Applying Constraint Satisfaction Techniques to Job Shop Scheduling

Cheng-Chung Cheng and Stephen F. Smith [1]
CMU-RI-TR-95-03

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

January 1995

[1] This research has been sponsored in part by the National Aeronautics and Space Administration, under contract NCC 2-531, by the Advanced Research Projects Agency under contract F30602-90-C-0119 and the CMU Robotics Institute.

Abstract

In this paper, we investigate the applicability of a constraint satisfaction problem solving (CSP) model, recently developed for deadline scheduling, to more commonly studied problems of schedule optimization. Our hypothesis is two-fold: (1) that CSP scheduling techniques provide a basis for developing high-performance approximate solution procedures in optimization contexts, and (2) that the representational assumptions underlying CSP models allow these procedures to naturally accommodate the idiosyncratic constraints that complicate most real-world applications. We focus specifically on the objective criterion of makespan minimization, which has received the most attention within the job shop scheduling literature. We define an extended solution procedure somewhat unconventionally by reformulating the makespan problem as one of solving a series of different but related deadline scheduling problems, and embedding a simple CSP procedure as the subproblem solver. We first present the results of an empirical evaluation of our procedure performed on a range of previously studied benchmark problems. Our procedure is found to provide strong cost/performance, producing solutions competitive with those obtained using recently reported shifting bottleneck search procedures at reduced computational expense. To demonstrate generality, we also consider application of our procedure to a more complicated, multi-product hoist scheduling problem. With only minor adjustments, our procedure is found to significantly outperform previously published procedures for solving this problem across a range of input assumptions.

Contents

List of Figures

List of Tables

1 Introduction

The problem of job shop scheduling to minimize makespan has been a subject of extensive investigation over the years and represents one of the most well-developed areas of deterministic scheduling theory. However, with few exceptions, research on job shop makespan minimization has been restricted to solution of relatively small problems under somewhat idealized representational assumptions. This is due, in large part, to a methodological bias toward development of optimal solution procedures. Our focus in this paper is on techniques for solving large job shop problems. Like several other recent efforts (Adams et al., 1988; Storer et al., 1992; Balas et al., 1993), we give up the guarantee of optimal solutions and instead concentrate on the development of efficient solution procedures that tend to optimize rather well. Our specific objective is to explore the potential of "constraint-posting" scheduling frameworks, which have originated from research in constraint satisfaction problem-solving (CSP) in the field of Artificial Intelligence, as a means for formulating and solving large makespan minimization problems. Generally speaking, CSP models would seem to offer an attractive approach to job shop scheduling problems, since they naturally accommodate more complex problem formulations (e.g., involving bounded-interval separation constraints between job steps, flexible duration constraints, sequence-dependent setups), and support a variety of systematic and local search techniques.

Job shop makespan minimization is a challenging problem. Though efficient, polynomial time solutions have been found for selected, restrictive versions of the problem, (e.g., two jobs, two machines (Jackson, 1956); two machines and unit processing times (Hefetz and Adiri, 1982)), it is well known that general problem formulations are strongly NP-hard. Work on optimal solution procedures for general makespan problems has focused most heavily on the development of implicit enumeration, or branch and bound, algorithms (Balas, 1969; Charlton and Death, 1970; Schrage, 1970; Florian et al., 1971; Lageweg et al., 1977; Barker and McMahon, 1985; Carlier and Pinson, 1989; Applegate and Cook, 1991; Brucker et al., 1992). This line of work has shown steady progress. The notorious 10-job, 10-machine problem originally posed by (Fisher and Thompson, 1963) was finally optimally solved by implicit enumeration in 1989 (Carlier and Pinson, 1989), and a more efficient branch and bound solution to this problem was subsequently reported in (Brucker et al., 1992). Yet, this improved solution procedure was not able to solve larger 20-job, 10-machine problems (running for over 3 days on a Sun 4/20 workstation). As observed by (Lawler et al., 1989), the applicability of implicit enumeration schemes is limited to relatively small problems, and their performance is quite sensitive to particular problem instances and initial upper bound values.

Other research has investigated heuristic approaches. Simple priority dispatching algorithms are the most representative and the most widely used in practical environments. (For a general introduction to priority rules, see (Panwalker and Iskander, 1977).) While dispatching algorithms are extremely fast and easy to implement, there are also drawbacks. The performance of any given rule is typically quite sensitive to problem characteristics, and it is generally quite difficult to find one that dominates in any particular environment. Such procedures are also susceptible to very poor performance in certain circumstances, due to the myopic nature of their decision-making.

With the rapid increase in computing power in recent years, a growing body of research has focused on development of more sophisticated heuristic methods, which incorporate various forms of search and aim at striking a better cost/performance trade-off than the extremes that are provided by dispatch and optimal solution procedures. One notable approach, reported in (Adams et al., 1988; Balas et al., 1993), emphasizes bottleneck tracking as a heuristic methodology for integrating optimal solutions to simpler, one-machine subproblems. A series of shifting bottleneck procedures have been defined which have demonstrated very strong performance on a range of previously published benchmarks, and provide a continuum of increasingly more accurate solution procedures at increasing computational expense. Other work has explored the use of various local search techniques as a basis for approximate solution, including simulated annealing (Matsuo et al., 1988; van Laarhoven et al., 1992), tabu search (Dell'Amico and Trubian, 1991; Taillard, 1993), genetic algorithms (Della Croce et al., 1992), and general neighborhood search (Storer et al., 1992). This work has also produced strong approximate results on previously studied benchmarks; it has also investigated and provided benchmark results for significantly larger makespan problems (Taillard, 1993).

In this paper, we propose a new heuristic procedure for solving job shop makespan minimization problems. Our approach is somewhat unconventional. We start from a base procedure called PCP (Precedence Constraint Posting) (Smith and Cheng, 1993), previously developed for efficient heuristic solution of job shop deadline scheduling problems. We reformulate the makespan problem as one of solving a series of different but related common deadline problems, and define an extended procedure, Multi-PCP, which employs PCP as a subproblem solver. PCP has three attractive properties in this context: (1) it relies on simple, computationally inexpensive CSP heuristics and thus is quite efficient, (2) it performs only a fixed amount of search, making its run time a predictable function of problem size parameters, and (3) the procedure can be straightforwardly generalized to accommodate more complex scheduling problem formulations. Our hypothesis is that CSP scheduling models such as PCP can provide a basis for heuristic makespan minimization procedures that offer good cost/performance and, at the same time, naturally extend to incorporate the more idiosyncratic constraints that must be enforced in many application environments.

To test the viability of our hypothesis, we first analyze the performance of Multi-PCP on previously studied benchmark problems, using the above mentioned shifting bottleneck procedures as a comparative base. On classical (small) benchmark problems, we find Multi-PCP to perform, on average, better than SB1 (the most efficient but least accurate shifting bottleneck procedure) and close to SB3 (more effective and more costly) with computation times equivalent to SB1. Closer analysis of results indicates that the relative performance of Multi-PCP depends significantly on the ratio of number of jobs to number of machines in the input problem; for problems with low job to machine ratios, Multi-PCP is found to consistently outperform SB1, in several cases outperform SB3, and, on a few problems, outperform all shifting bottleneck procedures. Conversely, Multi-PCP is found to be less effective on problems with high job to machine ratios. These performance trends are further confirmed through analysis of SB1 and Multi-PCP on more recently generated (larger) benchmark problems. However, at larger problem size levels, the sensitivity of SB1 to characteristics of specific problem instances also becomes much more apparent; computation times of SB1 are found to vary significantly from problem instance to instance, and the version of SB1 tested was unable to solve all problem instances. The computational cost of Multi-PCP, in contrast, is seen to be quite predictable and consistently much lower within all problem size categories.

To demonstrate the more general applicability of Multi-PCP, we also consider its application to a more idiosyncratic makespan minimization problem: the multi-product, hoist scheduling problem previously studied in (Yih, 1994). This problem is complicated by the need to enforce bounded interval constraints on both operation processing times and the allowable delay time between consecutive job steps, while additionally accounting for the time required to position the hoist for any material transfer. We develop simple extensions to the base PCP procedure to account for sequence-dependent hoist travel times, and introduce a simple problem decomposition scheme to improve the procedure's computational performance. Over a range of system configurations defined by the ratio of mean processing time to hoist speed, Multi-PCP is shown to consistently and significantly outperform Yih's previously reported procedure. Moreover, this performance improvement is seen to be invariant to the degree of flexibility in the manufacturing system, as defined by the tightness of the bounded interval constraints.

The remainder of the paper is presented as follows. In the next section, we first consider formulation of job shop deadline scheduling problem as a CSP and summarize the basic PCP procedure. In Section 3, we define the extended Multi-PCP procedure for makespan minimization. The results of our experimental study with classical job shop scheduling benchmarks are presented in Section 4. In Section 5, we consider application of Multi-PCP to the hoist scheduling problem. Finally, in Section 6, we summarize major points and conclusions.

1. Apply constraint propagation to establish the current set v_d of feasible values for each unassigned variable d;
2. If $v_d = \phi$ for any variable d, backtrack.
3. If no unassigned variables, or no consistent assignments for all variables, quit. Otherwise,
4. Select an unassigned variable d to assign.
5. Select a value from v_d to assign to d.
6. Go to step 1 .

Figure 1: Basic CSP Search Procedure

2 CSP Scheduling Models and the Job Shop Deadline Problem

Constraint satisfaction problem solving (CSP) has long been an area of active research within the field of Artificial Intelligence, and increasingly, CSP models and heuristics have been investigated as a means for solving scheduling problems (Cheng and Smith, 1994; Minton et al., 1992; Muscettola, 1993; Sadeh, 1991; Smith and Cheng, 1993). Generally speaking, a constraint satisfaction problem (CSP) is formulated as a triple $\{V, D, C\}$, where V is a set of decision variables, D a set of domains for the variables in V, and C a set of constraints on two or more variables in V. Basic CSP solution procedures construct solutions through depth first extension of partial assignments, according to the following basic search procedure given in Figure 1. Within this search procedure, step 1 is often referred to as *consistency enforcement*, and steps 3 and 4 are generally referred to as *variable ordering* and *value ordering* respectively. Specific CSP algorithms vary in the type and level of consistency enforcement that is employed (Mackworth, 1977; Haralick and Elliott, 1980), in the mechanisms incorporated for recovering from inconsistent search states (e.g., backjumping (Gaschnig, 1979), dependency-directed backtracking (Stallman and Sussman, 1977), dynamic backtracking (Ginsberg, 1994)), and in the heuristics utilized for variable and value ordering.

CSP scheduling research has generally emphasized development of more specialized algorithmic components, which take advantage of the structure of this particular class of problems. Much of this work has focused on variations of the job shop deadline problem. A job shop deadline problem involves synchronization of the production of n jobs in a facility with m machines, where (1) each job j requires execution of a sequence of operations within a time interval specified by its ready time r_j and deadline d_j, and (2) each operation O_i requires exclusive use of a designated machine M_i for a specified amount of processing time p_i. The objective is to determine a schedule for production that satisfies all temporal and resource capacity constraints. The job shop deadline problem is known to be NP-Complete (Garey and Johnson, 1979).

There are different ways to formulate this problem as a CSP. Most frequently, it has been formulated as a problem of finding a consistent set of start times for each operation of each job. Under this formulation, there are $n \times m$ decision variables, st_i, whose possible values are sets of start times. Most work with this CSP model has concentrated on techniques for exploiting resource capacity analysis as a means for restricting and directing search. "Look-ahead" heuristics for variable and value ordering, based on repeated computation of expected resource demand over time and identification of bottleneck intervals, have been shown to yield significant performance improvements over general CSP heuristics (Sadeh, 1991). Various contention analysis techniques have also been used to enhance consistency checking and early search space pruning capabilities (Sadeh, 1991), and to improve backtracking performance (Xiong et al., 1992). Other work has investigated the use of "iterative repair" search procedures (in contrast to the basic constructive search framework of Figure 1), which start with an infeasible initial solution and attempt to incrementally eliminate conflicts(Minton et al., 1992; Zweben et al., 1990). However, such "repair" techniques have been shown to perform rather poorly in comparison to constructive approaches (Muscettola, 1993).

