KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Economics and Applied Economics

Resource-constrained project scheduling for timely project completion with stochastic activity durations

Francisco Ballestin and Roel Leus

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 0723

Resource-constrained project scheduling for timely project completion with stochastic activity durations

Francisco Ballestín
Department of Statistics and Operations Research
Universidad Pública de Navarra, Spain
Tel. +34 948 169 216; Fax +34 948 169 204
Francisco.Ballestin@unavarra.es

Roel Leus
Department of Decision Sciences and Information Management
Katholieke Universiteit Leuven, Belgium
Tel. +32 16 32 69 67; Fax +32 16 32 67 32
Roel.Leus@econ.kuleuven.be

— July 2007 —

Resource-constrained project scheduling for timely project completion with stochastic activity durations

We investigate resource-constrained project scheduling with stochastic activity durations. Various objective functions related to timely project completion are examined, as well as the correlation between these objectives. We develop a GRASP-heuristic to produce high-quality solutions, using so-called descriptive sampling. The algorithm outperforms other existing algorithms for expected-makespan minimization. The distribution of the possible makespan realizations for a given scheduling policy is studied, and problem difficulty is explored as a function of problem parameters.

Keywords: project scheduling; uncertainty; GRASP.

1. Introduction

The larger part of the scientific literature on resource-constrained project scheduling focuses on the minimization of the project duration in a deterministic setting. The goal of the resource-constrained project scheduling problem (RCPSP) is to minimize the duration of a project subject to finish-start precedence constraints and renewable resource constraints. It is shown in Blazewicz et al. (1983) that the RCPSP, as a job-shop generalization, is NP-hard in the strong sense. A lot of exact and heuristic procedures have been proposed to construct workable baseline schedules that solve this deterministic RCPSP, see Demeulemeester and Herroelen (2002), Kolisch and Padman (2001) and Neumann et al. (2002) for recent overviews.

During project execution, however, project activities are often subject to considerable uncertainty, which results from many different possible sources: activities may take more or less time than originally estimated, resources may become unavailable, material may arrive behind schedule, workers may be absent, etc. In this article we examine the case where this uncertainty is important enough to be incorporated into the planning phase. The sources of variability in processing times are manifold; nevertheless, the main scheduling objectives are mostly functions of the activity starting- (or ending-)times, the project makespan being the single most-studied objective, next to other such as weighted earliness-tardiness and net present value of the project. This justifies a restriction to the study of uncertainty in processing times only, although many different sources may be at the basis of this variability. The stochastic resource-constrained project scheduling problem (stochastic RCPSP or

SRCPSP) is the stochastic equivalent of the RCPSP, where the durations of the activities are not known in advance but are represented as random variables. The probability distributions can be either objective (a risk situation) or result from subjective judgment (in the case of decision-theoretic uncertainty or even ignorance).

The SRCPSP usually aims at minimizing the expected makespan over a limited set of possible decisions to be taken during project execution. As coherently described by Stork (2001), an important new aspect comes into play on moving from the deterministic to the stochastic case: what is a solution to an SRCPSP-instance? A deterministic schedule does not necessarily contain enough information to make decisions during the execution of the project. Hence, a solution should define for each possible event that occurs within the execution of the project an appropriate action, typically the start of new activities. To make such decisions, one may want to exploit the information given by the current state of the project. In line with Igelmund and Radermacher (1983), among others, we call such a solution a (scheduling) *policy*.

A vast amount of literature exists on the so-called (generalized) PERT-problem, where no resource constraints are taken into consideration. These studies are usually concerned with the computation of certain characteristics of the project makespan (earliest project completion), mainly with exact computation, approximation and bounding of the distribution function and the expected value. Note that in this case, no real scheduling effort is required: all activities can be started when their predecessors are completed. For a review of research up until 1987, we refer to Adlakha and Kulkarni (1989). A recent computational study on bounding the makespan distribution, in which the most promising algorithms are compared, is given by Ludwig et al. (2001).

The work on the SRCPSP, however, has remained rather limited until now. There are only few computational publications on this problem: Igelmund and Radermacher (1983) and Stork (2001) report on experiments with branch-and-bound algorithms, while Golenko-Ginzburg and Gonik (1997) and Tsai and Gemmill (1998) develop greedy and local-search heuristics. Time/resource trade-offs with stochastic activity durations, in which the resource allocation influences the mean and/or the variance of the durations, are investigated in Gerchak (2000), Gutjahr et al. (2000) and Wollmer (1985).

The contributions of this article are sixfold: (1) we examine multiple possible objective functions for project scheduling with stochastic activity durations; (2) we show by computational experiments that these different objective functions are closely connected and that for most practical purposes, it suffices to focus on the minimization of the expected makespan; (3) we develop a GRASP-heuristic that produces high-quality solutions, outperforming existing algorithms for expected-makespan minimization; (4) the variance-reduction technique of descriptive sampling is applied and its benefits assessed; (5) the distribution of makespan realizations for a given scheduling policy is studied; and (6) problem difficulty is explored as a function of problem parameters.

The remainder of this article is organized as follows. Definitions and a detailed problem statement are provided in Section 2, followed by a discussion of the computational setup (Section 3). Section 4 presents the basic ingredients of our GRASP-algorithm. Our main computational results for the expected-makespan objective can be found in Section 5; the relationship between the expected makespan and some other objective functions is treated in Section 6. The distribution of makespan realizations is the subject of Section 7, and we try to characterize problem difficulty as a function of problem parameters in Section 8. Finally, a summary is given in Section 9.

2. Definitions and problem statement

This section contains a number of definitions (Section 2.1), a discussion of scheduling policies (Section 2.2), and a statement of the problems that we wish to solve (Section 2.3).

2.1 Definitions

A project consists of a set of activities $N = \{0, 1, \ldots, n\}$, which are to be processed without interruption on a number K of renewable resource types with availability a_k, $k = 1, \ldots, K$; each activity i requires $r_{ik} \in \mathbb{N}$ units of resource type k. The duration D_i of activity i is a random variable (r.v.); the vector $(D_0, D_1, \ldots, D_n)$ is denoted by $\mathbf{D}$. A is a (strict) partial order on N, i.e. an irreflexive and transitive relation, which represents technological precedence constraints. (Dummy) activities 0 and n represent start and end of the project, respectively, and are the (unique) least and greatest element of the partially ordered set (N, A). Activities 0 and n have zero resource usage and $Pr[D_i = 0] = 1$ for $i = 0, n$; for the remaining activities $i \in N \backslash \{0, n\}$ we assume that $Pr[D_i < 0] = 0$ ($Pr[e]$ represents the probability of event e). We associate the directed acyclic graph $G(N, A)$ with the partially ordered set (N, A).

We use lowercase vector $\mathbf{d} = (d_0, d_1, \ldots, d_n)$ to represent one particular realization (or sample, or scenario) of $\mathbf{D}$. Alternatively, when each duration D_i is a constant, we use the same notation $\mathbf{d}$. In the (deterministic) RCPSP, each duration D_i is a constant integer value. A solution for the RCPSP is a schedule $\mathbf{s}$, i.e., a vector of starting times $(s_0, s_1, \ldots, s_n)$ with $s_i \geq 0$ for all $i \in N$, that is both time-feasible and resource-feasible. Schedule $\mathbf{s}$ is called *time-feasible* if $s_i + d_i \leq s_j$ for all $(i, j) \in A$; $\mathbf{s}$ is said to be *resource-feasible* if, at any time t and for each resource type k, it holds that $\sum_{i \in \mathcal{A}(\mathbf{s},t)} r_{ik} \leq a_k$, where the *active set* $\mathcal{A}(\mathbf{s}, t) = \{i \in N | s_i \leq t < s_i + d_i\}$ contains the activities in $N \backslash \{0, n\}$ that are in progress at time t. The objective function in RCPSP is the project makespan s_n (which is to be minimized).

2.2 Scheduling policies

The execution of a project in the context of SRCPSP can best be seen as a dynamic decision process. A solution is now a *policy* Π, which defines *actions* at *decision times.* Decision times are typically $t = 0$ (the start of the project) and the completion times of activities. An action can entail the start of a set of activities that is precedence- and resource-feasible. A schedule is thus constructed gradually through time. A decision at time t may only use information that has become available before or at time t; this requirement is often referred to as the *non-anticipativity constraint.* As soon as all activities are completed, the activity durations are known, yielding a realization $\mathbf{d}$ of $\mathbf{D}$. Consequently, every policy Π may alternatively be interpreted (cfr. Igelmund and Radermacher, 1983; Stork, 2001) as a function $\mathbb{R}^n_{\geq} \to \mathbb{R}^n_{\geq}$ that maps given samples $\mathbf{d}$ of activity durations to vectors $\mathbf{s}(\mathbf{d}; \Pi) \in \mathbb{R}^n$ of feasible activity starting times (schedules); if no misinterpretation is possible we usually omit the identification of the policy and write $\mathbf{s}(\mathbf{d})$. For a given scenario $\mathbf{d}$ and policy Π, $s_n(\mathbf{d}; \Pi)$ denotes the makespan of the schedule. The most-studied objective for the SRCPSP is to select a policy Π^* within a specific class that minimizes $E[s_n(\mathbf{D}; \Pi)]$, with $E[\cdot]$ the expectation operator with respect to $\mathbf{D}$.

