SCHEDULING TO MAXIMIZE WORKER SATISFACTION AND ON TIME ORDERS

Alex J. Ruiz-Torres*
Facultad de Administración de Empresas
Universidad de Puerto Rico – Rio Piedras
San Juan, PR 00931-3332, USA
Phone: 787-764-8000 ext 87125
Email: alex.ruiztorres@upr.edu

Nelson Alomoto
Facultad de Ciencias Administrativas
Escuela Politécnica Nacional
Quito, 17-01-2759 Ecuador
E-mail: nelson.alomoto@epn.edu.ec,

Giuseppe Paletta
Dipartimento di Economia, Statistica e Finanza
Università della Calabria (UNICAL)
87036 Rende, CS, Italy
E-mail: g.paletta@unical.it

Eduardo Pérez
Ingram School of Engineering
Texas State University, 601 University Drive
San Marcos, TX 78666
E-mail: eduardopr@txstate.edu

Abstract

Two important managerial objectives in production planning are the maximization of the *on time* delivery of orders and the satisfaction of the workers. While the maximization of *on time* deliveries has been considered frequently in past production planning research, maximizing worker satisfaction has been typically ignored. The assignment of preferred tasks to workers is important since it results in a productive working environment with high worker performance and low turnovers. This paper presents a job scheduling model that considers both criteria simultaneously and derives solution approaches to generate non-dominated solutions. The solution approaches are examined under various experimental conditions to evaluate their performance. Finally a prototype tool developed as a *proof of concept* is presented.

Keywords: Parallel machines, late jobs, worker satisfaction, worker preference, completion times, on time orders, makespan, heuristics, non-dominated solutions, bi-criteria, evaluation methods.

SCHEDULING TO MAXIMIZE WORKER SATISFACTION AND ON TIME ORDERS

1. INTRODUCTION

The literature on scheduling problems is vast and diverse given the importance of effectively utilizing the limited resources available in manufacturing and service organizations. While the literature has often focused on problems that can be linked to "machine" type resources, the general framework used implies that many problems can also apply to the scheduling of people. For example, the health care sector has adopted techniques applied to manufacturing (including scheduling) to create real value by reducing waste and improving productivity, cost, quality, and the timely delivery of patient care services (Johnson et al., 2012). While *worker (staff) scheduling* variables can be very similar to those used to characterize machines, one that may not easily correlate is that of worker's preference/ satisfaction for the work they perform. Preference/ satisfaction is not equivalent to the ability or speed of performing a task, although they could be related. In general, workers like to perform tasks they are good/fast at and similarly, dislike tasks they are slow/ inefficient at. However, there are workers that prefer tasks with a high level of complexity or "nature", even when they may not perform them "fast/ efficiently", and similarly have low preference for another set of tasks they can perform quickly, but find them boring or repetitive.

This paper addresses a problem observed at a manufacturing facility that produces make to order windows and cabinetry. The problem is to find the best assignment of jobs to workers that meets customer order due dates while taking into consideration the preference of the workers (the level of satisfaction of the work assigned to them). The assignment of the preferred tasks to the workers will result in workers with higher morale, higher dedication to their work, and will reduce company turnover levels. In the setting under consideration there are multiple pending orders and each order is characterized by a due date, order type, a processing time, and a scheduling time window. Each worker has a preference level for each job type and the satisfaction level associated with performing a task impact the processing time of the job. Following the traditional scheduling framework, this problem is considered as an identical parallel machine scheduling problem. Two managerial objectives are considered: the maximization of the on time jobs and the maximization of worker satisfaction.

A large number of papers have been published that aim to minimize the number of late jobs, although the number focused on parallel machine settings is relatively small. Ho and Chang (1995) were the first to address the problem of minimizing the number of late jobs in parallel machines. The authors present and discuss the performance of several heuristics derived from the optimal single machine algorithm from Moore (1968). Lin and Jeng (2004) addressed the case of batch scheduling in parallel machines with two objectives: the minimization of the maximum lateness and the number of tardy jobs. They propose a dynamic programming approach and some heuristics to solve the problem. M'Hallah and Bulfin (2005) developed a branch and bound algorithm to minimize the weighted and un-weighted number of tardy jobs for the identical and unrelated parallel machine cases. A polynomial algorithm for the identical parallel machine case with identical processing times and number of tardy jobs objective is presented by Bornstein et al. (2005). In their research, the problem is formulated as a maximum flow network model. The dual resource problem of scheduling jobs on a set of parallel machines with the objective of minimizing the number of late jobs is investigated by Ruiz-Torres et al. (2007). In the addressed problem the speed of the machines depends on the allocation of a secondary resource that is allocated to the machines at the start of the schedule. The paper proposes a set of heuristics for the general case and an integer programming formulation to solve the case where the jobs are pre-assigned to the machines.

When workers submit their schedule preferences and the organizational work plan does not have any conflict, a win-win schedule can be generated without costing the organization (Alsheddy and Tsang, 2011). This ideal scenario is hard to find in practice; therefore different approaches have been suggested in order to maximize workers satisfaction. An integer programming model to maximize part-time employee satisfaction and meet the demand requirements per shift is presented by Mohan (2008). The part-time workers have availabilities, preferences for the shifts, and also a seniority level. Alsheddy and Tsang (2011) developed an empowerment scheduling model which enables employees to express their own schedule. The main idea of empowerment is to make the employees feel the fairness of the system by incorporating an automatic market-like mechanism that controls the violation cost of each worker's request. Shahnazari-Shahrezaei et al. (2012) argue that preferences may not be determined precisely and causes the worker scheduling problem to be of fuzzy nature. They present a fuzzy multi-objective mathematical model to solve a multi-skilled manpower scheduling problem considering imprecise target values of employers' objectives and employees' preferences. Akbari et al. (2012) present a linear programming model aimed to maximize workers' satisfaction while regarding workers' availability, productivity, priority preference, seniority level, and number of workers required. They use simulated annealing and variable neighborhood search for tackling this difficult combinatorial problem.

Another relevant problem variation is the consideration of resource cost as this can relate to worker preference. Research in parallel machines considering a cost function has been investigated by a few authors. Leyvand et al. (2010) present a convex resource consumption function and study scheduling problems with controllable processing times on parallel machines. The objectives are to maximize the weighted number of jobs that are completed exactly at their due date and to minimize the total resource allocation cost. Ruiz-Torres et al. (2010) study the case in which a concave cost function together with any regular performance measure are criteria for determining how many machines to use and how to optimally load them.

This research work contributes to the scheduling literature by analyzing a problem with a new measure of performance of importance in environments where people are the critical resource and where their satisfaction with their work assignment highly relevant. The research is also relevant as it addresses the maximization of two objective functions in a non-dominated format, while also considering time window constraints, an atypical formulation but of practical application (in the parallel machine literature minimizing the time window or makespan is typically an objective function, not a constraint).

The remaining of the paper is organized as follows. Section 2 provides the problem formulation and an example. Section 3 describes the solution approaches while section 4 presents the evaluation methodology. The experimental framework and the results are discussed in section 5, while a prototype application is presented in Section 6. A summary and directions for future work are included in Section 7.

2. PROBLEM FORMULATION AND EXAMPLE

The problem under consideration is described as follows. There are n jobs to be completed, $N = \{1, \ldots, n\}$ and w workers available $W = (1 \ldots w)$, with $w > 1$. Jobs are non-divisible and preemption is not allowed. Workers can only process one job at a time and cannot stand idle until the last job assigned to him/her has been finished. Each job j has a due date d_j and a process time of p_j. We assume $p_j \geq 1$, $d_j \geq p_j$, and processing times and due dates are integer values. Let P be the sum of all the jobs process times, $P = \sum_{j \in N} p_j$.

The satisfaction received by worker k when assigned job j is $s_{j,k}$. The problem considers two objectives simultaneously: one related to customer satisfaction (the maximization of *on time* orders) and one related to overall worker level of satisfaction. The first objective is a traditional objective of the scheduling literature as it is obviously equivalent to the minimization of the number of late jobs. Let c_j be the completion time of job j and u_j be a binary variable equal to 1 if $c_j > d_j$, and 0 otherwise. Let the percentage of on time jobs O_t equal $(n - \sum_{j\in N} u_j)/n$.

To the best of our knowledge there are no job based worker satisfaction criteria for multi-job multi-machine environments, although a related criteria is the minimization of total job costs (as an inverse to satisfaction). The models found in the literature relate worker satisfaction to meeting the preference of workers to be assigned to particular desired work shifts, and not to performing particular types/ sets of jobs. We thus propose that *a job assignment based satisfaction criteria* should consider several factors including the total level of satisfaction of the staff (which is similar to the total cost), and the relative or minimum satisfaction among workers (no similar element when we consider costs, there is no point in having balanced costs across resources).

It is important to note that focusing solely on the sum of worker satisfaction scores (equivalent to minimizing the cost of the resource to job assignments) could lead to highly unbalanced satisfaction levels between the workers. For example, if all jobs are of type q, and worker k has a high preference for type q jobs, and all other workers strongly dislike type q jobs, assigning all jobs to worker k would maximize the total satisfaction score. However worker k would in reality be highly unsatisfied given it was tasked to perform all the work. Thus as mentioned earlier, worker satisfaction for the complete set of workers would be associated with a balanced allocation of satisfaction. In our model which is based on the manufacturing setting described in section 1, the objective is to finds a schedule where none of the workers ends up below a satisfaction threshold level. It is proposed that from an overall worker satisfaction point of view, having assignments where all the worker satisfaction scores meet a minimum threshold level is better than some very satisfied workers and a few very unsatisfied workers (below a threshold), even when the later total satisfaction score (for the group) is higher than the first. For example, assume a satisfaction scale 1-7; 7 being very satisfied, 1 being very unsatisfied, and 4 is the threshold level (workers are "ok" with the assignment). There are four workers. Assume a schedule $\sigma 1$ where their job assignments result in satisfaction scores of 7, 7, 3,and 3; two *very happy* workers and two *unhappy* workers, and a total score of 20 (average of 5). Assume a schedule $\sigma 2$ with satisfaction scores of 4 for all the workers; all workers are "ok" with their assignments, and a total score of 16 (average of 4). Our model assumes that schedule $\sigma 2$ is preferred by management (and the workers). Having limits on the satisfaction scores is an obvious difference from the resource cost type objective function (a resource cost objective would select $\sigma 1$).

Let z_k be the satisfaction score associated with the jobs assigned to worker k. Let N_k be the set of jobs assigned to worker k, then $z_k = \sum_{j\in Nk} s_{j,k} / \sum_{j\in Nk} p_j$. As previously described, our model includes a constraint that limits the satisfaction score per worker (threshold level) for any of the workers. Let Z_{min} be the minimum allowed worker satisfaction score, thus $z_k \geq Z_{min}$ for all the workers. It is noted that the number of feasible solutions will shrink as Z_{min} increases, and no solutions may be found after Z_{min} reaches some instance specific level. The total average worker satisfaction is defined as $Z_{ave} = \sum_{k\in W} z_k / w$ and is our second measure of performance. Clearly Z_{ave} will always be greater than or equal Z_{min} for a feasible schedule.

