Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

STOCHASTIC ROLLOUT AND JUSTIFICATION TO SOLVE THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

Ningxiong Xu
Linda Nozick

Civil and Envir. Engr.
Hollister Hall, Cornell University
Ithaca, N.Y. 14853, U.S.A.

Orr Bernstein
Dean Jones

Sandia National Laboratories
PO Box 5800 MS 1138
Alburquerque, NM 87185-1138 U.S.A.

ABSTRACT

The key question addressed by the resource-constrained project scheduling problem (RCPSP) is to determine the start times for each activity such that precedence and resource constraints are satisfied while achieving some objective. Priority rule-based heuristics are widely used for large problems and more recently justification has been shown to be an important extension. Xu et al. further augments priority rule heuristics by creating rollout procedures and proves their effectiveness. However, that procedure generates just one schedule. We extend that method using sampling to generate a set of schedules using probabilistic techniques and select the best schedule from this sample. Using the 600 problem instances in PSLIB, we present empirical evidence that this procedure produces solutions that are better than the rollout procedure alone but at a computational cost.

1 INTRODUCTION

The key question addressed in the resource-constrained project scheduling problem (RCPSP) is to determine the start times for each activity such that precedence and resource constraints are satisfied while achieving some objective like shortest project duration or minimum resource investment. In the literature the RCPSP is commonly formulated as an integer programming problem for which the integer variables indicate the period in which an activity starts (or ends). This model has become an important management tool for many business activities. For example, production planning in make-to-order operations frequently requires the assignment of resources to small production lots, each of which has specific machining and labor needs. In these environments, the production planning problem bears considerable resemblance to the RCPSP, such as is common in construction operations or other activity-oriented situations.

Many authors have developed exact solution procedures for this problem formulation (or minor variations on it), (Brucker et al. (1999), Demeulemeester and Herroelen (1997), and Patterson (1984)) but most of those authors have also pointed out that it is impractical to solve this integer programming problem exactly for even moderately sized instances. Computationally, the RCPSP is known to be *NP*-hard (Blazewicz et al. 1983). For problems of the size generally experienced in practice it is necessary to resort to heuristics.

General reviews of efforts to address the RCPSP can be found in Morton and Pentico (1993), Özdamar and Ulusoy (1995), Herroelen et al. (1998), Węglarz (1999), Brucker, et al. (1999), Hartmann (1999), and Kolisch and Hartmann (2006). Many heuristic procedures have been developed to tackle realistic instances of this problem. Heuristics that have shown promise include the disjunctive arc method (Bell and Han (1991)), local constraints-based analysis (Ulusoy and Özdamar (1994), Kohlmorgen et al. (1999)), simulated annealing (Bouleimen and Lecocq (2003)), and other meta-heuristics (Lova et al. (2000), Alcaraz and Maroto (2001), Fleszar and Hindi (2004), Debels et al. (2006), and Valls et al. (2007, 2005, 2004, 2003)). Weiss (1988) and Deckro, et al. (1991) develop decomposition-based solution procedures. Weiss (1988) uses a Danzig-Wolfe decomposition to solve the LP relaxed version of a resource constrained multi-project scheduling problem. Deckro et al. (1991) and Möhring et al. (2003) develop Lagrangian heuristics. Nozick et al. (2002) develop an alternative formulation of the RCPSP using continuous variables for start times and create a heuristic solution procedure based on Generalized Benders Decomposition.

Heuristics based on priority rules are of utmost importance in solving large problems in the literature (Kolisch (1996)). Priority-based heuristics develop a schedule by adding activities one at a time to that schedule. A priority rule specifies, for a set of activities that are eligible to be scheduled at a particular point in the algorithm, the one to be placed on the schedule next. The priority values for each activity can be based on a number of factors, including activity duration, the difference between early and late start times, the number of successor activities, etc. For a discussion of efficient priority rules see Kolisch (1996).

