A Hybrid Constraint Programming / Local Search Approach to the Job-Shop Scheduling Problem

Jean-Paul Watson[1] and J. Christopher Beck[2]

[1] Discrete Math and Complex Systems Department,
Sandia National Laboratories,
Albuquerque, New Mexico, USA
jwatson@sandia.gov
[2] Department of Mechanical and Industrial Engineering,
University of Toronto, Toronto, Ontario, Canada
jcb@mie.utoronto.ca

Abstract. Since their introduction, local search algorithms – and in particular tabu search algorithms – have consistently represented the state-of-the-art in solution techniques for the classical job-shop scheduling problem. This is despite the availability of powerful search and inference techniques for scheduling problems developed by the constraint programming community. In this paper, we introduce a simple hybrid algorithm for job-shop scheduling that leverages both the fast, broad search capabilities of modern tabu search and the scheduling-specific inference capabilities of constraint programming. The hybrid algorithm significantly improves the performance of a state-of-the-art tabu search for the job-shop problem, and represents the first instance in which a constraint programming algorithm obtains performance competitive with the best local search algorithms. Further, the variability in solution quality obtained by the hybrid is significantly lower than that of pure local search algorithms. As an illustrative example, we identify twelve new best-known solutions on Taillard's widely studied benchmark problems.

1 Introduction

Local search algorithms for the traditional makespan-minimization formulation of the job-shop scheduling problem (JSP) have dominated the state-of-the-art for at least the past 15 years. These include Nowicki and Smutnicki's landmark TSAB tabu search algorithm [13], Balas and Vazacopoulos' guided local search algorithm [1], Nowicki and Smutnicki's follow-on i-TSAB tabu search algorithm [14], and most recently Zhang et al.'s hybrid tabu search / simulated annealing algorithm [25]. These algorithms are all built upon a foundation of one or more powerful, problem-specific move operators, which are able to efficiently identify promising feasible and high-quality solutions in the neighborhood of a given solution. Metaheuristic search strategies then leverage these move operators to perform global search for minimal-cost solutions; the complexity of these strategies ranges from simple tabu search (in the case of TSAB) to highly intricate hybridizations of tabu search, path relinking, and elite pool maintenance schemes (in the case of i-TSAB).

On established benchmark problems [19,20], the progression of local search algorithms has consistently established new upper bounds over time, perhaps leading one to question the utility of further research in the area. We address such criticism with the following observations. First, on the most difficult benchmark instances, there is no indication that the upper bounds are necessarily close to the optimal solutions, i.e., there likely remains significant room for performance improvements. Second, although the aforementioned local search algorithms collectively have established the best-known solutions to benchmark instances, no single algorithm can consistently generate these solutions. Consequently, improved algorithms yielding reductions in performance variability are desirable. Third, proportionally little research is dedicated to understanding why local search algorithms are so effective on the job-shop scheduling problem; developing such knowledge is foundational to consistently achieving high-performance in other problem domains.

Perhaps somewhat paradoxically, constraint programming (CP) algorithms are more commonly used than their local search counterparts to obtain solutions to real-world scheduling problems, e.g., using ILOG's Scheduler software library [17]. This is widely attributed to a combination of the ability to easily incorporate various idiosyncratic "side" constraints that are pervasive in real-world scheduling problems (such constraints can require significant redesign of local search algorithms) and to effectively deduce, via powerful domain-specific constraint propagators, the implications of various scheduling decisions. However, despite the level of research effort dedicated to the development of scheduling-specific constraint propagation and search techniques (e.g., see [2,4]), the performance of CP algorithms on the traditional JSP has significantly lagged that of their local search counterparts. To date, the strongest CP-based algorithm is solution-guided multi-point constructive search [3], although the performance of even this algorithm lags that of modern tabu search algorithms for the JSP [9] in terms of both time and final solution quality.

Hybridization of local search and CP on JSPs without side constraints does not, therefore, immediately appear to be a promising research direction. However, the following two unexplored aspects of these algorithms provide what we feel to be contrary evidence, and motivate the line of research developed in this paper:

– The strong propagation techniques in CP are more efficient in constrained search states. That is, the polynomial time inference algorithms are more likely to be able to find implied constraints, and to reduce the search space, in states that are already highly constrained. When a good solution has been found, strong "back-propagation" from the upper bound on the makespan results in such a highly constrained search state. Therefore, we conjecture that while CP is unable to competitively find good solutions, once given a good solution, it may be able to improve on it more quickly than a local search approach.
– A popular conceptualization of the power of modern local search algorithms is that they balance intensification with diversification [9]. Intensification, which can loosely be understood as searching "near" an existing good solution, is often implemented by repeatedly restarting search from a good solution that has been found earlier. Diversification, in contrast, tries to distribute the search effort in unexplored areas of the search space. It is often implemented by maintaining a varied set of

promising solutions and combining them in a variety of ways, such as via path relinking [7]. However, modern tabu search algorithms seem to do a relatively poor job of intensification. Watson [22] showed that a relatively small number of iterations after restarting from a good solution, tabu search is a considerable distance from the starting solution. Further, *a posteriori* analysis of algorithmic traces indicates that tabu search often fails to locate high-quality solutions that are quite close to high-quality solutions located by tabu search [22]. In contrast, solution-guided constructive search performs a much more focused search around its guiding solution [3]. Therefore, we conjecture that improved performance may result from using CP to strongly intensifying around a diverse set of high-quality solutions generated by tabu search.