Another important element of the model is the consideration of a specified deadline C_{bound} for the jobs, i.e. each job must finish not later than time C_{bound}. This model does not minimize the makespan C_{max}

= ***max*** $_{j\in N}$ c_j but considers the constraint that all jobs are completed before the deadline C_{bound}. The assumption is that this time window accounts for the available worker time, and that time not assigned to processing the jobs in N will be assigned to other tasks as training, equipment maintenance, and last minute/ "emergency" activities. As in the case of Z_{min}, having a very tight C_{bound} constant may result in no feasible schedules being found.

Let G be the set of all possible positions in a worker's schedule, and assuming all workers get at least one job there are at most $n - w + 1$ positions in a worker's schedule. The mathematical formulation for this problem is presented next. The decision variable x_{jki}, $j\in N$, $k\in W$, $i\in G$, is a binary variable that is equal to 1 if job j is assigned to worker k in position i, 0 otherwise. The decision variable u_{ki}, $k\in W$, $i\in G$, is a binary variable that is equal to 1 if the job assigned to worker k in position i is late, 0 otherwise. Let B be a big number.

2.1 Formulation

$$\text{Maximize } O_t = (n - \textstyle\sum_{i\in G, k\in W} u_{ki})/n \qquad (1)$$
$$\text{Maximize } Z_{ave} = \textstyle\sum_{k\in W} z_k / w \qquad (2)$$

$$\textstyle\sum_{j\in N} x_{jki} \leq 1 \qquad \forall\ i\in G, k\in W \qquad (3)$$
$$\textstyle\sum_{i\in G, k\in W} x_{jki} = 1 \qquad \forall\ j\in N \qquad (4)$$
$$x_{jki} \leq \textstyle\sum_{l\in N} x_{lk(i\text{-}1)} \qquad \forall\ j\in N, k\in W\ i\in G\backslash\{1\} \qquad (5)$$
$$m_{ki} = \textstyle\sum_{j\in N,\, l = 1..i} p_j \times x_{jkl} - \sum_{j\in N} d_j \times x_{jki} \qquad \forall\ k \in W, i\in G \qquad (6)$$
$$m_{k,i} \leq u_{ki} \times B + B\,(1\text{-}\textstyle\sum_{j\in N} x_{jki}) \qquad \forall\ k \in W, i \in G \qquad (7)$$
$$C_k = \textstyle\sum_{j\in N, i\in G} p_j \times x_{jki} \qquad \forall\ k \in W \qquad (8)$$
$$C_{bound} \geq C_k \qquad \forall\ k \in W \qquad (9)$$
$$z_k = \textstyle\sum_{i\in G, j\in N} s_{j,k} \times x_{jki} \;/\; C_k \qquad \forall\ k \in W \qquad (10)$$
$$z_k \geq Z_{min} \qquad \forall\ k \in W \qquad (11)$$
$$x_{jki} \in \{0, 1\} \qquad \forall\ j\in N, k\in W, i\in G \qquad (12)$$
$$u_{ki} \in \{0, 1\} \qquad \forall\ k\in W, i\in G \qquad (13)$$

The first two equations are the independent objective functions considered. Equation (3) states that to each position in each worker at most one job can be assigned. Equation (4) states that each job must be assigned just once to one position in one worker. Equation (5) guarantees continuous assignments In Equation (6), m_{ki} represents the tardiness of the job assigned to position i of the worker if $\sum_{j\in N} x_{jki}$=1 (i.e. a job is assigned to worker k at position i). In this case, a value of m_{ki} less than or equal to 0 indicates an *on-time* job, while a positive value indicates the job in that position/worker is late. Whereas if no job is assigned to that position of worker k (i.e. $\sum_{j\in N} x_{jki} = 0$), m_{ih} represents the workload of worker k. Equation (7) is used to set the binary variable u_{ki} to 1 if the job assigned to worker k in position i is late, 0 for on time job (or unused positions). Equation (8) establishes the workload for each worker and equation (9) limits the workload in each worker (C_k) by the maximum allowed (time windows bound). Equation (10) is used to set the satisfaction score for each worker and equation (11) constraints the satisfaction scores per worker by the minimum level. Equations (12-13) set up the binary variables. The problem has approximately $n^2(w$+1) binary variables, making it "computationally" challenging as n and w

increase. The problem of minimizing the number of late jobs has been shown to be NP-Hard (Ho and Chang 1995), thus any multi-criteria problem that considers this criteria is also NP-Hard.

2.2 Problem Example

To demonstrate the problem challenges and tradeoffs between the criteria, a simple example is presented based on the observed industrial/service case of a custom-made cabinetry and furniture operation. For this case, the level of satisfaction perceived by the worker is based on the job type. It is assumed that there are b job types and each worker k has a preference rating $r_{k,h}$ for job type h. A linear rating scale is proposed with a maximum value reflecting a highly desirable job type, and the lowest value a highly undesirable job type. Let r_{max} be the highest rating a job type can have, and r_{min} be the lowest rating any job type can have, and $r_{min} \geq 0$, $r_{max} \geq 0$, and $r_{max} > r_{min}$. The satisfaction associated with assigning job j of type h to worker k, $s_{j,k}$, is a function of p_j and $r_{k,h}$ and modeled as $s_{j,k} = r_{k,h} \times p_j$ which indicates a linear relationship between the satisfaction the worker gets from jobs and their processing duration. There are multiple ways for defining $s_{j,k}$. The approach used in this work is based on the observed manufacturing setting..

Table 1 presents the worker preference information while Table 2 presents the job information. Let $r_{max} = 7$ and $r_{min} = 1$; a $r_{k,h}$ value of 7 indicates a very strong preference for that type of job, a score of 1 indicates a strong dislike for this type of job and a score of 4 indicates “indifference” or an “acceptable” type of job. We assume an initial value of Z_{min} equal to 3 and C_{bound} equal to 42 (approximately a 10% of slack from P/w).

Table 1. Worker preference information

	Preference ratio ($r_{k,h}$)		
worker (k)	$h = 1$	$h = 2$	$h = 3$
1	5	1	5
2	4	7	2

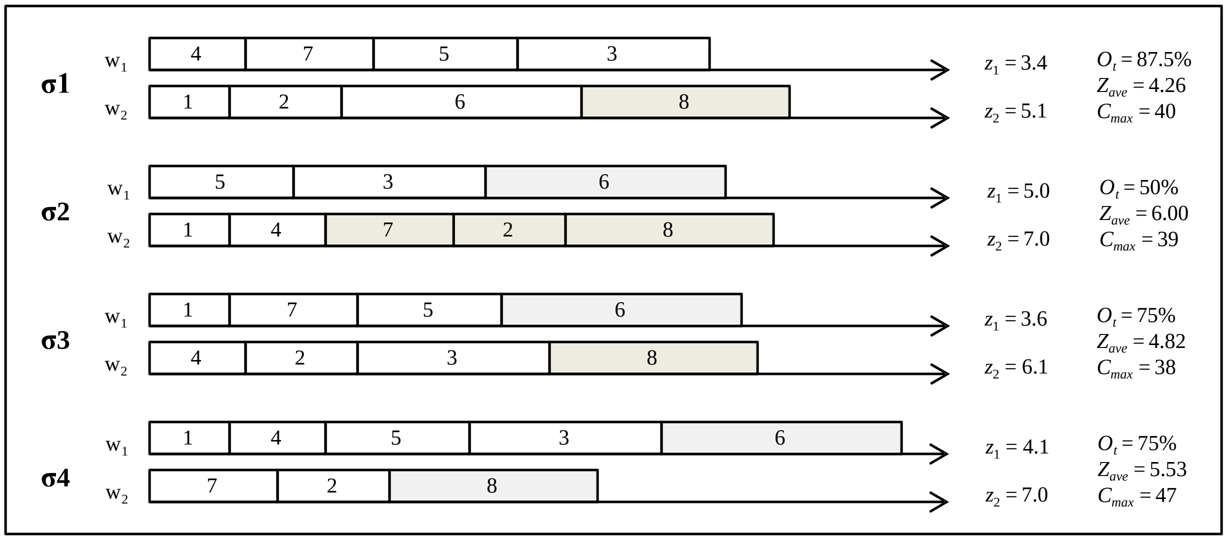
Table 2. Job information

Job (j)	h	p_j	d_j	s_{j1}	s_{j2}
1	2	5	8	5	35
2	2	7	15	7	49
3	1	12	35	60	48
4	2	6	14	6	42
5	3	9	23	45	18
6	3	15	27	75	30
7	2	8	14	8	56
8	2	13	17	13	91

Four possible schedules are presented in Figure 1. The four schedules are not intended to be the most “reasonable” / efficient schedules. Schedule σ1 is optimal for the percentage of on time jobs (no schedule can be generated with 100% on time jobs), but has a relatively low average satisfaction score Z_{ave} when compared to schedules σ1, σ2, and σ3. Furthermore, while it complies with the Z_{min} requirement, the satisfaction of worker 1 is below the “mid-point” (the desired level as discussed in the next paragraph). Schedule σ*2* has the highest possible value of Z_{ave} (e.g. 6.0), a value that is obtained by assigning jobs to the worker that has the highest preference for that type. While schedule σ2 is optimal for Z_{ave}, it is not a *good* schedule from a management standpoint given that it only completes 50% of the jobs on time. Schedule σ3 provides a balanced solution considering the on-time jobs and level of satisfaction performance measures, although worker 1 is still below the median of 4. This schedule is also makespan optimal.

If Z_{min} is increased to 4 (the threshold level desired by management) only schedule σ2 will meet the requirement. However if C_{bound} is decreased to 48 (30% of slack from P/w) schedule σ4 will become a solution (besides σ2) meeting Z_{min} with a relatively good customer service level (75% on time jobs). Clearly the tradeoff is a larger scheduling time window. Thus, based on this example and the four schedules presented, there are tradeoffs between the objective functions, and multiple solutions can be selected for implementation, all limited by minimum satisfaction levels for the workers and the size of the time window.

Figure 1. Four possible schedules



3. SOLUTION APPROACHES

The solution approaches proposed in this section are *worker to job* and *job to worker* assignment heuristics that combine various worker selection strategies with job selection strategies. Given that the number of late jobs in a single machine (worker) is optimally solved by Moore's algorithm (1968), the problem's key decision process is which jobs are assigned to which worker and not the sequence of jobs per worker. However, it is obvious that the assignment of jobs to workers will have an effect on the total number of jobs that are completed on time by each worker and in their satisfaction scores. We first present two heuristics to generate the baseline schedules followed by two improvement algorithms, the goal of the later to generate schedules that meet the C_{bound} and Z_{min} constraints.