Priority rules can be embedded in other heuristic methods. Priority rules embedded in genetic algorithms are developed in Leon and Ramamoorthy (1995), Özdamar (1999), and Hartmann (2002, 1998). Priority rules integrated with sampling schemes are discussed in Kolisch (1995) and Kolisch and Drexl (1996). Procedures based on the use of multiple priority rules within a single heuristic procedure are discussed in numerous papers including Boctor (1990) and Li and Willis (1992).

As Valls et al. (2005) show, double justification (for simplicity referred to as justification throughout the remainder of the paper) is a useful technique to improve the solution quality of a scheduling heuristic. A schedule is called a left schedule if no activity can be started earlier without delaying some other activity or violating the constraints. Similarly, a schedule is called as a right schedule if no activity can be finished later without advancing some other activity, violating the constraints, or increasing the project makespan. Given a schedule, the right (left) justification of an activity is to start the activity as late (early) as possible while the start times of the other activities remain unchanged. The right (left) justification of a schedule is achieved by making each activity to the right (left) in the decreasing (increasing) order of the finish (start) times of activities. Justification of a schedule consists of two steps: right justification and then left justification. It is easy to see that the solution quality of a schedule with justification is at least as good as the original one.

It is important to recognize that if an underlying combinatorial optimization problem can be stated as a sequential decision problem, a rollout algorithm can be created to improve the solution quality of a base heuristic, albeit at a computational cost (Bertsekas and Castanon (1999), Bertsekas et al. (1997)). Since priority rule solution procedures for the RCPSP are structured as sequential decision problems, rollout algorithms based on priority rule heuristics are quite natural. Xu et al. (2007) develops several such algorithms and provides estimates of the solution quality and computational burden of these heuristics using the PSLIB problem library (Kolisch and Sprecher 1996). They also compare the solution quality obtained from these procedures to other algorithms found in the literature and finds that these procedures are competitive with the best algorithms found in the literature. This paper extends one of the algorithms developed in Xu et. al (2007) by integrating it with sampling methods.

Section 2 describes the RCPSP and how stochastic rollout can be used to create a solutions to this problem. Section 3 focuses on the solution quality and computational cost of this procedure. Section 4 discusses conclusions and areas of further research.

2 PROBLEM DESCRIPTION AND STOCHSTIC ROLLOUT

The RCPSP can be stated as follows. We consider a single project schedule that consists of $j = 0,\ldots,J+1$ activities with a non-preemptive duration d_j periods. Activities 0 and $J+1$ represent the start and end of the project and thus $d_0 = d_{J+1} = 0$. The activities are partially ordered by precedence relations, where P_j is the set of immediate predecessors of activity j. An activity j is subject to two kinds of constraints: precedence and resource constraints, i.e., activity j cannot start before all its predecessors are finished and requires k_{jr} units of resource type r in each period when it is active. There are K_r units available of resource *r* in each period. The resource-constrained project scheduling problem is to identify the start time for each activity such that the project makespan is minimized, (i.e., the completion time of the project).

A priority rule schedule consists of two parts, a priority rule and a scheduling scheme. The scheduling scheme is used to construct, in a stage-wise fashion, a schedule by building on a partial schedule. The priority rule estimates, among the eligible activities, which activity should be scheduled at the current time point with the goal of obtaining the shortest project duration. Priority rules include criteria like latest finish time, and maximum total successors.

Rollout can be integrated with a priority rule heuristic by doing the following. At each point in the algorithm when a set of activities that are precedence and resource feasible are being considered to determine which one to put on the schedule at the current time point, evaluate the effects of adding each. This is done by placing each activity on the schedule (one at a time) at the current time point and completing the schedule using the priority rule heuristic. This creates an estimate of the makespan if that activity were placed on the schedule now. Justification can be used to refine this estimate. Once all of the activities have been evaluated in this manner the algorithm actually places on the schedule at the current time point the one for which the estimated makespan is the lowest. This creates a rollout procedure based on a priority rule and justification. Notice that these procedures require no parameters that must be tuned.