A well-known class of scheduling policies is the class of priority policies, which order all activities according to a priority list and, at every decision point t, start as many activities as possible in the order dictated by the list (in line with the parallel schedule generation scheme – parallel SGS – see Kolisch and Hartmann, 1999). These list-scheduling policies present a number of drawbacks. First of all, priority policies cannot guarantee an optimal schedule. Moreover, the change of activity durations may lead to so-called Graham anomalies (Graham, 1966) such as increasing project duration due to decreasing activity durations. Stork (2001) describes how, if we consider the interpretation of a policy as a function, these anomalies lead us to conclude that priority policies are neither monotone nor continuous.

Several other classes of policies have been examined by Stork (2001), most of which exhibit severe computational limitations; he concludes that, for larger instances, the only remaining alternative is to use the class of so-called activity-based policies, which is also the class that will be studied in this paper. An activity-based policy $\Pi(L)$ is also represented by a priority list L of the activities and, for a given sample $\mathbf{d}$, computes starting times by starting each activity in the order imposed by L as early as possible, with the side constraint that $s_i(\mathbf{d}) \leq s_j(\mathbf{d})$ if $i \prec_L j$. Elimination of this side constraint would yield a simple priority policy that suffers from the Graham anomalies, but the 'activity-based' point of view, rather than greedy 'resource-based', does away with this problem. Since these activity-based policies perform activity-incrementation rather than time-incrementation, they can alternatively be referred to as '(stochastic) serial SGS' (Ballestín, 2007).

2.3 Problem statement

The literature on project management abounds with motivations for reducing project lead times, including various first-mover advantages in new-product development (see, for instance, Smith and Reinertsen, 1991), advantages during the bidding process (Kerzner, 1998; Newbold, 1998; Xie et al., 2006), and incentive contracts that foresee a penalty for delayed completion or a bonus for early delivery (Bayiz and Corbett, 2005). Our focus on timely project completion by using some characterization of the makespan (which is a stochastic variable) as objective function, is therefore evident. We elaborate on this choice in the paragraphs below.

2.3.1 Individual projects

French (1988) describes four criteria for decision making under uncertainty, which for a minimization problem amount to (1) *minimax* (minimize the worst makespan realization that can occur), (2) *minimin* (minimize the best outcome that can occur, which is an optimistic approach, as opposed to the pessimistic minimax), (3) *minimax regret* (minimize the largest possible difference in makespan between the policy to be selected, and the optimal makespan for a given realization), and (4) minimize the objective *in expectation.* Scheduling with objectives (1) and (3) is studied in Kouvelis and Yu (1997); we do not adopt these objectives because (a) one normally needs discrete scenarios instead of continuous distributions, and

(b) the evaluation of the optimal objective function for constant durations should be easy in order to be able to produce computational results for average-size instances (in the case of Kouvelis and Yu: single-machine scheduling with total-flow-time objective and two-machine flow-shop with makespan objective). Since a practical decision maker is usually risk-averse, we also do not investigate objective (2).

In conclusion, when a project is to be executed in isolation (e.g. for internal clients), expected makespan is the most logical objective of the foregoing. Actually, it is well-known that the expectation criterion is most appropriate for a *risk-neutral* decision maker, and in order to account for possible risk averseness, rather than work with utility functions (with their inherent difficulty of estimation), one may put forward the use of constraints of the form $Pr[s_n(\mathbf{D}) \geq \delta] \leq p$, for given probability p and deadline δ, as an approximate representation of risk averseness, which represents the undesirability of exceptionally high makespan realizations, and which is comparable with downside-risk or Value-at-Risk (VaR) constraints in Finance (Ang et al., 2006; Jorion, 2000). Likewise, Schuyler (2001) also advocates conservatism when large potential gains and losses are associated with individual decisions.

A second way to account for risk averseness is to investigate the trade-off between the expected makespan $E[s_n(\mathbf{D})]$ and the makespan variance $\mathrm{var}[s_n(\mathbf{D})]$. This is in line with Portougal and Trietsch (1998) who suggest that "variance reduction should be introduced explicitly in the objective, while retaining the expected completion time as well". Similarly, Elmaghraby et al. (1999) also distinguish both mean and variance of the project duration as the two prime performance measures of concern. In Gutierrez and Paul (2001) the impact is examined of variability in activity durations on mean project duration, while Cho and Yum (1997) focus more on the sensitivity of makespan variability. Finally, knowledge of the entire distribution function of makespan realizations for a given policy is obviously also highly informative to the decision maker; we investigate this in a separate section (Section 7).

2.3.2 External clients

The foregoing discussion looked into the execution of a project in isolation. When a project deadline has been negotiated beforehand with external clients, however, it may be more useful to adapt the scheduling objective function in order to reflect possible applicable penalty structures (we do not focus on bonuses for early completion), which take the form either of a fixed charge, so that the objective function becomes $\max Pr[s_n \leq \delta]$ for a given deadline δ, or of a fixed charge *per unit-time overrun*, leading to $\min E[\max\{0; s_n - \delta\}]$ (see Gutjahr et al. (2000) for a model in a slightly different context that uses more general loss functions).

2.3.3 Problem statement: conclusion

In conclusion, we are left with three objective functions to investigate: $\max Pr[s_n \leq \delta]$, henceforth, in line with Portougal and Trietsch (1998), referred to as the *service-level* objective due to its similarity with inventory management (see, for instance, Silver et al., 1998); $\min E[s_n]$ (the *expectation* objective), and $\min E[\max\{0; s_n - \delta\}]$ (the *(expected) overrun* objective). For the benefit of risk averseness, a lower bound may be imposed on the service level, or an upper bound on the makespan variance. In the remainder of this text, we use the term *statistic* to refer to any function of s_n. Obviously, a decision maker may also be interested in possible trade-offs between the different suggested statistics. The correlation between the statistics will be studied in Section 6.

3. Computational setup

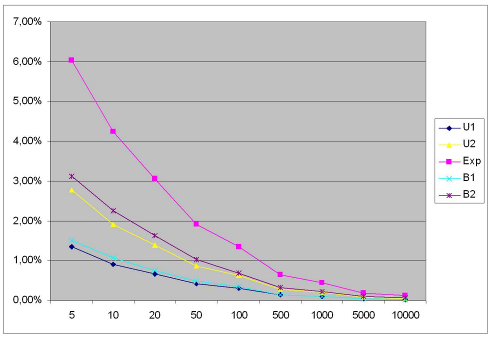
The analyses in the next sections are based on computational experiments using randomly generated datasets. The coding was performed in C using the Microsoft Visual C++ 6.0 programming environment, and the experiments were run on a Samsung X15 Plus portable computer with Pentium M processor with 1,400 MHz clock speed and 512 MB RAM, equipped with Windows XP. Our tests are performed on instances from the benchmark library PSPLIB, which contains instances of different size and with different characteristics of the deterministic RCPSP, and which were generated by the problem generator *ProGen* (Kolisch and Sprecher, 1996); the number of resource types $K \leq 4$. We only use the dataset containing 600 instances with 120 activities (commonly named 'j120'). The deterministic duration d_i^* for each activity $i \neq 0, n$ is an integer randomly chosen from $\{1, 2, \ldots, 10\}$.

We generate the probability distributions of the stochastic job processing times D_i (which are not created by the ProGen instance generator) in line with Stork (2001): we take the given deterministic processing time d_i^* of each job as expectation and we construct uniform, exponential and beta distributions. More specifically, we examine five distributions: (continuous) uniform on $[d_i^* - \sqrt{d_i^*}; d_i^* + \sqrt{d_i^*}]$ (subsequently referred to as case 'U1'); (continuous) uniform on $[0; 2d_i^*]$ ('U2'); exponential with expectation d_i^* ('Exp'); beta distribution with support $[d_i^*/2; 2d_i^*]$ and variance $= d_i^*/3$ ('B1'); and beta distribution with support $[d_i^*/2; 2d_i^*]$

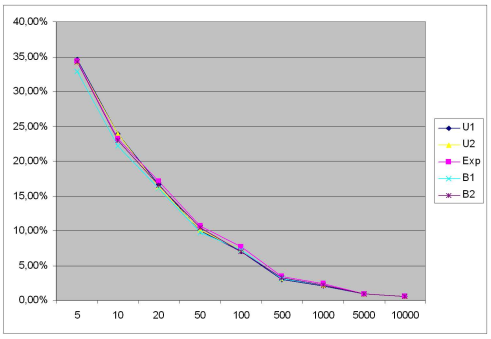
and variance = $d_i^{*2}/3$ ('B2'). These five distributions have variances of $d_i^*/3$, $d_i^{*2}/3$, d_i^{*2}, $d_i^*/3$ and $d_i^{*2}/3$, respectively. The distributions have been created so that U1 and B1 on the one hand and U2 and B2 on the other hand share the same variance.