The proposed solution approaches are iterative in nature and multiple schedules are generated in the process. We consider the analysis of the heuristics based on the non-dominated solution set each heuristic version generates. Considering two schedules, the set is non-dominated if $Z_{ave}(\sigma1) \geq Z_{ave}(\sigma2)$

and $O_t(\sigma 1) \leq O_t(\sigma 2)$ with at least one inequality being true, or $Z_{ave}(\sigma 1) \leq Z_{ave}(\sigma 2)$ and $O_t(\sigma 1) \geq O_t(\sigma 2)$ with at least one inequality being true. The evaluation approach for the non-dominated solution sets is discussed in the next section.

3.1 Notation

C_k	Current processing time load in worker k, i.e. $C_k = \mathbf{max}_{j \in N_k} c_j$.
n_k	Number of jobs assigned to worker k.
N_k	Set of jobs assigned to worker k.
N'	Set of all unassigned jobs.
W'	Set of available workers.
L	Set of temporarily un-assignable jobs.
Q	Set of schedules (solutions).
Bound	Current makespan bound.

3.2Heuristic BinIterative

Heuristic *BinIterative* first selects the worker to be scheduled, then the job to be assigned. The process does not guarantee that a feasible solution will be found (i.e. a solution that meets the Z_{min} and C_{bound} constraints), an issue addressed later by the improvement algorithms (and also in the evaluation methodology and prototype sections of the paper). There will be four variants of the *BinIterative* heuristic based on the rules used to select the workers and jobs. The process iterates the target completion time as to create multiple Pareto Efficient schedules. We note that unless mentioned, ties are solved arbitrarily.

```
Step 0.    Let $Q = \emptyset$ and $Bound = C_{bound}$.
Step 1.    Let $N' = N$, $L = \emptyset$, and $W' = W$.
Step 2.    Select worker $k$ from $W'$ by worker rule (rule 1: $k = \textbf{arg max}_{k \in W'}\{\sum_{v \in N'} s_{v,k}\}$; rule 2: $k =$
           $\textbf{arg min}_{k \in W'}\{\sum_{v \in N'} s_{v,k}\}$). Remove worker $k$ from $W'$.
Step 3.    Select job $j$ from $N'$ by job rule (rule 1: $j = \textbf{arg max}_{j \in N'}\{s_{j,k}/p_v: p_j \leq Bound - C_k\}$ solving
           ties by smallest $\{d_j\}$; rule 2: $j = \textbf{arg min}_{j \in N'}\{d_j: p_j \leq Bound - C_k\}$, solving ties by largest
           $\{s_{j,k}/p_j\}$). If no job is found then go to Step 7.
Step 4.    Remove job $j$ from $N'$ and assign job $j$ to worker $k$: $N' = N' - \{j\}$ and $N_k = N_k \cup \{j\}$. Order
           the jobs in $N_k$ by non-decreasing due date, this is temporary sequence with $f$ late jobs.
Step 5.    If $f > 0$ then generate $n_k$ sequences by temporary removing one at a time each of the jobs
           assigned to worker $k$ (each temporary sequence would have $n_k - 1$ jobs). Select the
           sequence that has no late jobs, breaking ties by maximum $z_k$. Let $l$ be the job from $N_k$ not
           included in the selected sequence. Let $N_k = N_k - \{l\}$ and $L = L \cup \{l\}$.
Step 6.    If $N' \neq \emptyset$ then go to Step 3.
Step 7.    Let $N' = N' \cup L$ and $L = \emptyset$. If $W' \neq \emptyset$ and $N' \neq \emptyset$ go to Step 2.
Step 8.    If $N' \neq \emptyset$ then let $W^*$ be the set of all workers $h$ from $W'$ with $z_h < Z_{min}$. Select $j$ from $N'$
           by $j = \textbf{arg min}_{j \in N'}\{d_j\}$.Generate a set of temporary sequences by assigning job $j$ to each
           worker $h$ from $W^*$, ordering each temporary sequence by Moore's algorithm. Select the
           worker $k$ with temporary sequence with smallest increase in late jobs, solve ties by smallest
           $\{C_h\}$, then by largest $\{z_h\}$.
```

Step 9	Remove j from N' and assign to worker k. Sort jobs in N_k by Moore's algorithm.
Step 10.	If $N' \neq \varnothing$ then go to Step 8.
Step 11.	Add the generated schedule to set Q. Remove all dominated schedules from Q.
Step 12.	If $Bound > \lceil P / w \rceil$ then let $Bound = Bound - 1$ and go to Step 1. Else End.

The *BinIterative* heuristic is further explained next. Step 0 initializes the target processing load bound for all workers (*Bound*) and the set of solutions (Q) to an empty set. Step 1 defines the set of unassigned jobs (N'), the set of un-assignable jobs (L), and the set of available workers (W'). In Step 2 the *worker to load* (i.e. k) is selected based on one of two rules: rule 1 selects the worker with the highest overall satisfaction for the unscheduled jobs, and rule 2 selects the worker with the least total satisfaction for the unscheduled jobs. Step 3 selects the job to be assigned to the *worker to load* based on two rules: rule 1selects the job with maximum contribution to the worker's satisfaction and rule 2 selects the job with earliest due date. In all cases, adding this job must not violate the *Bound* limitation. If a job is found, then in Step 4, the job is added to the *worker to load* and the sequence is ordered by the job's due date. Step 5 generates multiple sequences for the *worker to load* when adding the selected job results in at least one late job. Each temporary sequence is based on removing one of the jobs assigned to the *worker to load*. The sequence with no late jobs and maximum satisfaction is selected and the removed job placed on the un-assignable set of jobs (L). Step 6 serves as a loop control, where if there are still unassigned jobs the process returns to Step 3 (to select the next job to be assigned). Step 7 is reached when the temporary set of unassigned jobs (N') is empty (jobs have been assigned to workers or to set L) or if no job was found to meet the constraints in Step 3. In Step 7, the set of unassigned jobs is redefined to include the jobs that were un-assignable (L) in this loop, and if this set is not empty and there are workers that have not been assigned jobs, the process returns to Step 2 (select next worker to be assigned jobs). Step 8 is reached when all workers have received an initial assignment of jobs, but there are still some unassigned jobs. In this step the set of workers that do not meet the Z_{min} constraint is created and a job and worker selected by four cases that prioritize workers that do not meet the Z_{min} constraint and job assignments that do not violate the C_{Bound} restriction. However, in Step 9 assignments can be made that violate the C_{bound} constraint. Step 8 is repeated until all unassigned jobs have been assigned to one of the workers (Step 10 serves as loop control). Step 11 adds the generated schedule to the set of solutions and dominated solutions are eliminated. Step 12 redefines the target bound and restarts the process, limited by a perfect balance across the workers.

3.3 Heuristic JobIterative

This heuristic first selects the job to be scheduled and considers all available workers simultaneously for its assignment. As in the case of *BinIterative*, the process does not guarantee a solution that meets the Z_{min} and C_{bound} constraints. Four versions of this heuristic are also implemented.

Step 0.	Let $Q = \varnothing$ and $Bound = C_{bound}$.
Step 1.	Let $N' = N$.
Step 2.	Select job j from N' by *job rule* (*job rule 1*: $j = \textbf{\textit{arg max}}_{j \in N',\, h \in W'} \{s_{j,h} / p_j\}$, solving ties by smallest $\{d_j\}$; *job rule 2*: $j = \textbf{\textit{arg min}}_{j \in N',\, h \in W'} \{d_j\}$, solving ties by largest $\{s_{j,h} / p_j\}$).
Step 3.	Generate w' temporary sequences by assigning job j to each worker k from W', ordering each temporary sequence by Moore's algorithm.

worker selection rule 1: Select the worker k with the sequence that has the smallest increase in late jobs with a workload that satisfies the bound, $k = \textbf{arg min}_{k \in W'} \{u'_k - u_k: C_k \leq Bound\}$, solve ties by largest $\{z_k\}$, then by largest $\{C_k\}$. If k is not found, select the worker k with the sequence that has the smallest increase in late jobs, $k = \textbf{arg min}_{k \in W'} \{u'_k - u_k\}$, solve ties by smallest $\{C_k\}$, then by largest $\{z_k\}$.

worker selection rule 2: Select the worker k with temporary sequence with largest satisfaction and satisfies the bound, $k = \textbf{arg max}_{k \in W'} \{z_k : C_k \leq Bound\}$, solve ties by smallest $\{u'_k - u_k\}$, then by largest C_k. If k is not found, select the worker k with the sequence that has the maximum satisfaction, $k = \textbf{arg max}_{k \in W'} \{z_k\}$, solve ties by smallest C_k then by smallest $\{u'_k - u_k\}$.

Step 4. Remove j from N' and assign to worker k. Sort jobs in N_k by Moore's algorithm.
Step 5. If $N' \neq \emptyset$ then go to Step 2.
Step 6. Add the generated schedule to set Q. Remove all dominated schedules from Q.
Step 7. If $Bound > \lceil P / w \rceil$ then let $Bound = Bound - 1$ and go to Step 1. Else End.

The *JobIterative* heuristic is further clarified next. Step 0 initializes the target processing load bound for all workers (*Bound*) and the set of solutions (Q). Step 1 defines the set of unassigned jobs (N') and the set of available workers (W'). Step 2 selects the job to be assigned based on two rules: rule 1 selects the job with maximum contribution to the worker's satisfaction (any of the available workers) and rule 2 selects the job with earliest due date. Step 3 determines the worker that will be assigned the current job based on one of two rules: rule 1 the assignment that maximizes on time jobs and rule 2, the assignment that maximizes worker satisfaction. Step 4 assigns the job to the selected worker and Step 5 updates the set of available workers serves as a loop control, where if there are still unassigned jobs the process returns to Step 2 (to select the next job to be assigned). Steps 6 and 7 are similar to Steps 11 and 12 of *BinIterative*.

4.4 Improvement Heuristics

Both of the just described heuristics aim at meeting problem constraints and maximizing the objective functions, however the steps in these heuristics do not guarantee that the z_{min} and C_{bound} constraints will be met. Instead the processes guarantee that a schedule where all the jobs in N are scheduled in W is built. This subsection presents two algorithms based on neighborhood search that are used to "move" each of the solutions in a set Q to meet the mentioned constraints.

Heuristic Meet_C_{bound}
Input: Schedule with all jobs from N assigned to the workers in W.
Step 1. Let X be an empty set of schedules.
Step 2. Select k from W; $k = \textbf{arg max}_{k \in W} \{C_k\}$. If $C_k \leq C_{bound}$ then End.
Step 3. Let $\Phi = C_k$ and $W^* = W - \{k\}$.
Step 4. Generate temporary schedules by

Step 4.1 Single job removals from k: Removing each job j from worker k and assigning j to each worker h in W^*, considering only cases where $C_h + p_j \leq C_{bound}$; ordering all the jobs in workers h and k by Moore's algorithm. A maximum of $|N_k| \times (w - 1)$ temporary schedules are generated in this step; add all temporary schedules to set X.