Xu et al. (2007) develops several additional rollout procedures but shows that rollout using latest finish time with justification is very effective at reasonable computational cost.

There have been a significant number of priority rule heuristics that have been extended to use sampling methods (Hartmann, 1999). This creates procedures for which the selection of the activity to put on the schedule at the current time point is done randomly with a bias based on the priority value instead of simply taking the activity with the best priority value. Thus each time the priority rule heuristic is used a potentially different schedule is created. The schedule with the smallest makespan is then ultimately selected.

We can extend the rollout procedure described above to include sampling by considering all the activities that are precedence- and resource-feasible at the current time point, computing the estimate of the makespan using rollout and justification and then, instead of simply placing the one on the schedule with the smallest makespan, randomly select the activity to place on the schedule biased by the estimates of makespan. Since this selection is probabilistic each time we execute the procedure a potentially different schedule is created.

There are many probability functions that could be used in selecting the activity to place on the schedule given the estimated makespan for each activity. For the purposes of this research we investigate two functions. The first function simply says that we take the one with the smallest estimated makespan with a fixed probability *X* and if that activity is not selected randomly select one of the others with equal probability.

A second probability model can be created by focusing on the fact that at the beginning of the planning horizon our estimates of makespan are likely to be less accurate than those created for activities at the end of the planning horizon. Therefore it may make sense to use a slightly lower probability of selecting the activity with the smallest estimated makespan to place on the schedule at the beginning of the planning horizon and to slowly increase it as the algorithm advances. We explore this by using the formula given below.

$$p_j = a + j(b-a)/J \qquad (1)$$

where a is the probability to use at the beginning of the planning horizon, b is the maximum value to be used at the end of the planning horizon, j is the activity number and J is the number of activities in the project (dummy activities excluded). This assumes that the activities have been numbered in a way that is highly correlated with the order in which they will be executed over the planning horizon. In practice this order can be developed based on the precedence constraints.

The deterministic rollout procedure has no parameters but when sampling is added parameters become necessary. Using the first probability mechanism there are two parameters, *X* and *N,* the probability of selecting the activity with the smallest estimated makespan and the number of times to run the procedure, respectively. If the probability mechanism given in (1) is used then three parameters are needed: *a*, *b* and *N*. The next section will focus on the impact of selecting different values for these parameters. It also contains a comparison of these algorithms to others found in the literature.

3 COMPUTATIONAL RESULTS

This section is devoted to a computation assessment of the stochastic rollout procedures described in Section 2. The algorithms have been coded in C and implemented on a 1.8 GHz Pentium® PC running Linux. In order to evaluate the stochastic rollout procedures, we compare the solution quality and computational time of these procedures to other algorithms found in the literature. Xu et al. (2007) demonstrates that LFT is among the best priority rules to use with rollout and that the parallel scheduling scheme is computationally much faster than the serial scheduling scheme. Hence we focus on latest finish time (LFT) as the priority rule and use a parallel scheduling scheme.

The test problems used are the six hundred projects each with 120 activities contained in PSLIB, a test problem library developed to test scheduling algorithms (Kolisch and Sprecher 1996). For some of the problems in the library the optimal solution is not known: therefore we measure the average percentage deviation from the resource unconstrained critical path (which is a lower bound on the shortest feasible project duration) and the best upper bound found to date (which is an upper bound on the shortest feasible project duration).

Figure 1 gives the average percentage deviation from the critical path lower bound using a constant probability of selection for the activity with the lowest estimated makespan where that probability is 50%, 80%, 90%, 95% or 99% for 1 to 100 iterations (*N*). Figure 2 gives a similar comparison except it is based on the average percentage deviation from the best upper bound. It is useful to notice that better solutions are obtained when the probability of selecting the activity with the smallest estimated makespan is very high. For example the smallest probability considered, 50%, is substantially worse than the other values.