The PCP scheduling framework (Smith and Cheng, 1993; Cheng and Smith, 1994) and other recent work in CSP scheduling (Aarts and Smith, 1994; Boddy and Goldman, 1994; Harvey, 1994; Muscettola, 1993) alternatively start from a problem representation akin to a disjunctive graph formulation (Balas, 1969). The problem is instead assumed to be one of establishing sequencing constraints between those operations contending for the same resource. We define a set of decision variables $Ordering_{ij}$ for each (O_i, O_j) such that $M_i = M_j$, which can take on two possible values: $O_i \prec O_j$ or $O_j \prec O_i$. In this case, the search proceeds by incrementally "posting" new precedence relations into an underlying temporal constraint graph and propagating the consequences of each new constraint to verify consistency.

There appear to be several pragmatic advantages to this second approach. By deferring commitment on specific start times until they are forced by problem constraints, a larger set of possible extensions is retained, reducing the likelihood of arriving at an inconsistent state. Likewise, from a solution robustness perspective, precise start time decisions are delayed (if possible) until needed at execution time. Finally, formulation as an ordering problem provides a more convenient search space in which to operate. The basic insight underlying PCP is that the search benefits provided by look-ahead analysis of resource contention over time can be obtained through much simpler, local analysis of sequencing flexibility. In (Smith and Cheng, 1993), a configuration of variable and value ordering heuristics based on measures of temporal slack are shown to yield very competitive problem solving performance to currently dominant contention-based approaches (Sadeh, 1991; Muscettola, 1993) at a small fraction of the computational cost.

2.1 Problem Representation

The PCP scheduling model can be formalized more precisely as a type of *general temporal constraint network (GTCN)* (Meiri, 1991). In brief, a GTCN T consists of a set of variables $\{X_1, ..., X_n\}$ with continuous domains, and a set of unary or binary constraints. Each variable represents a specific temporal object, either a time point (e.g., a start time st_i or an end time et_i) or an interval (e.g. an operation O_i). A constraint C may be qualitative or metric.

A qualitative constraint C is represented by a disjunction $(X_i\, q_1\, X_j) \vee ... \vee (X_i\, q_k\, X_j)$, alternatively expressed as a relation set $X_i\, \{q_1, ..., q_k\}\, X_j$, where q_i represents a *basic qualitative constraint*. Three types of basic qualitative constraints are allowed:

1. interval to interval constraints - The GTCN definition of (Meiri, 1991) includes Allen's 13 basic temporal relations (Allen, 1983): *before, after, meets, met-by, overlaps, overlapped-by, during, contains, starts, started-by, finishes, finished-by,* and *equal*. For convenience, we additionally include the relations *before-or-meets* and *after-or-met-by*, which represent the union of relation pairs (*before, meets*) and (*after, met-by*) respectively (Bell, 1989).

2. point to point constraints - The relations identified in (Vilain and Kautz, 1986), denoted by the set $\{<, =, >\}$, are allowable here.

3. point to interval or interval to point constraints - In this case, the 10 relations defined in (Ladkin and Maddux, 1989) are specifiable, including *before, starts, during, finishes, after*, and their inverses.

A metric constraint C is represented by a set of intervals $\{I_1, ..., I_k\} = \{[a_1, b_1], ..., [a_k, b_k]\}$. Two types of metric constraints are specifiable. A unary constraint C_i on point X_i restricts X_i's domain to a given set of intervals, i.e. $(X_i \in I_1) \vee ... \vee (X_i \in I_k)$. A binary constraint C_{ij} between points X_i and X_j restricts the feasible values for the distance $X_j - X_i$, i.e., $(X_j - X_i \in I_1) \vee ... \vee (X_j - X_i \in I_k)$. A special time point X_0 can be introduced to represent the "origin". Since all times are relative to X_0, each unary constraint C_i can be treated as a binary constraint C_{0i}.[1]

[1] In Section 5, we extend the above definition of qualitative, interval to interval constraints to incorporate metric quantifiers, which is necessary to model sequence-dependent setup times. We ignore this complication for now to simplify presentation of the basic PCP procedure.

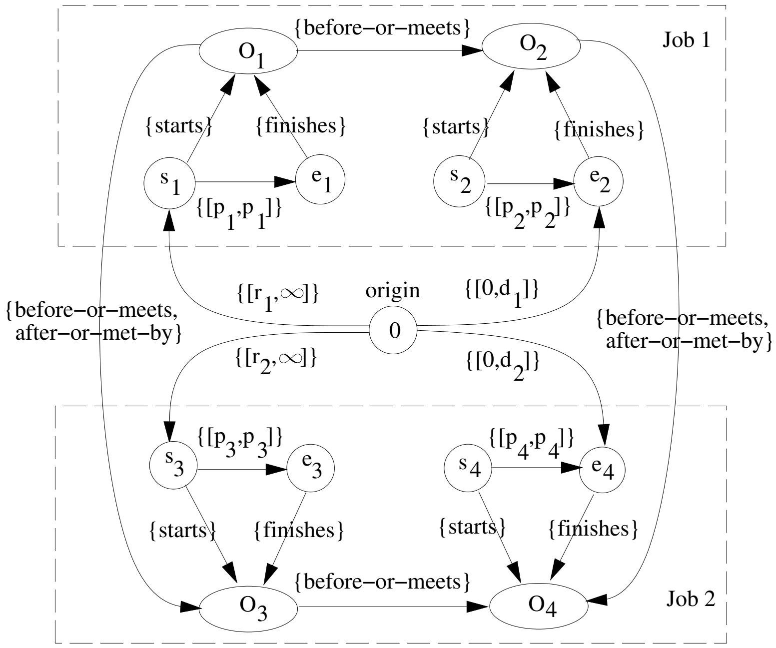

Figure 2: Constraint Graph for simple 2 job, 2 machine problem

A GTCN forms a *directed constraint graph*, where nodes represent variables, and a edge $i \rightarrow j$ indicates that a constraint C_{ij} between variables X_i and X_j is specified. We say a tuple $X = (x_1, ..., x_n)$ is a *solution* if X satisfies all qualitative and metric constraints. A network is *consistent* if there exists at least one solution. Figure 2 depicts the constraint graph for a simple 2 job, 2 machine deadline scheduling problem.

An enumerative scheme for solving a GTCN is given in (Meiri, 1991). Let a *labeling* of a general temporal constraint network, T, be a selection of a single disjunct (relation or interval) from each constraint specified in T. In the graph of Figure 2 there are 4 possible labelings, owing to the {*before-or-meets, after-or-met-by*} relation sets introduced to avoid resource contention between operation pairs (O_1, O_2) and (O_3, O_4). Since any basic qualitative constraint can be translated into at most four metric constraints (Kautz and Ladkin, 1991) (e.g., O_i *before-or-meets* O_j translates to $et_i \leq st_j$), any labeling of T defines a Simple Temporal Problem (STP) network - a metric network containing only single interval constraints (Dechter et al., 1991). T will be consistent if and only if there exists a labeling whose associated STP is consistent.

For any STP network, we can define a directed edge-weighted graph of time points, G_d, called a *distance graph*. An STP is consistent if and only if the corresponding distance-graph G_d has no negative weight cycles. The *minimal network* of the STP can be specified by a complete directed graph, called the *d-graph*, where each edge, $i \rightarrow j$, is labeled by the shortest path length, sp_{ij}, from point i to point j in G_d (Dechter et al., 1991). An STP network can be solved in $O(n^3)$ time by the Floyd-Warshall's all-pairs shortest-paths algorithm, where n is the number of variables in the STP network.

Thus, a simple, complete procedure for solving a GTCN is to enumerate all labelings, solve each corresponding STP and combine results. We can increase the efficiency of this enumeration procedure by running a backtracking search over a *meta-CSP* network, whose variables correspond to arcs in the GTCN that can be labeled in more than one way and whose domains are simply the set of possible labelings. In the case of the deadline scheduling problem, this leads to the set of decision variables $V = \{Ordering_{ij}\}$ previously identified, and a worst case complexity of $\mathcal{O}(n^3 2^{|V|})$.

2.2 The PCP Procedure

The PCP scheduling model (Smith and Cheng, 1993; Cheng and Smith, 1994) augments this basic backtracking search procedure to incorporate simple analysis of the temporal flexibility associated with each sequencing decision that must be made. This analysis is utilized in two ways:

1. to specify dominance conditions that allow identification of unconditional decisions and early search space pruning, and
2. to provide heuristic guidance for variable and value ordering.

Each of these extensions is summarized below.

Specification and use of dominance conditions in PCP derives directly from the concept of *Constraint-Based Analysis* (CBA) originally developed in (Erschler et al., 1976; Erschler et al., 1980). This work utilized calculations of the temporal slack associated with an unordered operation pair to distinguish among cases where neither ordering alternative, just one ordering alternative, or either alternative remains feasible. For example, if $slack(O_i \prec O_j) = lft_j - est_i - (p_i + p_j) < 0$ then O_i cannot be sequenced before O_j. These conditions are applied to detect and post any "forced" sequencing constraints at each step of the search, and to detect inconsistent solution states.

In (Cheng and Smith, 1994), these dominance conditions are generalized to account for the wider range of constraints that are specifiable in a GTCN. Suppose $Ordering_{ij}$ is a currently unassigned variable in the meta-CSP network, and consider the d-graph associated with the current partial solution. Let s_i, e_i, s_j, and e_j be the start and end points respectively of operations O_i and O_j, and further assume sp_{ij} is the shortest path length from e_i to s_j and sp_{ji} is the shortest path length from e_j to s_i. Then, four mutually exclusive cases can be identified:

Case 1. If $sp_{ij} \geq 0$ and $sp_{ji} < 0$, then $O_i \prec O_j$ must be selected.

Case 2. If $sp_{ji} \geq 0$ and $sp_{ij} < 0$, then $O_j \prec O_i$ must be selected.

Case 3. If $sp_{ji} < 0$ and $sp_{ij} < 0$, then the partial solution is inconsistent.

Case 4. If $sp_{ji} \geq 0$ and $sp_{ij} \geq 0$, then either ordering relation is still possible.

We note that the "slack-based" dominance conditions of (Erschler et al., 1976) represent a special case of the above conditions; under classical job shop scheduling assumptions (i.e., fixed processing times, simple job precedence constraints) $slack(O_i \prec O_j) = sp_{ij}$. However, many practical scheduling problems require satisfaction of more complex temporal constraints. For instance, manufacturing processes sometimes place limits on the amount of time that can elapse between consecutive job steps - e.g., if metal is heated for subsequent shaping, then shaping must occur before the metal cools. Similarly, a chemical bath operation may not necessitate a fixed amount of time but rather require a minimum processing time to be productive and a maximum time to avoid damage. Under such more complex modeling assumptions, shortest path information provides stronger dominance criteria.

The second distinguishing aspect of PCP is its use of sequencing flexibility analysis for variable and value ordering, which dictates how the search should proceed in the undecided states (case 4 above). Intuitively, in situations where several $Ordering_{ij}$ decisions remain to be made, each with both possibilities still open, we would like to focus attention on the decision that has the least amount of sequencing flexibility. Since the posting of any precedence constraint is only likely to further reduce possibilities for sequencing other operation pairs, delaying the currently most constrained decision will only increase chances of arriving at an infeasible solution state. With respect to making the selected ordering decision, we intuitively prefer the ordering relation that leaves the search with the most degrees of freedom.

One very simple estimate of the sequencing flexibility associated with a given $Ordering_{ij}$ is the minimum shortest path length, $\omega_{ij} = min(sp_{ij}, sp_{ji})$, which gives rise to a variable ordering heuristic that selects the $Ordering_{ij}$ with the minimum ω_{ij}. This heuristic makes reasonable sense; at each step, the decision which is closest to becoming forced is taken. However, its exclusive reliance on ω_{ij} values can lead to problems. Consider two ordering decisions $Ordering_{ij}$ with associated shortest path lengths $sp_{ij} = 3$ and $sp_{ji} = 100$, and $Ordering_{kl}$ with $sp_{kl} = 4$ and $sp_{lk} = 4$. In this case, there are only limited possibilities for feasibly resolving $Ordering_{kl}$ and deferring this decision may well eliminate them, while a feasible assignment to $Ordering_{ij}$ is not really in any jeopardy.

To hedge against these situations, PCP instead bases variable ordering decisions on a slightly more complex notion of *biased* shortest path length. Specifically, $bsp_{ij} = sp_{ij}/\sqrt{S}$ and $bsp_{ji} = sp_{ji}/\sqrt{S}$ are computed, where $S = \min\{sp_{ij}, sp_{ji}\}/\max\{sp_{ij}, sp_{ji}\}$ estimates the degree of similarity between the two values sp_{ij} and sp_{ji}. The sequencing flexibility associated with a given decision $Ordering_{ij}$ is redefined to be $\omega_{ij} = min(bsp_{ij}, bsp_{ji})$, and the decision selected during variable ordering is the decision with the minimum ω_{ij}. The value ordering heuristic utilized in PCP simply selects the ordering relation implied by $max(bsp_{ij}, bsp_{ji})$, i.e. the sequencing constraint that retains the most temporal flexibility is posted.