Step 4.2 Pairwise exchanges: Exchanging each job j assigned to worker k with each job i assigned to worker h, for all workers h from W^*, considering only the exchanges where $C_h + p_j - p_i \leq C_{bound}$, ordering all the jobs in workers h and k by Moore's algorithm. A maximum of $|N_k| \times |N - N_k|$ temporary schedule are generated in this step; add all temporary schedules to set X.

Step 5. Select the schedule in set X with the largest C_k , $C_k \leq C_{bound}$, solving ties by smallest O_t, then by Z_{ave}. If a schedule is found then this is the current schedule and go to Step 1.

Step 6. Select the schedule in set X with the smallest C_k, solving ties by smallest O_t, then by Z_{ave}. If $C_k < \Phi$ then this is the current schedule and go to Step 1.

Step 7. End.

The *Meet_C*$_{bound}$ heuristic is further clarified next. Step 1 defines a set of empty schedules (X). Step 2 determines if there is a worker with a workload that exceeds the C_{bound} constraint, and if this is not the case the heuristic ends, otherwise this worker becomes the *worker to improve* (i.e. k). Step 3 defines a variable to track the current makespan and a set of workers that meet the C_{bound} constraint. Step 4 generates schedules with all the pairwise interchanges between the *worker to improve* and all other workers. Step 5 selects from the schedules generated in Step 4 the schedule that results in an exchange where both "participating" workers have loads that meet the C_{bound} constraint. If such a schedule is found the process goes back to Step 1, else Step 6 selects from the schedules generated in Step 4 the schedule where the interchange results in a reduction in the load of the *worker to improve* (worker k) and where the load in the *other* "participating" worker does not exceed C_{bound}. If such a schedule is found the process goes back to Step 1. Step 7 is reached if no improvement was found and the current schedule cannot be enhanced further.

Heuristic Meet_Z$_{min}$

Input: Schedule with all jobs from N assigned to the workers in W and $C_{max} \leq C_{bound}$.

Step 1. Select k from W; $k = \textbf{arg min}_{k \in W} z_k$. If $z_k \geq Z_{min}$ then End.

Step 2. Let $\Phi = z_k$, $W' = W - \{k\}$.

Step 3. Let X be an empty set of schedules.

Step 4. Generate temporary schedules by

Step 4.1 Single job removals from k: Removing each job j from worker k and assigning j to each worker u in W', considering only cases where $s_{j,k} / p_j < Z_{min}$, $C_u + p_j \leq C_{bound}$, $(s_{j,u} + z_u \times C_u)/(C_u + p_j) \geq Z_{min}$; then ordering all the jobs in workers u and k by Moore's algorithm. A maximum of $|N_k| \times (w - 1)$ temporary schedules are generated in this step; add all temporary schedules to set X.

Step 4.2 Single job insertions into k: Removing each job i assigned to worker u and assigning this job to worker k, for all workers u from W', considering only cases where $s_{i,k} / p_i > Z_{min}$ and $C_k + p_i \leq C_{bound}$, $(z_u \times C_u - s_{i,u} \ /(C_u - \ p_i) \geq Z_{min}$; then ordering all the jobs in workers u and k by Moore's algorithm. A maximum of $|N - N_k|$ temporary schedules are generated in this step; add all temporary schedules to set X.

Step 4.3 Pairwise exchanges: Exchanging each job j assigned to worker k with each job i assigned to worker u, for all workers u from W', considering only the exchanges where $s_{i,k} / p_i > s_{j,k} / p_j$, $C_u + p_j - p_i \leq C_{bound}$, $v_k - p_j + p_i \leq C_{bound}$, $(s_{j,u} - s_{i,u} + z_u \times C_u)/(C_u + p_j - p_i) \geq Z_{min}$; then ordering all the jobs in workers u and k by Moore's algorithm. A maximum of $|N_k| \times |N - N_k|$ temporary schedule are generated in this step; add all temporary schedules to set X.

Step 5. Select the schedule in set X with largest O_t and $z_k \geq Z_{min}$, solving ties by largest Z_{ave}. If no schedule in X is found, select the schedule with largest z_k (where $z_k > \Phi$); solving ties by largest O_t, then by Z_{ave}. If a schedule is found then this is the current schedule and go to Step 1.

Step 6. End.

The *Meet_*Z_{min} heuristic is further clarified next. Step 1 determines if there is a worker with a satisfaction below the Z_{min} constraint, and if this is not the case the heuristic ends, otherwise this worker becomes the *worker to improve* (i.e. k). Step 2 defines a variable to track the satisfaction score of the *worker to improve* and Step 3 defines a set of empty schedules (X). Step 4 generates all the schedules with single job insertions/removals and pairwise interchanges between the *worker to improve* and all other workers; but only schedules that will meet the C_{bound} constraint. Step 5 selects from the schedules generated in Step 4 the schedule that results in an insertion/removal/exchange where the satisfaction of the *worker to improve* meets the Z_{min} constraint, and if there are multiple schedules the one with maximum on time jobs. If no schedule generated in Step 4 results in a satisfaction score for the *worker to improve* that meets the Z_{min} constraint, the schedule with the largest satisfaction for that worker would be selected, given its an improvement over the initial value. If a schedule is found, the process goes back to Step 1. Step 6 is reached if no improvement was found and the current schedule cannot be enhanced further (thus not a feasible schedule).

4.5 Overall solution process

The overall solution process is as follows.

1. Implement a main heuristic, *BinIterative* (BI) or *JobIterative* (JI), with a specific job and worker rule. This generates a non-dominated solution set (ie. Q). For example, implementing *BinIterative* with job rule (jr) 1 and worker rule (wr) 1 will result in the non-dominated solution set: $Q^{BI\text{-}jr1\text{-}wr1}$.
2. For each solution in the set Q implement *Meet_*C_{bound}. Eliminate all solutions in Q that do not meet the C_{bound} constraints and any dominated solutions.
3. For each solution in each set Q implement *Meet_*Z_{min}. Eliminate all solutions in Q that do not meet the Z_{min} constraint and any dominated solutions.

It is possible that a set Q is empty at the end of this process, thus the heuristic was unable to generate a feasible solution. The process is repeated eight times, resulting in a total of eight non-dominated solution sets: $Q^{\text{BI-jr1-wr1}}$, $Q^{\text{BI-jr2-wr1}}$, …, $Q^{\text{BI-jr2-wr2}}$, $Q^{\text{JI-jr1-wr1}}$,…, $Q^{\text{JI-jr2-wr2}}$.

4. EVALUATION METHODOLOGY

The evaluation of heuristics considering non-dominated solution sets in bi-criteria problems has been addressed using several methods (Ruiz-Torres and Barton, 2001, Carlyle et al., 2003, Ruiz-Torres and Lopez, 2004). In this paper we use a modified version of the DEV method proposed in Ruiz-Torres and

Barton (2001) which is based on the deviation from the solution set generated by a heuristic to each solution in the benchmark set.

A basic element of the evaluation methodology is the comparison of the solution set generated by a heuristic versus the benchmark set of solutions for a problem instance. This benchmark set of solutions is populated by either the optimal solutions or the best found (which could include none, some, or all of the optimal solutions). In problems with a small search space where for example full enumeration could be used to generate all the solutions to a problem instance, the benchmark set would be composed of all the optimal non-dominated solutions for a problem instance. For problem instances where finding the optimal set of solutions is unfeasible (ie. extensive computational times) the benchmark set is populated by the combination of the solution sets from all heuristics. The later applies in this research and let the benchmark set β equal the non-dominated solutions from $\{Q^{\text{BI-jr1-wr1}} \cup \ldots \cup Q^{\text{BI-jr2-wr2}} \cup Q^{\text{JI-jr1-wr1}} \cup \ldots \cup Q^{\text{JI-jr2-wr2}}\}$. It is assumed that β is populated by at least one feasible solution, otherwise the problem instance is defined as unsolved.

The DEV method in Ruiz-Torres and Barton (2001) evaluates each of the solutions in a heuristic set by measuring their deviation from the solutions in the benchmark set. The method assumes each solution in Q can be compared to each solution in β by worsening the solution in Q if it does not fall within the dominated region. Modifications to DEV were needed as it was assumed that Q was always populated by at least one solution, which as mentioned earlier, may not be true in this problem. In this version, and as in the FDH based method used in Ruiz-Torres and Lopez (2004), the efficiency of the solutions is measured instead of the deviation, where the maximum value is 1, and when no solution populates Q, the efficiency is 0. The efficiency of a solution α from β when compared to a solution ω from Q is $e_{\alpha,\omega}$ and determined by $(\boldsymbol{min}\,[O_t(\omega)/O_t(\alpha), 1] + \boldsymbol{min}\,[Z_{ave}(\omega)/Z_{ave}(\alpha), 1])/2$. The efficiency score for a solution α from β given set Q is based on the best performing solution from Q, thus $\boldsymbol{max}_{\,\omega \in Q}\, e_{\alpha,\omega}$. The average efficiency $E_{Q\beta}$ of set Q when compared to set β is $\sum_{\alpha \in \beta} [\, \boldsymbol{max}_{\,\omega \in Q}\, e_{\alpha,\omega}] \,/\, |\beta|$. It is noted that both Z_{ave} and O_t have minimum levels above 0 and therefore avoids divisions by 0 (in our implementation $z_{min} > 0$ and O_t will always be $\geq w/n$ as at least there will be w on time jobs given $d_j \geq p_j$ for all jobs).

An example is used to illustrate the evaluation methodology. Assuming three heuristics with sets Q^{h1} = {(67%, 3.3), {50%, 5.2}, {30%, 6.2}}, Q^{h2} = {{90%, 4.1}, {30%, 5.2}}, and Q^{h3} = {{70%, 4.6}, {50%, 5.6}, {25%, 5.9}}. By combining the three sets and eliminating the dominated solutions the benchmark set is obtained, β = {{90%, 4.1}, {70%, 4.6}, {50%, 5.6}, {30%, 6.2}}. The efficiency calculations for sets Q^{h1} and Q^{h2} are presented in Tables 3 and 4 respectively. The average efficiency for the three heuristic sets are 0.9, 0.93 and 0.96 respectively, thus Q^{h3} is the best set of the three under the proposed evaluation method.

