constrained project scheduling: A survey of recent developments. *Comp and Opns Res* **25**: 279–302.
54 Demeulemeester E and Herroelen W (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Mgmt Sci* **38**: 1803–1818.
55 Demeulemeester E and Herroelen W (1997). New benchmark results for the resource-constrained project scheduling problem. *Mgmt Sci* **43**: 1485–1492.
56 Schwindt C (1996). Generation of resource-constrained project scheduling problems with minimal and maximal time lags. Report WIOR-489, Institut für Wirtschafstheorie und Operations Research, Universität Karlsruhe.
57 De Reyck B and Herroelen W (1998). A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations. *Eur J Opl Res* **111**: 125–174.
58 Demeulemeester E, Elmaghraby SE and Herroelen WS (1996). Optimal procedures for the discrete time/cost trade-off problem in project networks. *Eur J Opl Res* **88**: 50–68.
59 Demeulemeester E, De Reyck B, Foubert B, Herroelen W, and Vanhoucke M (1998). New computational results on the discrete time/cost trade-off problem in project networks. *J Opl Res Soc* **49**: 1153–1163.
60 Demeulemeester E, De Reyck B and Herroelen W (1997). The discrete time/resource trade-off problem in project networks—A branch-and-bound approach. Research Report 9717, Department of Applied Economics, Katholieke Universiteit Leuven (Belgium).
61 De Reyck B, Demeulemeester E and Herroelen W (1998). Local search methods for the discrete time/resource trade-off problem in project networks. *Nav Res Logist* **45**: 533–578.
62 Herroelen W, Demeulemeester E and Van Dommelen P (1996). An optimal recursive search procedure for the deterministic unconstrained max-npv project scheduling problem. Research Report 9603, Department of Applied Economics, Katholieke Universiteit Leuven (Belgium).
63 De Reyck B and Herroelen W (1998). An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations. *Comp and Opns Res* **25**: 1–17.
64 Cooper DF (1976). Heuristics for scheduling resource-constrained projects: An experimental comparison. *Mgmt Sci* **22**: 1186–1194.

Received April 1998;
accepted October 1998 after one revision