2.3 More Efficient, Approximate Procedures

The dominance conditions and variable/value ordering heuristics that distinguish the basic PCP procedure do not, of course, change the exponential worst case behavior of the backtracking search required to guarantee completeness. Given our pragmatic interest in solving large problems, we thus introduce two less-costly, approximate solution procedures for later use. The first variant is simply defined as a backtrack-free version of the basic PCP procedure. In particular, total reliance is placed on the ability of the search to move directly to a feasible solution; if Case 3 above is ever encountered (i.e., no feasible ordering for a given ordering decision), the search simply terminates in failure (and does not produce a solution). The effectiveness of this partial solution procedure, which we will refer to as "Simple PCP" below, was demonstrated in (Smith and Cheng, 1993) on a set of previously published CSP scheduling benchmark problems.

We also define a second variant, referred to below as "Simple PCP with Relaxation" which extends Simple PCP in the following manner. Whenever an ordering decision is recognized as Case 3, the unresolvable decision is set aside, and the search is allowed to proceed with other, still resolvable ordering decisions. Once all feasibly resolvable decisions have been made, the set U of unresolvable (Case 3) decisions is then reconsidered. For each $Ordering_{ij}$ in U, deadlines d_i and d_j are relaxed (increased) by $|max(sp_{ij}, sp_{ji})|$ and the corresponding precedence relation (which is now feasible) is posted. This second approximate procedure thus always produces a solution, albeit one that may not satisfy all original problem constraints. Both approximate procedures can be seen to have worst case time complexity of $\mathcal{O}n^3|V|$, where $|V|$ is the number of ordering decisions that must be made.

3 MULTI-PCP

While constraint satisfaction scheduling procedures such as PCP have been effectively applied to complex deadline scheduling problems, their applicability to more commonly studied problems of schedule optimization is not obvious. In this section, we focus specifically on the problem of makespan minimization and propose one possible approach to incorporating these techniques.

Our approach is motivated by the concept of problem duality exploited in the MULTIFIT algorithm (Coffman et al., 1978) in the context of multiprocessor scheduling. Suppose that we are given an instance of a makespan problem, denoted by $\Pi_M(I)$ where

I represents the problem data associated with this problem instance. If we know the minimum makespan for $\Pi_M(I)$ to be C^*_{max}, then we can reduce $\Pi_M(I)$ to a special deadline problem $\Pi_D(I,d)$, where each job is assigned a 0 ready time and a common deadline d, with $d = C^*_{max}$. For any $d \geq C^*_{max}$, we are assured that a feasible solution to $\Pi_D(I,d)$ exists. More important, C^*_{max} defines a unique common deadline such that for $d < C^*_{max}$, $\Pi_D(I,d)$ has no feasible solution. This dual relationship between problems $\Pi_M(I)$ and $\Pi_D(I,d)$ implies that the makespan problem $\Pi_M(I)$ can be reformulated as a problem of finding the smallest common deadline, d_{min}, for which $\Pi_D(I,d)$ has a feasible solution.

Given an algorithm for optimally solving the deadline problem $\Pi_D(I,d)$, it is straightforward to construct an search procedure for determining d_{min} (and its associated schedule). We start with known upper and lower bounds d_U and d_L on the common deadline d_{min}; at each step, we attempt to solve $\Pi_D(I,d)$ for $d = (d_U + d_L)/2$. If a feasible schedule is found, d_U becomes d; otherwise, d_L becomes d. We continue the search until $d_U = d_L$, retaining the schedule with the best makespan as we go.

There is a complication, however, in utilizing this binary search procedure in conjunction with a heuristic deadline scheduling procedure. In particular, the search may fail to yield the best solution if the deadline scheduling procedure does not ensure *monotonicity* in solution results across an interval of common deadlines. This property implies that if a feasible solution cannot be found for a given common deadline d_1, then a solution will also not be found for any common deadline $d_2 < d_1$, and likewise if a solution is found for a given d_1, then a solution will also be found for any $d_2 > d_1$. It is not difficult to construct examples which demonstrate that neither of the simple, one-pass PCP procedures defined in Section 2.3 possess this property, and consequently the assumptions underlying use of binary search are no longer valid. For this reason, we instead define our extended makespan minimization procedure in terms of a more conventional k-iteration search; the approximate PCP procedure (either variant) is applied k times with different common deadlines evenly distributed between d_L and d_U. While k-iteration search obviously also provides no guarantee of finding the optimal solution, empirical analysis has indicated that, with proper selection of k, use of k-iteration search leads to consistently better makespan minimization performance.

The only remaining issue concerns initial establishment of upper and lower bounds on d_{min}. A lower bound d_L is provided by the procedure originally described in (Florian et al., 1971), where each machine is sequenced independently in order of earliest operation start times and the maximum job completion time is then selected. An upper bound d_U can be obtained through application of one or more priority dispatch rules. In the experiments reported below, a set of six priority rules - SPT, LPT, LFT, EFT, MOR, and LOR - were applied, taking the best makespan generated as d_U. The complete algorithm, referred to as MULTI-PCP, is given in Figure 3.

1. Compute upper bound d_U and lower bound d_L.
   Set $Best\_Makespan = d_U$.
   Set $I = 0$.

2. If $I \geq k$, stop.
   Otherwise, set common deadline $d = d_U - I(d_U - d_L)/k$.

3. Apply PCP (either variant) and compute the makespan M.
   If $M < Best\_Makespan$, set $Best\_Makespan = M$.
   If $Best\_Makespan = d_L$, stop. We have the optimal solution.
   Otherwise, set $I = I + 1$. and go to 2.

Figure 3: The Multi-PCP procedure

4 A Benchmark Performance Study

In this section, we empirically analyze the performance of Multi-PCP on two sets of job shop scheduling benchmark problems previously studied within the literature. The first problem set, referred to below as the "small" problem set, consists of 39 job shop problems with sizes varying from 6-job by 6-machine to 15-job by 15-machine. The first three problems, Mt06, Mt10, and Mt20, are the long standing problems of (Fisher and Thompson, 1963). The remainder are taken from the 40 problems originally created by (Lawrence, 1984); of these 40 problems, we include only the 36 problems for which optimal solutions have been obtained. The second set of benchmark problems, which we designate as the "large" problem set, are the problems more recently defined by (Taillard, 1993). This set consists of 80 larger job shop problems with sizes ranging from 15-job by 15-machine to 100-job by 20-machine. For each problem in this set, Taillard reported the "best solution" obtained with a tabu search procedure that was run for extended time intervals.

We take as a principal comparative base, the shifting bottleneck family of procedures, SB1, SB3 and SB4 (Adams et al., 1988; Balas et al., 1993), which provides a series of increasingly more accurate approximate procedures for makespan minimization at increasingly greater computational expense. On the small problem set, we compare the performance of each of these procedures and Multi-PCP in terms of two measures: % deviation of generated solutions from the optimal solution and amount of computation time required for solution. Results for SB1 were obtained on a Sun SPARC 10 workstation using an implementation kindly provided to us by Applegate and Cook (for detail please see (Applegate and Cook, 1991)). Results for SB3 and SB4 were taken from (Balas et al., 1993), with the reported Sun SPARC 330 computation times translated to reflect expected performance on a SPARC 10. Multi-PCP

Table 1: % deviation from optimal solution for Multi-PCP, SB1, SB3, and SB4 across small benchmark problem categories

Job x Machine	Multi-PCP		Multi-PCP w/ Relax		SB1		SB3		SB4	
	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ
mt06	0.00	-	0.00	-	7.27	-	0.00	-	0.00	-
mt10	2.04	-	2.04	-	2.37	-	5.48	-	1.08	-
mt20	8.76	-	2.32	-	5.41	-	2.92	-	2.92	-
10 x 5	1.53	1.58	1.47	1.46	1.59	1.84	1.44	2.05	1.44	2.05
15 x 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20 x 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10 x 10	2.44	1.75	2.44	1.75	4.94	5.32	3.16	2.33	2.25	1.17
15 x 10	4.29	2.34	3.78	1.99	6.34	2.60	2.72	1.91	2.72	1.91
20 x 10	3.12	2.65	3.12	2.65	6.57	7.26	1.37	1.23	0.96	1.26
30 x 10	0.30	0.58	0.30	0.58	0.00	0.00	0.00	0.00	0.00	0.00
15 x 15	4.15	0.73	4.15	0.73	6.16	2.10	3.09	0.92	3.01	0.97
All	1.93	1.20	1.72	1.15	3.01	2.39	1.51	1.05	1.24	0.92

computation times were also obtained on a SPARC 10. All procedures considered were implemented in C. For the large problem set, we alternatively compute the % deviation from the best tabu search solution produced by (Taillard, 1993) as the performance criterion. Cost/performance comparison across these problems is restricted to Multi-PCP and SB1, since results have not been reported for SB3 and SB4 on this problem set.

We consider the performance of two versions of the Multi-PCP procedure, defined by incorporating either Simple PCP or Simple PCP with Relaxation as the base CSP scheduling procedure (see Section 2.3). These two configurations are referred to below as “Multi-PCP” and “Multi-PCP with Relaxation” respectively. In both configurations, the bound k on the number of iterations performed was set to 8 for all runs.

4.1 Computational results on the small benchmark problems

Tables 1 and 2 summarize the performance results obtained on the small benchmark problem set. Performance is indicated individually for the 3 problems of Fisher and Thompson and aggregated according to problem size for the 36 problems of Lawrence. Associated computation times are given in Table 2. Computation times were found to be identical for both Multi-PCP configurations at the level of precision given in Table 2 and are thus listed only once. Detailed results for each individual problem are given in Tables 5, 6, and 7 in Appendix A.

Considering first the performance of the two Multi-PCP configurations tested, we see that augmenting the base PCP procedure to produce ”relaxed” deadline solutions

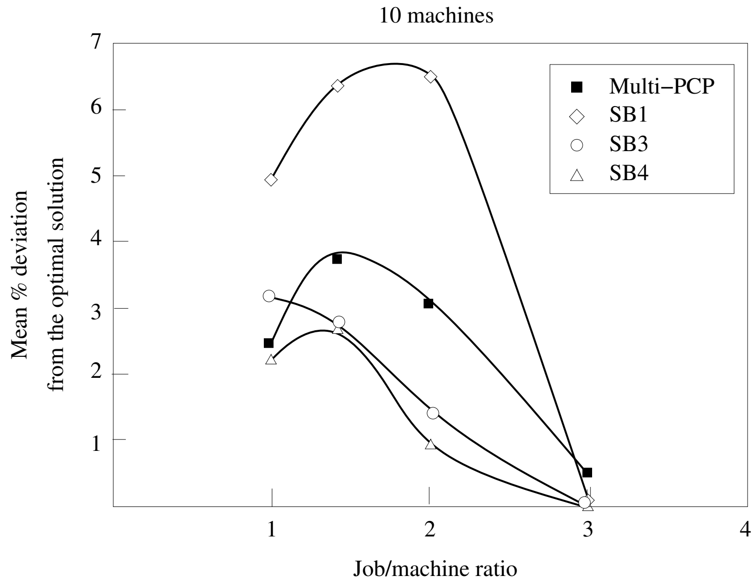

Figure 4: Mean % deviation from optimal solution for increasing job/machine ratio on small benchmark problems

when feasible solutions are not found yielded improved solutions in only a small number of problems. In these isolated cases, however, the improvement provided by the extended procedure was sometimes substantial; for the mt20 problem of Fisher and Thompson, % deviation from the optimum was reduced from 8.76 to 2.32, matching the best solution found for this problem by any of the shifting bottleneck procedures. Since the extended Multi-PCP with Relaxation procedure incurs virtually no additional computational cost, we restrict attention to the results obtained with this configuration in our discussion below.

As shown in Table 1, the makespan minimization performance of Multi-PCP on the small problem set falls within the performance continuum defined by the shifting bottleneck procedures. On average, Multi-PCP is seen to perform better than SB1 and very close to SB3, with SB4 yielding the best overall makespan performance. Relative performance was found to vary across different problem subsets. On the three classic Fisher and Thompson problems, Multi-PCP found equivalent or better solutions than both SB1 and SB3 in all cases, and failed to match the performance of SB4 in just one case. There is little difference in performance on the very small, 6-machine problems; all procedures produce optimal or near optimal solutions in these problem categories.