Figures 3 and 4 present the same comparisons but for the probably model given by equation (1) when *a* and *b* are (90%, 99%) and (95%,99%). The results are similar for both and modestly better than using a constant probability model. For example, the average percentage deviation for a constant probability of 95% when *N* is 100 is about 33.8%. When the (95%, 99%) probability model is used that average drops to about 33.7%. Similarly for the average percentage deviation from the best upper bound for a constant probability of 95% when *N* is 100 is about 2.46%. When the (95%, 99%) probability model is used that average drops to about 2.41%.

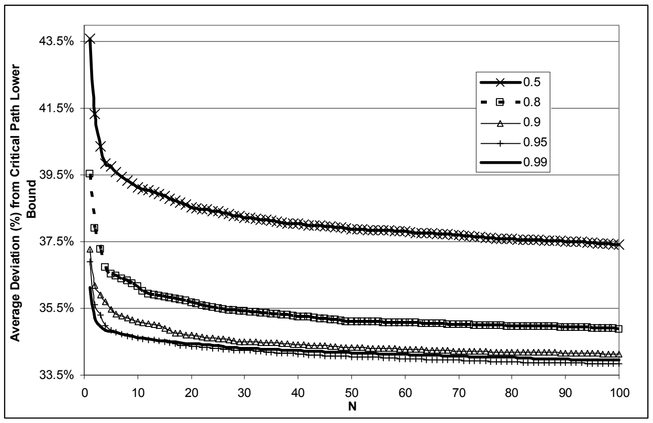

Figure 1: Average (%) deviation from the critical path lower bound using a constant probability of selection

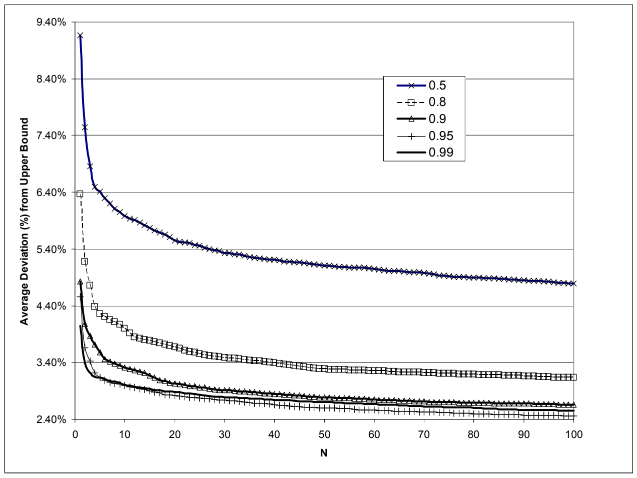

Figure 2: Average (%) deviation from the upper bound for a constant probability of selection

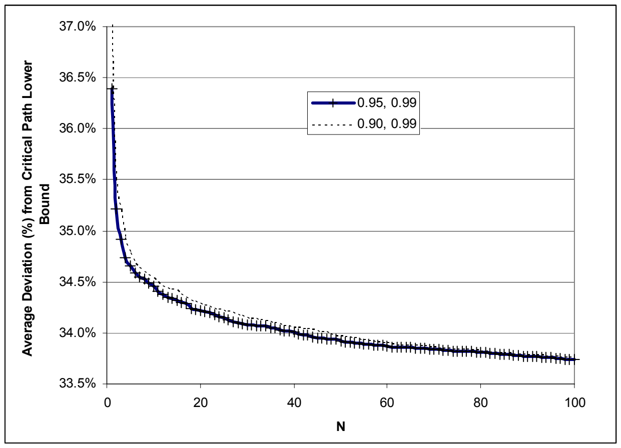

Figure 3: Average (%) deviation from the critical path lower bound using the model given in equation (1)

