# The Challenge of Sequence-Dependent Setups: Proposal for a Scheduling Competition Track on One Machine Sequencing Problems

**Vincent A. Cicirello**
Computer Science and Information Systems
The Richard Stockton College of New Jersey
Pomona, NJ 08240
cicirelv@stockton.edu

**Abstract**

Designing a scheduling competition to attract researchers from the several fields interested in scheduling problems seems a challenging, and highly worthwhile effort. In this paper, we propose a design for one possible track of this proposed scheduling competition. Specifically, we propose a track aimed at one machine sequencing problems. We argue that any such track must include problems with sequence-dependent setups. Our proposed single machine sequencing track would additionally include a spectrum of objective functions of increasing optimization difficulty under sequence-dependent setups. We also offer a problem instance generator along with a set of benchmark problem instances for one potential competition problem—the weighted tardiness scheduling problem with sequence-dependent setups.

## Introduction

Designing a scheduling competition to attract researchers with interests in a broad set of problem solving approaches seems a challenging, and highly worthwhile effort. It seems an important next step in a series of recent events related to bringing scheduling researchers from different fields such as AI and OR together. For example, in August of 2007 the third installment of a new conference series, the Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA) (MISTA 2003 2007), takes place. MISTA's aim since its inception in 2003 is to bring together researchers from the numerous fields interested in scheduling. Or for example, in the UK, there is the relatively recently formed Inter-disciplinary Scheduling Network (University of Nottingham 2007) supported by the UK's Engineering and Physical Sciences Research Council (EPSRC), which is a network of researchers from academia and industry as well as industrial practitioners engaged in scheduling related research and development. There have been reasonable success at bringing AI and OR researchers together in the past in the field of constraint programming such as the International Conference on Integration of AI and OR Techniques in Constraint Programming (CP-AI-OR) which grew from workshop size from 1999 to 2003 into an annual conference beginning in 2004 (CP-AI-OR 1999 2006).



Likewise, there have also been recent events aimed at bringing AI scheduling researchers together with those in the field of AI planning, such as the AAAI 2005 Workshop on Integrating Planning into Scheduling (AAAI 2005). Perhaps a more prominent example of emphasizing the importance of bringing AI planning and scheduling researchers together was when the predecessor of ICAPS changed its name from the International Conference on AI Planning Systems (AIPS) to the International Conference on AI Planning and Scheduling in 2000, retaining the AIPS acronym but increasing emphasis on the interaction of planning and scheduling (AIPS 2000 2002). Additionally, the AIPS 2000 conference included for the first (and only) time a scheduling competition (AIPS 2000). As we begin to formulate a scheduling competition as an interdisciplinary community, we should perhaps look back to see why this past attempt was a one-time thing. This paper does not attempt to answer that question.

Any successful scheduling competition must acknowledge that there is a great diversity not just in the type of approaches taken to solving scheduling problems, but also in the nature of those problems. One possible organization of a competition would be to include multiple tracks aimed at the different categories of scheduling problem. In this paper, we propose a design for such a track for a scheduling competition. Specifically, we propose a track aimed at one machine sequencing problems. We argue that any such track must include problems with sequence-dependent setups. Our proposed single machine sequencing track would additionally include a spectrum of objective functions of increasing optimization difficulty under sequence-dependent setups. We also offer a problem instance generator along with a set of benchmark problem instances for one potential competition problem—the weighted tardiness scheduling problem with sequence-dependent setups.

## Why Sequence-Dependent Setups

*Setup time* refers to a length of time that must be spent preparing a machine prior to processing a job (Morton & Pentico 1993). If all jobs are identical, or if the setup time only depends on the job that the machine is being setup for, but not on the previously processed job, then we can say that the setups are *sequence-independent*. If the setups are sequence-independent, then the problem can be trans-

formed to essentially remove them from the problem (e.g., adding the setup times to the process times). When the setup time of a job depends on the job that is processed immediately before it on the machine than the setups are *sequence-dependent*. Allahverdi et al as well as Zhu and Wilhelm offer comprehensive reviews of these and other considerations pertaining to setup costs (Allahverdi, Guptab, & Aldowaisan 1999; Zhu & Wilhelm 2006).