Consideration of results on the larger, 10-machine problem categories reveals perhaps the most significant comparative performance trend. These results are graphically depicted in Figure 4 in terms of increasing ratio of number of jobs to number of machines in the input problem. For problems with low job to machine ratios, Multi-PCP exhibits its strongest comparative performance. In the case of the 10x10 problem

Table 2: CPU time (in seconds) for Multi-PCP, SB1, SB3, and SB4 across small benchmark problem categories

Job x Machine	Multi-PCP		SB1		SB3		SB4	
	mean	σ	mean	σ	mean	σ	mean	σ
mt06	0.05	-	0.12	-	0.78	-	1.45	-
mt10	0.38	-	0.72	-	2.21	-	7.76	-
mt20	1.38	-	0.28	-	1.99	-	3.62	-
10 x 5	0.13	0.09	0.12	0.01	0.40	0.20	0.56	0.28
15 x 5	0.22	0.26	0.15	0.02	0.12	0.04	0.12	0.04
20 x 5	0.07	0.07	0.16	0.03	0.14	0.04	0.14	0.04
10 x 10	0.26	0.03	0.67	0.13	1.61	0.16	3.20	0.25
15 x 10	1.04	0.10	1.20	0.18	3.41	0.98	5.74	2.52
20 x 10	2.51	0.13	1.61	0.26	3.86	2.27	7.50	5.35
30 x 10	4.85	3.96	2.58	0.58	4.13	1.76	4.13	1.76
15 x 15	1.29	0.17	4.55	1.30	13.34	0.49	26.79	1.05
Average	1.19	0.60	1.21	0.31	2.96	0.74	5.29	1.41

category, Multi-PCP performed better on average than both SB1 and SB3, and very close to SB4. Conversely, Multi-PCP was found to be less effective (comparatively) on problems with high job to machine ratios. On the 30x10 problems (which turn out to be the easiest 10-machine problems for all procedures), all three shifting bottleneck procedures were able to obtain optimal solutions, whereas Multi-PCP failed to find the optimum for 2 of the 5 problems in this category (see Table 6).

From a computational perspective, Table 2 shows that the solution times achieved on this problem set by Multi-PCP are comparable overall to those of SB1. Table 2 also confirms the increasingly higher computational demands of SB3 and SB4; on the 15-machine problems, the average solution times for SB3 and SB4 are approximately 10 and 20 times larger (respectively) than those of Multi-PCP.

4.2 Computational results on the large benchmark problems

Table 3 extends the performance comparison of Multi-PCP and SB1 to the larger problem set of Taillard. Again, performance results are summarized by problem size category, in this case relative to the best solutions reported by Taillard's tabu search procedure. Figure 5 graphically depicts average solution performance by increasing job-to-machine ratio as before for both 15-machine and 20-machine problem categories. We note that due to excessive memory requirements, the SB1 implementation was able to solve only 9 of the 10 problems in each of the 50x15 and 50x20 problem sets, and was unable to solve any of the largest 100x20 problems on our Sun SPARC10 workstation. The average performance numbers listed in Table 3 and Figure 5 for these categories

Table 3: % deviation from the best solution for Multi-PCP and SB1 across large benchmark problem categories

Job x Machine	Multi-PCP		Multi-PCP w/ Relax		SB1	
	mean	σ	mean	σ	mean	σ
15 x 15	5.74	0.74	5.57	0.78	9.00	2.05
20 x 15	7.52	1.82	7.27	1.49	10.15	2.14
30 x 15	9.88	2.57	9.65	2.47	8.38	3.06
50 x 15∗	7.39	2.31	7.21	2.26	2.66†	1.57
20 x 20	7.60	1.54	7.33	1.33	9.98	2.29
30 x 20	11.76	2.43	11.64	2.37	13.05	2.55
50 x 20∗	8.77	0.96	8.34	1.06	5.33†	1.82
100 x 20	4.88	1.41	4.88	1.41	- ‡	-
All∗	8.38	1.72	8.14	1.65	8.36	2.21

† SB1 able to solve nine out of ten problems.
‡ SB1 unable to solve any of the 100x20 problems.
∗ Average performance is measured with respect to problems solved by both procedures.

reflect only those problems that were successfully solved. Corresponding average computation times by problem category are given in Table 4. Detailed performance on each individual problem is given in Tables 8, 9, and 10 in Appendix A.

Ignoring the scalability problems encountered with the tested SB1 implementation, the results at larger problem sizes make much more explicit the comparative performance trends observed at the 10-machine problem level. Multi-PCP is seen to consistently outperform SB1 at low job-to-machine ratios, while the inverse is true at high job-to-machine ratios. Both procedures achieve increasingly better solutions at higher job-machine ratios (consistent with Taillard's observation that these problems are easier), but in no cases does either Multi-PCP or SB1 achieve the best solutions generated by extended Tabu search.

Examination of relative computational costs (Table 4) indicates some additional scalability trends and tradeoffs. Multi-PCP was found to consistently produce solutions in less computation time than SB1; the largest differential (roughly 4 times as fast) was observed in the problem categories with the smallest job-to-machine ratios, and, on average, Multi-PCP obtained solutions in about half as much CPU time as SB1. Multi-PCP was also found to be much more predictable with respect to computational cost. As shown in Table 4, the variance in Multi-PCP solution times across all problem categories was extremely low in comparison to SB1.

It is clear that the simple heuristics embedded in Multi-PCP are too weak to compete with the extended tabu search of Taillard. At the same time, Taillard's best solutions were obtained by running the tabu search procedure 3 to 4 times for a large,

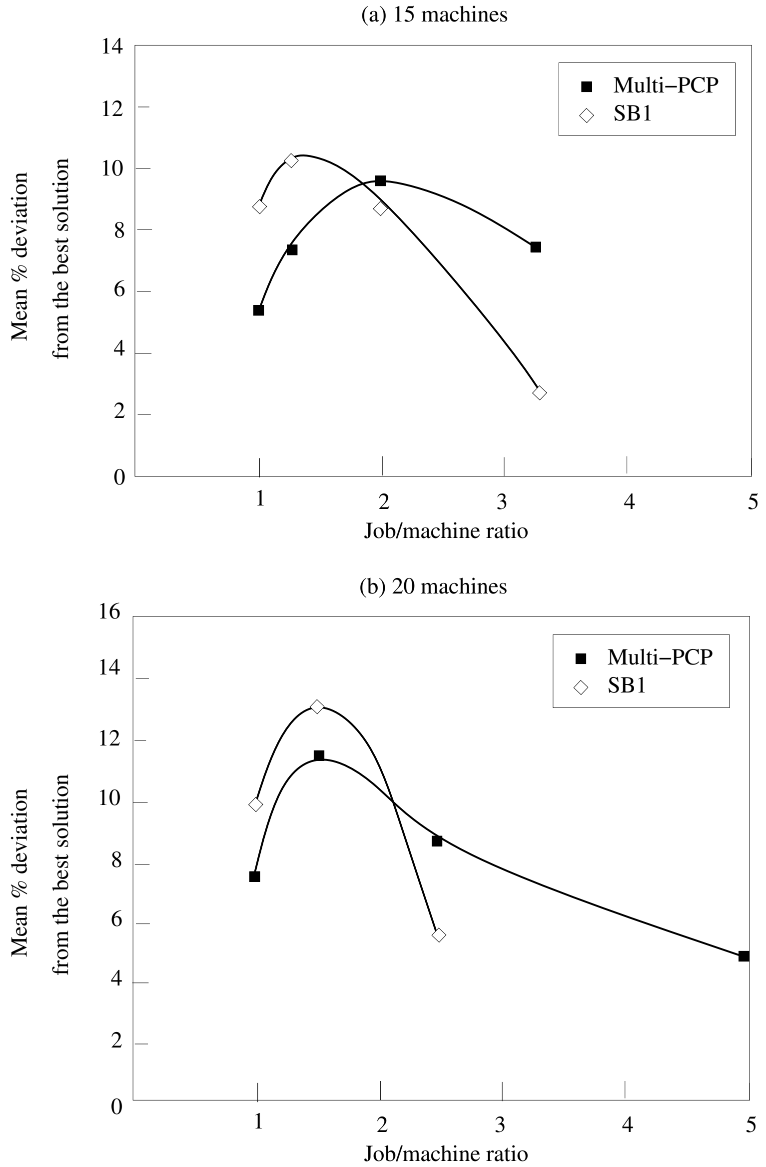

Figure 5: Mean % deviation from best solution for increasing job/machine ratio on large benchmark problems

Table 4: Mean and standard deviation of CPU seconds for procedures, Multi-PCP and SB1, performed on the large benchmark problems

Job x Machine	Multi-PCP mean	Multi-PCP σ	SB1 mean	SB1 σ
15 x 15	1.20	0.08	5.10	1.39
20 x 15	3.42	0.33	7.63	1.06
30 x 15	11.90	0.85	14.68	1.72
50 x 15	68.11	7.12	141.33	104.84
20 x 20	3.73	0.32	15.64	2.91
30 x 20	15.51	0.77	31.58	4.03
50 x 20	94.90	6.28	165.83	94.75
100 x 20	857.36	38.43	-	-

pre-determined number of iterations [2]. Given the additional (but incomplete) computational details provided in (Taillard, 1989) on incremental solution progress and accounting for computing hardware differences, it would appear that Multi-PCP consistently obtains its solution in less time than it takes Taillard's procedure to obtain a solution of comparable quality (particularly on problems with low job-to-machine ratios). Though this computational advantage is lowered by the more efficient tabu procedure recently reported in (Nowicki and Smutnicki, 1994), Multi-PCP would nonetheless seem to offer leverage as a solution seeding mechanism to such iterative search procedures.

5 More Complicated Problem Formulations

The above study relates the performance of Multi-PCP to state-of-the-art makespan minimization procedures; perhaps somewhat surprising, it shows that Multi-PCP's use of a CSP scheduling model in conjunction with fairly simple search control heuristics leads to respectable cost/effectiveness (although certainly not overwhelming all previously reported results). In this section, we consider a complementary issue: the broader applicability of CSP-based solution approaches, and Multi-PCP specifically, to more idiosyncratic problem formulations. As indicated earlier, real-world applications are often complicated by additional temporal synchronization and resource usage constraints, and solution procedures which rely on problem structure that is peculiar to the canonical job shop problem formulation are of little use in such contexts. CSP scheduling models, alternatively, are based on very general representational assumptions and naturally extend to accommodate richer problem formulations. We

[2] Provisions are made for early termination upon detection of an optimal solution. As indicated in Appendix A, optimal solutions were found for 1 of the 10 15x15 problems and for 9 of the 10 problems in each of the 15x50 and 20x100 categories. In all other problems, there was no early termination of the search.

demonstrate this generality below by applying the Multi-PCP procedure to another, less-structured makespan minimization problem: the multi-product version of the hoist scheduling problem studied in (Yih, 1994).

5.1 The Hoist Scheduling Problem

The hoist scheduling problem finds its origin in printed circuit board (PCB) electroplating facilities. In brief, a set J of jobs, $J = \{J_1, ..., J_n\}$ each require a sequence of chemical baths, which take place within a set M of m chemical tanks, $M = \{1, ..., m\}$. Execution of a particular chemical bath operation O_i requires exclusive use of tank m_i. The processing time of any O_i required for a job j is not rigidly fixed; instead there is a designated minimum time, p_i^{min}, that j must stay in the tank for the bath to accomplish its intended effect and a maximum time, p_i^{max}, over which product spoilage occurs. All jobs move through the chemical tanks in the same order, though a given job may require only a subset of the baths and thus "skip" processing in one or more tanks along the way. All job movement through the facility is accomplished via a single material handling hoist, H, which is capable of transporting a job initially into the system from the input buffer, from tank to tank, and finally out of the system into the output buffer. H can grip only a single job at a time, moves between any two adjacent stations (input buffer, tanks, or output buffer) at constant speed s, and has constant loading and unloading speeds, L and U, at any tank or buffer. The facility itself has no internal buffering capability; thus jobs must be moved directly from one tank to the next once they have entered the system. The objective is to maximize facility throughput (or equivalently minimize makespan) subject to these process and resource constraints.