The remainder of this paper is organized as follows. We begin in Section 2 with a brief discussion of the job-shop scheduling problem and the benchmark instances used in our analysis. The foundational algorithms for our hybrid approach – iterated simple tabu search (i-STS) and solution-guided multi-point constructive search (SGMPCS) – and our simple hybrid are described in Section 3. Our experimental methodology is introduced in Section 4, followed by a description of empirical performance results in Section 5. Section 6 details some implications of our results, followed by our conclusions in Section 7.

2 Problem Description and Benchmark Instances

We consider the well-known $n \times m$ static, deterministic JSP in which n jobs must be processed exactly once on each of m machines [5]. Each job i ($1 \leq i \leq n$) is routed through each of the m machines in a pre-defined order π_i, where $\pi_i(j)$ denotes the jth machine ($1 \leq j \leq m$) in the routing order of job i. The processing of job i on machine $\pi_i(j)$ is denoted o_{ij} and is called an operation. An operation o_{ij} must be processed on machine $\pi_i(j)$ for an integral duration $\tau_{ij} > 0$. Once initiated, processing cannot be preempted and concurrency on individual machines is not allowed, i.e., the machines are unit-capacity resources. For $2 \leq j \leq m$, o_{ij} cannot begin processing until $o_{i(j-1)}$ has completed processing. The scheduling objective is to minimize the makespan C_{max}, i.e., the maximal completion time of the last operation of any job. Makespan-minimization for the JSP is NP-hard for $m \geq 2$ and $n \geq 3$ [6].

An instance of the $n \times m$ JSP is uniquely defined by the set of nm operation durations τ_{ij} and n job routing orders π_i. In nearly all benchmark instances, the τ_{ij} are uniformly sampled from the interval $[1, 99]$, while the π_i are given by random permutations of the integer sequence $1, \ldots, m$. As discussed in Section 4, our experimental results are generated using a subset of Taillard's benchmark instances, specifically those labeled `ta11` through `ta50` [19]. This subset contains 10 instances of each of the following problem sizes: 20×15, 20×20, 30×15, and 30×20. We have selected these instances because they are widely studied (all competitive algorithms introduced since 1995 have been tested on these instances), are known to be very challenging (even state-of-the-art algorithms fail to consistently find solutions with makespans equal to the best known solutions), and there remains "headroom" for improvement in best-known makespans (as illustrated by the often large gap between those values and the best-known lower bounds). For these same reasons, we ignore the easier instances in Taillard's problem suite, in addition to many historical instances (e.g., the "ft", "la", and "orb" instances) for which modern JSP algorithms can consistently locate optimal solutions.

3 Algorithms

In this section, we discuss the two foundational algorithms in detail before presenting the simple hybridization we investigate in this paper.

3.1 Iterated Simple Tabu Search

Beginning with an early approach by Taillard [21], tabu search algorithms have consistently represented the state-of-the-art in obtaining high-quality solutions for the JSP. A variety of researchers have introduced tabu search algorithms of increasing effectiveness and complexity. Specific algorithmic advances of note in this progression include the introduction of (1) the highly restrictive $N5$ critical path-based move operator [13], (2) search intensification mechanisms in conjunction with sets of "elite" or high-quality solutions [13], and (3) search diversification mechanisms in the form of path relinking [14]. These techniques are simultaneously embodied in Nowicki and Smutnicki's i-TSAB algorithm, which has represented the state-of-the-art since 2003. With the exception noted below, the sole competitor is a hybrid tabu search / simulated annealing algorithm introduced by Zhang et al. [25]. The Zhang et al. algorithm uses simulated annealing to generate an initial set of elite solutions, which are then processed via tabu search-driven intensification. The primary differences between the Zhang et al. algorithm and i-TSAB are the lack of an explicit diversification mechanism (path relinking is used in i-TSAB) and the use of the $N6$ move operator introduced by Balas and Vazacopoulos [1] (i-TSAB employs the $N5$ move operator).

Although remarkably effective, i-TSAB is an extremely intricate and complex algorithm. Such complexity is a significant drawback to researchers, as in practice it impedes reproducibility, adoption, and subsequent study. In the specific case of i-TSAB, the intricacy makes it difficult to assess the contribution of the various algorithmic components to its overall performance. Toward this goal, we previously introduced a simplified version of i-TSAB, which we denote iterated simple tabu search, or i-STS [9]. As discussed below, i-STS contains the key algorithmic ingredients of i-TSAB while reducing the overall complexity and maintaining near-equivalent performance.

A basic tabu search lies at the core of i-STS, built around the $N5$ move operator. Short-term memory is used to prevent inversion of recently swapped pairs of adjacent operations on a critical path. Following [21], the tabu tenure is periodically and randomly sampled from a fixed interval $[L, U]$. Search in i-STS proceeds in two phases. In the first phase, the basic tabu search algorithm is executed for a small, fixed number of iterations from each of $|E|$ different random initial solutions. The best solution from each iteration-limited run is saved, and forms the initial set E of elite solutions.