Tables 1 and 2 present a comparison of the performance of the stochastic rollout algorithms with the best heuristics found in the literature. Table 1 focuses on algorithms which measure computational performance based on the number of schedules generated. The first pair of columns gives solution quality and an estimate of computational time when 1,000 schedules are to be generated. The second pair of columns does the same but when the number of schedules to be generated is between 5,000 and 10,000. Table 2 focuses on algorithms that are more heavily focused on computation time rather than the number of schedules created. In order to develop the comparison, we scale the computation times that were given for the different heuristics on their respective computing platforms to an estimate of computation time those heuristics would require on a 1.8 GHz machine. Scaling by clock speed only will tend to over-estimate the performance gains to be had by moving to a more powerful platform, because as the clock speeds have increased, computer memory systems have not kept up. Since many of these algorithms require large data structures, the speed of the memory will limit the increases in computational speed that are actually achieved (Wulf and McKee 1995). However, these projected run-times are useful in performing a general comparison across the algorithms considered.

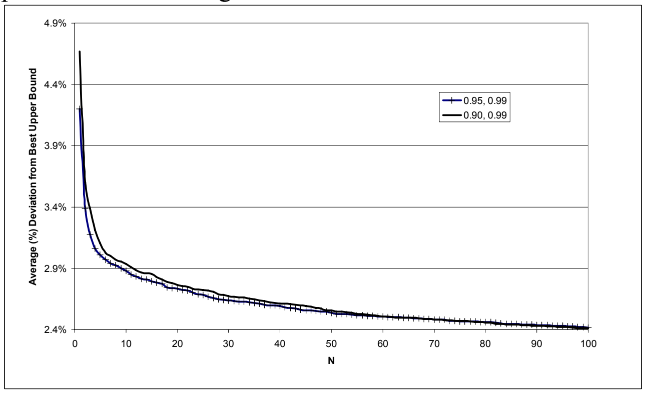

Figure 4: Average (%) deviation from the upper bound using the model given in equation (1)

We have also included a few algorithms for which the computation time is not known. Of course it is much harder to draw conclusions about the relative computational performance of those algorithms. As can be seen from the table, the stochastic rollout procedure is very competitive with the solution procedures in the literature in term of solution quality and time.

Valls et al. (2005) reports that for 5000 schedules generated using Hartmann (1998) requires 1.6 seconds on an 400 MHz PC (0.36 on a 1.8 GHz PC). The most computationally intensive part of this algorithm is the application of the serial generation scheme to the ordering of tasks produced by the crossover and mutation operations hence we can approximate that the right and left justification of each of these orderings requires about $2.2x10^{-4}$ seconds on a 1.8

GHz PC. Of course, this is an over-estimate of the true computation time. The stochastic rollout algorithm with *N=1* requires about 3.3x10^{-4} seconds to justify each schedule (both right and left). This indicates that it may be possible to substantially improve our computation times. However, this could also be an indication that we have significantly overestimated the performance improvement to be gained by moving to a faster processor.

Table 1: Comparison based on number of schedules, computation time and solution quality.

Heuristics	**1,000 Schedules**		**5,000 to 10,000 Schedules**	
	Avg. Dev. (%) from CP Lower Bound	**Time (Sec.)**	**Avg. Dev. (%) from CP Lower Bound**	**Time (Sec.)**
Valls et al. (2005) Justification	35.39	-	33.24	0.36
Valls et al. (2007) Hybrid GA	34.07	-	32.54	0.45
Debels et al. (2006) Scatter Search/Electromagnetism Heuristic	35.22	0.12	33.1	0.65
Debels and Vanhoucke (2005) Electromagnetism Meta-heuristic			33.94	-
Hartmann (2002) Self-Adapting GA Heuristic	37.19	-	35.39	1.04
Hartmann (1994) Activity List GA Heuristic	39.37	-	36.74	1.05
Tormos and Lova (2003) Hybrid Multi-path Heuristic	35.98	0.51	35.30	2.55
Tormos and Lova (2001) Hybrid Multi-path Heuristic	36.32	0.65	35.62	3.31
Merkle et al. (2002) Ant Colony Heuristic			35.43	6.94
Bouleimen and Lecocq (2003) SA Activity List Heuristic	42.81	-	37.68	-

Table 2: Comparison based on computation time and solution quality.