Most previous work in hoist scheduling has considered simplified versions of this problem. The single-product, hoist scheduling problem has received the most attention. In this special case, the problem can be reduced to one of finding a minimum length cycle of hoist operations, which can then be repeated over time; several algorithms for generating optimal (or near-optimal) cyclic schedules have been reported (Phillips and Unger, 1976; Lieberman and Turksen, 1981; Shapiro and Nuttle, 1988; Lei and Wang, 1991; Armstrong et al., 1994). In (Yih and Thesen, 1991; Yih et al., 1993), a hoist scheduling problem involving a multi-product facility is considered, but without permitting variance in job routings (i.e. no tank skipping). To our best knowledge, only (Yih, 1994) has reported procedures for solving the general hoist scheduling problem defined above.

5.2 Representation as a Constraint Graph

The GTCN formalism introduced in Section 2.1 requires only slight extension to model the hoist scheduling problem. Let's first consider representation of the process constraints of any given job j. For each individual operation O_i in j's process sequence, we define three constraint graph nodes: two time points, representing O_i's start and end points, and an interval, representing O_i itself. A given pair of start and end points are related to its corresponding interval through use of the qualitative constraints *starts* and *finishes* respectively. The values ultimately assigned to the time points of any O_i in the constraint graph will represent O_i's scheduled start and finish times.

The duration of a given operation O_i is modeled by specifying a metric constraint between its start and end points. There are two cases, corresponding to the two types of operations that must be interleaved to process any given job. If O_i is a required tank operation, the constraint $\{[p_i^{min}, p_i^{max}]\}$ is specified to enforce minimum and maximum allowable times in tank m_i. If alternatively, O_i is a hoist (transport) operation, then the constraint is instead a function of the distance to be traversed and hoist loading, traveling and unloading speeds. The constraint $\{[mt, mt]\}$ is specified in this case, where $mt = L + s * (destination_i - origin_i) + U$. For convenience, we assume a correspondence between a given tank's index and its location, and assign indices 0 and $m+1$ to the system's input and output buffers respectively to complete this mapping.

The process sequence for any given job j is specified by temporally relating the intervals defined for j's constituent tank and transport operations. Since the end of any given operation O_i must, by definition, coincide with the start of O_{i+1}, the *meets* constraint is used to establish this linkage. O_i *meets* O_{i+1} implies that $et_i = st_{i+1}$. Disjunctive constraints on resource usage are specified as relation sets between operations that require the same resource. Again there are two cases. For tank operations O_i and O_j where $m_i = m_j$, the constraint $\{$*before, after*$\}$ is introduced. Note that the relation set used for this purpose in the canonical job shop problem, $\{$*before-or-meets, after-or-met-by*$\}$, is not correct here, since it is physically impossible to switch between tank operations without intermediate loading and unloading. Synchronization of competing hoist operations is the only remaining issue. In this case, however, basic qualitative relations are insufficient, as they do not allow us to account for the "setup" time that may be required to position the hoist at the loading location.

To overcome this limitation, we extend our representation of qualitative constraints to optionally include a metric quantifier. For our purposes here, it is sufficient to include only the following two extended relations: *before-or-meets[lagtime]* and *after-or-met-by[lagtime]*, where $lagtime \geq 0$ designates a minimum metric separation between the related intervals. Thus, whereas the constraint O_i *before-or-meets* O_j implies $et_i \leq st_j$, the extended constraint O_i *before-or-meets[h_{ij}]* O_j implies $et_i + h_{ij} \leq st_j$. For each pair of hoist operations O_i and O_j belonging to different jobs, we specify the constraint O_i $\{$*before-or-meets[h_{ij}], after-or-met-by[h_{ji}]*$\}$ O_j, where $h_{ij} = s * |destination_i - origin_j|$

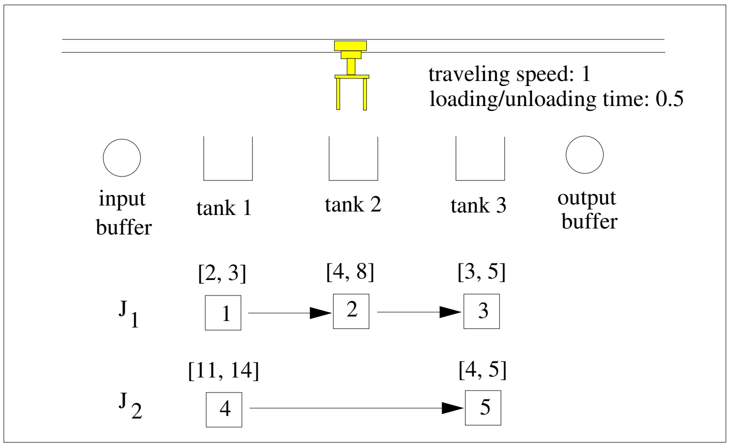

Figure 6: Example 1

and $h_{ji} = s * |destination_j - origin_i|$.

To illustrate, Figure 6 shows a simple example problem (taken from (Yih, 1994)) involving 2 jobs to be processed in a 3 tank system. Each operation is displayed below the tank that is required, and gives its minimum and maximum processing times. The corresponding constraint graph model is given in Figure 7 (with qualitative relations abbreviated as follows: starts (s), finishes(f), meets (m), before (b), after (bi), before-or-meets(bm), after-or-met-by (bmi)).

5.3 PCP Extensions

Given a GTCN model of a hoist problem, we can distinguish two sets of disjunctive constraints that must be resolved during scheduling: those that relate competing tank operations, and those relating pairs of hoist operations. In both cases, the set of possible values that might be selected/assigned differs from the {*before-or-meets, after-or-met-by*} relation set used earlier in the canonical job shop model; and proper treatment of these extended relation sets necessitates some adjustment to the base PCP constraint satisfaction procedure that underlies Multi-PCP.

Recall from Section 2.2, that PCP relies on shortest path lengths as a basic indicator of sequencing flexibility. In essence, the shortest path from et_i to st_j for operations O_i and O_j, designated sp_{ij}, indicates the current maximum feasible separation between these two points. In the case of a {*before-or-meets, after-or-met-by*} relation set, this is a reliable accounting of remaining flexibility. If $sp_{i,j} \geq 0$ then O_i *before-or-meets* O_j is still feasible and vice versa (providing the basis for dominance conditions); if $sp_{ij} < sp_{ji}$ then there is less flexibility remaining to sequence O_i before O_j than there is to do the opposite. However, in the context of sequence-dependent setups, shortest path lengths provide only a partial and distorted estimate of sequencing flexibility.

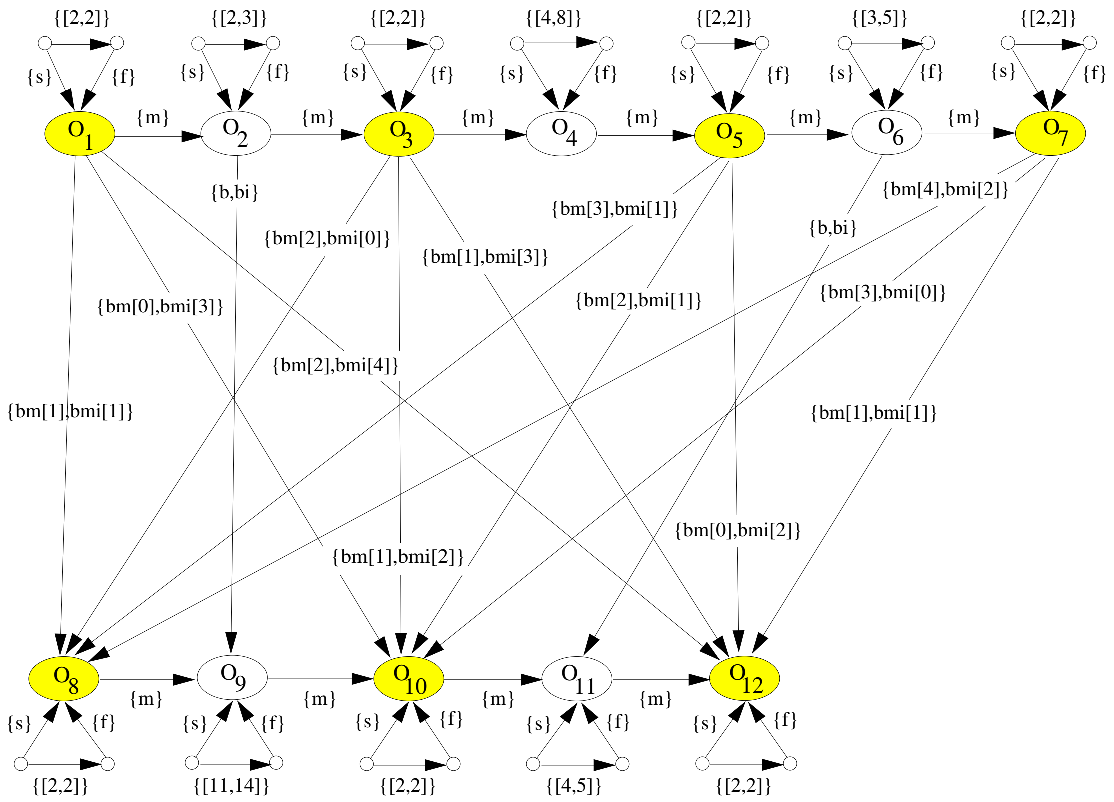

Figure 7: The constraint graph for example 1

To solve this problem, we generalize the basic measure of flexibility in PCP to incorporate sequence-dependent lag times. Assume h_{ij} to be the lag time required if O_i is processed before O_j, and h_{ji} be the lag time required if O_j is processed before O_i (i.e., the constraint specified in the network is O_i {*before-or-meets*[h_{ij}], *after-or-met-by*[h_{ji}]} O_j). We revise the dominance conditions and search control heuristics specified in Section 2.2 by simply substituting the extended calculation $(sp_{ij} - h_{ij})$ for sp_{ij} and, likewise, substituting $(sp_{ji} - h_{ji})$ for sp_{ji}. Note that these revised definitions also accommodate the {*before, after*} relation set required for synchronizing tank operations (in which case, h_{ij} and h_{ji} are both set to the smallest possible temporal increment), as well as the basic {*before-or-meets, after-or-met-by*} relation set (where $h_{ij}, h_{ji} = 0$).

5.4 Performance Results

In this section, we adapt the Multi-PCP procedure defined in Section 3 to incorporate the extensions discussed above, and examine its performance characteristics. We follow the same general experimental design of (Yih, 1994), and similarly consider hoist scheduling in a PCB electroplating facility with 5 chemical tanks. All problems defined consisted of 100 jobs, each with randomly generated routings and tank processing time constraints, and all assumed to be simultaneously available. Since material flow is uni-directional, differences in job routings correspond to which and how many tanks are skipped. Experiments were conducted to evaluate performance along two dimensions relating to facility constraints and operation: first as function of the relative speed of the hoist to mean tank processing time, and second as a function of the degree of flexibility provided by tank processing time constraints. To calibrate results, problems were also solved using the hoist scheduling procedure previously developed by Yih (Yih, 1994). Both procedures were implemented in C and run on a Sun SPARC 10 workstation. Before describing the experiments in more detail and presenting results, we briefly summarize Yih's approach and the hoist scheduling implementation of Multi-PCP.

5.4.1 Yih's Approach

In (Yih, 1994), two heuristic procedures for hoist scheduling are proposed and evaluated. First, to provide a baseline for comparison, a simple "basic algorithm" is specified. In brief, the basic algorithm constructs a feasible schedule one job at a time, by repeatedly extending a partial solution to include the schedule of an additional job. In adding any given job into the schedule, only minimum tank processing times are considered. The job's first tank operation is tentatively scheduled to start as early as possible and subsequent operations in the routing are sequentially checked for feasibility with respect to resource constraints. If a constraint conflict is detected (due either to tank or hoist unavailability), the first operation is delayed by an amount sufficient to avoid the conflict and the feasibility check is repeated. The process terminates as soon as a conflict free start time for the job is found, and the partial solution is extended to include the corresponding schedule.

The second proposed procedure, which we will refer to below as the "Yih94" algorithm, extends the basic algorithm to exploit any scheduling flexibility that is provided by maximum tank processing time constraints. Specifically, the job scheduling cycle is augmented as follows. Upon detection of a hoist availability conflict during the feasibility checking process, a backward pass is first made through the routing to enlarge the durations of previous tank operations where maximum processing time constraints permit. If the conflict is eliminated by this action, then the forward feasibility check is resumed. If not, the job start time is delayed as before. Experimental results reported in (Yih, 1994) indicated that this enhanced algorithm produced schedules with distinguishably lower makespan than those generated by the basic algorithm across a variety of system configurations. Owing to its simple, myopic nature, the Yih algorithm also operates very efficiently; 100 job, 5 tank problems are solved in seconds with our implementation.