Sequence-dependent setups commonly appear in real-world scheduling problems (e.g., (Adler *et al.* 1993; Chiang, Fox, & Ow 1990; Morley & Schelberg 1993; Morley 1996)). Unfortunately, however, they are often ignored during the development of algorithms. The vast majority of work on sequencing problems assume that setups are sequence-independent, usually without acknowledging the possibility that they may be a factor in the problem. For example, a recent search of scholar.google.com produced 4190 documents that include both the words sequence-dependent and scheduling; whereas scholar.google.com finds 1,160,000 documents with the term scheduling. That is, less than a half percent (0.36%) of the scheduling articles, books, etc indexed by scholar.google.com consider sequence-dependent setups.[1] Using Google to search the Citeseer repository produced 383 out of 75,700 scheduling articles that consider sequence-dependent setups (approximately 0.5%). Furthermore, if you relax the search to the terms scheduling and setup, a mere 1.77% of the scheduling articles indexed by scholar.google.com, and less than one percent (0.9%) of the scheduling documents in Citeseer, even mention setup times.

Sen and Bagchi discuss the significance of the challenge that sequence-dependent setups pose for exact solution procedures (Sen & Bagchi 1996). Specifically, they discuss how sequence-dependent setups induce a non-order-preserving property of the search problem's evaluation function. At the time of their writing, exact solution procedures such as A*, Branch-and-Bound algorithms, or GREC (Sen & Bagchi 1996) for sequencing problems with sequence-dependent setups were limited to solving instances with no more than approximately 25-30 jobs. Problem instances of larger size require turning to approximate or heuristic algorithms.

## Objective Functions

Consider single machine sequencing problems with the following characteristics. Specifically, we are given a set of jobs $J = \{j_0, j_1, \ldots, j_N\}$. Each of the jobs $j$ has a weight $w_j$, duedate $d_j$, and process time $p_j$. Furthermore, $s_{i,j}$ is defined as the amount of setup time required immediately prior to the start of processing job $j$ if it follows job $i$ on the machine. It is not necessarily the case that $s_{i,j} = s_{j,i}$. The 0-th job is the start of the problem ($p_0 = 0$, $d_0 = 0$, $s_{i,0} = 0$, $w_0 = 0$). Its purpose is for the specification of the setup time of each of the jobs if sequenced first. The sequence-dependent nature of the setup times is a primary source of problem difficulty.

We now list a series of possible objective functions that could be employed within a single machine sequencing track of the proposed scheduling competition. They are listed in order of what is believed to be increasing level of difficulty. The proposed competition track could require entered schedulers to be capable of optimizing any of the following objective functions, where the objective function would be supplied to the scheduler during the execution of the competition along with the instance. The performance of the schedulers in the competition track would be evaluated on each of the objective functions. Runners-up would include the best performing schedulers for each of the objective functions, but the ultimate winner of the track would be the best all around single machine scheduler.

In the following objective functions, the completion time $c_j$ of a job $j$ is equal to the sum of the process times and setup times of all jobs that come before it in the sequence plus the setup time and process time of the job itself. Specifically, let $\pi(j)$ be the position in the sequence of job $j$. Define $c_j$ as:

$$c_j = \sum_{i,k \in J, \pi(i) <= \pi(j), \pi(i) = \pi(k)+1} p_i + s_{k,i}. \quad (1)$$

**Objective Function 1: Makespan.** The makespan objective is to sequence the set of jobs $J$ on the machine to minimize:

$$C_{\max} = \max_{j \in J} c_j. \quad (2)$$

When setups are independent of sequence, this can be trivially computed for a one machine problem without temporal constraints (i.e., simply add the process times). However, for the sequence-dependent setup problem, this is equivalent to the wandering salesperson problem which is very closely related to the traveling salesperson problem both of which are NP-Hard (Papadimitriou & Steiglitz 1998).

**Objective Function 2: Weighted Lateness.** The weighted lateness objective is to sequence the set of jobs $J$ on a machine to minimize:

$$L = \sum_{j \in J} w_j L_j = \sum_{j \in J} w_j (c_j - d_j), \quad (3)$$

where $L_j$ is the lateness of job $j$. More often, weighted lateness is expressed as weighted completion times with:

$$L = \sum_{j \in J} w_j c_j, \quad (4)$$

since the schedule that optimizes weighted completion times also optimizes weighted lateness. The duedates in the weighted lateness function are superfluous.