Exact evaluation of the statistics of our interest (expectation, variance, ...) is overly time-consuming (for the expected makespan, for instance, this amounts to the PERT problem, which is well-known to constitute a formidable computational challenge, see Hagstrom (1988)), which is why we approximate these values by means of simulation. Similar decisions in the context of scheduling under uncertainty have been made by Ballestín (2007), Leus and Herroelen (2004), Möhring and Radermacher (1989) and Stork (2001), among others. A solution in this article is an activity-based policy $\Pi(L)$, which is represented by an activity list L. An approximation of any statistic $g(L)$ associated with $\Pi(L)$ is based on sampling a number of realizations from $\mathbf{D}$; the number of replications is a parameter. Stork (2001), for instance, states that, for the expected makespan, 200 samples turn out to provide a reasonable trade-off between precision and computational effort.

For our computational experiments we have examined the standard deviation of the percentage deviation of simulated versus 'true' makespan (the last one obtained from a high number (25000) of runs) over the dataset. For a justification of the standard deviation as an accuracy measure, and for a discussion of the convergence of the estimate towards the true value as the number of samples increases, see Kleywegt et al. (2001). Leus and Herroelen (2004) note that the number of simulation runs corresponding with the same standard deviation decreases with the number of activities; this approach has the advantage of reducing (relative) simulation effort for larger problem instances. For instances with 120 activities (the only project size we work with), our observations are summarized in Figure 1, where the accuracy of our estimate of the expectation and standard deviation of makespan is depicted; the corresponding graph for the service level is very similar to Figure 1(a), the graph for the expected overrun is close to Figure 1(b). Especially in Figure 1(a) we see that the same number of replications leads to less accurate results for distributions with higher variability. We conclude that 1000 replications lead to a (lack of) accuracy of below 1% for the expected makespan. For 1000 replications, Figure 2 shows the quality of the approximation of the due dates corresponding with different service levels (estimated based on the order statistics). We learn from the U-shaped graphs that it is easier to approximate due dates corresponding with service levels in the middle of the interval $[0; 100\%]$ than in the tails; we come back to this in Section 6.



(a) Expected makespan.



(b) Makespan standard deviation.

Figure 1: Accuracy, measured using the standard deviation of the percentage error, as a function of the number of replications.

Ballestín (2007) shows that using fewer scenarios to calculate the approximation of each activity list during the algorithm favors the calculation of more policies and leads to better solutions at the end of the procedure. Those scenarios were calculated using random sampling, as is common in most algorithms that work with scenarios. In this paper we use the so-called *descriptive sampling* (Saliby, 1990, 1997), which is one particular variance-reduction technique. Concretely, 1000 replications is the default number we work with for computing correlations and fitting distributions (Sections 6, 7 and 8). When our focus is on the highest-quality solutions obtainable with a given computational effort, on the other hand, we consider the number of replications as a parameter of the algorithm. In our computational results in Section 5, we opt for 10 replications (which was the best tested number in Ballestín, 2007), and we provide empirical evidence that such a low number is preferable.

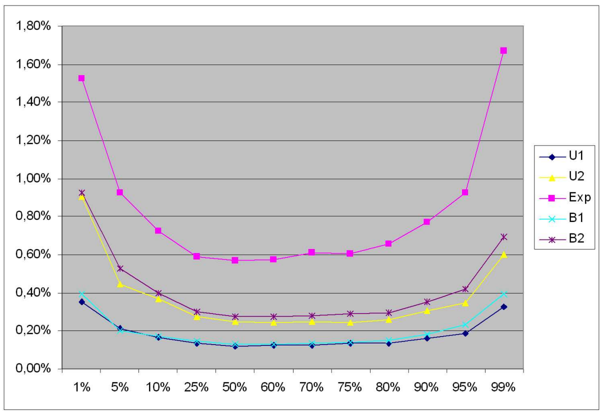


Figure 2: Accuracy of approximation of the service level for various due dates.

4. GRASP

Below, we discuss GRASP a a general heuristic procedure (Section 4.1) and we describe the overall structure of our search procedure for SRCPSP-solutions (Section 4.2).

4.1 GRASP as a general-purpose metaheuristic

A greedy randomized adaptive search procedure (GRASP) is a multi-start or iterative process (Feo and Resende, 1995, 2000; Aiex et al., 2002). Each GRASP-iteration consists of two phases: in a construction phase, a feasible solution is produced and in a local-search phase, a local optimum in the neighborhood of the constructed solution is sought. The best overall solution is kept as the result.

In the construction phase, a feasible solution is iteratively constructed, one element at a time. The basic construction phase in GRASP is similar to the semi-greedy heuristic proposed independently by Hart and Shogan (1987). At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list $\mathcal{C}$ with respect to a greedy function $\mathcal{C} \rightarrow \mathbb{R}$. This function measures the (myopic) benefit of selecting each element. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a GRASP resides in the fact that we choose one of the best candidates in the list, but not necessarily the top candidate; the list of best candidates is called the restricted candidate list. It is almost always beneficial to apply a local-search procedure to attempt to improve each constructed solution.

4.2 Adapting GRASP to our setting

The global structure of our GRASP-implementation is represented as Algorithm 1. Our basic algorithm maintains a set $\mathsf{EliteSet}$ of elite solutions (activity lists), containing the best solutions so far encountered. At each iteration of the algorithm, the solutions in $\mathsf{EliteSet}$ are used to create a new activity list with the procedure $\mathsf{BuildActList}$. Subsequently, a schedule $\mathbf{s}^*$ is built by applying a job-based policy with the mean durations $\mathbf{d}^*$ to this list. After a local-search procedure that attempts to improve this deterministic schedule, we re-translate schedule $\mathbf{s}^*$ into an activity list by means of function $\mathsf{ScheduleToList}$, ordering activities by starting times. This yields the representation of a new activity-based policy, whose objective

Algorithm 1 GRASP: global algorithmic structure

```
EliteSet = $\varnothing$
while TerminationCriterion not met do
    $L$ = BuildActList(EliteSet)
    $\mathbf{s}^* = \mathbf{s}(\mathbf{d}^*, \Pi(L))$
    $\mathbf{s}^*$ = LocalSearch($\mathbf{s}^*$)
    $L$ = ScheduleToList($\mathbf{s}^*$)
    Evaluate the activity-based policy $\Pi(L)$
    if $L$ is better than the worst solution $L'$ in EliteSet then
        EliteSet = (EliteSet $\setminus L'$) $\cup L$
    end if
end while
Return the best solution found
```

function is then evaluated (using simulation). If it is better than the worst solution in the elite set, we erase that solution and include the new one.

We have implemented a local-search procedure based on the concept of *justification.* The (double) justification of a schedule consists in first scheduling the activities as late as possible in non-increasing order of their finish times and then scheduling the activities of the obtained solution as soon as possible in non-decreasing order of their start times. In the RCPSP and some of its generalizations, justification has proved to be very efficient: the makespan of a solution is never worse after justification, and often lower. The technique is based on principles described in Li and Willis (1992) and Özdamar and Ulusoy (1996), and Valls et al. (2005) show that incorporating this technique in several different heuristic algorithms for the RCPSP could improve their quality without an increase in the computation times; this implementation is referred to as 'LS1'. Additionally, in order to further improve the quality of the solution, we also investigate the application of a two-point crossover with as input the schedules before and after justification (the resulting method is called 'LS2').

At each iteration of BuildActList (see Algorithm 2) an eligible activity is selected, until we obtain a complete activity list. An activity is called eligible when all its predecessors have been selected. A greedy way to choose the activity could be to use the best solution found so far as the *reference* for this selection – that is, to select the eligible activity that comes first in that activity list. In order to randomize the selection, we will randomly choose among the elite set the solution that will serve as the reference. An elite solution remains the reference in the following nit $\in$ [nitmin ; nitmax] iterations (randomly chosen).

To add more randomness, we also include the possibility that the eligible activity is

Algorithm 2 BuildActList

```
$i = 0$; EligibleSet $= \{0\}$; nit $= 0$
while $i < n$ do
  if nit $= 0$ then
    reference = SelectSolution
    if reference $\neq$ "LFT", "random" then
      nit $\in$ [nitmin ; nitmax]
    end if
  else
    nit = nit $-1$
  end if
  Select an activity $j$ from EligibleSet according to the reference
  $L(i) = j$; $i = i + 1$
end while
Return the activity list $L$
```

chosen according to its latest finish time or that it is chosen randomly; these latter two options are only applied in a small fraction pLFT and pRandom, respectively; in this case nit = 0 (other values have been examined but lead to worse results). In the first iterations of our GRASP, when the elite set is not full, we set the values of pLFT and pRandom to 95% and 5%, respectively.

In order to introduce diversity in the procedure, we have included the possibility of a yet different reference solution in the function SelectSolution (see Algorithm 3), namely the *inverse* of a list in EliteSet, with probability pInverse, denoted as inv(). In this case, a list L from EliteSet is chosen, and the next activity in BuildActList is an eligible activity with highest position in L.