5.4.2 A Multi-PCP Implementation for Hoist Scheduling

To adapt the Multi-PCP procedure defined in Section 3 for hoist scheduling, we augment the "Simple PCP" procedure to incorporate the extended dominance conditions and search control heuristics discussed above, and specify new sub-procedures for determining lower and upper bounds, d_L and d_U, on the common deadline interval considered within the k-iteration search. To establish d_L, we simply compute the minimum total required processing time (including hoist operations) for each job and take the maximum. To establish d_U, we run Yih's "basic algorithm" and take the makespan of its result. As in the job shop scheduling experiments, the number of iterations, k, is set to 8 and not varied.

There is one further pragmatic consideration. Given the more complex structure of the constraint graph required to model the hoist scheduling problem, it is not possible to validly exploit the special-case, "slack-based" computations of (Erschler et al., 1976) (see Section 2.2) as a surrogate for shortest-path length calculations within PCP (as it is, for example, in the case of canonical job shop problem formulations).[3] This significantly raises the computational overhead of constraint propagation; preliminary experiments with smaller, 50 job, 5 tank problems were found to yield run times of about 30 minutes.

To provide a more computationally competitive alternative to Yih's algorithm, a simple problem decomposition method, conceived initially by (Ashour, 1967) and extensively studied in (Hirabayashi et al., 1994), is introduced. Under this scheme, a solution is generated by:

1. Partitioning the total set of n jobs into $\lfloor n/b \rfloor$ subproblems with $\geq b$ jobs,
2. Solving each subproblem independently with Multi-PCP to obtain $\lfloor n/b \rfloor$ partial schedules, and
3. Randomly combining the $\lfloor n/b \rfloor$ partial schedules to produce an n job schedule.

In the experiments reported below, a subproblem size of $b = 10$ was uniformly adopted. For these 100 job, 5 tank problems, overall solution time was about 100 seconds.

[3] Interestingly, the need for explicit shortest path length computations is *not* a consequence of the extensions introduced to accommodate sequence-dependent setups, but is due instead the presence of bounded interval constraints between time points, i.e., minimum and maximum processing time constraints. Slack-based computations remain a legitimate computational simplification, for example, in solving canonical job shop problems with sequence-dependent setups.

5.4.3 Sensitivity to Relative Hoist Speed

One dimension along which PCB Electroplating system configurations can vary is in the relative speeds of hoist and tank operations. Accordingly, the sensitivity of scheduling performance to this ratio provides one interesting basis for comparative analysis. We first contrast the performance of Multi-PCP and Yih94 on problem sets designed to vary the ratio $\gamma = \hat{p}^{min}/s$, where $\hat{p}^{min}$ is the mean minimum processing time of tank operations and s is the speed of the hoist in moving between adjacent system locations. A total of 100 problems were randomly generated according to the following experimental design:

- For each problem, each job routing was generated by first choosing some number of tanks to skip from U[0,4], and then selecting (without replacement) which tanks to skip.
- 10 different problem sets, each consisting of 10 problems, were generated by drawing minimum processing times for tank operations from the following 10 uniform distributions: U[5,15], U[15,25], U[25,35], U[35,45], U[45,55], U[55,65], U[65,75], U[75,85], U[85,95], and U[95,105].
- For a given tank operation O_i, p_i^{max} was defined as $\rho \times p_i^{min}$, where ρ is the tolerance factor. ρ was held constant at 1.0 in this experiment.
- For all problems, we assumed a hoist travel speed s of 2 and hoist loading or unloading time of 1.5. Thus, the 10 problem sets reflect γ ratios of 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50.

To characterize results, we take the same approach as in (Yih, 1994); we use the "basic algorithm" to provide a benchmark solution and compute the *improvement rate* provided by both Multi-PCP and Yih94. The precise calculation is:

$$Improvement\ Rate = \frac{Basic_M - X_M}{Basic_M},$$

where $Basic_M$ is the makespan of the basic algorithm's solution, and X_M is the makespan of the solution generated by procedure X, $X \in \{$Multi-PCP, Yih94$\}$.

Figure 8 graphically displays the average results obtained for each problem set with Multi-PCP and Yih94, ordered by increasing γ ratio. Both procedures are seen to generate the largest improvement for values of γ in the range of [10,25], with improvement rates degrading as γ becomes larger or smaller. In the case of Yih94, no improvement is obtained at either of the extreme points tested. Multi-PCP, alternatively, yields an improvement rate of 8% at the smallest γ value, and as γ becomes increasingly larger, its improvement rate stabilizes at about 15%. Across all experiments, Multi-PCP is seen

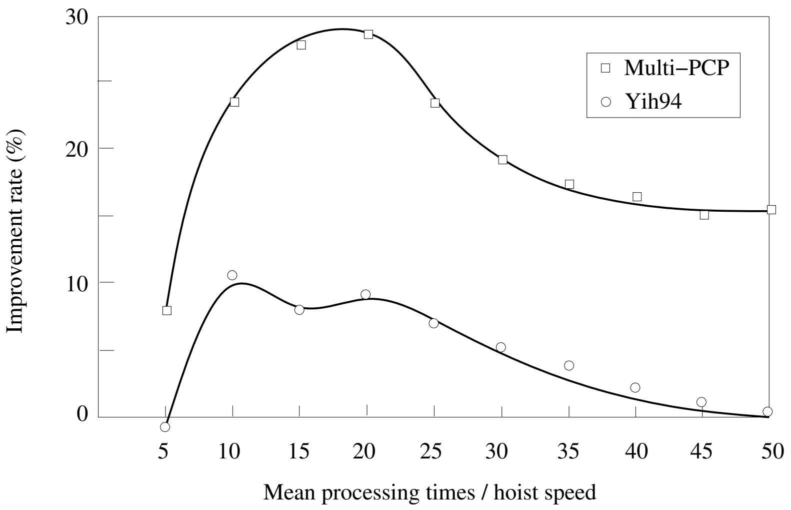

Figure 8: Solution improvement rates for increasing ratio of mean processing time to hoist speed

to produce solutions that, on average, are 15% better (in relation to the benchmark solution) than those obtained with Yih94.

The behavior of Yih94 at extreme γ ratios is predictable. As the hoist is operated at increasingly slower speeds in relation to tank processing time, hoist availability becomes an increasingly dominant constraint. The use of processing time flexibility becomes increasingly insufficient, by itself, as a means of overcoming hoist delays, and the behavior of Yih94 increasingly degenerates to that of the basic algorithm. Alternatively, at increasingly faster relative hoist speeds, hoist availability becomes increasingly less constraining. Opportunities to exploit processing time flexibility correspondingly decrease, again leading to increased reliance on the basic algorithm.

5.4.4 Sensitivity to Duration Flexibility

To consider how processing time flexibility affects scheduling performance, the initial 100-problem experimental design was repeated for three additional settings of ρ, the tolerance parameter used to determine p_i^{max} for any generated tank operation O_i. Figures 9 (a) and (b) show the performance results obtained with Multi-PCP and Yih94, respectively, for problems generated with ρ settings of 0.0 (no flexibility), 0.5, 1.0 (the original setting) and 2.0.

As expected, the performance of both procedures is seen to improve as the duration flexibility provided by minimum and maximum processing time constraints is increased. Multi-PCP does a better job of exploiting duration flexibility across all configurations. At $\gamma = 15$, the difference is most extreme. With this system configuration, Multi-PCP

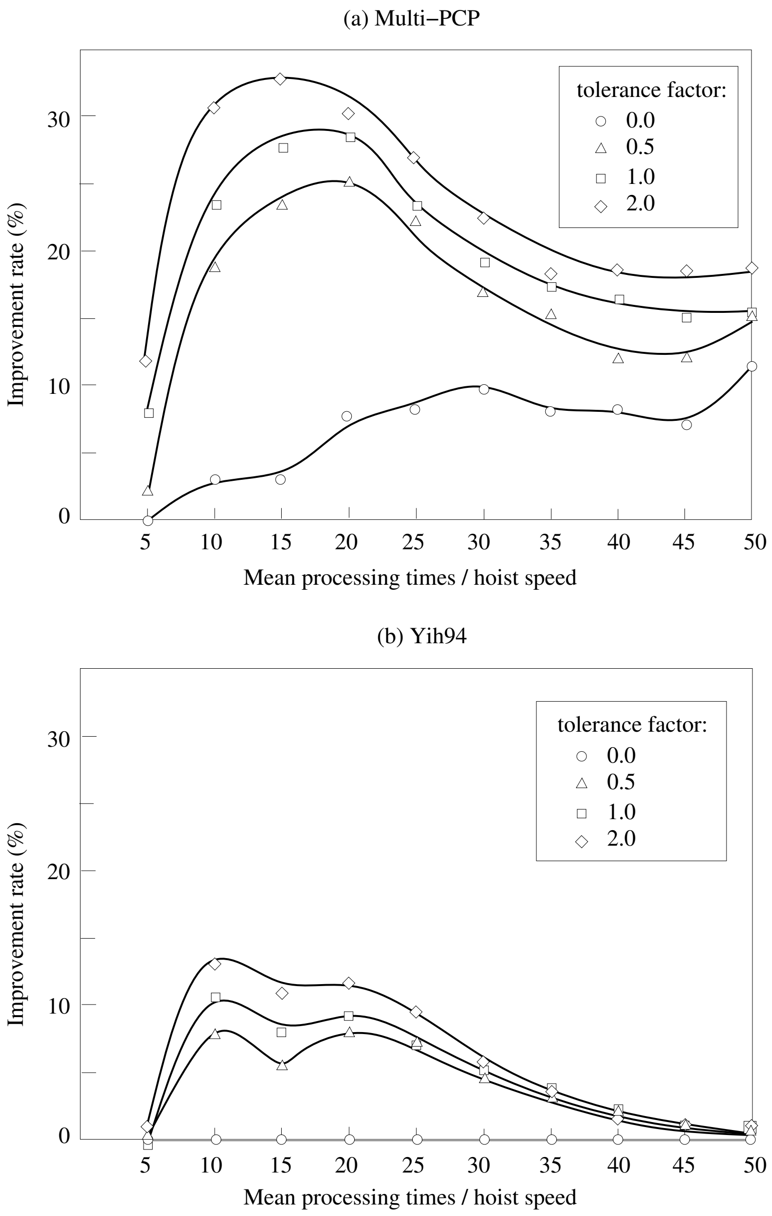

Figure 9: Solution improvement rates as processing time flexibility is varied

obtained a 30% increase in solution improvement rate as ρ is raised from 0 to 2 while the corresponding increase in improvement rate observed for Yih94 over the same problem sets was just 6%. Interestingly, Multi-PCP is found to improve the benchmark solution even in absence of processing time flexibility ($\rho = 0$) for all problem sets except the extreme $\gamma = 5$ problem set, where hoist availability is most problematic; in this case Multi-PCP's improvement rate increases to 10% as hoist availability becomes less and less constraining. Yih94, of course, yields no improvement at $\rho = 0$, since it degenerates to the basic algorithm when there is no processing time flexibility.

More generally, the results obtained in these experiments illustrate the difficulty of accommodating complex, interacting constraints within scheduling procedures that proceed via explicit commitment to specific start times. In the case of the Yih94 algorithm, a priori design decisions are made as to when it is productive to extend processing times versus delay job starts and what specific extend or delay decisions should be taken. Most frequently, however, the best decision actually involves some combination of these two alternatives. This tradeoff is not considered by Yih94, and, while extensions to consider additional types of decisions would appear possible, it is not clear how one could extend such a "fixed times" solution approach to generally address this tradeoff in a cost/effective manner. Multi-PCP, alternatively, does not suffer from this limitation. By operating instead in the space of sequencing decisions, this tradeoff is naturally and directly considered. There is no need to design the algorithm to reason explicitly about the types of constraints involved; decision-making can instead be based strictly on their emergent influence on the evolving partial solution.

6 Summary and Conclusion

In this paper, we have examined the applicability of constraint satisfaction problem solving (CSP) techniques to the long studied problem of makespan minimization, arguing their utility as a basis for the design of cost/effective scheduling procedures which naturally accommodate the idiosyncratic constraints that complicate real-world applications. We reformulated the makespan minimization problem as a search for the least feasible common deadline d, and proposed a simple search procedure to approximate d by solving a series of deadline scheduling problems. A CSP procedure called PCP, which constructs a schedule by incrementally adding precedence constraints to a temporal constraint graph representing the input problem, was embedded as the core, deadline scheduling sub-procedure. The overall procedure, referred to as Multi-PCP, was then empirically evaluated to assess its performance characteristics.