Heuristics	**Average Deviations (%) from Critical Path Lower Bound**	**Time (Seconds)**
Xu et al. (2007) 1-Rollout with LFT with Justification	35.11	0.27
Stochastic Rollout with N=10 and (95%,99%)	**34.5**	**2.7**
Valls et al(2003) Critical Activity Reordering Heuristic	34.53	3.78
Stochastic Rollout with N=20 and (95%,99%)	**34.2**	**5.4**
Mohring et al. (2003) Lagrangian Heuristic	36.00	8.1
Stochastic Rollout with N=30 and (95%,99%)	**34.08**	**8.1**
Valls et al (2004) Population Based Approach	31.58	13.21
Artigues et al. (2003) TS-Network Flow Heuristic	36.16	16.16
Sprecher (2002) Network Decomposition Heuristic	39.29	42.28
Nonobe and Ibaraki (2002) Tabu Search	34.99	107.55
Fleszar and Hindi (2004) Neighborhood Search	33.1	122.14
Palpant et al. (2004) Neighborhood Search	32.41	265.65

Figures 2 and 4 measure performance based on the upper bound for the 120 task problem instances in PSLIB. It is difficult to use this metric to compare our algorithms to those in the literature because over time these bounds improve as researchers find better solutions. Debels et al. (2006) published in 2006 does provide this measure for their algorithm using the upper bounds that were available in 2004. They report a percentage deviation of 3.24% for 1,000 schedules and 1.91% for 5000 schedules. We are using the upper bounds available as of December 2006. For stochastic rollout with (95% ,99%) for *N* equal to 5 and 10 we get about 3% and 2.88%, respectively. Again, it is difficult to compare these results with those given in Debels et al (2006) but the stochastic rollout algorithms are competitive.

4 CONCLUSIONS AND FUTURE RESERACH

This paper has illustrated how to extend rollout algorithms using sampling to solve the RCPSP. The key idea that underlies these algorithms is that at each point in the algorithm when there is a set of activities that are all resource and precedence feasible a computation is done to estimate the makespan when each of the activities is placed on the schedule. The activity to actually place on the schedule is then randomly selected based on these estimates. Since rollout produces excellent solutions it is important that the probability model used often places the activity on the schedule with the smallest estimated makespan but infrequently a different activity is selected. One of the key benefits of these procedures is the very small number of parameters that are needed. This makes these algorithms very easy to use.

Stochastic rollout does provide an increase in solution quality but at a non-trivial increase in computation cost. Mostly this is because the initial rollout solutions are very high quality. Future research to combine rollout (and potentially stochastic rollout) with local search is likely to prove very valuable because it is likely that either the optimal solution or a better sub-optimal solution is in relatively close proximity to the first few rollout solutions generated.

REFERENCES

Alcaraz J. and C. Maroto. 2001.A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research* 102:83-109.

Artigues R., P. Michelon, and S. Reusser. 2003. Insertions techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research* 149:249-267.

Bell C. E. and J. Han. 1991. A new heuristic solution method in resource-constrained project scheduling. *Naval Research Logistics* 38:315-331.

Bertsekas D. P. and D. A. Castanon. 1999. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics 9* 1:89-108.

Bertsekas D. P., J. N. Tsitsiklis, and C. Wu. 1997. Rollout algorithms for combinatorial optimization. *Journal of Heuristics* 3:245-262.