In the second phase of i-STS, the elite solutions, E, are iteratively processed by both intensification and diversification mechanisms, each selected at any given iteration with respective probabilities p_i and p_d, $p_i + p_d = 1$. To perform search intensification, a single elite solution $e \in E$ is selected at random and an iteration-limited tabu search is executed from e. Due to tie-breaking during move selection, facilitated by the pervasiveness of plateaus of equally fit neighboring solutions in the JSP [23], different trajectories can locate solutions of variable quality. If a solution e' with a lower makespan than e is located, e' replaces e in E. To perform diversification, two elite solutions $e_1, e_2 \in E$ are selected at random. Path relinking is then performed to generate a solution e' that is approximately equi-distant from both e_1 and e_2. Iteration-limited tabu search is then executed from e', as is performed in the intensification process. If a solution e'' is identified with a lower makespan than e_1, then e'' replaces e_1 in E. The second phase of i-STS continues until an aggregate number of basic tabu search iterations M have been executed, with the best solution $e \in E$ returned upon completion.

3.2 Solution-Guided Multi-point Constructive Search

Solution-guided multi-point constructive search (SGMPCS) is a recently proposed algorithm that combines constructive tree search, randomized restart, and heuristic guidance from good solutions found earlier in the search [3]. The basic approach is a CP tree search with a limit on the number of dead-ends ("fails") that are encountered before restarting. Each tree search is guided by using an existing sub-optimal solution as a value-ordering heuristic. Once a variable to be assigned has been chosen (see below), the value chosen is the one in the guiding solution, provided that value is still in the domain of the chosen variable. Otherwise, any other value-ordering heuristic may be used. As in i-STS, a small set of "elite" solutions is maintained, one of which is chosen with uniform probability to guide a given tree search. When a tree search exhausts its fail limit, it returns the best solution it has found (if any). That solution, if it exists, replaces the guiding solution in the elite pool.

Beck [3] showed that SGMPCS has strong, but not state-of-the-art, performance on job shop scheduling, makespan minimization problems. While finding significantly better solutions than chronological backtracking and randomized restart (using the same propagators, heuristics, and, in the latter case, fail limit sequences), SGMPCS was not able to perform as well as i-STS.

Details. A simplified version of SGMPCS is used in this paper. This version fixes a number of the parameters in the full algorithm. As the version presented here is a particular parametrization of the full version, we continue to refer to it as SGMPCS. Readers interested in the full version are referred to Beck [3].

Pseudocode for SGMPCS is shown in Algorithm 1. The algorithm initializes a set, E, of elite solutions and then enters a while-loop. In each iteration, a chronological backtracking search is guided with a randomly selected elite solution (line 6). If a solution, s, is found during the search, it replaces the starting elite solution, r. Each individual search is limited by a fail bound: a maximum number of fails that can be incurred. The entire process ends when the problem is solved, proved insoluble within one of the tree searches, or when some overall bound on the computational resources (e.g., CPU time or number of fails) is reached.

More formally, a search tree is created by asserting a series of choice points of the form: $\langle V_i = x \rangle \vee \langle V_i \neq x \rangle$, where V_i is a variable and x is the value assigned to V_i.

Algorithm 1. SGMPCS: Solution-Guided Multi-Point Constructive Search

```
SGMPCS():
1 initialize elite solution set $E$
2 while not solved and termination criteria unmet do
3     $r$ := randomly chosen element of $E$
4     set upper bound on cost function
5     set fail bound, $b$
6     $s$ := search($r$, $b$)
7     if $s$ is better than $r$ then
8         replace $r$ with $s$
9 return best($E$)
```

SGMPCS can use any variable-ordering heuristic to choose the variable to assign. The choice point is formed using the value assigned in the guiding solution or, if the value in the guiding solution is inconsistent, a heuristically chosen value. Let a guiding solution, r, be a set of variable assignments, $\{\langle V_1 = x_1\rangle, \langle V_2 = x_2\rangle, \ldots, \langle V_m = x_m\rangle\}, m \leq n$, where n is the number of decision variables. Let $dom(V_i)$ be the set of possible values (i.e., the *domain*) of variable V_i. The variable-ordering heuristic has complete freedom to choose a variable, V_i, to be assigned. If $x_i \in dom(V_i)$, where $\langle V_i = x_i\rangle \in r$, the choice point is made with $x = x_i$. Otherwise, if $x_i \notin dom(V_i)$, any value-ordering heuristic can be used to choose $x \in dom(V_i)$.

At line 4 in the pseudocode, an upper bound is placed on the cost function for the subsequent search. We use two different methods in our experiments. The *local upper bound* is one less than cost of the guiding solution (i.e., cost(r)-1). The *global upper bound* is one less than the best solution that has been found (i.e., cost(best(E))-1). Intuitively, the local upper bound allows a more heuristic search since local improvements will be accepted into the elite set while the global upper bound tries to maximize the impact of the inference algorithms as it always searches in the most constrained space possible.