β	Q^{h1}	***min*** $[O_t(\omega)/O_t(\alpha), 1]$	***min*** $[Z_{ave}(\omega)/Z_{ave}(\alpha), 1]$	$e_{\alpha,\omega}$	***max*** $_{\omega \in Q}\, e_{\alpha,\omega}$
α = {90%, 4.1}	ω = {67%, 3.3}	0.74	0.80	0.77	0.78
	ω = {50%, 5.2}	0.56	1.00	0.78	
	ω = {30%, 6.2}	0.33	1.00	0.67	
α = {70%, 4.6}	ω = {67%, 3.3}	0.96	0.72	0.84	0.86
	ω = {50%, 5.2}	0.71	1.00	0.86	
	ω = {30%, 6.2}	0.43	1.00	0.71	
α = {50%, 5.6}	ω = {67%, 3.3}	1.00	0.59	0.79	0.96
	ω = {50%, 5.2}	1.00	0.93	0.96	
	ω = {30%, 6.2}	0.60	1.00	0.80	
α = {30%, 6.2}	ω = {67%, 3.3}	1.00	0.53	0.77	1.00
	ω = {50%, 5.2}	1.00	0.84	0.92	
	ω = {30%, 6.2}	1.00	1.00	1.00	

Table 3. Efficiency calculations for set Q^{h1}

β	Q^{h2}	***min*** $[O_t(\omega)/O_t(\alpha), 1]$	***min*** $[Z_{ave}(\omega)/Z_{ave}(\alpha), 1]$	$e_{\alpha,\omega}$	***max*** $_{\omega \in Q}\, e_{\alpha,\omega}$
α = {90%, 4.1}	ω = {90%, 4.1}	1.00	1.00	1.00	1.00
	ω = {30%, 5.2}	0.33	1.00	0.67	
α = {70%, 4.6}	ω = {90%, 4.1}	1.00	0.89	0.95	0.95
	ω = {30%, 5.2}	0.43	1.00	0.71	
α = {50%, 5.6}	ω = {90%, 4.1}	1.00	0.73	0.87	0.87
	ω = {30%, 5.2}	0.60	0.93	0.76	
α = {30%, 6.2}	ω = {90%, 4.1}	1.00	0.66	0.83	0.92
	ω = {30%, 5.2}	1.00	0.84	0.92	

Table 4. Efficiency calculations for set Q^{h2}

5. COMPUTATIONAL EXPERIMENTS

The computational experiments are based on previous research in parallel machines which consider as factors the number of jobs and workers (typically referred to as the number of machines) and due date tightness. Factors relevant to the particular problem under study are the number of product types, the worker to job types preference ratings, and the tightness of the C_{bound} and Z_{min} constraints.

The factor w is considered at two levels: 5 and 10, while the number of jobs is based on the n/w ratio, equal to 10 and 20, thus the minimum n is 50 and the maximum is 200. The job's process times are generated by a discrete uniform distribution (***DU***) with range 1 to 99, thus an average process time of 50. The due date tightness is based on the due date tightness ratio (*ddtr*) as in Ho and Chang (1995); and a higher *ddtr* would result in a larger number of late jobs. Let $D_{max} = 50 \times n / (w \times ddtr)$, and the due date of a job be determined by p_j+ ***DU***(0, D_{max}). This due date assignment method guarantees that each job will be on-time if placed at the start of the schedule of any worker, therefore there are at least w on time jobs. We consider *ddtr* at 1 and 3, which are in line with the experiments by Ho and Chang (1995) and were selected given in pilot experiments result in "reasonable" on time percentages.

In regards to the satisfaction related parameters, we maintain the range used in the example (r_{max} = 7 and r_{min} = 1). The number of job types b is considered at two levels, 5 and 10 and the probability that a job is of a type is 1/b. The preference rating between each worker k job type t_j, $r_{k,tj}$ is generated by a discrete function of the form: [probability, preference rating value, probability, preference rating value,

…]. We define three cases of this function: case 1: [5%, 1, 10%, 2, 20%, 3, 30%, 4, 20%, 5, 10%, 6, and 5%, 7]; case 2: [20%, 1, 25%, 2, 15%, 3, 15%, 4, 10%, 5, 10%, 6, and 5%, 7]; and case 3: [5%, 1, 10%, 2, 10%, 3, 15%, 4, 15%, 5, 25%, 6, and 20%, 7]. The first case representing an environment where a majority of the job types provides an acceptable level of satisfaction and only a few are "hated"/ disliked or "loved"/preferred, the second case represents the environment where a majority of the job types are "hated"/ disliked, and a minority "loved"/preferred, and the third case where a majority of the job types are "loved"/preferred and a minority "hated"/disliked. Clearly in terms of difficulty (as to meet the Z_{min} constrain) case 2 is the "hardest", while case 3 is the "easiest". Twenty five problem instances are created for each of the combinations, therefore a total of 1,200 problems.

The final two experimental factors are the *tightness* of the C_{bound} and Z_{min} constraints. We consider C_{bound} at two levels, defined in each instance by 105%P/w and 101%P/w, and Z_{min} at two levels: 3 and 4. Therefore each instance is solved four times, under each of the four combinations of C_{bound} and Z_{min}. This results in a total of 192 combinations.

5.1 Shop Indicators for the benchmark solution sets

As discussed earlier, the benchmark solution set is the non-dominated solution set for a problem instance which combines the schedules generated by all the heuristics. Each benchmark set for a problem instance will have a particular number of solutions, and a range of O_t and Z_{ave} values. The average results by experimental variable are presented in Table 5, noting that all values are based on the instances where at least one feasible schedule was generated (thus a benchmark set exists). The table presents the averages per experimental factor for the number of instances solved (out of 25 per experimental combination), the number of benchmark solutions, the lowest and highest on time percentage, and the lowest and highest average satisfaction. In terms of the average number of instances that were solved, three factors play a noticeable role; the number of product types (b), the cases used to generate the preference ratings, and the minimum satisfaction level (Z_{min}). As there are fewer product types, there are fewer relationships between workers and products and thus fewer alternatives in the scheduling process, thus limiting what can be done to meet the C_{bound} and Z_{min} constraints. Similarly, as more of the relationships between workers and types are of "dislike" (case 2), the possibility of generating schedules that meet the Z_{min} constraint is reduced, and alternative, as there are more "like" relationships, the probabilities increase (at least one feasible schedule was generated for a 100% of the case 3 experiments. Finally, it is obvious that as Z_{min} decreases (and/or C_{bound} decreases) the possibility of finding feasible schedules is reduced. The experiments with Z_{min} = 4, case 2, and b =5 had as an average 10.75 instances solved, thus under those experimental conditions more than half of the problems none of the heuristics was able to generate even a single schedule that met the C_{bound} and Z_{min} constraints.

Three factors affected the average number of benchmark solutions, a problem metric which is relevant as it gives a wider set of options to the decision makers. As the problem size increased, defined by a higher number of workers and jobs to schedule, the number of alternative schedules increased, which is an intuitive result. Another intuitive result was that a larger C_{bound} resulted in a higher number of benchmark solutions, this as it allows more flexibility in allocation jobs among the workers (e.g. having late jobs assigned to workers with high preference ratings for those types at the end of the schedule while some other workers have no assigned jobs). When w = 5, n/w = 10 and C_{bound} = 1%, the average number of benchmark solutions is 2.23, while when w = 10, n/w = 20 and C_{bound} = 5%, the average number of benchmark solutions is 5.69. Regarding the lowest and highest on time percentages, only the *ddtr* parameter had an effect, which is expected as it controls the due date tightness. Also as expected, only the cases used to generate the preference had an effect on the average satisfaction scores. It is interesting to note that a higher Z_{min} did not increase the range for the average satisfaction scores (Z_{ave}).

Factor	Level	Number of instances solved	Number of benchmark solutions	lowest *Ot%*	highest *Ot%*	lowest Z_{ave}	highest Z_{ave}
w	5	22.8	3.1	73%	77%	4.9	5.3
	10	22.7	4.8	71%	77%	5.0	5.5
n/*w*	10	22.4	3.3	72%	77%	5.0	5.4
	20	23.1	4.5	71%	77%	4.9	5.4
ddtr	1	22.8	4.1	90%	96%	5.0	5.4
	3	22.8	3.8	53%	58%	4.9	5.4
b	5	21.5	4.0	71%	77%	4.9	5.3
	10	24.1	3.9	72%	77%	5.0	5.5
$r_{k, tj}$	case 1	24.3	4.0	71%	77%	4.7	5.2
	case 2	19.0	3.6	72%	77%	4.5	4.9
	case 3	25.0	4.1	71%	77%	5.6	6.2
Z_{min}	3	24.2	4.1	71%	77%	4.9	5.4
	4	21.4	3.8	72%	77%	5.0	5.4
C_{bound}	101%*P*/*w*	22.4	3.5	72%	76%	4.9	5.4
	105%*P*/*w*	23.1	4.3	71%	77%	5.0	5.4

Table 5. Performance indicators for the benchmark set.

5.2 Heuristic Performance - Efficiency Scores and Percentage of Benchmark Solution

The efficiency score results per experimental factor for the eight heuristics are presented in Table 6. Based on those instances that each heuristic was able to solve, the two best performing heuristics are BI-jr1-wr1 and BI-jr2-wr1 with overall efficiency scores of 0.977 and 0.980, followed by BI-jr1-wr2 with an average score of 0.965. The remaining five heuristic have very similar efficiency averages, ranging from 0.957 to 0.959.

Three of the experimental factors have an effect on the best performing heuristic: the number of workers, the congestion ratio, and the number of product types. At *w* = 5, heuristic BI-jr2-wr1 dominates while at *w* = 10, both BI-jr1-wr1 and Bi-jr2-wr1 outperform the rest. The results for the *ddtr* factor are interesting as at *ddtr* = 1 a heuristic based on the *JobIterative* approach dominates, in this case performing at a similar level to BI-jr1-wr1. Finally, in regards to the number of product types, at *b* = 5 there is no notable difference between BI-jr1-wr1 and BI-jr2-wr1, while at *b* = 10, heuristic BI-jr2-wr1 outperforms all others.

		BinIterative(*BI*)				*JobIterative*(*JI*)			
Factor	Level	jr1-wr1	jr1-wr2	jr2-wr1	jr2-wr2	jr1-wr1	jr1-wr2	jr2-wr1	jr2-wr2
w	5	0.976	0.960	**0.984**	0.956	0.958	0.956	0.960	0.956
	10	**0.978**	0.969	0.977	0.960	0.955	0.961	0.958	0.961
n/w	10	0.978	0.969	**0.979**	0.960	0.954	0.954	0.957	0.954
	20	0.976	0.960	**0.982**	0.956	0.959	0.963	0.961	0.963
ddtr	1	**0.982**	0.959	0.974	0.949	0.976	0.962	**0.982**	0.962
	3	0.972	0.970	**0.987**	0.967	0.937	0.955	0.935	0.955
b	5	**0.979**	0.968	0.977	0.956	0.959	0.962	0.962	0.963
	10	0.975	0.961	**0.984**	0.960	0.954	0.955	0.956	0.954
$r_{k,\,tj}$	case 1	0.979	0.965	**0.981**	0.951	0.960	0.962	0.960	0.959
	case 2	0.975	0.960	**0.982**	0.957	0.959	0.958	0.957	0.954
	case 3	0.976	0.969	**0.979**	0.966	0.950	0.955	0.960	0.962
Z_{min}	3	0.974	0.960	**0.979**	0.954	0.952	0.955	0.955	0.956
	4	0.980	0.969	**0.982**	0.962	0.961	0.962	0.963	0.961
C_{bound}	101%P/w	0.977	0.964	**0.980**	0.959	0.958	0.959	0.961	0.959
	105%P/w	0.977	0.965	**0.980**	0.957	0.955	0.958	0.957	0.958
Overall		**0.977**	**0.965**	**0.980**	**0.958**	**0.957**	**0.959**	**0.959**	**0.958**

Table 6. Efficiency results.