Blazewicz J., J. K. Lenstra, and Kan A.H.G. Rinnooy. 1983. Scheduling Subject to Resource Constraints: Classification and Complexity. *Discrete Applied Mathematics* 5:11-24.

Boctor F. F. 1990. Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research* 49:3-13.

Bouleimen P. and H. Lecocq. 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research* 149:268-281.

Brucker P., A. Drexl, R. Mohring, K. Neumann, and E. Pesch. 1999. Resource-constrained project scheduling: notation, classification, models, and methods. *European Journal of Operational Research* 112:3-41.

Brucker P., S. Knust, A. Schoo, and O. Thiele. 1998. A branch-and-bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research* 107:272-288.

Debels D., B. D. Reyck, R. Leus, and M. Vanhoucke. 2006. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research* 169:638-653.

Debels D. and M. Vanhoucke. 2005. The electromagnetism meta-heuristic applied to the resource-constrained project scheduling problem. *Artificial Evolution*:259-270.

Deckro R. F., J. E. Hebert, and W. A. Verdini. 1991. A decomposition approach to multi-project scheduling. *European Journal of Operational Research* 51:110-118.

Demeulemeester E. and W. S. Herroelen. 1997. New benchmark results for the resource-constrained project scheduling problem. *Management Science* 43:1485-1492.

Fleszar K. and K. Hindi. 2004. Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research* 155:402-413.

Hartmann S. 2002. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics* 49:433-448.

Hartmann S. 1999. *Project scheduling under limited resources*. Berlin, Germany, Springer.

Hartmann S. 1998. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics* 45:733-750.

Herroelen W. S., Reyck B. De, and Demeulemeester E. 1998. Resource-constrained project scheduling: a survey of recent developments. *Computers and Operations Research* 25:279-302.

Kohlmorgen U., Schmeck H., and Hasse K. 1999. Experiences with fine grained parallel genetic algorithms. *Annals of Operations Research* 90:203-219.

Kolisch R. 1996. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management* 14:179-192.

Kolisch R. 1996. Serial and parallel resource-constrained project scheduling methods: revisited: theory and computation. *European Journal of Operational Research* 90: 320-333.

Kolisch R. 1995. Project scheduling under resource constraints: efficient heuristics for several problem classes. *Physica-Verlag*.

Kolisch R. and A. Drexl. 1996. Adaptive search for solving technique for resource-constrained project scheduling problems. *Naval Research Logistics* 43:23-40.

Kolisch R. and S. Hartmann. 2006. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research* 174:23-37.

Kolisch R. and A. Sprecher. 1996. PSLIB-A project scheduling problem library. *European Journal of Operational Research* 96:205-216.

Leon V. J. and B. Ramamoorthy. 1995. Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *OR Spektrum* 17:173-182.

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.

Lova A., C. Maroto, and P. Tormos. 2000. A multicriteria heuristic method to improve resource allocation in multiproject scheduling. *European Journal of Operational Research* 127:408-424.

Merkle D., M. Middendorf, and H. Schmeck. 2002. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions Evolutionary Computation* 6:333-346.

Möhring R. H., A. S. Schulz, F. Stork, and M. Uetz. 2003. Solving project scheduling problems by minimum cut computations. *Management Science* 49(3):330-350.

Morton T. E. and D. W. Pentico. 1993. Heuristic scheduling systems. *John Wiley and Sons*.

Nonobe K. and T. Ibaraki. 2002. Formulation and tabu search algorithm for the resource constrained project scheduling problem (PCPSP). In: Ribeiro, C.C., Hansen, P. (Eds.), Essays and Surveys in Metaheuristics. *Kluwer Academic Publishers, Boston* :557-588.

Nozick L., M. Turnquist, L. List, C. Lawton, D. Jones, S. Wright, and E. Keljaard. 2002. Production Planning for a Project Job Shop, with Application to Disassembly, Evaluation and Maintenance of Nuclear Weapons. *Production Planning and Control* 13:187-198.