Given a large enough fail limit (line 5), an individual search can exhaust the search space. Therefore, completeness depends on the policy for setting the fail limit. In our experiments, we will use two fail sequence polices: Fixed and Luby. The Fixed limit simply uses a constant fail-limit for each search. Obviously, such a policy is not complete. The Luby limit is an evolving sequence that has been shown to be the optimal sequence for satisfaction problems under the condition of no knowledge about the solution distribution [12]. The sequence is as follows: 1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, ... That is, the fail limit for the first and second searches is 1, for the third search is 2, and so on. Following [24] and our own preliminary experimentation, we multiply the elements of the sequence by a fixed constant. As the sequence increases without limit, a single search will eventually have a fail limit that is sufficient to search the entire search space and therefore the overall algorithm using the Luby fail limit is complete.

SGMPCS is a general framework for constructive tree search. To apply SGMPCS to the JSP, solutions are encoded using the well known disjunctive graph representation.

Texture-based heuristics [4] are used to identify a machine and time point with maximum contention among the operations and to then choose a pair of unordered operations. The heuristic is randomized by specifying that the $\langle$machine, time point$\rangle$ pair is chosen with uniform probability from the top 10% most critical pairs. The ordering found in the guiding solution is asserted. Note that because the decisions are binary, the pair in the solution must be locally consistent, otherwise the pair of operations would already be sequenced in the opposite order. The standard constraint propagation techniques for scheduling [15,10,11] (available via the ILOG Scheduler library) are also used.

3.3 A Very Simple Hybrid Approach

Given the complexities of the two foundational algorithms, our approach to hybridization in this paper is the most concise that we could envision. We begin by executing i-STS for a fixed number of total iterations of the underlying tabu search algorithm. The intent of this phase is to quickly generate a set of high-quality solutions. At the end of these iterations, the best $|E|$ solutions that have been found are used as the initial elite set for SGMPCS (line 1 of the SGMPCS pseudocode). SGMPCS is then run for a fixed, comparatively larger (in contrast to i-STS) CPU time and the best solution found is returned.

There is clearly much more we could do. However, this simple (and perhaps, simple-minded) hybrid directly and concisely addresses the two motivations we had for this research, as described in Section 1.

4 Experimental Methodology

Our analysis is based on multiple runs of the hybrid on each of the Taillard benchmark instances we consider. Each run consists of executing i-STS for 5M iterations, which requires a few minutes of CPU time, depending on the problem size. For each run, we use an elite pool size of 8, and $p_i = p_d = 0.5$; these parameter settings were chosen based on prior empirical studies of both i-TSAB [14] and i-STS [9]. All remaining free parameters of i-STS are set identically to that reported in [9]. At the end of 5M iterations, the best $|E|$ elite solutions are used as the starting elite set for SGMPCS. SGMPCS is then run for 30 CPU minutes and reports the best solution found. Each problem instance is run 10 times independently for a given parameter configuration. i-STS is implemented in C++ while SGMPCS uses ILOG Scheduler 6.5 (also in C++). All code was compiled using the GNU gcc compiler. Experiments were executed on a cluster containing 2GHz Dual Core AMD Opteron 270 nodes, each with 2GB RAM running Red Hat Enterprise Linux 4.

Given an experimental setup for a problem instance (10 runs per instance, fixing the behavior of i-STS), we next consider the various configurations of the remaining free SGMPCS parameters. Following [3], we perform a full factor experimental design, considering: (1) $|E| \in \{1, 4, 8\}$ (for SGMPCS; in i-STS, $|E|$ is fixed to 8), (2) local and global upper bounds, and (3) a fixed fail limit of 500, in addition to scaled Luby limits with parameters 100 and 200. We observe that when $|E| = 1$, local and global upper bounding strategies are equivalent. Consequently, we execute a set of 10 runs on each of Taillard's instances under fifteen distinct configurations of SGMPCS.

5 Results

Our analysis is broken into four components: parameter sensitivity (Section 5.1), performance relative to state-of-the-art algorithms for the JSP (Section 5.2), best-known upper bounds (Section 5.3), and proving optimality (Section 5.4).

5.1 Parameter Sensitivity

We first analyze the performance of the various SGMPCS parameterizations relative to one another, in an effort to determine parameter sensitivity and a sole candidate for comparison of the hybrid algorithm with other state-of-the-art JSP algorithms. We performed a two-way (factorial) ANOVA on the resulting data.[1] The three independent variables are elite pool size, fail limit, and upper bound method, while the sole dependent variable is the makespan of the best solution obtained. The outcome of this experiment indicated that there were *no* significant main *or* interaction effects between the SGMPCS parameters and the quality of the final solution obtained. The largest p value obtained was for $|E|$, and was equal to 0.499. This result is in direct contrast to [3], in which all main factors and interactions were statistically significant. The sole difference in the experimental designs is the mechanism used to initialize the elite solution set for SGMPCS (i-STS versus random solutions). It appears that while different parameterizations of SGMPCS influence the degree to which the algorithm can improve upon random initial solutions, this sensitivity disappears once solution quality is "sufficiently" good, e.g., as is the case for i-STS solutions.