Algorithm 3 SelectSolution

```
Draw $p \in [0; 1]$
if $p <$ pLFT then
  reference = "LFT"
else if $p <$ pLFT + pRandom then
  reference = "random"
else
  A reference solution is randomly drawn from EliteSet
  if $p <$ pLFT + pRandom + pInverse then
    reference = inv(reference)
  end if
end if
Return reference solution
```

5. Computational results for the expected makespan

In this section we compare different versions of our GRASP-implementation with expected-makespan objective in order to evaluate the quality of the overall algorithm and its individual elements (Section 5.1), and we provide a comparison of our algorithmic performance with other recently proposed algorithms (Section 5.2).

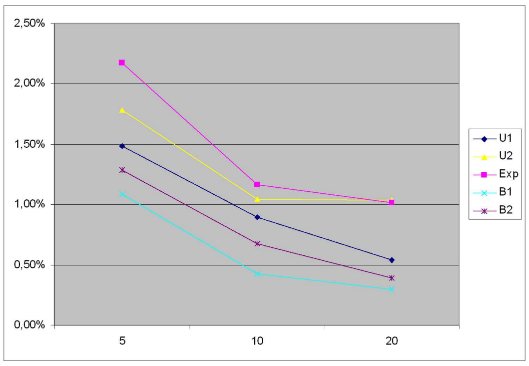
5.1 Details of our GRASP-implementation

The quality of an algorithm is measured by the percentage distance of $E[s_n(\mathbf{D};\Pi(L))]$ (approximated using 1000 replications, independent from the ones used in the optimization phase) from the critical-path length of the project with deterministic mean durations d_i^*. These percentages are averaged over all the instances of the set j120. In the literature on heuristics for the deterministic RCPSP, it is common (see, e.g., Hartmann and Kolisch, 2000) to impose a limit on the number of generated schedules, for ease of comparison of different algorithms regardless of the computer infrastructure. We work with two limits on the number of schedules: 5000 and 25000. Due to the particularities of job-based policies (an activity cannot be scheduled before an already scheduled activity), it turns out (see Ballestín, 2007) that job-based policies are about twice as fast as the deterministic serial SGS. Consequently, we count one scheduling pass of a job-based policy as 0.5.

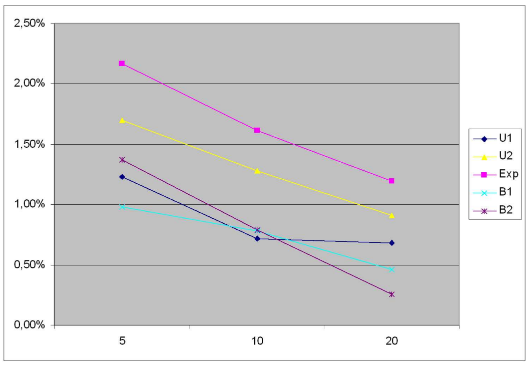
The first line of Table 1 shows the results of the final version of our algorithm, simply called 'GRASP', where pInverse = 0, pRandom = 0.05 and LS2 is used. The second line, labelled 'Basic', pertains to an implementation without local search and without descriptive

Distribution	U1		U2		Exp		B1		B2	
# schedules	5000	25000	5000	25000	5000	25000	5000	25000	5000	25000
GRASP	46.84	45.21	72.58	70.95	114.42	112.37	47.17	45.60	75.97	74.17
Basic	50.57	48.68	76.55	74.78	118.72	117.20	50.70	48.99	79.49	77.64
Basic+DS	50.12	48.33	75.76	73.83	117.36	115.34	50.49	48.61	78.96	77.04
GRASP-LS1	49.17	47.73	76.68	74.93	121.07	119.02	49.75	48.14	80.50	78.62
Inverse	46.93	45.35	72.67	70.97	114.40	112.46	47.25	45.58	76.00	74.22
100 iter	49.81	46.73	75.36	71.76	116.76	112.36	50.20	47.18	78.63	75.00
500 iter	53.04	49.53	79.59	75.05	122.69	116.16	53.40	49.89	83.06	78.26
1000 iter	53.79	51.15	80.50	76.95	123.70	118.77	54.15	51.51	83.97	80.28

Table 1: Percentage distance of $E[s_n(\mathbf{D};\Pi(L))]$ (approximated using 1000 replications) from the critical-path length of the project with deterministic mean durations d_i^*.



(a) 5000 schedules.



(b) 25000 schedules.

Figure 3: Computational results for descriptive sampling.

sampling. 'Basic+DS' refers to the inclusion of descriptive sampling (DS) in Basic. 'GRASP-LS1' is GRASP with LS1 instead of LS2. In 'Inverse' we set pInverse = 0.05 and pRandom = 0, to study whether the use of solutions in EliteSet to attain diversity is useful. Finally, the last three lines of the table give the computational performance of GRASP with 100, 500 and 1000 replications rather than just 10.

We observe that the biggest improvement comes from changing from LS1 to LS2. The straightforward use of justification improves the quality of Basic+DS only when activity-duration variability is low (U1-B1), but worsens it in the other cases. However, LS2 manages to improve the results in all cases. The use of elite solutions in an inverse manner to introduce diversity does not add anything to the results. It is our opinion nevertheless that it is a good way to diversify the search and we will in the future try to look for other ways to implement this idea. A low number of replications also turns out to perform considerably better (as was hinted at at the end of Section 3).

Finally, the inclusion of the DS also (slightly) improves the results of the basic algorithm. We have computed the improvement in the average deviation from the critical-path length that is obtained by the incorporation of DS (compared to random sampling) when we work with 5, 10 and 20 replications; the results for a schedule limit of 5000 and 25000 schedules can be found in Figure 3. The gain obtained by DS tends to decrease as the number of replications increases, both for 5000 and 25000 schedules, and this trend is independent of the duration distribution. Specifically, the gain is up to 2% with five replications for some distributions, making the incorporation of DS clearly worthwhile. Perhaps more important than the technique of DS itself is the fact that this opens a new area of research in heuristic algorithms with replications, namely regarding the implementation of the sampling: to the best of our knowledge, up until now only random sampling has been used. Based on the foregoing data, we can recommend the use of descriptive rather than random sampling in heuristic search especially when the number of replications is low, in which case the potential improvement appears to be significant.

5.2 Comparison with state-of-the-art algorithms

We are now ready to compare our GRASP-algorithm with other SRCPSP-algorithms from the literature. First of all, we consider the genetic algorithm (GA) of Ballestín (2007), where the same dataset and schedule limit are used, with distributions U1, U2 and Exp. Table 2 provides a comparison; we can see that GRASP outperforms the GA in all cases. If we look back to Table 1, even our Basic algorithm does better than the GA, which emphasizes the improvement obtained by adding the descriptive sampling and LS2.

Secondly, we consider the tabu search (TS) and simulated annealing (SA) of Tsai and Gemmill (1998), where algorithmic performance is evaluated on the Patterson dataset (Patterson, 1984), with adaptations for obtaining stochastic (beta) activity durations. As a measure of the quality of their algorithms, the authors report the deviation from an approximate lower bound. Table 3 shows their results obtained on a personal computer with 166 MHz; SA2 and TS2 differ from SA1 and TS1 only in the parameters settings; the two final columns contain the results of our GRASP-algorithm on the same problem set. The GRASP-algorithm with a limit of 5000 schedules outperforms both the SA and the TS in quality and in time, even if we take into account the difference in computer infrastructure.

Distribution	U1		U2		Exp	
# schedules	5000	25000	5000	25000	5000	25000
GA	51.94%	49.63%	78.65%	75.38%	120.22%	116.83%
GRASP	46.84%	45.21%	72.58%	70.95%	114.42%	112.37%

Table 2: Comparison between GRASP and GA.

Algorithm	SA1	SA2	TS1	TS2	GRASP (5000)	GRASP (25000)
Above approximate lower bound	3.40%	2.27%	3.71%	2.54%	2.01%	1.96
Average time (s)	10.804	21.414	5.834	11.290	0.92	4.24

Table 3: Comparison between GRASP (with 5000 and 25000 schedules) and the algorithms of Tsai and Gemmil (1998).

Distribution	beta	uniform	normal
Heuristic 1	433.88	448.49	448.85
Heuristic 2	447.98	461.35	461.58
GRASP (5000)	408.75	427.64	422.04
GRASP (25000)	403.16	424.28	415.40

Table 4: Expected makespan for the instance from Golenko-Ginzburg and Gonik (1997).

The small difference between 5000 and 25000 schedules might be due to the fact that the solutions found are near-optimal.

Golenko-Ginzburg and Gonik (1997) test their algorithms only on one instance with 36 activities and a single resource type; Table 4 contains their and our results for this instance for three duration distributions. The authors do not report on running times for the procedures but only point out that the algorithm that uses an exact procedure to solve consecutive multi-dimensional knapsack problems (Heuristic 1) needs much more time than the algorithm that solves these problems heuristically (Heuristic 2). Obviously, no strong conclusions can be drawn based on only one instance, but the difference between the algorithms is quite large, especially with Heuristic 2. Stork (2001) also tests his exact algorithm (branch-and-bound) on the same instance, but only for the uniform distribution. He obtains an expected makespan of (rounded) 434 when the branch-and-bound is truncated.