To calibrate its cost/effectiveness as an approximate solution procedure, the performance of Multi-PCP was contrasted with that of recently developed shifting bottleneck and tabu search procedures on previously studied benchmark problem sets. Despite its fairly simple heuristics, Multi-PCP was found to perform quite well. On the Fisher and

Thompson/Lawrence benchmarks, Multi-PCP produced overall results comparable to SB3 at a computational cost comparable to SB1 (Applegate's version). Multi-PCP was found to perform best on problems with low job-to-machine ratios, achieving performance comparable to SB4 on the 10x10 problems. On the larger Taillard benchmarks, Multi-PCP's sensitivity to job-to-machine ratio was further confirmed; on problems with low ratios, Multi-PCP consistently produced better solutions than Applegate's SB1 (the only shifting bottleneck procedure we were able to compare), while the inverse was observed at higher ratios (for those problems that the SB1 implementation was able to successfully solve). In no cases, did Multi-PCP (or SB1) produce solutions as good as those obtained with Taillard's tabu search procedure. However, Multi-PCP solutions were obtained in less time than it took to generate comparable solutions within the tabu search, suggesting its potential as a seeding mechanism for such extended search procedures.

To demonstrate the generality of the approach, we also considered application of Multi-PCP to the multi-product, hoist scheduling problem previously studied by Yih. This problem requires satisfaction of a more complicated set of constraints than does the canonical job shop scheduling formulation of the benchmark problems, including simultaneity in the starts and ends of interleaved material transport and manufacturing operations, non-rigid processing times, and sequence-dependent hoist travel times. It is in this problem context that the power of the underlying constraint satisfaction model is most evident. With only minor extension to account for sequence-dependent setups, Multi-PCP was shown to be directly applicable to this more idiosyncratic makespan minimization problem. Though computational considerations necessitated use of a problem decomposition scheme to achieve realistic problem scale, experimental results nonetheless indicated uniform, sizable improvement in solution quality over other known solution procedures across a wide range of system configurations.

References

Aarts, R. J. and Smith, S. F. (1994). A high performance scheduler for an automated chemistry workstation. In *Proceedings of 1994 European Conference on Artificial Intelligence, Amsterdam.*

Adams, J., Balas, E., and Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391 – 401.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 11(26):832–843.

Applegate, D. and Cook, W. (1991). A computational study of the job-shop scheduling problem. *ORSA Journal of Computing*, 3(2):149 – 156.

Armstrong, R., Lei, L., and Gu, S. (1994). A bounding scheme for deriving the minimal cycle time of a single-transporter n-stage process with time-window constraints. *European Journal of Operational Research*, 78:130–140.

Ashour, S. (1967). A decomposition approach for the machine scheduling problem. *International Journal of Production Research*, 6:109–122.

Balas, E. (1969). Machine sequencing via disjunctive graphs: An implicit enumerated algorithm. *Operations Research*, 17:941–957.

Balas, E., Lenstra, J. K., and Vazacopoulos, A. (1993). The one machine problem with delayed precedence constraints and its use in job shop scheduling. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, #MSRR-589(R).

Barker, J. and McMahon, G. B. (1985). Scheduling the general job-shop. *Management Science*, 31(5):594–598.

Bell, C. E. (1989). Maintaining project networks in automated artificial intelligence planning. *Management Science*, 35(10):1192 – 1214.

Boddy, M. S. and Goldman, R. P. (October, 1994). Empirical results on scheduling and dynamic backtracking. In *Proceedings of the 3rd International Symposium on Artificial Intelligence, Robotics and Automation for Space, Pasadena, CA*.

Brucker, P., Jurisch, B., and Sievers, B. (1992). A branch and bound algorithm for the job-shop scheduling problem. Technical report, Osnabrucker Schriften zur Mathematik, Universitat Osnabruck.

Carlier, J. and Pinson, E. (1989). An algorithm for solving the job-shop problem. *Management Science*, 35(2):164–176.

Charlton, J. M. and Death, C. C. (1970). A generalized machine-scheduling algorithm. *Operational Research Quarterly*, 21:127 – 134.

Cheng, C. and Smith, S. F. (1994). Generating feasible schedules under complex metric constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence, Seattle, Washington.*

Coffman, E. G., Garey, M. R., and Johnson, D. S. (1978). An application of bin-packing to multiprocessor scheduling. *SIAM J. Comput.*, 7:1–17.

Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, 49:61–95.

Della Croce, F., Tadei, R., and Volta, G. (1992). A genetic algorithm for the job shop problem. Technical report, D.A.I. Politecnico di Torino, Italy.

Dell'Amico, M. and Trubian, M. (1991). Applying tabu-search to the job shop scheduling problem. Technical report, Politecnico di Milano, Italy.

Erschler, J., Roubellat, F., and Vernhes, J. P. (1976). Finding some essential characteristics of the feasible solutions for a scheduling problem. *Operations Research*, 24:772–782.

Erschler, J., Roubellat, F., and Vernhes, J. P. (1980). Characterizing the set of feasible sequences for n jobs to be carried out on a single machine. *European Journal of Operational Research*, 4:189–194.

Fisher, H. and Thompson, G. L. (1963). Probabilistic learning combinations of local job-shop scheduling rules. In *Industrial Scheduling*. J. F. Muth, G. L. Thompson (eds), Prentice-Hall, Englewood Cliffs, NJ.

Florian, M., Trepant, P., and McMahon, G. B. (1971). An implicit enumeration algorithm for the machine sequencing problem. *Management Science*, 17(12):B–782–B–792.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman Company.

Gaschnig, J. (1979). Performance measurement and analysis of certain search algorithms. Technical report, Carnegie Mellon University.

Ginsberg, M. L. (1994). Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46.

Haralick, R. M. and Elliott, G. L. (1980). Consistency in networks of relations. *Artificial Intelligence*, 14:263–313.

Harvey, W. D. (1994). Search and jobshop scheduling. Technical report, Computer Science Department, Standford University, and Computational Intelligence Research Laboratory, University of Oregon, CIRL TR 94-1.

Hefetz, H. and Adiri, I. (1982). An efficient optimal algorithm for the two machine unit-time jobshop schedule-length problem. *Mathematics of Operations Research*, 7:354–360.

Hirabayashi, N., Nagasawa, H., and Nishiyama, N. (1994). A decomposition scheduling method for operating flexible manufacturing systems. *International Journal of Production Research*, 32(1):161–178.

Jackson, J. R. (1956). An extension of johnson's results on job lot scheduling. *Navel Research Logistics Quarterly*, 3:201–203.

Kautz, H. and Ladkin, P. B. (1991). Integrating metric and qualitative temporal reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence, Anaheim, CA.*, pages 241–246.

Ladkin, P. B. and Maddux, R. D. (1989). On binary constraint networks. Technical report, Kestrel Institute, Palo Alto, CA.

Lageweg, B., Lenstra, J. K., and Kan, A. H. G. R. (1977). Job-shop scheduling by implicit enumeration. *Management Science*, 24(4):441 – 450.

Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R., and Shmoys, D. B. (1989). Sequencing and scheduling: Algorithms and complexity. Technical report, Report Centre Mathematics and Computer Science, Amsterdam.

Lawrence, S. (1984). Resource constraint project scheduling: An experimental investigation of heuristic scheduling techniques. Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University.

Lei, L. and Wang, T. J. (1991). The minimum common-cycle algorithm for cyclic scheduling of two hoists with time window constraints. *Management Science*, 37(12):1629–1639.

Lieberman, R. W. and Turksen, I. B. (1981). Crane scheduling problems. *AIIE Transactions*, 13(4):304–311.

Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118.

Matsuo, H., Suh, C. J., and Sullivan, R. S. (1988). A controlled search simulated annealing method for the general job shop scheduling problem. Technical report, WP03-04-88, Department of Management, Graduate School of Business, University of Texas, Austin.

Meiri, I. (1991). Combining qualitative and quantitative constraints in temporal reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence, Anaheim, CA.*, pages 260–267.

Minton, S., Johnston, M. D., Philips, A. B., and Laird, P. (1992). Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205.

Muscettola, N. (1993). Scheduling by iterative partition of bottleneck conflicts. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence Applications, Orlando, FL.*

Nowicki, E. and Smutnicki, C. (1994). A fast taboo search algorithm for the job shop problem. Technical report, Technical University of Wroclaw, Institute of Engineering Cybernetics, ul. Janiszewskiego 11/17, 50-372 Wroclaw, Poland.

Panwalker, S. S. and Iskander, W. (1977). A survey of scheduling rules. *Operations Research*, 25:45 – 61.

Phillips, L. W. and Unger, P. S. (1976). Mathematical programming solution of a hoist scheduling problem. *AIIE Transactions*, 8(2):219–225.

Sadeh, N. (1991). Look-ahead techniques for micro-opportunistic job shop scheduling. Technical report, CMU-CS-91-102, School of Computer Science, Carnegie Mellon University.

Schrage, L. (1970). Solving resource-constrained network problems by implicit enumeration - nonpreemptive case. *Operations Research*, 18:263 – 278.

Shapiro, G. W. and Nuttle, H. L. W. (1988). Hoist scheduling for a pcb electroplating facility. *IIE Transactions*, 20(2):157–167.

Smith, S. F. and Cheng, C. (1993). Slack-based heuristics for constraint satisfaction scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence, Washington, DC.*, pages 139 – 144.

Stallman, R. M. and Sussman, G. J. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9(2):135–196.

Storer, R. H., Wu, S. D., and Vaccari, R. (1992). New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38(10):1495 – 1509.

Taillard, E. (1989). Parallel taboo search technique for the jobshop scheduling problem. Technical report, ORWP 89/11, Department de Mathematiques, Ecole Polytechnique Federale De Lausanne, Lausanne, Switzerland.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278 – 285.

van Laarhoven, P. J. M., Aarts, E. H. L., and Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research*, 40(1):113 – 125.

Vilain, M. and Kautz, H. (1986). Constraint propagation algorithms for temporal reasoning. In *Proceedings of the Fourth National Conference on Artificial Intelligence, Philadelphia, PA.*, pages 377–382.

Xiong, Y., Sadeh, N., and Sycara, K. (1992). Intelligent backtracking techniques for job shop scheduling. In *Proceedings of the Third International Conference on Principle of Knowledge Representation, Cambridge, MA.*

Yih, Y. (1994). An algorithm for hoist scheduling problems. *International Journal of Productions Research*, 32(3):501–516.

Yih, Y., Liang, T., and Moskowitz, H. (1993). Robot scheduling in a circuit board production line: A hybrid or/ann approach. *IIE Transactions*, 25(2):26–33.

Yih, Y. and Thesen, A. (1991). Semi-markov decision models for real-time scheduling. *International Journal of Productions Research*, 29(11):2331–2346.