Next, we examine the absolute difference in performance between the various algorithm configurations. For a given problem instance and solution makespan M, we define the relative error as $RE = (M - LB)/LB * 100$, where LB is the largest known lower bound for the instance. For our analysis, we take LB from [20], where such values have been recorded by Taillard since the introduction of these problem instances. Consider a specific parameterization of SGMPCS in our hybrid algorithm, and one of the following statistics defined over the set of 10 runs on a given problem instance: best makespan, average makespan, and worst makespan. The mean relative error, or MRE, for the given parameterization and makespan statistic is then computed simply as the mean RE taken over the 40 problem instances.

The resulting MRE statistics for our hybrid algorithm are shown in Table 1. Boldfaced entries indicate that the corresponding SGMPCS parameterization yielded the best performance with respect to the given makespan statistic. We exclude results for parameterizations with the Luby 100 fail limit, as they generally, though marginally, under-perform the respective Luby 200 fail limit strategy. Consistent with the two-way ANOVA analysis, the differences in all makespan statistics are minimal in absolute terms, varying at most by 0.20% in any given column. Although no parameterization stands out as a winner, the parameterization with $|E| = 8$, a fixed fail limit of 500,

[1] All statistical analyses were performed using the publicly available R software package.

Table 1. Mean relative error (MRE) statistics for the i-STS / SGMPCS hybrid on Taillard's benchmark instances for various parameter configurations. Bold-faced entries in an MRE column indicate the configuration obtaining the best performance.

$\|E\|$	Fail Limit	Upper Bound	Best MRE	Mean MRE	Worst MRE
1	Fixed 500	Local/Global	3.146	3.490	3.897
1	Luby 200	Local/Global	3.178	3.502	3.886
4	Fixed 500	Local	3.134	3.392	3.705
4	Luby 200	Local	3.143	3.424	3.718
4	Fixed 500	Global	3.129	3.408	3.726
4	Luby 200	Global	3.136	3.425	3.746
8	Fixed 500	Local	3.123	**3.369**	3.706
8	Luby 200	Local	3.142	3.397	**3.691**
8	Fixed 500	Global	**3.103**	3.400	3.709
8	Luby 200	Global	3.148	3.409	3.694

Table 2. MRE statistics for i-TSAB, Zhang et al.'s hybrid tabu search / simulated annealing algorithm, and our hybrid i-STS / SGMPCS algorithm, on Taillard's benchmark instances

Instance	Best	i-TSAB	Zhang		Hybrid		
Group	Known		Best	Mean	Best	Mean	Worst
ta11-20	2.29	2.81	2.37	2.92	2.26	2.45	2.83
ta21-30	5.38	5.68	5.44	5.97	5.52	5.71	6.00
ta31-40	0.46	0.78	0.55	0.93	0.50	0.68	0.85
ta41-50	4.02	4.70	4.07	4.84	4.22	4.63	5.15
Overall	3.04	3.49	3.11	3.67	3.13	3.37	3.71

and the local upper bound obtained the most consistent performance, as measured in terms of average makespan of solutions obtained. For this reason, we emphasize this parameterization of SGMPCS in subsequent analyses.

5.2 Performance Relative to the State-of-the-Art

Having established the relative insensitivity of our hybrid algorithm performance to SGMPCS parameter settings, we now analyze performance relative to state-of-the-art search algorithms for the JSP. We select two baselines for comparison: Nowicki and Smutnicki's i-TSAB tabu search algorithm [14] and Zhang et al.'s hybrid tabu search / simulated annealing algorithm [25]. The i-TSAB algorithm represents the state-of-the-art from 2003 onwards, while Zhang et al.'s algorithm is a recently introduced competitor. A single "winner" is not easily determined, lacking carefully controlled experiments and availability of the source code of the two algorithms. However, it is clear from published MRE performance analysis that these two algorithms are superior to all predecessors. Finally, we do not compare the performance of our hybrid with that of previously published CP algorithms, e.g., [16], as those algorithms have not historically proved competitive on the standard JSP; to the best of our knowledge, SGMPCS is the best-performing pure CP algorithm for the JSP, as reported in [3].

As indicated previously in Section 5.1, we consider the performance of our hybrid algorithm obtained with the best overall *mean* performance, obtained with $|E| = 8$, a fixed fail limit of 500, and the local upper bound. While it may be argued that the comparison should be based on the mean MRE obtained across all parameter settings, Nowicki and Smutnicki document significant parameter tuning in the development of i-TSAB, and Zhang et al. undoubtedly performed similar experimentation, although it is not explicitly documented in [25]. The MRE performance statistics for the two comparative baselines and our hybrid algorithm are shown in Table 2; in addition, we compute the MRE for the best-known solutions recorded in [20] as of November 30, 2007. Unfortunately, Nowicki and Smutnicki [14] only report results for a single run of i-TSAB, complicating interpretation. Absent a rigorous alternative, we treat the corresponding MRE results as representative of mean i-TSAB performance. The Zhang et al. statistics are taken over 10 independent runs of their algorithm on each problem instance. Without the actual sample populations, it is not possible to make statistical inferences regarding the relative superiority of the Zhang et al. algorithm and our hybrid algorithm. Consequently, we proceed with a qualitative analysis.