6. Correlations and trade-offs

In this section we investigate how the different statistics (expected makespan, probability of meeting a due date, ...) behave relatively to one another. To this aim we examine 1500 solutions per project instance, generated by the GA from Ballestín (2007). We do not use the GRASP-algorithm that is the subject of this paper in order to guarantee that we do not influence the outcome of the results and because the GA is a good algorithm (for expected-makespan minimization), outperforming other state-of-the-art algorithms.

6.1 Expected makespan versus service level

The first relationship to be examined is that between service level and expected makespan, the first being a probability, the second a measure of schedule length. It is tempting to simply investigate the correlation between $E[s_n]$ and $Pr[s_n \leq \delta]$, for given values of δ. This approach has some disadvantages, however. First of all, it is difficult to choose appropriate due dates δ

probability	U1	U2	Exp	B1	B2
50%	99.91%	99.60%	97.66%	99.89%	99.45%
	0.11	0.37	1.67	0.12	0.45
75%	99.84%	99.35%	96.94%	99.83%	99.22%
	0.18	0.60	1.96	0.19	0.58
90%	99.62%	98.41%	93.81%	99.61%	98.21%
	0.42	1.74	4.38	0.35	1.33
99%	98.21%	94.07%	73.91%	97.89%	92.62%
	1.96	5.11	13.72	1.92	5.00

Table 5: Coefficients of determination for the relationship between due date and expected makespan, for four service levels (in the first column): average (first value) and standard deviation (second value) over the dataset. The columns correspond with the five duration distributions.

for an entire dataset: it may be more appropriate to have (a) different value(s) per instance. Second, since the service level is expressed as a percentage, it is not the most convenient quantity to compute correlations with, because quite a number of solutions may have 0% or 100% service level. Therefore we calculate for each instance the due date δ associated with given service levels and investigate the correlation between δ and $E[s_n]$. Table 5 contains the coefficient of determination, which characterizes the linear relation between the two quantities (note that this coefficient is the square of the correlation coefficient).

We observe that the more variance the distribution has, the less correlation is present: the distribution with smallest correlation is clearly Exp, followed by U2 and B2. The correlation is smallest in the tail and larger in the 'middle' of the domain. We conclude that, for most practical purposes, it is *not* necessary to work with service levels: optimal (or good) expected-makespan solutions *automatically* perform very well on the service-level objective. Only perhaps in some extreme cases will it be interesting to optimize the service level instead of the expected makespan, namely for very high levels ($\geq$ 99%) and large duration variability. One should also take into account that the lesser correlations in these cases may be due in part to the difficulty in estimating the service levels (simulation of rare events) (cfr. Section 5, especially Figure 2).

As a result of the foregoing conclusion, we will also not spend attention anymore to service-level bounds as a representation of risk averseness: assuming near-perfect correlation, either the (unconstrained) solution with minimum expected makespan respects the service-level bound, or no solution exists that offers a service level above the threshold. One additional remark is in order here: we noted in Section 3 that for comparable precision, one

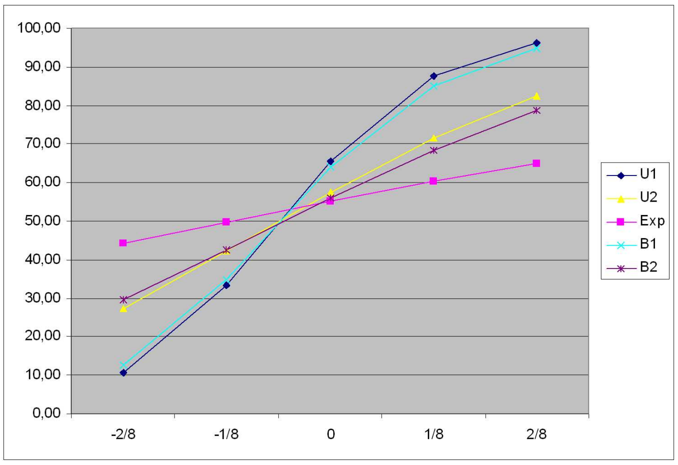


Figure 4: The trade-off of due date (abscissa) versus service level (ordinate). Five due dates are considered, namely $\mathsf{min} - (2/8)(\mathsf{max} - \mathsf{min})$, $\mathsf{min} - (1/8)(\mathsf{max} - \mathsf{min})$, min, $\mathsf{min} + (1/8)(\mathsf{max} - \mathsf{min})$ and $\mathsf{min} + (2/8)(\mathsf{max} - \mathsf{min})$, where 'min' and 'max' are the lowest and highest makespan realization of a good GA-solution (for a high number of replications).

needs a considerably higher number of replications for the service level than for the expected makespan, so we can anticipate that, ceteris paribus, the same number of replications will tend to favor the selection of better solutions in the case of the makespan objective.

We are now ready investigate the trade-off of due date versus service level. The results are displayed in Figure 4; in line with the previous paragraphs, optimal (or at least high-quality) service levels are set via expected-makespan optimization. We observe close similarities between the distributions with similar variance (U1-B1 and U2-B2). We can also see how the graphs 'flatten out' as the variability increases. Specifically, the service level changes drastically when the deadline changes for U1 and B1. For these low-variability distributions a clear 'S'-shape is discerned, which shows that both for very high and very low service levels, the necessary improvement in average makespan to obtain a given service-level improvement is higher than in the 'bulk' of the makespan spread; we presume that this is so because less solutions correspond with very high and low makespans. The trend is almost linear for U2, B2 and Exp, with shallowest slope for Exp. We point out that these figures are averages over 600 instances, so the behavior may be different for individual instances.

6.2 Expected makespan versus variance

We first include a discussion on delivery dates (Section 6.2.1) and then present computational results (Section 6.2.2).

6.2.1 Delivery dates

For the benefit of risk averseness, one of the options envisaged is to impose lower bounds on the makespan variance. In principle we can simply eliminate a solution if it does not respect the constraint. A problem that may occur is that, since our search procedure looks for objective-function improvements, it may generate *only* non-permissible solutions. We therefore proceed as follows: the *altered* makespan s_n^{alt} corresponding with an activity list L is obtained as

$$s_n^{alt}(\mathbf{D}; \Pi(L)) = \max\{s_n(\mathbf{D}; \Pi(L)); \Delta\} \tag{1}$$

where Δ is an artificial *delivery date* for the schedule; higher Δ leads to higher expected makespan but lower variance. For a given upper bound on the variance we find the lowest value of Δ such that the bound is respected (via binary search). Since we evaluate the performance measures by means of sampling, the computations corresponding with Equation (1) are straightforward (the max-operator is applied to known numbers for each sample). As an example, for four makespan realizations $s_n = 100$, 101, 103 and 104, Table 6 contains the quantities $s_n^{alt}(\mathbf{d}, \Pi(L))$. One activity list leads to multiple pairs $(E[s_n^{alt}], \text{var}[s_n^{alt}])$, dependent on Δ. For a given threshold (upper bound) on the variance, however, only one of those pairs comes out best, namely the pair with lowest $E[s_n^{alt}]$ such that $\text{var}[s_n^{alt}]$ does not exceed the threshold.

6.2.2 Computational results for the relationship variance/expected makespan

In this section we investigate the relationship between the expected makespan and the makespan variance. For the same dataset as in Section 6.1, we obtain results quite different than before: average coefficients of determination are between 62% and 69% (see Table 7). Interestingly, the highest values occur for the exponential, while these were lowest for the service level (see Table 5). This latter phenomenon might be due in part to the fact

	$\Delta =$	99	100	101	106
$s_n = 100$		100	100	101	106
101		101	101	101	106
103		103	103	103	106
104		104	104	104	106

Table 6: Altered makespan s_n^{alt} corresponding with four different values for the delivery date Δ. Each column contains one sample of altered makespans.

U1	U2	Exp	B1	B2
62.12%	63.43%	68.67%	64.19%	65.50%
17.36	17.17	14.08	15.94	16.60

Table 7: Coefficients of determination for the relationship between standard deviation and expected makespan: average (first value) and standard deviation (second value) over the dataset. The columns correspond with the five duration distributions.

that the expectation of an exponential variable is equal to its standard deviation, so that for a given longest path of activities in the schedule, higher expectation will also imply higher variance (although the same is true, but to a lesser extent, for the other distributions; in the context of resource-constrained scheduling, one should also generally avoid to overly focus the attention on paths).