Sen and Bagchi specified 7 general classes of scheduling objective in increasing order of difficulty (Sen & Bagchi 1996). The first 3 classes pertained only to problems with sequence-independent setups. The next 4 categories pertained to problems with sequence-dependent setups. The first of the latter set (category D) is if the optimization objective is a linear function of the job completion times. The weighted lateness objective is a clear example of this type of objective function. It is also an objective function that has

[1] Search executed on June 13, 2007. No attempt has been made to remove duplicated documents from the search results.

received little, if any, attention within the scope of a problem with sequence-dependent setups. For example, only 3 articles are return from a search of scholar.google.com that discuss both sequence-dependent setups and the weighted lateness objective. In all 3 cases, the articles are actually considering the weighted tardiness objective (discussed below) under sequence-dependent setups; and weighted lateness is simply mentioned as a related objective function. It is proposed here as an "easier" problem level for the competition, although under sequence-dependent setups it is certainly not an easy problem and is in fact NP-Hard. If setups are independent of sequence, this objective function can be very easily optimized (Morton & Pentico 1993).

**Objective Function 3: Weighted Tardiness.** The next proposed objective function for this track of the scheduling competition is that of weighted tardiness. The weighted tardiness objective is to sequence the set of jobs $J$ on a machine to minimize:

$$T = \sum_{j \in J} w_j T_j = \sum_{j \in J} w_j \max(c_j - d_j, 0), \qquad (5)$$

where $T_j$ is the tardiness of job $j$.

The weighted tardiness objective is encountered in a number of real-world applications, including turbine component manufacturing (Chiang, Fox, & Ow 1990), the packaging industry (Adler *et al.* 1993), among others (Morton & Pentico 1993). It is a scheduling objective function that Morton and Pentico indicate to be very hard even if setups are independent of job sequence (Morton & Pentico 1993)—this "easier" case is actually NP-Hard (Garey & Johnson 1979). Within the categorization of Sen and Bagchi, it is unclear just where weighted tardiness fits. They do not discuss how things like the max function effect their categorization scheme. It is not really a linear function of the job completion times, although it is piecewise linear. Therefore, it would seem to belong in the hardest of Sen and Bagchi's categories (category G: more complex functions of the job completion times) (Sen & Bagchi 1996). Exact optimal solutions can be found by branch-and-bound for instances of at most 40-50 jobs (e.g., (Potts & van Wassenhove 1985)), but are considered impractical for instances that are larger than this (Narayan, Morton, & Ramnath 1994). For larger instances, heuristic or metaheuristic approaches are preferred (e.g., (Congram, Potts, & van de Velde 2002; Narayan, Morton, & Ramnath 1994; Potts & Van Wassenhove 1991; Crauwels, Potts, & Van Wassenhove 1998)).

Although there are exact approaches for optimizing this objective function when setups are independent of sequence (or non-existent), all current approaches for the problem when setups are sequence-dependent are either heuristic or metaheuristic. For example, there are some dispatch scheduling heuristics for the sequence-dependent setup version of the weighted tardiness problem such as ATCS (Lee, Bhaskaran, & Pinedo 1997) as well as the heuristic of (Raman, Rachamadugu, & Talbot 1989). Both of these are rather ad hoc modifications of the well-known R&M dispatch policy (Rachamadugu & Morton 1982) for the setup-free version of the problem. Additionally, there have been several recent metaheuristics for the problem. Lee et al, in addition to specifying the ATCS heuristic, suggested a local hill climbing algorithm to apply to the dispatch solution (Lee, Bhaskaran, & Pinedo 1997). Cicirello and Smith developed a value-biased stochastic sampling algorithm to expand the search around ATCS; and also benchmarked their approach with several other heuristic search algorithms (Cicirello & Smith 2005). Most recently, a permutation-based genetic algorithm using the Non-Wrapping Order Crossover operator (Cicirello 2006) and a simulated annealing algorithm (Cicirello 2007) have both improved upon a number of the best known solutions to several benchmark instances.