First, we compare the performance of our hybrid with that of i-TSAB. On all but the ta21-30 problem group, the hybrid outperforms i-TSAB in terms of mean MRE. Overall, the hybrid outperforms i-TSAB by 0.12% in terms of mean MRE; again, we are treating the individual i-TSAB samples as representative of mean performance. While the percentage advantage is small in absolute terms, we observe that due to the difficulty of these instances, apparently small differences have historically differentiated state-of-the-art algorithms from second-tier competitors. Although we cannot rigorously determine whether our hybrid performance dominates that of i-TSAB, it is clear that the performance is, *at a minimum*, indistinguishable.

Next, we compare the performance of our hybrid with that of Zhang et al.'s algorithm, hereafter referred to simply as Zhang's algorithm. In terms of mean MRE, the hybrid algorithm dominates the Zhang algorithm both overall and in each problem subgroup; overall, the advantage is 0.30%. In terms of best MRE, each algorithm dominates on two of the four problem groups, with Zhang holding a slight 0.02% advantage overall. Of particular interest is the excellent worst MRE performance of our hybrid algorithm. On two of the problem groups, the worst MRE of the hybrid is better than the *mean* MRE of the Zhang algorithm. Overall, the hybrid worst MRE performance is only slightly worse that the Zhang mean MRE performance, with a difference of only 0.04%. Clearly, a significant advantage of our hybrid algorithm is the consistency of the state-of-the-art performance, which is often elusive (e.g., in the case of the Zhang algorithm) on very difficult benchmark problems.

A major issue in comparative assessment of state-of-the-art algorithms for the JSP involves quantification of computational effort. In addition to issues involving the use of disparate computing hardware, software engineering decisions and coding skill make such comparisons notoriously problematic. We do not address these issues here. Rather, we observe that from analyses of published performance reports [14,25], all three test algorithms were executed on modern computing hardware (Pentium III or greater) and the allocated run-times on the larger problem instances were all within a factor of three.

Table 3. The makespan of new best-known solutions identified by the hybrid i-STS / SGMPCS algorithm for Taillard's benchmark problems

Instance	Prev. Best-Known	New Best-Known	Instance	Prev. Best-Known	New Best-Known
ta11	1359	1357	ta19	1335	1332
ta21	1644	1643	ta24	1646	1645
ta32	1795	1794	ta34	1829	1828
ta40	1674	1671	ta41	2018	2006
ta46	2015	2011	ta47	1903	1899
ta49	1967	1966	ta50	1926	1924

Finally, an obvious question is: Does our hybrid outperform the basic i-STS algorithm? In other words, does the premature termination of i-STS followed by SGMPCS outperform the full-length i-STS algorithm. Drawing from our previously analysis of i-STS [9], the best and mean MREs of i-STS are respectively 3.30% and 3.55%. From Table 2, this represents an under-performance of of 0.17% and 0.18% for best and worst MRE, respectively, relative to our hybrid algorithm. Such large differences provide very strong evidence that our hybrid algorithm significantly outperforms the original i-STS baseline, as is confirmed by subsequent non-parametric two-sample tests.

5.3 Best-Known Upper Bounds

Given the strong performance of our hybrid i-STS / SGMPCS algorithm, it is worth noting that various runs, under various parameterizations of SGMPCS, yielded a remarkable twelve new best-known solutions to Taillard's benchmark instances. Although our main research goal is not to enter "horse-race" competitions of the type that are particularly common in Operations Research [8], the ability of an algorithm to establish new best-known solutions in a given domain is a common (albeit heuristic, because it fails to account for factors such as run-time, coding ability, machine, and related factors) benchmark for establishing the state-of-the-art in performance. At the very least, the ability of an algorithm to establish new best-known solutions with reasonable computing effort provides strong evidence of general effectiveness. Consequently, we record both the previous and our newly obtained best-known solutions to Taillard's benchmark instances in Table 3. Of particular note is the ability of our algorithm to establish new best-knowns for five of the ten 30×20 instances, which are among the most difficult JSP benchmarks – especially given that we did not scale allocated CPU time in proportion to problem instance size.

For the single parameterization of SGMPCS used in Table 2 ($|E| = 8$, Fixed 500, local upper bound), Table 4 indicates the number of problem instances in each group for which the best solution found by the parameterization over its 10 runs is better than, equals, or is worse than the best-known solutions. The best-known solutions are the lowest makespans in Taillard's table [20] and Zhang et al's results [25]. As can be observed, this single parameterization of the hybrid algorithm is able to meet or improve upon the current best known solutions in 33 of the 40 instances. This is an impressive result given that the best-known solutions are the best solutions found by a wide variety of algorithms rather than those of a single algorithm.

Table 4. The number of instances in each group for which the best solution found by the hybrid (with parameters $|E| = 8$, Fixed 500, local upper bound) is better than, equal to, or worse than the current best known. Each instance group contains 10 instances.