From the foregoing we conclude that these correlations are not satisfactorily high for us to neglect the variance, and we anticipate that an actual expectation/variance trade-off exists. This trade-off is examined in Figure 5, where the expected makespan obtained by the GRASP-algorithm is plotted as a function of an upper bound imposed on the variance (actually on the standard deviation). Seven such bounds are enforced. Let 'min' represent the minimum standard deviation over all solutions that were examined by the GA in a search for the minimum expected makespan (unconstrained). The bounds are determined as $k \times \mathsf{min}$, with $k = 0, \frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}$ and 1. The graphs are very similar in shape for all five distributions, but there are differences, mainly in the 'jumps' in the expected makespan corresponding with a step from one k-value to the next. For example, the jump in U1 is five units of expected makespan from $k = \frac{1}{4}$ to $k = 1$, while it is 45 for Exp. Consequently, the

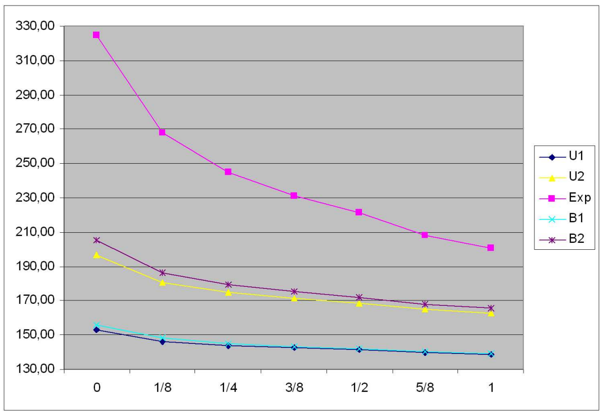


Figure 5: The trade-off of variance (abscissa) versus expected makespan (ordinate).

decision maker needs to 'sacrifice' a considerable increase in makespan expectation if he/she wants to restrict the makespan variance in the cases where the variances of the individual activity durations are high (especially Exp, also U2 and B2). However, this loss is not very important when the variances of the D_i are low (U1 and B1), unless the restriction is severe ($k = 0, \frac{1}{8}, \frac{1}{4}$). This observation was to be expected intuitively; in this section we have been able to show the existence of this phenomenon numerically. Also, and contrary to the due-date/service-level trade-off, the experimental trade-off curves for this case are convex.

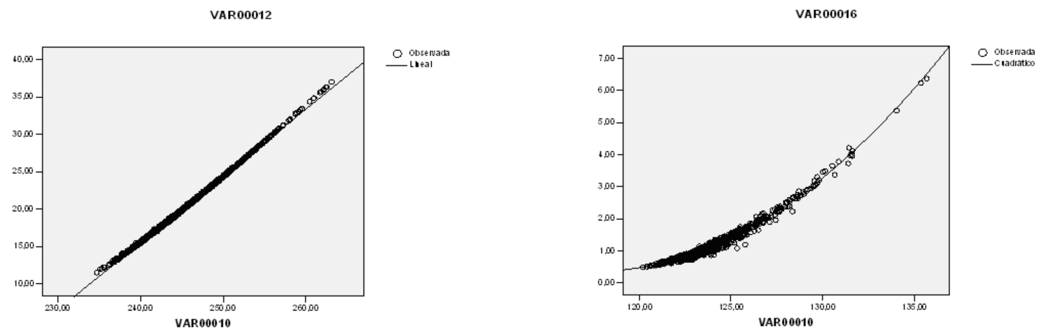
6.3 Expected makespan versus expected overrun

We investigate the minimization of the expected overrun $E[\max\{0; s_n - \delta\}]$ once the manager has fixed a deadline δ. We are obviously mainly interested in δ-values such that

$$\min_{\mathbf{d}}\{s_n(\mathbf{d};\Pi)\} < \delta < \max_{\mathbf{d}}\{s_n(\mathbf{d};\Pi)\}, \tag{2}$$

where optimization in the first and third term is performed over all possible duration-realization vectors $\mathbf{d}$ in the sample we work with, and Π is any job-based policy (otherwise, either we have an optimal objective of zero, or we simply minimize expected makespan). In order to examine these cases, we compute 'minmin' as the minimum of the minimization term in Equation (2) taken over all policies Π examined by the GA, and similarly 'minmax' as the minimum of the maximization term in (2). We wish to investigate especially $\delta \in]\mathsf{minmin}; \mathsf{minmax}[$; other values often turn out to admit policies with *all* makespan realizations either higher or lower than the deadline. More specifically, we work with three values for δ, namely $\delta_1 = \mathsf{minmin} + (\mathsf{minmax} - \mathsf{minmin})/2$, $\delta_2 = \mathsf{minmin} + 5(\mathsf{minmax} - \mathsf{minmin})/8$ and $\delta_3 = \mathsf{minmin} + 3(\mathsf{minmax} - \mathsf{minmin})/4$.

We find that, dependent on the value of the deadline, two types of relation between the expected makespan and the expected overrun can be encountered, either linear or quadratic; this is illustrated in Figure 6. A high determination coefficient in a linear or in a quadratic relationship between $E[s_n]$ and $E[\max\{0; s_n - \delta\}]$ would imply that for all practical purposes, we can suffice with optimization of the first quantity in order to optimize also the second. It turns out that on fitting a quadratic equation to the data (which obviously includes a linear relationship), the determination coefficients are above 95% in all cases, and the majority even above 99%; for the sake of brevity, we omit the table displaying all coefficients.



(a) A linear example. (b) A non-linear example.

Figure 6: Expected makespan (abscissa) versus expected overrun (ordinate).

7. The distribution of the makespan realizations of a given policy

In a deterministic setting, the decision maker knows exactly when the project will finish and can make decisions based on this information. In a stochastic environment, he/she disposes only of the expected makespan, which is a very limited piece of information knowing that many different makespan realizations can actually occur. Clearly, knowledge of the entire distribution of possible makespan realizations $G(t; \Pi(L)) : \mathbb{R} \rightarrow [0; 1] : G(t; \Pi(L)) = Pr[s_n(\mathbf{D}; \Pi(L)) \leq t]$ is much more informative. Our goal in this section is to better understand the shape of this distribution. We provide two paragraphs, one with the detailed computations (Section 7.1) and one with some conclusions (Section 7.2).

7.1 A detailed study of the distribution of makespan realization of a given policy

Our first step is to calculate two descriptive measures for the shape and symmetry of the makespan distribution of a policy, its skewness and its kurtosis. Approximations for these values are collected in Table 8. The first (second) line shows the average of the (absolute) skewness of the different instances. The third line represents the fraction of the instances with positive skewness. The remaining lines display the same results for the kurtosis.

The data coming from both uniforms are very symmetric, and even the number of instances with positive and negative skewness is around 50%. The absolute value of the kurtosis is small, although there are more instances in which the kurtosis is negative. At any rate, if we pay attention to the absolute values, the data could come from a normal distribution

Distribution	U1	U2	Exp	B1	B2
skewness	0.006	-0.009	0.492	0.143	0.014
|skewness|	0.067	0.069	0.492	0.148	0.074
% inst.>0	51.33%	43.67%	100.00%	92.00%	52.67%
kurtosis	-0.030	-0.037	0.440	0.023	-0.051
|kurtosis|	0.128	0.122	0.450	0.130	0.133
% inst.>0	36.50%	35.67%	92.67%	49.33%	33.00%

Table 8: Skewness and kurtosis for the different distributions.

(which has zero skewness and kurtosis). Interestingly, the increase in the variability from U1 to U2 hardly affects the measures. The situation is slightly different for the beta distributions. The figures of B2 are very similar to those of U1 and U2, but for B1 we observe data that are less symmetric (long right-sided tail), yet more unbiased in the case of positive and negative skewness. We can still consider the normal distribution as a possible model for these data. Finally, data coming from the exponential have a long right-sided tail (positive skewness) and a prominent peak (positive kurtosis). The normal should not be able to capture these data.

The next step in our attempt to characterize the distribution of the makespan realizations is to find a known distribution that adequately fits the obtained data, for which many choices are possible. Based on the previous measures and histograms of the data (e.g. Figure 7 for instance j1201_1 if the durations stem from U1), we have tried to fit a normal distribution $\mathcal{N}(\mu, \sigma^2)$ to the data, with μ the average of the makespan realizations and σ^2 the (sample)

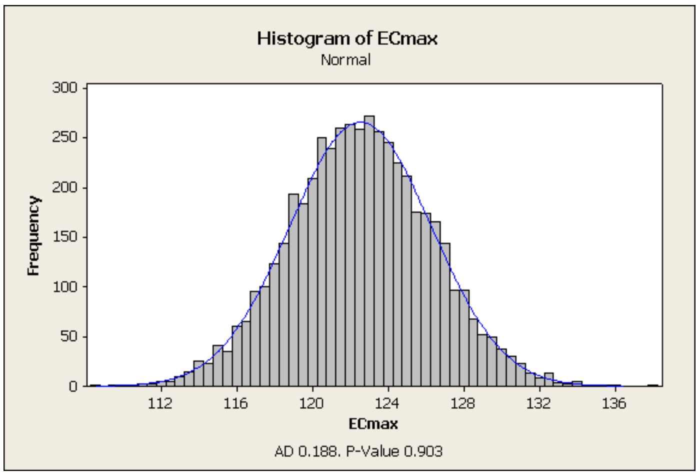


Figure 7: Histogram of the different makespan realizations of a solution for j1201_1.