**Objective Function 4: Weighted Squared Tardiness.** The weighted squared tardiness objective is to sequence the set of jobs $J$ on a machine to minimize:

$$T^2 = \sum_{j \in J} w_j T_j^2 = \sum_{j \in J} w_j \max(c_j - d_j, 0)^2. \qquad (6)$$

There has been limited research into scheduling with sequence-dependent setups to optimize the weighted squared tardiness objective (e.g., (Sun, Noble, & Klein 1999)). In Sen and Bagchi's categorization, the weighted squared tardiness objective would seem at first to belong in category E which includes objectives that are quadratic functions in the completion times of the jobs (Sen & Bagchi 1996). But as with the weighted tardiness objective, the max function inside the sum seems to increase the difficulty in optimizing this objective beyond what it would be without the max. In any event, this is likely a more difficult objective function to optimize under sequence-dependent setup constraints.

## Weighted Tardiness Problem Instance Generator

During his PhD dissertation research, the author implemented a problem instance generator for the weighted tardiness scheduling problem with sequence-dependent setups (Cicirello 2003). This instance generator is an implementation of a procedure described by Lee et al and used in the analysis of Lee et al's dispatch scheduling policy ATCS (Lee, Bhaskaran, & Pinedo 1997). Cicirello's instance generator has since been refined and reimplemented in Java and is available on the web (http://loki.stockton.edu/~cicirelv/benchmarks/). Although implemented to generate problem instances for the weighted tardiness objective, it can also be used without modification to generate instances for the other objectives.

Each problem instance is characterized by three parameters: the due-date tightness factor $\tau$; the due-date range factor $R$; and the setup time severity factor $\eta$. These parameters are defined as follows:

$$\tau = 1 - \frac{\bar{d}}{\tilde{C}_{\max}} \qquad (7)$$

$$R = \frac{d_{\max} - d_{\min}}{\tilde{C}_{\max}} \qquad (8)$$

$$\eta = \frac{\bar{s}}{\bar{p}} \qquad (9)$$

where $\bar{d}$, $\bar{p}$, and $\bar{s}$ are the average duedate, average process time, and average setup time, $d_{\max}$, $d_{\min}$ are the maximum and minimum duedates, and $\tilde{C}_{\max}$ is an estimation of the makespan $C_{\max}$ (or completion time of the last job). Computing the actual makespan for the instance is NP-Hard, thus the estimator suggested by Lee et al is used: $\tilde{C}_{\max} = n(\bar{p} + \beta\bar{s})$ where $n$ is the number of jobs in the problem instance. This estimator for the makespan amounts to a sum of the process times of the jobs which is the same regardless of sequence, and an estimate of the sum of the setup times.

Lee et al provide experimental data for setting the value of $\beta$ for 4 different size problems (20, 40, 60, and 80 job instances). Cicirello's original implementation of the instance generator was restricted to generating instances of those 4 sizes. One of the refinements of the new Java implementation is fitting a curve to Lee et al's reported data to extrapolate appropriate values of $\beta$ for other size problem instances. Specifically, a small 3 node feedforward neural net (2 hidden sigmoid nodes, 1 output sigmoid) with the number of jobs, $N$, as input was fitted to the data to minimize sum of squared errors (0.000825). The result is the following definition of $\beta$ as a function of the number of jobs:

$$\beta(N) = \frac{1}{1 + e^{(1.0949132 - 1971.6253 \cdot A(N) - 8.1243637 \cdot B(N))}}, \tag{10}$$

with

$$A(N) = \frac{1}{1 + e^{(7.168150953 + 0.040112027 \cdot N)}} \tag{11}$$

and

$$B(N) = \frac{1}{1 + e^{(-10.58867025 + 2.400027877 \cdot N)}}. \tag{12}$$

The processing times of the jobs of the instances produced by the generator are uniformly distributed over the interval $[50, 150]$, with $\bar{p} = 100$. The mean setup time $\bar{s}$ is then determined from $\eta$ and the setup times are uniformly distributed in the interval $[0, 2\bar{s}]$. The duedate of a job is uniformly distributed over $[\bar{d}(1 - R), \bar{d}]$ with probability $\tau$ and uniformly distributed over $[\bar{d}, \bar{d} + (C_{\max} - \bar{d})R]$ with probability $1 - \tau$. The weights of the jobs are distributed uniformly over $[0, 10]$.