Table 7 presents the results for the average percentage of benchmark solutions generated by each heuristic. The percentages add to more than 100% as more than one heuristic can generate one of the benchmark solutions. All averages above 15% are in bold, and as can be easily noted heuristics BI-jr1-wr1, BI-jr1-wr2, and BI-jr2-wr1 generate a majority of the benchmark solutions. However, the sum of the percentages for these three heuristic is always less then 100%, thus the other heuristics do generate solutions that are part of the benchmark sets. Heuristic JI-jr2-wr1 contributes an overall 14% of the benchmark solutions, and its percentage related to several of the experimental variables, in particular the congestion ratio, where at *ddtr* = 1 it generates 27% of the benchmark solutions, while at *ddtr* = 1 it generates 2% of the benchmark solutions.

		BinIterative(*BI*)				*JobIterative*(*JI*)			
Factor	Level	jr1-wr1	jr1-wr2	jr2-wr1	jr2-wr2	jr1-wr1	jr1-wr2	jr2-wr1	jr2-wr2
w	5	**24%**	**15%**	**36%**	8%	8%	4%	**15%**	5%
	10	**28%**	**21%**	**30%**	10%	7%	5%	14%	5%
n/w	10	**26%**	**21%**	**31%**	11%	8%	3%	**16%**	4%
	20	**25%**	**16%**	**34%**	8%	7%	5%	13%	6%
$ddtr$	1	**28%**	**17%**	**19%**	4%	14%	4%	**27%**	5%
	3	**23%**	**19%**	**46%**	14%	2%	4%	2%	5%
b	5	**28%**	**20%**	**31%**	9%	7%	6%	**15%**	6%
	10	**23%**	**16%**	**34%**	9%	9%	3%	14%	4%
$r_{k,\ tj}$	case 1	**26%**	**16%**	**34%**	4%	9%	5%	10%	3%
	case 2	**26%**	**19%**	**35%**	8%	12%	5%	**16%**	5%
	case 3	**25%**	**19%**	**30%**	**16%**	2%	3%	**18%**	6%
Z_{min}	3	**25%**	**16%**	**32%**	8%	7%	4%	13%	5%
	4	**26%**	**20%**	**34%**	10%	9%	5%	**15%**	5%
C_{bound}	101%P/w	**26%**	**18%**	**32%**	11%	8%	5%	**17%**	6%
	105%P/w	**25%**	**18%**	**33%**	8%	7%	4%	12%	4%
Overall		**26%**	**18%**	**33%**	**9%**	**8%**	**4%**	**14%**	**5%**

Table 7. Percentage of benchmark solutions results.

5.3 Heuristic Performance – Instances solved

The efficiency and percentage of benchmark solutions metrics do not tell the complete picture of heuristic performance as these are based on the problem instances that each of the heuristic solves (generates at least one schedule). Table 8 presents the average number of instances that each heuristic solves per experimental combination (maximum is 25). Heuristics JI-jr2-wr1 and JI-jr2-wr2 are the best performers, finding solutions to an average of 22.3 and 22.2 instances per experimental point (out of 25). When we compare it to the best performing heuristic in terms of efficiency, BI-jr1-wr1 and BI-jr2-wr1, it solves on average one more problem instance per experimental point. The most relevant experimental factor when considering this metric is the cases used to generate the preference rating, where in case 2 (higher percentage of dislike relationships), heuristic JI-jr2-wr1 solves on average 3.3 more instances per experimental point than BI-jr2-wr1 (which is the best performer in terms of the efficiency score for case 2).

		BinIterative(*BI*)				*JobIterative*(*JI*)			
Factor	Level	jr1-wr1	jr1-wr2	jr2-wr1	jr2-wr2	jr1-wr1	jr1-wr2	jr2-wr1	jr2-wr2
w	5	21.0	21.7	20.8	21.9	21.1	21.3	**22.3**	22.1
	10	21.4	21.8	20.6	21.6	21.4	21.7	**22.4**	22.3
n/w	10	20.4	20.9	20.0	21.1	20.4	20.9	**21.7**	21.6
	20	21.9	22.6	21.4	22.4	22.1	22.1	**22.9**	22.8
ddtr	1	20.6	21.4	20.4	21.6	20.5	21.1	**22.6**	22.3
	3	21.8	22.1	20.9	22.0	22.0	21.9	22.0	**22.1**
b	5	19.6	20.3	19.0	20.2	19.6	19.9	**20.9**	20.7
	10	22.8	23.2	22.4	23.3	22.9	23.1	**23.7**	**23.7**
$r_{k, tj}$	case 1	23.0	23.6	22.5	23.6	23.0	23.3	**23.9**	23.8
	case 2	15.7	16.7	14.8	16.8	15.9	16.3	**18.1**	17.8
	case 3	24.8	**24.9**	24.8	**24.9**	24.8	**24.9**	**24.9**	**24.9**
Z_{min}	3	23.4	23.7	23.0	23.7	23.3	23.6	**24.0**	**24.0**
	4	19.0	19.8	18.4	19.8	19.2	19.4	**20.7**	20.4
C_{bound}	101%P/w	19.9	20.8	19.4	20.9	20.0	20.5	**21.7**	21.5
	105%P/w	22.4	22.7	22.0	22.6	22.5	22.5	**23.0**	22.9
Overall		**21.2**	**21.7**	**20.7**	**21.8**	**21.3**	**21.5**	**22.3**	**22.2**

Table 8. Percentage of benchmark solutions results.

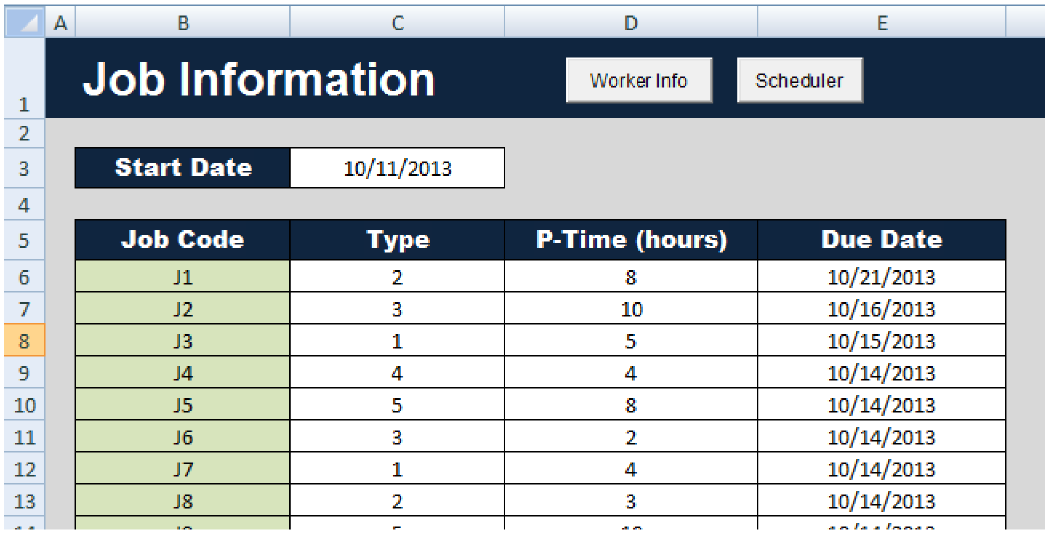
5.4 Concluding Remarks about the heuristics

These results led to the conclusion that no single heuristic can be used to generate the benchmark set and at least a few methods should be used to solve any problem instance. Heuristics BI-jr1-wr1, BI-jr1-wr2, and BI-jr2-wr1 have consistently high efficiency values, although they have limitations in finding feasible solutions, in particular in the condition of many "dislike" worker to product type relationships (case 2). On the other hand, heuristics JI-jr2-wr1 and JI-jr2-wr2 are the best performers in terms of finding solutions for a majority of the instances, but have low efficiency scores when compared to the *BinIterative* approaches.

6. SAMPLE APPLICATION

To demonstrate the problem and production planning process a prototype was developed in Excel ® and Visual Basic for Applications ®. The prototype includes the key information components as well as the outputs that the production planner would use. As for most prototypes, the goal is to demonstrate capabilities and determine areas where additional features may be needed, and it is not intended to be a fully functional software. The implemented prototype used five of the heuristics in the generation of the benchmark sets to be presented to the scheduler (all versions of BinIterative and Ji-J2-W1) and was limited to six workers, forty jobs and five product types.

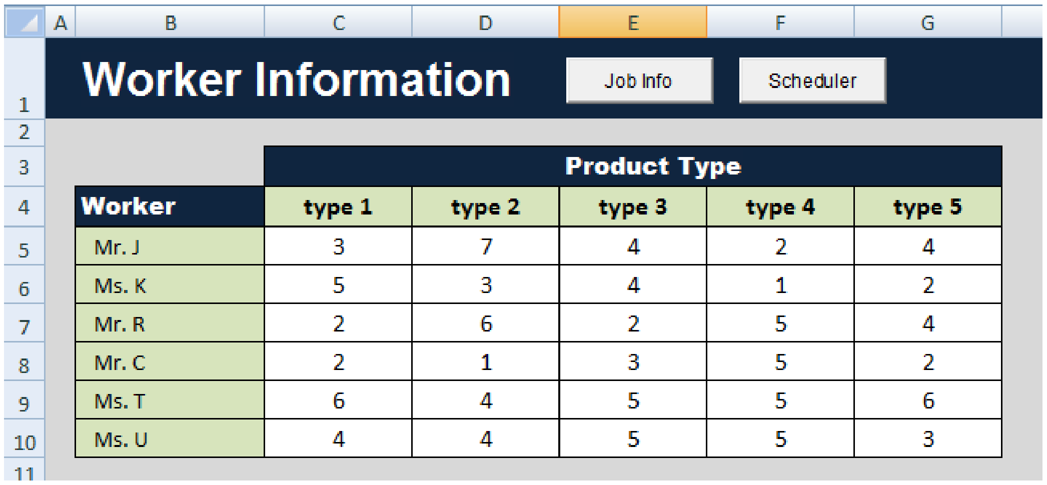
Figures 2 and 3 present the Excel sheets that collect the information related to the workers and to the jobs. Some of the assumptions of the prototype is that the process times are assumed in hours (integer), schedule starts at time 0 of the start date (a calendar date) for all workers, due dates are based on calendar dates (job is due at the end of that calendar date), work is performed 8 hours per day, work can span across multiple days, and there is no work on the weekends. A warning is made to the planner if the start date or any due date is set for a weekend date.