Distribution	U1	U2	Exp	B1	B2
% A-D < 5% crit. val.	92.67%	92.67%	1.00%	70.00%	90.67%
Average A-D stat.	0.404	0.413	3.032	0.648	0.433
Maximum A-D stat.	1.41	1.82	7.868	3.065	2.409
% inst. A-D > 1	2.33%	2.00%	96.00%	15.00%	3.33%

Table 9: A-D values and hypothesis tests for the different distributions.

variance. One method to determine whether a certain distribution appropriately fits the data is a hypothesis contrast or test:

$$H_0 : G = \mathcal{N}(\mu, \sigma^2) \qquad \text{(null hypothesis)}$$

$$H_1 : G \neq N(\mu, \sigma^2) \qquad \text{(alternative hypothesis)}$$

The way to contrast the hypothesis is by calculating a statistic of a sample of G. We will use the Anderson-Darling (A-D) statistic A^2 (D'Agostino, 1986; Linnet, 1988):

$$A^2 = -N - \frac{1}{N}\sum_{i=1}^{N}(2i-1)(\ln(G(y_i) - \ln(G(y_{N+1-i})))),$$

where $\{y_i\}_{i=1}^{N}$ are N (not necessarily different) makespan realizations obtained from N replications. This test can be applied for any distribution, although critical values of the A-D statistic under the null hypothesis have only been tabulated for a limited number of distributions. Table 9 shows for which fraction of instances (out of the 600) the A-D value is smaller than the 5% critical value, together with the average and the maximum of the statistic, and the percentage of instances in which it is larger than 1. According to the table, the normal distribution can be used to model the distribution of the makespan realizations of the solution given by the GRASP, at least in most instances of U1, U2 and B2, and also in many instances (but quite fewer than in B2) of B1. We should add that the average of the A-D statistic in U1, U2 and B2 corresponds to a significance level larger than 0.2 and that many instances lead to a A-D statistic associated with levels over 0.5 (some are 0.8 and 0.9). The table also states that the normal is of no use at all in the case of Exp.

When data do not pass a normality test, it is common practice to transform them and apply the test to the transformed data. We have applied the Box-Cox transformation, given by

$$x = f_\lambda(y) = \begin{cases} (y^\lambda - 1)/\lambda, & \lambda \neq 0; \\ \ln y, & \lambda = 0. \end{cases}$$

Distribution	U1	U2	Exp	B1	B2
% A-D $<$ 5% crit. val.	94.00%	94.67%	95.67%	95.00%	94.50%
Average A-D stat.	0.324	0.327	0.316	0.322	0.327
Maximum A-D stat.	1.105	1.106	0.956	1.782	1.304
% inst. A-D $>$ 1	0.17%	0.33%	0.00%	0.50%	0.50%

Table 10: A-D values and hypothesis tests for the different distributions, for the transformed data.

The parameter λ is found via maximum likelihood. Table 10 contains the same information as Table 9 but with respect to the transformed data (note that the critical values are lower than in the original test, which is not always taken into account in statistical packages). We see from the table that via this transformation we are able to adequately fit most of the instances for all five the duration distributions: all averages of the A-D statistic are very similar and correspond to significance levels larger than 0.2. Obviously, the optimal λ associated to many instances in the uniforms and betas correspond to values around 1 ($\lambda = 1$ corresponds with no transformation).

Our intention is not to perform an exhaustive search of the one distribution that best fits the data obtained by the solution of the GRASP, because the result may change if we alter the parameters of the GRASP (especially the number of replications or the limit on the number of schedules). Our goal has been to show that it is generally possible to find known distributions that fit reasonably well the makespan realizations of a given solution (a given policy).

7.2 Conclusions

Knowledge of an approximate distribution of the makespan realizations of an implemented policy provides the decision maker with the possibility to compute values such as $Pr[a \leq s_n \leq b]$ for any a, b, and therefore constitutes a valuable piece of managerial information. In a study of the stochastic flow shop, Dodin (1996) uses Monte-Carlo sampling to determine the makespan distribution of a given sequence. He argues that, for different duration distributions, the makespan of a given sequence becomes approximately normal when a large number of jobs are involved (and positively skewed in case of large variance), but only based on a visual observation of the shape of the sample distributions, so without hypothesis testing.

A number of cautionary remarks are in order in interpreting our results. Firstly, we have performed many tests and therefore we can expect to find contrasts where the test fails even when H_0 is true: if we produce a large number of random samples of the same size from the same $\mathcal{N}(\mu, \sigma^2)$, around 5% of them would fail the A-D contrast with a critical value of 5%. Secondly, if a sample passes a test, this does not mean that H_0 is correct, only that the available information is not useful to reject it. However, in our cases we have other information that supports our choice of distribution, like the histograms and the small averages of the A-D statistic. In conclusion, we cannot with certainty state that the data stem from (transformed) normal distributions, but these latter do seem to provide good approximations.

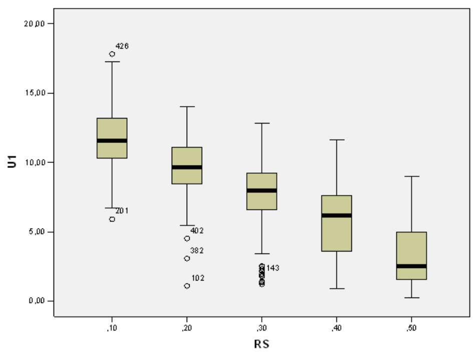
Under the foregoing caveat, and with a reasonable amount of certainty, we can say that the solutions obtained by the GRASP in the Exp-case do not follow a normal distribution and that their distributions are therefore quite different from those in the case of e.g. U1, which can be mostly modeled by a normal distribution.

8. Phase transitions

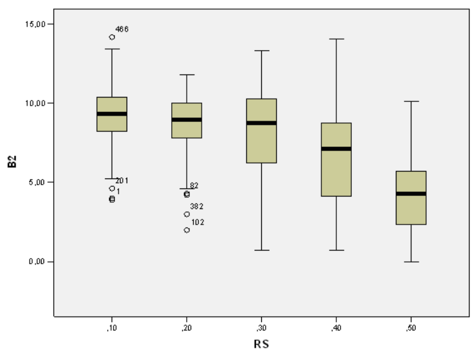
In an article published in 1999, Herroelen and De Reyck, following recent work in artificial intelligence (Hayes, 1997; Huberman and Hogg, 1987), study so called *phase transitions*, where they observe the varying difficulty of project scheduling as a function of a number of problem parameters. More specifically, the difficulty of problems is expressed in terms of the running time needed by an exact algorithm, and they obtain 'bell-shaped' curves with an 'easy-hard-easy' pattern for some of the problem parameters, and a steady hard-easy pattern for other parameters. In a similar way as Herroelen and De Reyck, we will investigate in this section whether an instance is easy or hard and try to explain this difficulty in terms of some parameters.

In determining the difficulty of an instance, however, we cannot just use the running time of our algorithm to measure whether an instance is easy or hard, because our heuristic spends more or less the same time for each instance, since it uses a limit on the number of schedules created during the computations. We therefore look for other ways to evaluate the difficulty of an instance. Our measure will be the percentage improvement from the average objective-function value of the initial population to the best (GRASP-)solution, with which we effectively obtain a measure for the spread of the objective-function values.

The first explanatory variable that we investigate is the *resource strength RS* (Cooper, 1976), which is a measure for the distribution of the resource requirements among the activ-



(a) Distribution U1.



(b) Distribution B2.

Figure 8: Problem difficulty as a function of *RS*.

ities; our results are illustrated in the box plots in Figure 8. For U1, we obtain a hard-easy transition, which is in concordance to what has been observed in the RCPSP; a similar graph results for B1. When we work with the rest of the duration distributions, however, which have a larger variance (U2, B2 and Exp), the pattern is somewhat different: the hard-easy transition only occurs after a 'level' hard interval.

Secondly, we turn our attention to *order strength OS* (Mastor, 1970), which is a measure for the density of the precedence network. In the deterministic RCPSP, lower *OS* corresponds with higher computational effort, which is rather intuitive since more sequencing decisions remain to be made. The difficulty of the instances is depicted as a function of *OS* for all the problem instances in Figure 9, for distribution U1 (the plots are very similar for the other distributions). To our surprise, no clear relationship can be distinguished. We conjecture that this is so because in the SRCPSP the prime source of difficulty resides in the stochasticity

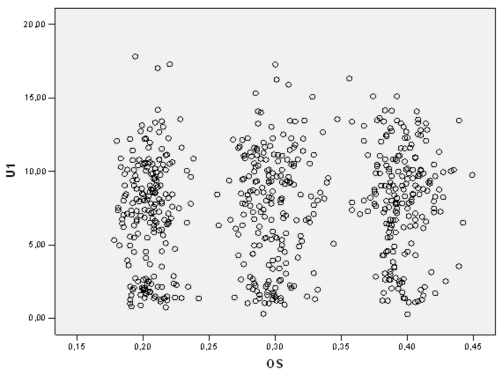


Figure 9: Problem difficulty as a function of *OS* for U1.

of the activity durations rather than in the ex-ante sequencing part of the computations. One might argue that the spread in the objective-function values (our measure of difficulty) need not be 100% proportional to the difficulty of the problem; we have therefore tried other measures of difficulty, including an attempt to quantify the curvature of the plot of the objective-function value as a function of the iteration count, but this has not led to different results. Finally, other explanatory variables such as *resource factor RF*, *network complexity NC* (Pascoe, 1966; Kolisch and Sprecher, 1996) and variability of the activities' durations have also been examined but did not yield interesting outcomes either.