Instance Group	# New Best	# Equal Best Known	# Worse
ta11-20	0	10	0
ta21-30	1	6	3
ta31-40	3	7	0
ta41-50	5	1	4
Overall	9	24	7

Table 5. The number of runs (out of 10) for which the hybrid algorithm found and proved the optimal solution. Note that no use is made of existing lower bounds for these proofs.

Hybrid Parameters	ta14	ta31	ta34	ta35	ta36	ta38	ta39
$\|E\| = 8$, Fixed 500, Local	10	10	1	0	10	4	10
$\|E\| = 8$, Luby 200, Local	10	10	1	1	10	2	9

5.4 On Proving Optimality

Unlike previous state-of-the-art algorithms for JSP, our hybrid (when using the Luby bound) is a complete algorithm. It is therefore possible to find and prove an optimal solution directly rather than based on previously known lower bounds. Even when using a fixed fail limit, it may be possible to find a proof of optimality on some instances.

Table 5 displays the number of runs (out of 10) for which two parameterizations of SGMPCS were able to find and prove optimal solutions. That is, in each case, an individual tree search exhausted the search space without reaching its fail limit. The other parameterizations had similar performance. Again, we observe relatively consistent performance in proving optimality across different runs of the same parameterization and instance. Of particular note is ta34 as, according to [20], this is the first time that the optimal solution for this problem has been found and proved.

6 Discussion

Our strong empirical results are somewhat surprising given the very simple nature of the hybrid algorithm. We have achieved state-of-the-art results by running two strong, but not necessarily state-of-the-art, algorithms in sequence, using the best solutions found by the first algorithm to initialize the second. While the underlying algorithms are complex, the effort to hybridize them consisted almost entirely of writing a translation between the solution representations of the two algorithms.

As noted in Section 1, this work was motivated by non-formalized ideas about differences in the search styles of the two foundational algorithms. While our results are positive, it is important to note that this paper *does not test* these ideas. The ideas need to be examined through careful formalization and experimental design. It is possible that there are other underlying explanations of our results, unrelated to these motivations. More rigorous testing of these ideas will be the focus on follow-on research.

There are a number of other interesting observations arising from this study, which we address in the balance of this section.

The Performance of $|E| = 1$*:* The strong performance of the hybrid with $|E| = 1$ is quite surprising, though it is consistent with previous SGMPCS results [3]. When $|E| = 1$, the best solution found by i-STS is used as the only elite solution for SGMPCS. We would expect that such an undiversified search would result in high variance: if we were unlucky, the elite solution would not be in the vicinity of a better solution. However, the results in Table 1 show that the worst MRE for $|E| = 1$ is not significantly worse than that for the other values of $|E|$. We believe that for an explanation we will need to understand more about both the distribution of solutions in the JSP search space and the behavior of SGMPCS in searching that space.

The Impact of Solution Guidance: The main innovation in SGMPCS is the use of elite solutions to guide constructive search. To evaluate the importance of this aspect of the algorithm, we also ran the hybrid but replaced SGMPCS with chronological backtracking and randomized restart. In both cases, we ran i-STS for 5M iterations, as above. In the case of chronological backtracking, one less than the cost of the best solution found by i-STS was used as the upper bound on the cost function and the same randomized heuristics and propagators described above were used. The only differences are that the value ordering was always determined by the min-slack heuristic [18] and there was no restarting of the search (i.e., the fail limit was infinite). For randomized restart, the upper bound on the cost function, the propagators, and the heuristics were identical to that of chronological backtracking. We experimented with the same three fail limit sequences used for SGMPCS.

Overall, performance was poor. Over the 400 runs of chronological backtracking (10 runs per instance), an improvement over the i-STS starting solution was found in only one run. The improvement reduced the makespan by one time-unit. Similarly, over the 1200 runs of randomized restart (10 runs by 3 fail limits by 40 instances), an improvement was only found in 6 runs. Again each of these improvements only reduced the makespan by one time-unit.

The randomized restart results are particularly interesting because the only difference with SGMPCS (with global upper bound) is the value-ordering heuristic. We conclude, therefore, that elite solution guidance is *a* critical component of the hybrid.

7 Conclusions and Future Research Directions

Historically, the performance of constraint programming approaches – despite the availability of strong, domain-specific propagation and heuristic search techniques – has lagged that of local search algorithms on the classical job-shop scheduling problem. We introduced a simple hybrid algorithm that leverages the broad search capabilities of a high-performance tabu search algorithm for the JSP (i-STS) with the domain-specific inference capabilities of the state-of-the-art constraint programming algorithm for the

JSP (SGMPCS). The performance of the hybrid algorithm is at least competitive with the two state-of-the-art algorithms for the JSP: Nowicki and Smutnicki's i-TSAB tabu search algorithm and Zhang et al.'s hybrid tabu search / simulated annealing algorithm. While various factors outside our immediate control prevent us from making a more rigorous and precise statement regarding relative performance, we additionally observe that our hybrid algorithm was able to locate 12 new best-known solutions to Taillard's notoriously difficult benchmark instances, providing additional evidence of the effectiveness of our approach. Further, our hybrid algorithm provides two additional advantages over the i-TSAB and Zhang et al. algorithms. First, we demonstrate that performance is largely insensitive to the choice of the fundamental parameters underlying the algorithm. Second, and perhaps most importantly, the hybrid is able to *consistently* achieve excellent performance, e.g., the worst-case performance is roughly equivalent to the mean performance of the Zhang algorithm.