Özdamar L. 1999. A genetic algorithm approach to a general category project scheduling problem. *IEEE Transactions* 29:44-59.

Özdamar L. and G. Ulusoy. 1995. A survey on the resource-constrained project scheduling problem. *IIE Transactions* 27:254-586.

Palpant M., C. Artigues, P. Michelon. 2004. LSSPER: The resource-constraint project scheduling problem with large neighborhood search. *Annals of Operations Research* 131:237-257.

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

Sprecher A. 2002. Network decomposition techniques for resource-constrained project scheduling. *Journal of the Operational Research Society* 53:405-414.

Tormos P. and A. Lova. 2003. An efficient multi-pass heuristic for project scheduling with constrained resources. *International Journal of Production Research* 41: 1071-1086.

Tormos, P. and A. Lova. 2001. A competitive heuristic solution technique for resource-constrained project scheduling. *Annals of Operations Research* 102:65-81.

Ulusoy G. and L. Özdamar. 1994. A constrained-based perspective in resource constrained project scheduling. *International Journal of Production Research* 32: 693-705.

Valls V., F. Ballestín, and S. Quintanilla. 2007. A hybrid genetic algorithm for the RCPSP. *European Journal of Operational Research*, (Accepted)

Valls V., F. Ballestín, and S. Quintanilla. 2005. Justification and RCPSP: A technique that pays.; *European Journal of Operational Research* 165:375-386.

Valls V., F. Ballestín, and S. Quintanilla. 2004. A population-based approach to the resource-constrained project scheduling problem. *Annals of Operations Research* 131: 305-324.

Valls V., S. Quintanilla, and F. Ballestín. 2003. Resource-constrained project scheduling: a critical activity reordering heuristic. *European Journal of Operational Research* 149:282-301.

Weiss E. 1988. An optimization based heuristic for scheduling parallel project networks with constrained renewable resources. *IIE Transactions* 20:137-143.

Węglarz J. (Ed.) Project scheduling: recent models, algorithms and applications. *Kluwer, Dordrecht 1999.*

Wulf W. and S. McKee. 1995. Hitting the memory wall: implications of the obvious. *ACM Computer Architecture News* 23(1):20-24.

Xu N., S. McKee, L. Nozick, and R. Ufomata. 2007. Augmenting priority rule heuristics with justification and rollout to solve the resource-constrained project scheduling problem, accepted at *Computers and Operations Research*.

AUTHOR BIOGRAPHIES

NINGXIONG XU is research associate in the School of Civil and Environmental Engineering at Cornell University. He earned his Ph.D. in Operations Research from Stanford University in 2001 His research interests are the development and application of optimization to tools to supply chain and infrastructure systems problems. His email address is nx22@cornell.edu **.**

LINDA NOZICK is a professor in the School of Civil and Environmental Engineering at Cornell University. Her research interests are in the development of mathematical

models for use in the management of complex systems. She has a particular interest in systems that can be represented mathematically as networks, including transportation and logistics systems, civil infrastructure networks and project networks. Her web page can be found at <http://www.cee.cornell.edu/fbxk/fcbo.cfm?pid=194>. Her email address is lkn3@cornell.edu.

ORR BERNSTEIN is a member of technical staff in the Operations Research & Computational Analysis Group at Sandia National Laboratories. He earned a Master's of Engineering degree in Computer Science from Cornell University in 2004. He specializes in the development of robust, distributed decision support applications, which incorporate a wide variety of optimization, heuristic and machine learning techniques to solve large-scale combinatorial problems. His email address is oyberns@sandia.gov.

DEAN JONES is a distinguished member of technical staff in the Operations Research & Computational Analysis Group at Sandia National Laboratories. He received his Master's degree in Applied Mathematics from the University of New Mexico in 1992. He specializes in the application of large-scale network optimization solution techniques to a variety of domains including critical infrastructures and transportation. His email address is dajones@sandia.gov.