Zweben, M., Deale, M., and Gargan, R. (1990). Anytime rescheduling. In *Proceedings of DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control, Morgan Kaufmann Pub.*

Appendix A

Table 5: Makespan results for small benchmark problem set

Problem	Job x Machine	Optimal Value	Multi-PCP	Multi-PCP w/ Relax	SB1	SB3	SB4
mt06	6 x 6	55	55	55	59	55	55
mt10	10 x 10	930	949	949	952	981	940
mt20	20 x 5	1165	1267	1192	1228	1199	1199
la1	10 x 5	666	666	666	666	666	666
la2		655	670	670	684	667	667
la3		597	619	617	605	626	626
la4		590	600	600	603	593	593
la5		593	593	593	593	593	593
la6	15 x 5	926	926	926	926	926	926
la7		890	890	890	890	890	890
la8		863	863	863	863	863	863
la9		951	951	951	951	951	951
la10		958	958	958	958	958	958
la11	20 x 5	1222	1222	1222	1222	1222	1222
la12		1039	1039	1039	1039	1039	1039
la13		1150	1150	1150	1150	1150	1150
la14		1292	1292	1292	1292	1292	1292
la15		1207	1207	1207	1207	1207	1207
la16	10 x 10	945	982	982	1076	961	961
la17		784	787	787	829	796	796
la18		848	886	886	855	866	861
la19		842	852	852	863	902	878
la20		902	922	922	918	932	922
la22	15 x 10	927	984	965	968	954	954
la23		1032	1041	1041	1071	1032	1032
la24		935	983	983	1015	976	976
la25		977	1026	1026	1061	1012	1012
la26	20 x 10	1218	1272	1272	1393	1239	1224
la28		1216	1275	1275	1281	1245	1245
la30		1355	1356	1356	1355	1355	1355
la31	30 x 10	1784	1784	1784	1784	1784	1784
la32		1850	1850	1850	1850	1850	1850
la33		1719	1722	1722	1719	1719	1719
la34		1721	1744	1744	1721	1721	1721
la35		1888	1888	1888	1888	1888	1888
la36	15 x 15	1268	1321	1321	1326	1319	1319
la37		1397	1446	1466	1471	1425	1425
la39		1233	1286	1286	1301	1278	1278
la40		1233	1272	1272	1347	1266	1262

Table 6: % deviation from optimal solution for small benchmark problem set

Problem	Job x Machine	Multi-PCP	Multi-PCP w/ Relax	SB1	SB3	SB4
mt06	6 x 6	0	0	7.27	0	0
mt10	10 x 10	2.04	2.04	2.37	5.48	1.08
mt20	20 x 5	8.76	2.32	5.41	2.92	2.92
la1	10 x 5	0	0	0	0	0
la2		2.29	2.29	4.43	1.83	1.83
la3		3.69	3.35	1.34	4.86	4.86
la4		1.69	1.69	2.20	0.51	0.51
la5		0	0	0	0	0
la6	15 x 5	0	0	0	0	0
la7		0	0	0	0	0
la8		0	0	0	0	0
la9		0	0	0	0	0
la10		0	0	0	0	0
la11	20 x 5	0	0	0	0	0
la12		0	0	0	0	0
la13		0	0	0	0	0
la14		0	0	0	0	0
la15		0	0	0	0	0
la16	10 x 10	3.92	3.92	13.86	1.69	1.69
la17		0.38	0.38	5.74	1.53	1.53
la18		4.48	4.48	0.83	2.12	1.53
la19		1.19	1.19	2.49	7.13	4.28
la20		2.22	2.22	1.77	3.33	2.22
la22	15 x 10	6.15	4.10	4.42	2.91	2.91
la23		0.87	0.87	3.78	0	0
la24		5.13	5.13	8.56	4.39	4.39
la25		5.02	5.02	8.60	3.58	3.58
la26	20 x 10	4.43	4.43	14.37	1.72	0.49
la28		4.85	4.85	5.35	2.38	2.38
la30		0.07	0.07	0	0	0
la31	30 x 10	0	0	0	0	0
la32		0	0	0	0	0
la33		0.17	0.17	0	0	0
la34		1.34	1.34	0	0	0
la35		0	0	0	0	0
la36	15 x 15	4.18	4.18	4.57	4.02	4.02
la37		4.94	4.49	5.30	2.00	2.00
la39		4.30	4.30	5.52	3.65	3.65
la40		3.16	3.16	9.25	2.68	2.35

Table 7: Computation times (in seconds) for small benchmark problem set

Problem	Job x Machine	Multi-PCP	SB1	SB3	SB4
mt06	6 x 6	0.05	0.12	0.78	1.45
mt10	10 x 10	0.38	0.72	2.21	7.76
mt20	20 x 5	1.38	0.28	1.99	3.62
la1		0.05	0.12	0.08	0.13
la2		0.23	0.13	0.51	0.81
la3	10 x 5	0.17	0.10	0.43	0.68
la4		0.19	0.11	0.37	0.63
la5		0.02	0.13	0.61	0.61
la6		0.03	0.15	0.09	0.09
la7		0.53	0.17	0.11	0.11
la8	15 x 5	0.48	0.18	0.20	0.20
la9		0.05	0.12	0.11	0.11
la10		0.02	0.15	0.10	0.10
la11		0.03	0.18	0.19	0.19
la12		0.03	0.12	0.12	0.12
la13	20 x 5	0.05	0.18	0.13	0.13
la14		0.03	0.13	0.09	0.09
la15		0.20	0.20	0.18	0.18
la16		0.27	0.48	1.34	3.59
la17		0.25	0.65	1.57	2.97
la18	10 x 10	0.23	0.62	1.74	3.05
la19		0.30	0.78	1.67	3.30
la20		0.23	0.80	1.71	3.09
la22		1.15	1.10	3.59	6.52
la23	15 x 10	1.10	1.28	1.99	1.99
la24		0.95	1.40	3.92	7.19
la25		0.97	1.00	4.15	7.25
la26		2.55	1.33	4.55	10.77
la28	20 x 10	2.37	1.83	5.71	10.41
la30		2.62	1.67	1.33	1.33
la31		5.77	3.04	3.45	3.45
la32		0.12	2.38	2.74	2.74
la33	30 x 10	8.42	1.67	4.63	4.63
la34		8.63	3.04	6.99	6.99
la35		1.30	2.77	2.86	2.86
la36		1.08	4.07	14.01	28.14
la37	15 x 15	1.23	2.93	13.22	26.84
la39		1.47	5.78	12.84	25.59
la40		1.37	5.40	13.30	26.58

Table 8: Makespan results for large benchmark problem set

Job x Machine	Best Value	Multi-PCP	Multi-PCP w/ Relax	SB1	Job x Machine	Best Value	Multi-PCP	Multi-PCP w/ Relax	SB1
	1231*	1280	1280	1360		2064	2323	2323	2343
	1252	1322	1308	1367		1983	2205	2205	2199
	1223	1288	1288	1289		1896	2165	2165	2123
	1181	1260	1253	1289		2031	2269	2252	2393
15 x 15	1234	1306	1306	1359	30 x 20	2032	2215	2215	2262
	1243	1320	1320	1314		2057	2269	2269	2329
	1228	1307	1307	1322		1947	2104	2104	2202
	1221	1289	1289	1345		2005	2277	2277	2191
	1289	1366	1366	1437		2013	2337	2330	2270
	1261	1334	1334	1331		1973	2188	2188	2301
	1376	1522	1488	1481		2760*	2948	2948	2853
	1381	1495	1495	1503		2756*	2976	2964	2806
	1367	1478	1478	1521		2717*	2898	2891	2761
	1355	1452	1452	1540		2839*	2905	2905	2839
20 x 15	1366	1486	1486	1532	50 x 15	2689	2944	2944	-
	1371	1478	1478	1511		2781*	3023	3023	2924
	1480	1609	1609	1605		2943*	3178	3156	2987
	1432	1503	1503	1532		2885*	3131	3131	2969
	1361	1440	1440	1504		2655*	2943	2938	2755
	1373	1441	1441	1535		2723*	2904	2904	2828
	1663	1764	1764	1814		2921	3177	3177	3098
	1626	1739	1739	1776		3002	3311	3244	3237
	1574	1729	1687	1822		2835	3059	3047	2978
	1660	1773	1773	1822		2775	2982	2951	2879
20x 20	1598	1742	1742	1847	50 x 20	2800	3061	3061	2978
	1679	1759	1759	1861		2914	3188	3187	3089
	1704	1862	1862	1857		2895	3165	3165	3105
	1626	1749	1749	1809		2835	3048	3048	2918
	1635	1761	1761	1771		3097	3419	3419	-
	1614	1743	1743	1729		3075	3349	3349	3163
	1770	2015	2001	2017		5464*	5804	5804	-
	1853	2033	2019	2061		5181*	5378	5378	-
	1855	2053	2053	1987		5568*	5885	5885	-
	1851	2071	2071	1996		5339*	5553	5553	-
30 x 15	2007	2106	2106	1965	100 x 20	5392*	5743	5743	-
	1844	2031	2016	1965		5342*	5733	5733	-
	1822	1954	1954	1976		5436*	5621	5621	-
	1714	1853	1853	1915		5394*	5618	5618	-
	1824	2010	2010	1890		5358*	5563	5563	-
	1723	1931	1931	1838		5213	5413	5413	-

* These indicate optimal values.

Table 9: % deviation from best solution for large benchmark problem set

Job x Machine	Multi-PCP	Multi-PCP w/ Relax	SB1	Job x Machine	Multi-PCP	Multi-PCP w/ Relax	SB1
15 x 15	3.98	3.98	10.48	30 x 20	12.55	12.55	13.52
	5.59	4.47	9.19		11.20	11.20	10.89
	5.31	5.31	10.55		14.19	14.19	11.97
	6.69	6.10	9.14		11.72	10.88	17.77
	5.83	5.83	10.13		9.01	9.01	11.32
	6.19	6.19	5.71		10.31	10.31	13.22
	6.43	6.43	7.65		8.06	8.06	13.10
	5.57	5.57	10.16		13.57	13.57	9.28
	5.97	5.97	11.48		16.10	15.75	12.77
	5.79	5.79	5.55		10.90	10.90	16.62
20 x 15	10.61	8.14	7.63	50 x 15	6.81	6.81	3.37
	8.25	8.25	8.83		7.98	7.55	1.81
	8.12	8.12	11.27		6.66	6.40	1.62
	7.16	7.16	13.65		2.32	2.32	0.00
	8.78	8.78	13.65		9.48	9.48	-
	7.80	7.80	10.21		8.70	8.70	5.14
	8.72	8.72	8.45		7.99	7.24	1.50
	4.96	4.96	6.98		8.53	8.53	2.91
	5.80	5.80	10.51		10.85	10.66	3.77
	4.95	4.95	11.80		6.65	6.65	3.86
20 x 20	6.07	6.07	9.08	50 x 20	8.76	8.76	6.06
	6.95	6.95	9.23		10.29	8.06	7.83
	9.85	7.18	9.59		7.90	7.48	4.97
	6.81	6.81	9.76		7.46	6.34	3.75
	9.01	9.01	15.58		9.32	9.32	6.36
	4.76	4.76	10.84		9.40	9.37	6.01
	9.27	9.27	8.98		9.33	9.33	7.25
	7.56	7.56	11.25		7.51	7.51	2.93
	7.71	7.71	8.32		10.40	10.40	-
	7.99	7.99	7.13		8.91	8.91	2.86
30 x 15	13.84	13.05	13.95	100 x 20	6.22	6.22	-
	9.71	8.96	11.23		3.80	3.80	-
	10.67	10.67	7.12		5.69	5.69	-
	11.89	11.89	7.83		4.01	4.01	-
	4.93	4.93	6.63		6.51	6.51	-
	10.14	9.33	6.56		7.32	7.32	-
	7.24	7.24	8.45		3.40	3.40	-
	8.11	8.11	11.73		4.15	4.15	-
	10.20	10.20	3.62		3.83	3.83	-
	12.07	12.07	6.67		3.84	3.84	-

Table 10: Computation times (in seconds) for large benchmark problem set

Job x Machine	Multi-PCP	SB1	Job x Machine	Multi-PCP	SB1
	1.15	4.08		14.65	28.32
	1.18	5.88		14.05	37.95
	1.17	6.48		15.67	35.42
	1.13	4.52		16.32	28.97
15 x 15	1.33	3.22	30 x 20	14.85	34.05
	1.20	5.05		16.52	32.73
	1.17	3.37		15.87	34.70
	1.27	6.88		15.60	30.47
	1.33	6.93		15.68	28.30
	1.10	4.55		15.90	24.92
	3.53	7.98		73.88	269.91
	3.05	9.62		72.20	139.21
	3.15	6.65		66.68	316.89
	3.18	6.20		72.46	47.40
20 x 15	3.67	8.42	50 x 15	64.68	-
	3.32	6.97		50.43	46.26
	3.63	7.77		70.48	37.30
	4.10	7.77		75.31	175.06
	3.54	8.43		68.66	49.80
	3.06	6.48		66.28	190.11
	3.40	11.62		93.43	125.86
	4.32	18.48		91.43	150.18
	3.40	15.92		89.01	251.82
	3.77	13.90		90.74	98.21
20 x 20	3.78	13.75	50 x 20	103.90	382.80
	3.80	13.78		105.43	90.95
	4.05	13.52		95.35	97.71
	3.77	15.53		100.61	145.39
	3.22	19.73		91.33	-
	3.83	20.20		87.72	149.58
	12.28	12.45		944.45	-
	12.54	15.97		832.63	-
	10.98	14.32		870.70	-
	11.95	12.11		798.67	-
30 x 15	10.00	14.17	100 x 20	865.13	-
	12.59	16.20		846.10	-
	11.60	15.72		836.57	-
	12.77	17.68		864.87	-
	12.43	14.15		877.18	-
	11.82	14.02		837.33	-