## Weighted Tardiness Benchmark Instances

In addition to the instance generator, the set of benchmark problem instances used by (Cicirello 2003; Cicirello & Smith 2005; Cicirello 2006; 2007) among others, are available on the web (http://loki.stockton.edu/~cicirelv/benchmarks/). This benchmark set includes 120 problem instances with 60 jobs each. The problem set is characterized by the following parameter values: $\tau = \{0.3, 0.6, 0.9\}$; $R = \{0.25, 0.75\}$; and $\eta = \{0.25, 0.75\}$. For each of the twelve combinations of parameter values, there are 10 problem instances. Generally speaking, these 12 problem sets cover a spectrum from loosely to tightly constrained problem instances.

Table 1: Current best known solutions for instances with: Loose Duedates, Narrow Duedate Range, Mild Setups

| Instance | Best | First Found By |
|---|---|---|
| 1 | 790 | SA-H |
| 2 | 5824 | SA-H |
| 3 | 1936 | GA |
| 4 | 6840 | SA-H |
| 5 | 5017 | SA-R |
| 6 | 7824 | SA-H |
| 7 | 3933 | SA-H |
| 8 | 298 | SA-R |
| 9 | 7059 | SA-H |
| 10 | 2125 | SA-H |

Table 2: Current best known solutions for instances with: Loose Duedates, Narrow Duedate Range, Severe Setups

| Instance | Best | First Found By |
|---|---|---|
| 11 | 5088 | SA-H |
| 12 | 0 | LDS |
| 13 | 6147 | VBSS-HC |
| 14 | 3761 | SA-R |
| 15 | 2039 | SA-R |
| 16 | 5559 | SA-R |
| 17 | 387 | SA-R |
| 18 | 1918 | SA-R |
| 19 | 239 | SA-H |
| 20 | 3805 | SA-R |

**File Format.** Each benchmark instance is stored in a separate file according to the following file format:

```
Problem Instance: <instance number>
Problem Size: <number of jobs>
Begin Generator Parameters
Tau: <tau>
R: <R>
Eta: <eta>
P_bar: <average process time>
P_MIN: <minimum process time>
P_MAX: <maximum process time>
S_bar: <average setup time>
MAX_WEIGHT: <maximum weight value>
C_max: <makespan estimate>
D_bar: <average duedate>
End Generator Parameters
Begin Problem Specification
Process Times:
<process time for job 0>
...
<process time for job n-1>
Weights:
<weight for job 0>
...
<weight for job n-1>
Duedates:
```

Table 3: Current best known solutions for instances with: Loose Duedates, Wide Duedate Range, Mild Setups

| Instance | Best | First Found By |
|---|---|---|
| 21 | 0 | LDS |
| 22 | 0 | LDS |
| 23 | 0 | LDS |
| 24 | 1092 | SA-H |
| 25 | 0 | SA |
| 26 | 0 | LDS |
| 27 | 57 | SA-R |
| 28 | 0 | GA |
| 29 | 0 | LDS |
| 30 | 215 | SA-R |

Table 4: Current best known solutions for instances with: Loose Duedates, Wide Duedate Range, Severe Setups

| Instance | Best | First Found By |
|---|---|---|
| 31 | 0 | LDS |
| 32 | 0 | LDS |
| 33 | 0 | LDS |
| 34 | 0 | LDS |
| 35 | 0 | LDS |
| 36 | 0 | LDS |
| 37 | 1008 | SA-R |
| 38 | 0 | LDS |
| 39 | 0 | LDS |
| 40 | 0 | LDS |

```
<duedate for job 0>
...
<duedate for job n-1>
Setup Times:
<i> <j> <setup for job j if after i>
// i=-1 indicates setup if j is first
End Problem Specification
```

**Current Best Known Solutions.** Tables 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12 list the current best known solutions to the benchmark instances. Specifically, each table lists the instance number, the current best known solution (for the weighted tardiness objective), and the algorithm that first found that solution. Note that other algorithms may also have found that solution. We simply list here the first one that did. Algorithms are abbreviated as follows:

- SA-H: A recent simulated annealing algorithm (Cicirello 2007).
- SA-R: A second version of that recent simulated annealing algorithm (Cicirello 2007).
- GA: A permutation-based genetic algorithm using the Non-Wrapping Order Crossover operator and an insertion mutator (Cicirello 2006).
- VBSS-HC: A multistart hill climber seeded with starting configurations generated by Value Biased Stochastic Sampling (Cicirello & Smith 2005).
- VBSS: Value Biased Stochastic Sampling (Cicirello & Smith 2005).
- LDS: Limited Discrepancy Search truncated after exhausting all search trajectories with 2 or less discrepancies from the ATCS heuristic solution (Cicirello & Smith 2005). It was used in the original empirical evaluation of VBSS and VBSS-HC.
- SA: A simulated annealing algorithm also used in the original empirical evaluation of VBSS and VBSS-HC (Cicirello & Smith 2005).

Table 5: Current best known solutions for instances with: Moderate Duedates, Narrow Duedate Range, Mild Setups

| Instance | Best | First Found By |
|---|---|---|
| 41 | 71242 | SA-H |
| 42 | 59493 | SA-H |
| 43 | 147737 | SA-H |
| 44 | 36265 | SA-H |
| 45 | 59696 | SA-R |
| 46 | 36175 | SA-R |
| 47 | 74389 | SA-H |
| 48 | 65129 | SA-R |
| 49 | 79656 | SA-R |
| 50 | 32777 | SA-R |

Table 6: Current best known solutions for instances with: Moderate Duedates, Narrow Duedate Range, Severe Setups

| Instance | Best | First Found By |
|---|---|---|
| 51 | 54707 | SA-H |
| 52 | 100793 | SA-H |
| 53 | 94394 | SA-R |
| 54 | 123558 | VBSS |
| 55 | 72420 | SA-R |
| 56 | 80258 | SA-H |
| 57 | 68535 | SA-H |
| 58 | 46978 | SA-R |
| 59 | 56181 | SA-H |
| 60 | 68395 | SA-R |

The problem instances and the best known solutions to them are organized according to the 12 classes of instances as follows:

- Table 1: Loose Duedates, Narrow Duedate Range, Mild Setups
- Table 2: Loose Duedates, Narrow Duedate Range, Severe Setups
- Table 3: Loose Duedates, Wide Duedate Range, Mild Setups
- Table 4: Loose Duedates, Wide Duedate Range, Severe Setups
- Table 5: Moderate Duedates, Narrow Duedate Range, Mild Setups

Table 7: Current best known solutions for instances with: Moderate Duedates, Wide Duedate Range, Mild Setups

| Instance | Best | First Found By |
|---|---|---|
| 61 | 76769 | SA-R |
| 62 | 44781 | SA-H |
| 63 | 76059 | SA-H |
| 64 | 93079 | SA-H |
| 65 | 127713 | SA-R |
| 66 | 59717 | SA-H |
| 67 | 29394 | SA-R |
| 68 | 22653 | SA-R |
| 69 | 71534 | SA-H |
| 70 | 76140 | SA-R |

Table 8: Current best known solutions for instances with: Moderate Duedates, Wide Duedate Range, Severe Setups

| Instance | Best | First Found By |
|---|---|---|
| 71 | 155036 | SA-R |
| 72 | 49886 | SA-H |
| 73 | 30259 | SA-H |
| 74 | 32083 | SA-R |
| 75 | 21602 | SA-R |
| 76 | 57593 | SA-H |
| 77 | 35380 | SA-H |
| 78 | 21443 | SA-H |
| 79 | 121434 | SA-H |
| 80 | 20221 | SA-H |

- Table 6: Moderate Duedates, Narrow Duedate Range, Severe Setups
- Table 7: Moderate Duedates, Wide Duedate Range, Mild Setups
- Table 8: Moderate Duedates, Wide Duedate Range, Severe Setups
- Table 9: Tight Duedates, Narrow Duedate Range, Mild Setups
- Table 10: Tight Duedates, Narrow Duedate Range, Severe Setups
- Table 11: Tight Duedates, Wide Duedate Range, Mild Setups
- Table 12: Tight Duedates, Wide Duedate Range, Severe Setups

In this set of benchmark instances, loose duedates, moderate duedates, and tight duedates refer to values of the duedate tightness factor $\tau$ of 0.3, 0.6, and 0.9, respectively. Narrow duedate range and wide duedate range refer to values of the duedate range factor of 0.25 and 0.75, respectively. Mild setups and severe setups refer to values of the setup time severity factor $\eta$ of 0.25 and 0.75, respectively.