While this paper focuses on the introduction and analysis of a hybrid algorithm in terms of performance, our original motivation was to better understand why constraint programming algorithms for the JSP – in particular, SGMPCS – generally underperform their local search counterparts. Although it is now clear that SGMPCS has a niche relative to local search in state-of-the-art algorithms for the JSP, we have only begun preliminary investigations into understanding this niche and how SGMPCS exploits it. For example, we have preliminary evidence that SGMPCS acts primarily as an intensification mechanism for the elite solutions generated by i-STS, and is empirically more efficient than tabu search in that role. The insensitivity of SGMPCS performance to parameter settings also raises a number of issues, for example, the need to better understand why elite pool size is not a major factor in CP-based search, while it appears fundamental in local search. Overall, the present contribution establishes the hybrid i-STS / SGMPCS algorithm as an interesting test subject; future research will analyze these and other questions raised by this performance analysis.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada, the Canadian Foundation for Innovation, the Ontario Research Fund, Microway, Inc., and ILOG, S.A.. Sandia is a multipurpose laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000

References

1. Balas, E., Vazacopoulos, A.: Guided local search with shifting bottleneck for job-shop scheduling. Management Science 44(2), 262–275 (1998)
2. Baptiste, P., Le Pape, C., Nuijten, W.: Constraint-based Scheduling. Kluwer Academic Publishers, Dordrecht (2001)
3. Beck, J.C.: Solution-guided multi-point constructive search for job shop scheduling. Journal of Artificial Intelligence Research 29, 49–77 (2007)
4. Beck, J.C., Fox, M.S.: Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics. Artificial Intelligence 117(1), 31–81 (2000)

5. Blażewicz, J., Domschke, W., Pesch, E.: The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research 93(1), 1–33 (1996)
6. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1(2), 117–129 (1976)
7. Glover, F., Laguna, M., Martí, R.: Scatter search and path relinking: Advances and applications. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics, Kluwer Academic Publishers, Dordrecht (2003)
8. Hooker, J.N.: Testing heuristics: We have it all wrong. Journal of Heuristics 1, 33–42 (1996)
9. Howe, A.E., Watson, J.P., Whitley, L.D.: Deconstructing nowicki and smutnicki's i-tsab tabu search algorithm for the job-shop scheduling problem. Computers and Operations Research, Anniversary Focused Issue on Tabu Search 33(9), 2623–2644 (2006)
10. Laborie, P.: Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. Artificial Intelligence 143, 151–188 (2003)
11. Le Pape, C.: Implementation of resource constraints in ILOG Schedule: A library for the development of constraint-based scheduling systems. Intelligent Systems Engineering 3(2), 55–66 (1994)
12. Luby, M., Sinclair, A., Zuckerman, D.: Optimal speedup of Las Vegas algorithms. Information Processing Letters 47, 173–180 (1993)
13. Nowicki, E., Smutnicki, C.: A fast taboo search algorithm for the job shop problem. Management Science 42(6), 797–813 (1996)
14. Nowicki, E., Smutnicki, C.: An advanced tabu search algorithms for the job shop problem. Journal of Scheduling 8(2), 145–159 (2005)
15. Nuijten, W.P.M.: Time and resource constrained scheduling: a constraint satisfaction approach. PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology (1994)
16. Nuijten, W.P.M., Le Pape, C.: Constraint-based job shop scheduling with ILOG Scheduler. Journal of Heuristics 3, 271–286 (1997)
17. Scheduler. ILOG Scheduler 6.5 User's Manual and Reference Manual. ILOG, S.A (2007)
18. Smith, S.F., Cheng, C.C.: Slack-based heuristics for constraint satisfaction scheduling. In: Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI 1993), pp. 139–144 (1993)
19. Taillard, E.D.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285 (1993)
20. Taillard, É.D.: (November 2007), `http://ina.eivd.ch/collaborateurs/etd/default.htm`
21. Taillard, É.D.: Parallel taboo search technique for the jobshop scheduling problem. Technical Report ORWP 89/11, DMA, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland (1989)
22. Watson, J.-P.: On metaheuristic "Failure Modes": A case study in tabu search for job-shop scheduling. In: Proceedings of the Fifth Metaheuristics International Conference (2005)
23. Watson, J.P.: Empirical Modeling and Analysis of Local Search Algorithms for the Job-Shop Scheduling Problem. PhD thesis, Department of Computer Science, Colorado State University (2003)
24. Wu, H., van Beek, P.: On universal restart strategies for backtracking search. In: Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming, pp. 681–695 (2007)
25. Zhang, C.Y., Li, P., Rao, Y., Guan, Z.: A very fast TS/SA algorithm for the job shop scheduling problem. Computers and Operations Research 35(1), 282–294 (2008)