Start Date	10/11/2013

Job Code	Type	P-Time (hours)	Due Date
J1	2	8	10/21/2013
J2	3	10	10/16/2013
J3	1	5	10/15/2013
J4	4	4	10/14/2013
J5	5	8	10/14/2013
J6	3	2	10/14/2013
J7	1	4	10/14/2013
J8	2	3	10/14/2013

Figure 2. Prototype tool snapshot: Job Information sheet.



	Product Type				
Worker	type 1	type 2	type 3	type 4	type 5
Mr. J	3	7	4	2	4
Ms. K	5	3	4	1	2
Mr. R	2	6	2	5	4
Mr. C	2	1	3	5	2
Ms. T	6	4	5	5	6
Ms. U	4	4	5	5	3

Figure 3. Prototype tool snapshot: Worker Information sheet.

Figure 4 presents the left side of the Schedule Manager worksheet. In this page the planner will call for the heuristics to generate production plans. Note that initially the area to the right is empty. The grid on the prototype allows the planner to select the desired level of minimum utilization and minimum satisfaction level for the workers. It is obvious that clicking on the *top left* quadrant of the grid (a button) will provide the less constrained set of schedules, thus the best options in terms of on time performance and average satisfaction.

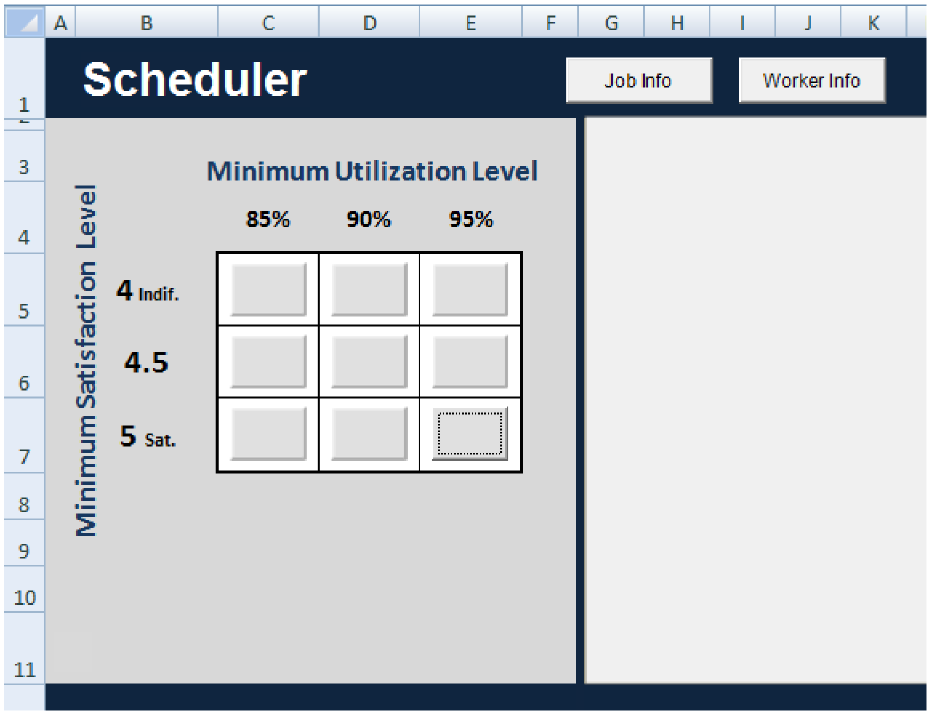
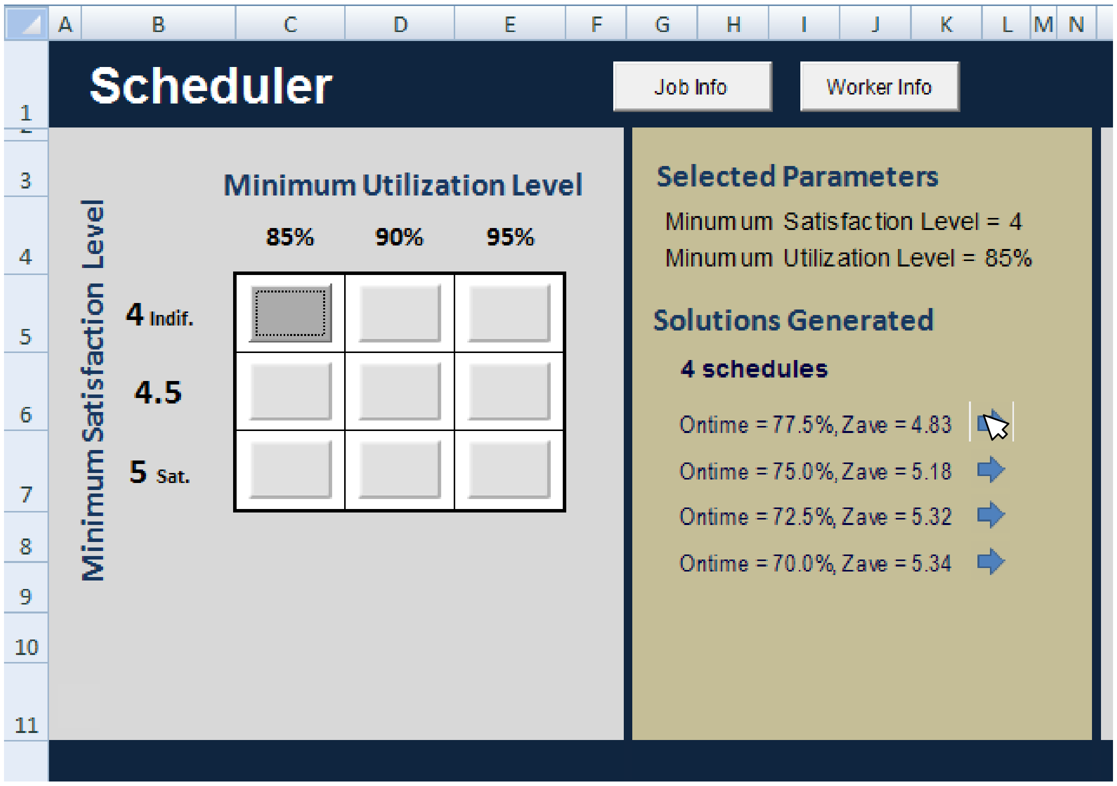


Figure 4. Prototype tool snapshot: Scheduler worksheet – start of process

Figure 5 illustrates what occurs when the user clicks on one of the buttons of the grid, in this case on the top left button. A new display area lists the selected parameters and the solutions generated. The planner can then view any of the schedules generated by the system by clicking on the blue arrows. Figure 6 presents the left side of the sheet (after the planner clicks on one of the blue arrows) and a schedule is shown, in this case the schedule with 70% on time jobs and Z_{ave} = 4.83. The prototype displays the schedule by calendar days and differentiates on time jobs from late jobs by their color. An additional feature is that the user can select to see the job assignments that have very low preference ratings (Figure 7). Finally Figures 8 and 9 present the list of schedules generated when other buttons on the grid to the left are selected. In the case presented in Figure 8, at a minimum satisfaction level of 4.5 and 95% utilization, two schedules were generated and the highest on time percentage is 72.5%, while as shown in Figure 9, at the tightest constraints of minimum satisfaction and utilization (5 and 95% respectively), no schedules that satisfy these constraints were found.

Additional prototype features suggested by possible users include a grid that shows all the schedules simultaneously and the ability to manipulate job to worker assignments on the schedule as to test single jobs moves in order to eliminate very low satisfaction assignments.



Selected Parameters

Minumum Satisfaction Level = 4

Minumum Utilization Level = 85%

Solutions Generated

4 schedules

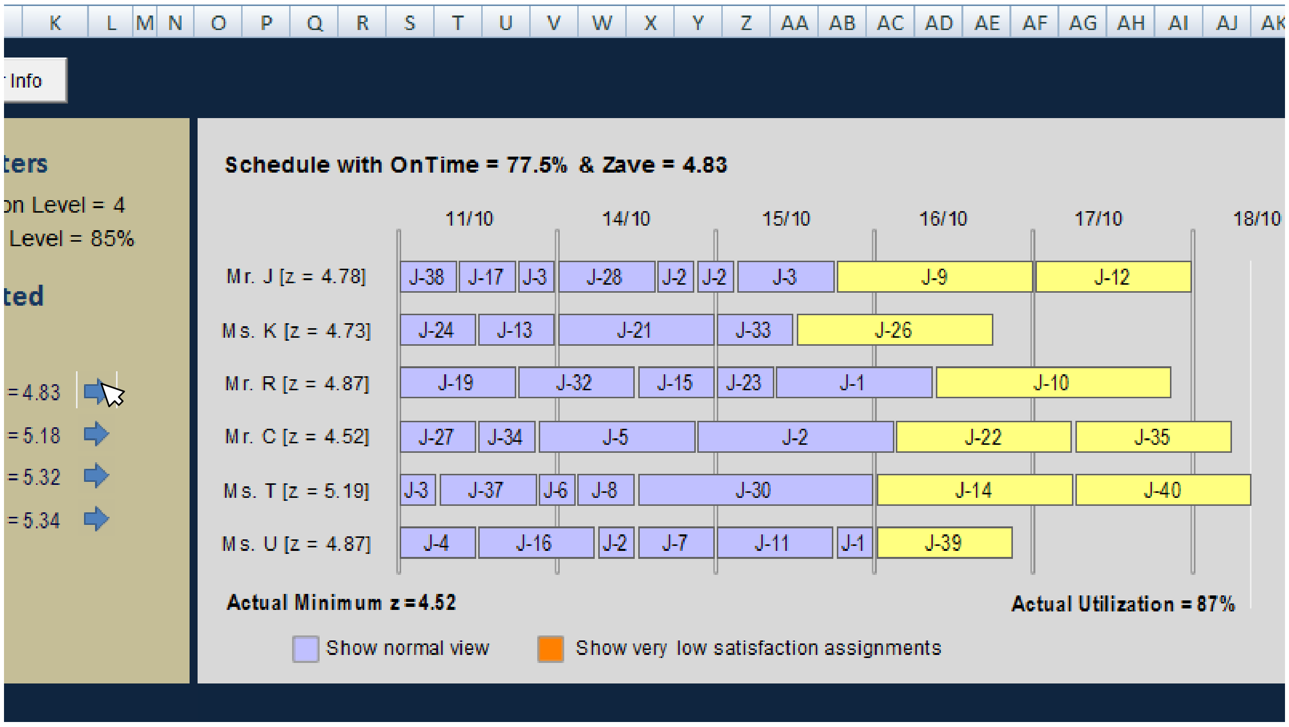
Ontime = 77.5%, Zave = 4.83

Ontime = 75.0%, Zave = 5.18

Ontime = 72.5%, Zave = 5.32

Ontime = 70.0%, Zave = 5.34

Figure 5. Prototype tool snapshot: Schedule Manager worksheet – a list of generated schedules based on the planner's selected parameters.