9. Summary

This article has investigated the incorporation of explicit recognition of variability into project planning by developing activity-based scheduling policies for the stochastic RCPSP. We have examined multiple possible objective functions for project scheduling with stochastic activity durations, and we have shown by means of computational experiments that these different objective functions are closely connected and that for most practical purposes, it suffices to focus on the minimization of the expected makespan. We have proposed a GRASP-heuristic that produces high-quality solutions, outperforming the currently available procedures. The variance-reduction technique of descriptive sampling is applied and its benefits assessed. Finally, we have also studied the distribution of the makespan realizations for a given scheduling policy, and we have explored problem difficulty as a function of problem parameters, this latter topic under the header of 'phase transitions'.

Acknowledgments

This work was supported by research project G.0578.07 of the Research Foundation – Flanders (FWO) (Belgium).

References

Adlakha, V.G. and V.G. Kulkarni. 1989. A classified bibliography of research on stochastic PERT networks: 1966-1987. *INFOR* **27**(3), 272-296.

Aiex, R.M., M.G.C. Resende and C.C. Ribeiro. 2002. Probability distribution of solution time in GRASP: an experimental investigation. *Journal of Heuristics* **8**(3), 343–373.

Ang, A., J. Chen and Y. Xing. 2006. Downside risk. *The Review of Financial Studies* **19**, 1191–1239.

Ballestín, F. 2007. When it is worthwile to work with the stochastic RCPSP? *Journal of Scheduling* **10**(3), 153–166.

Bayiz, M. and C.J. Corbett. 2005. Coordination and incentive contracts in project management under asymmetric information. *Working Paper CC31, Anderson Graduate School of Management, University of California, Los Angeles.*

Blazewicz, J., J. Lenstra and A. Rinnooy-Kan. 1983. Scheduling subject to resource constraints – classification and complexity. *Discrete Applied Mathematics* **5**, 11-24.

Cho, J.G. and B.J. Yum. 1997. An uncertainty importance measure of activities in PERT networks. *International Journal of Production Research* **35**, 2737–2757.

Cooper, D.F. 1976. Heuristics for scheduling resource-constrained projects: an experimental investigation. *Management Science* **22**(11), 1186–1194.

D'Agostino, R.B. 1986. Tests for the normal distribution. In: R.B. D'Agostino and M.A. Stephens (eds.). *Goodness-of-Fit Techniques*, pp. 367–419. Marcel Dekker.

Demeulemeester, E. and W. Herroelen. 2002. *Project Scheduling - A Research Handbook.* Kluwer Academic Publishers, Boston.

Dodin, B. 1996. Determining the optimal sequences and the distributional properties of their completion times in stochastic flow shops. *Computers & Operations Research* **23**(9), 829–843.

Elmaghraby, S.E. 1977. *Activity networks: project planning and control by network models.* Wiley.

Elmaghraby, S.E., Y. Fathi and M.R. Taner. 1999. On the sensitivity of project variability to activity mean duration. *International Journal of Production Economics* **62**, 219–232.

Feo, T.A. and M.G.C. Resende. 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization* **6**, 109–133.

Feo, T.A., M.G.C. Resende and S. Smith. 1994. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research* **42**, 860–878.

Festa, P. and M.G.C. Resende. 2000. GRASP: an annotated bibliography. *Technical Report, AT&T Labs Research, Florham Park, NJ 07733.*

French, S. 1988. *Decision Theory. An Introduction to the Mathematics of Rationality.* Ellis Horwood Limited.

Gerchak, Y. 2000. On the allocation of uncertainty-reduction effort to minimize total variability. *IIE Transactions* **32**, 403–407.

Golenko-Ginzburg, D. and A. Gonik. 1997. Stochastic network project scheduling with non-consumable limited resources. *International Journal of Production Economics* **48**, 29–37.

Graham, R.L. 1966. Bounds on multiprocessing timing anomalies. *Bell System Technical Journal* **45**, 1563–1581.

Gutierrez, G.J. and A. Paul. 2001. Robustness to variability in project networks. *IIE Transactions* **33**, 649–660.

Gutjahr, W.J., C. Straus and E. Wagner. 2000. A stochastic branch-and-bound approach to activity crashing in project management. *INFORMS Journal on Computing* **12**(2), 125-135.

Hagstrom, J.N. 1988. Computational complexity of PERT problems. *Networks* **18**, 139–147.

Hart, J.P. and A.W. Shogan. 1987. Semi-greedy heuristics: an empirical study. *Operations Research Letters* **6**, 107–114.

Hartmann, S. and R. Kolisch. 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research* **127**, 394–407.

Hayes, B. 1997. Can't get no satisfaction. *American Scientist* **85**(2), 108–112.

Herroelen, W. and B. De Reyck. 1999. Phase transitions in project scheduling. *Journal of the Operational Research Society* **50**, 148–156.

Huberman, B.A. and T. Hogg. 1987. Phase transitions in artificial intelligence systems. *Artificial Intelligence* **33**(2), 155–171.

Igelmund, G. and F.J. Radermacher. 1983. Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks* **13**, 1–28.

Jorion, P. 2000. *Value at Risk: The Benchmark for Controlling Market Risk.* McGraw-Hill.

Kerzner, H. 1998. *Project Management.* Wiley.

Kleywegt, A.J., A. Shapiro and T. Homem-De-Mello. 2001. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* **12**(2), 479–502.

Kolisch, R. and S. Hartmann. 1999. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In: J. Weglarz (ed.). *Project Scheduling – Recent Models, Algorithms and Applications*, pp. 147–178. Kluwer Academic Publishers, Boston.

Kolisch, R. and R. Padman. 2001. An integrated survey of deterministic project scheduling. *Omega* **29**(3), 249–272.

Kolisch, R. and A. Sprecher. 1996. PSPLIB - a project scheduling problem library. *European Journal of Operational Research* **96**, 205–216.

Kouvelis, P. and G. Yu. 1997. *Robust Discrete Optimization and Its Applications.* Kluwer Academic Publishers.

Leus, R. and W. Herroelen. 2004. Stability and resource allocation in project planning. *IIE Transactions* **36**,667–682.

Linnet, K. 1988. Testing normality of transformed data. *Applied Statistics* **37**(2), 180–186.

Li, K.Y. and R.J. Willis. 1992. An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research* **56**, 370–379.

Ludwig, A., R.H. Möhring and F. Stork. 2001. A computational study on bounding the makespan distribution in stochastic project networks. *Annals of Operations Research* **102**, 49–64.

Mastor, A.A. 1970. An experimental investigation and comparative evaluation of production line balancing techniques. *Management Science* **16**, 728–745.

Möhring, R.H. and F.J. Radermacher. 1989. The order-theoretic approach to scheduling: the stochastic case. In R. Slowinski and J. Weglarz (eds.). *Advances in Project Scheduling*, Chapter III.4. Elsevier.

Neumann, K., C. Schwindt and J. Zimmermann. 2002. *Project Scheduling with Time Windows and Scarce Resources.* Springer.

Newbold, R.C. 1998. *Project Management in the Fast Lane.* The St. Lucie Press/APICS Series on Constraints Management.

Özdamar, L. and G. Ulusoy. 1996. A note on an iterative forward/backward scheduling technique with reference to a procedure by Li and Willis. *European Journal of Operational Research* **89**, 400–407.

Pascoe, T.L. 1966. Allocation of resources – CPM. *Revue Française de Recherche Opérationnelle* **38**, 31–38.

Patterson, J.H. 1984. A comparison of exact approaches for solving the multiple constrained resource project scheduling problem. *Management Science* **30**, 854–867.

Portougal, V. and D. Trietsch. 1998. Makespan-related criteria for comparing schedules in stochastic environments. *Journal of the Operational Research Society* **49**, 1188–1195.

Saliby, E. 1990. Descriptive sampling: a better approach to Monte Carlo simulation. *Journal of the Operational Research Society* **41**(12), 1133–1142.

Saliby, E. 1997. Descriptive sampling: an improvement over Latin hypercube sampling. In: S. Andradóttir, K.J. Healy, D.H. Withers and B.L. Nelson (eds.). *Proceedings of the 1997 Winter Simulation Conference*, pp. 230–233.

Schuyler, J. 2001. *Risk and Decision Analysis in Projects.* Project Management Institute.

Silver, E.A., D.F. Pyke and R. Peterson. 1998. *Inventory Management and Production Planning and Scheduling.* John Wiley & Sons.

Smith, P.G. and D.G. Reinertsen. 1991. *Developing Projects in Half the Time.* Van Nostrand Reinhold.

Stork, F. 2001. Stochastic resource-constrained project scheduling. *Ph.D. thesis, Technische Universität Berlin.*

Tsai, Y.-W. and D.D. Gemmill. 1998. Using tabu search to schedule activities of stochastic resource-constrained projects. *European Journal of Operational Research* **111**, 129–141.

Valls, V., F. Ballestín and S. Quintanilla. 2005. Justification and RCPSP: a technique that pays. *European Journal of Operational Research* **165**(2), 375–386.

Wollmer, R.D. 1985. Critical path planning under uncertainty. *Mathematical Programming Study* **25**, 164-171.

Xie, G., J. Zhang and K.K. Lai. 2006. Risk avoidance in bidding for software projects based on life cycle management theory. *International Journal of Project Management* **24**, 516–521.