Table 9: Current best known solutions for instances with: Tight Duedates, Narrow Duedate Range, Mild Setups

| Instance | Best | First Found By |
|---|---|---|
| 81 | 385918 | SA-H |
| 82 | 410550 | SA-H |
| 83 | 459939 | SA-R |
| 84 | 330186 | SA-R |
| 85 | 557831 | SA-R |
| 86 | 364474 | SA-R |
| 87 | 400264 | SA-R |
| 88 | 434176 | SA-R |
| 89 | 411810 | SA-H |
| 90 | 403623 | SA-R |

Table 10: Current best known solutions for instances with: Tight Duedates, Narrow Duedate Range, Severe Setups

| Instance | Best | First Found By |
|---|---|---|
| 91 | 344428 | SA-R |
| 92 | 363388 | SA-R |
| 93 | 410462 | VBSS |
| 94 | 334180 | SA-H |
| 95 | 524463 | SA-R |
| 96 | 464403 | LDS |
| 97 | 418995 | SA-H |
| 98 | 532519 | VBSS |
| 99 | 374607 | SA-R |
| 100 | 441888 | VBSS-HC |

## Conclusions

In this paper, the design of a scheduling competition track focused on one machine sequencing problems was proposed. We argued that any such track should include scheduling problems with sequence-dependent setups. Sequence-dependent setups arise in a number of real-world scheduling problems, but are often ignored in the design of scheduling algorithms. Setups that are dependent upon the ordering of the jobs lead to sequencing problems that are far more difficult. For example, prior research shows that algorithms guaranteed to find the exact optimal solutions are largely limited to solving instances with no more than 25-30 jobs, at least for practical purposes, if setups are sequence-dependent even for "easy" objective functions. Coupling this with hard objective functions such as weighted tardiness offer a worthy challenge problem for a competition. To facilitate the design of this track further, the author has made an existing set of benchmark instances, along with a problem instance generator available on the web.

## Acknowledgments

The preparation of this paper was supported by the Richard Stockton College of New Jersey's Research and Professional Development Program. Thanks also go to the anonymous reviewers who provided valuable input.

Table 11: Current best known solutions for instances with: Tight Duedates, Wide Duedate Range, Mild Setups

| Instance | Best | First Found By |
|---|---|---|
| 101 | 353575 | SA-R |
| 102 | 495094 | SA-H |
| 103 | 380170 | VBSS |
| 104 | 358738 | SA-R |
| 105 | 450806 | SA-R |
| 106 | 457284 | SA-H |
| 107 | 353564 | SA-H |
| 108 | 462675 | SA-R |
| 109 | 413918 | SA-H |
| 110 | 419014 | SA-R |

Table 12: Current best known solutions for instances with: Tight Duedates, Wide Duedate Range, Severe Setups

| Instance | Best | First Found By |
|---|---|---|
| 111 | 348796 | SA-H |
| 112 | 375952 | SA-H |
| 113 | 261795 | SA-H |
| 114 | 471422 | SA-H |
| 115 | 460225 | VBSS |
| 116 | 537593 | SA-H |
| 117 | 507188 | SA-H |
| 118 | 357575 | LDS |
| 119 | 581119 | SA-H |
| 120 | 399700 | VBSS-HC |

## References


AAAI. 2005. AAAI 2005 workshop on integrating planning into scheduling. http://www.aaai.org/Library/Workshops/ws05-06.php.

Adler, L.; Fraiman, N. M.; Kobacker, E.; Pinedo, M.; Plotnitcoff, J. C.; and Wu, T. P. 1993. BPSS: a scheduling system for the packaging industry. *Operations Research* 41:641–648.

AIPS. 2000–2002. The international conference on AI planning and scheduling. http://www.icaps-conference.org/.

AIPS. 2000. AIPS2000 scheduling competition. http://www-aig.jpl.nasa.gov/public/aips00/AIPS-SchedComp.html.

Allahverdi, A.; Guptab, J.; and Aldowaisan, T. 1999. A review of scheduling research involving setup considerations. *Omega: International Journal of Management Science* 27:219–239.

Chiang, W. Y.; Fox, M. S.; and Ow, P. S. 1990. Factory model and test data descriptions: OPIS experiments. Technical Report CMU-RI-TR-90-05, The Robotics Institute, Carnegie Mellon University.

Cicirello, V. A., and Smith, S. F. 2005. Enhancing stochastic search performance by value-biased randomization of heuristics. *Journal of Heuristics* 11(1):5–34.

Cicirello, V. A. 2003. *Boosting Stochastic Problem Solvers Through Online Self-Analysis of Performance*. Ph.D. Dissertation, The Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Also available as technical report CMU-RI-TR-03-27.

Cicirello, V. A. 2006. Non-wrapping order crossover: An order preserving crossover operator that respects absolute position. In M. Keijzer et al., ed., *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'06)*, volume 2, 1125–1131. ACM Press.

Cicirello, V. A. 2007. On the design of an adaptive simulated annealing algorithm. Submitted for review to the CP 2007 Workshop on Autonomous Search.

Congram, R. K.; Potts, C. N.; and van de Velde, S. L. 2002. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing* 14(1):52–67.

CP-AI-OR. 1999–2006. International conference on integration of AI and OR techniques in constraint programming (CP-AI-OR). http://www-sop.inria.fr/coprin/cpaior04/.

Crauwels, H. A. J.; Potts, C. N.; and Van Wassenhove, L. N. 1998. Local search heuristics for the single machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing* 10(3):341–350.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Company.

Lee, Y. H.; Bhaskaran, K.; and Pinedo, M. 1997. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions* 29:45–52.

MISTA. 2003–2007. Multidisciplinary international scheduling conference: Theory and applications (MISTA). http://www.mistaconference.org/.

Morley, D., and Schelberg, C. 1993. An analysis of a plant-specific dynamic scheduler. In *Proceedings of the NSF Workshop on Intelligent, Dynamic Scheduling for Manufacturing Systems*, 115–122.

Morley, D. 1996. Painting trucks at general motors: The effectiveness of a complexity-based approach. In *Embracing Complexity: Exploring the Application of Complex Adaptive Systems to Business*, 53–58. The Ernst and Young Center for Business Innovation.

Morton, T. E., and Pentico, D. W. 1993. *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management*. John Wiley and Sons.

Narayan, V.; Morton, T.; and Ramnath, P. 1994. X-Dispatch methods for weighted tardiness job shops. GSIA Working Paper 1994-14, Carnegie Mellon University, Pittsburgh, PA.

Papadimitriou, C. H., and Steiglitz, K. 1998. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications.

Potts, C. N., and van Wassenhove, L. N. 1985. A branch and bound algorithm for the total weighted tardiness problem. *Operations Research* 33(2):363–377.

Potts, C. N., and Van Wassenhove, L. N. 1991. Single machine tardiness sequencing heuristics. *IIE Transactions* 23(4):346–354.

Rachamadugu, R. V., and Morton, T. E. 1982. Myopic heuristics for the single machine weighted tardiness problem. Working Paper 30-82-83, GSIA, Carnegie Mellon University, Pittsburgh, PA.

Raman, N.; Rachamadugu, R. V.; and Talbot, F. B. 1989. Real time scheduling of an automated manufacturing center. *European Journal of Operational Research* 40:222–242.

Sen, A. K., and Bagchi, A. 1996. Graph search methods for non-order-preserving evaluation functions: Applications to job sequencing problems. *Artificial Intelligence* 86(1):43–73.

Sun, X.; Noble, J. S.; and Klein, C. M. 1999. Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. *IIE Transactions* 31(2):113–124.

University of Nottingham. 2007. Inter-disciplinary scheduling network. http://www.asap.cs.nott.ac.uk/iol/is-network/.

Zhu, X., and Wilhelm, W. E. 2006. Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions* 38(11):987–1007.