Schedule with OnTime = 77.5% & Zave = 4.83

Mr. J [z = 4.78] J-38 J-17 J-3 J-28 J-2 J-2 J-3 J-9 J-12

Ms. K [z = 4.73] J-24 J-13 J-21 J-33 J-26

Mr. R [z = 4.87] J-19 J-32 J-15 J-23 J-1 J-10

Mr. C [z = 4.52] J-27 J-34 J-5 J-2 J-22 J-35

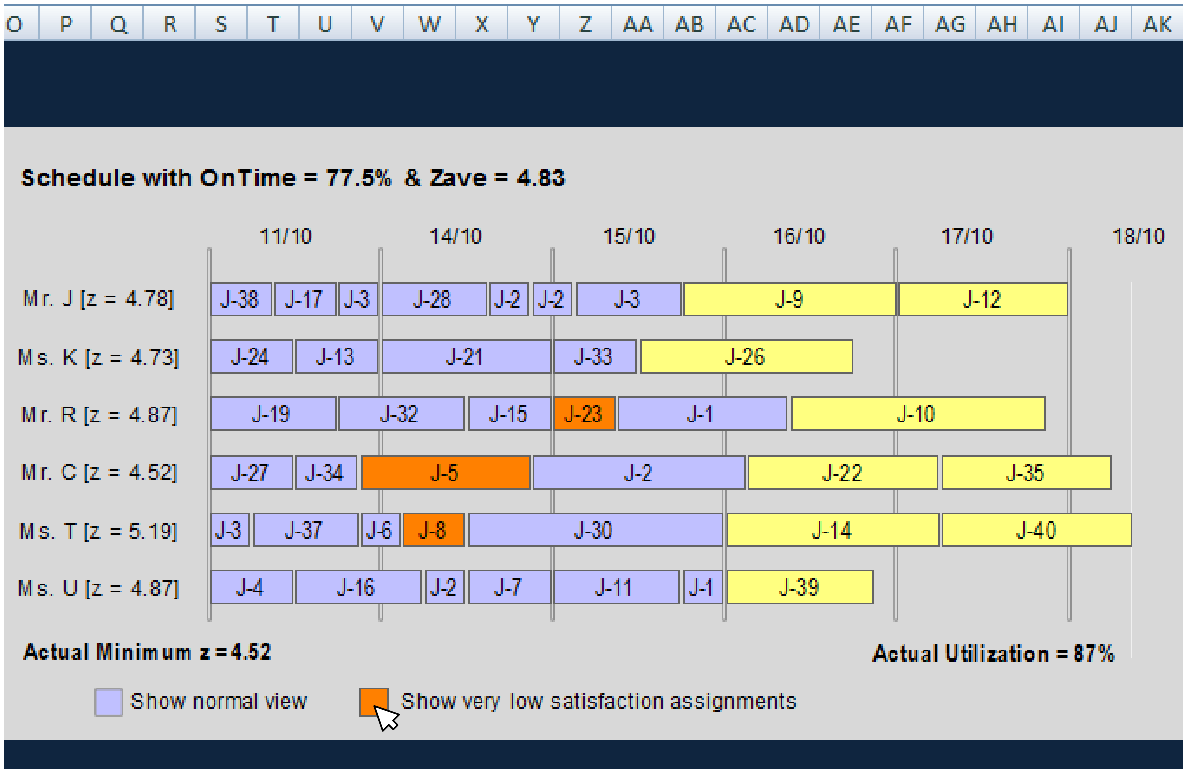
Ms. T [z = 5.19] J-3 J-37 J-6 J-8 J-30 J-14 J-40

Ms. U [z = 4.87] J-4 J-16 J-2 J-7 J-11 J-1 J-39

Actual Minimum z = 4.52

Actual Utilization = 87%

Figure 6. Prototype tool snapshot: Schedule Manager worksheet – showing Gantt chart for the schedule selected by the planner.

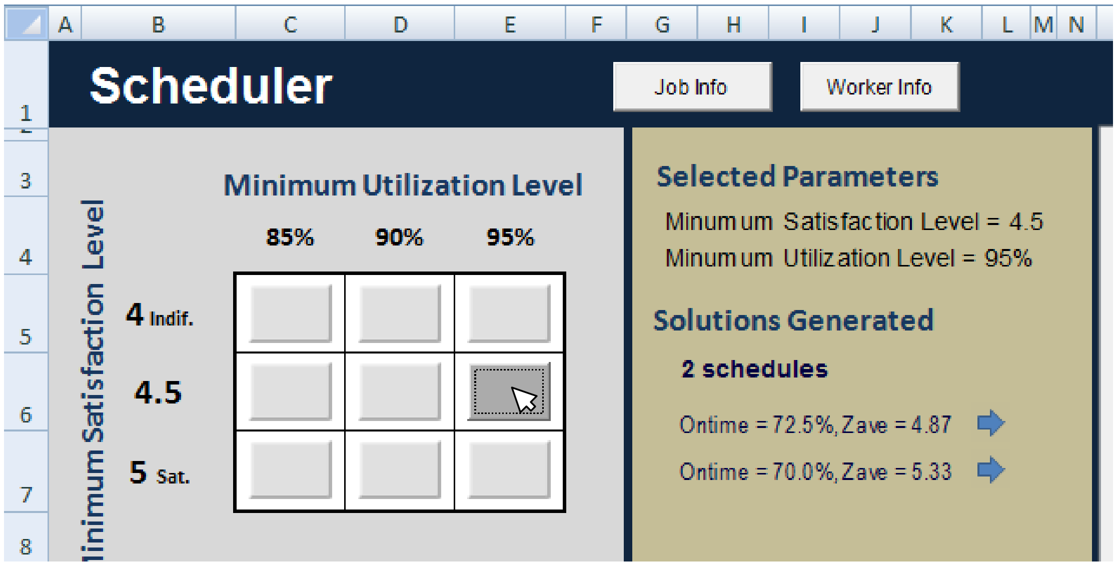

O P Q R S T U V W X Y Z AA AB AC AD AE AF AG AH AI AJ AK
11/10
14/10
15/10
16/10
17/10
18/10
Mr. J [z = 4.78]: J-38 J-17 J-3 J-28 J-2 J-2 J-3 J-9 J-12
Ms. K [z = 4.73]: J-24 J-13 J-21 J-33 J-26
Mr. R [z = 4.87]: J-19 J-32 J-15 J-23 J-1 J-10
Mr. C [z = 4.52]: J-27 J-34 J-5 J-2 J-22 J-35
Ms. T [z = 5.19]: J-3 J-37 J-6 J-8 J-30 J-14 J-40
Ms. U [z = 4.87]: J-4 J-16 J-2 J-7 J-11 J-1 J-39
Show normal view
Show very low satisfaction assignments

Schedule with OnTime = 77.5% & Zave = 4.83

Actual Minimum z = 4.52

Actual Utilization = 87%

Figure 7. Prototype tool snapshot: Schedule Manager worksheet – feature that shows job assignments with very low satisfaction ratings on the Gantt chart.


A B C D E F G H I J K L M N
1 3 4 5 6 7 8
Scheduler
Job Info
Worker Info
Minimum Utilization Level
85%
90%
95%
Minimum Satisfaction Level
4 Indif.
4.5
5 Sat.

Selected Parameters

Minumum Satisfaction Level = 4.5

Minumum Utilization Level = 95%

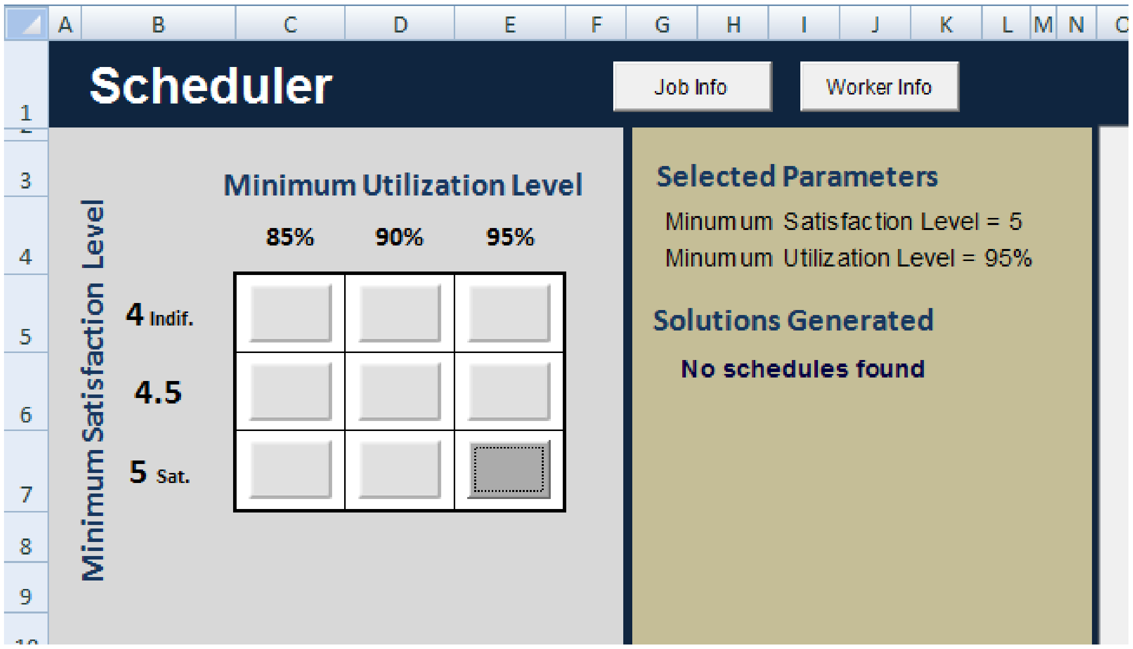
Solutions Generated

2 schedules

Ontime = 72.5%, Zave = 4.87

Ontime = 70.0%, Zave = 5.33

Figure 8. Prototype tool snapshot: Schedule Manager worksheet – change in the parameters (constraints) resulting in a smaller set of feasible schedules.



Scheduler

Minimum Utilization Level

Selected Parameters

Minumum Satisfaction Level = 5

Minumum Utilization Level = 95%

Solutions Generated

No schedules found

Figure 9. Prototype tool snapshot: Schedule Manager worksheet – case where no schedules are found that satisfy the parameters (constraints) selected by the planner.

7. SUMMARY AND FUTURE WORK

This research proposes a scheduling model that considers the preference that workers have towards the work is assigned to them. This consideration is not typically taken into account in the scheduling literature, yet is highly relevant given in most cases happy workers equates to higher productivity (Taris and Schreurs, 2009). The model aims to maximize two objectives simultaneously, one related to customer service (the on time completion of tasks) and the total worker satisfaction. The model also considers an efficiency related constraint (the maximum completion time of all tasks), and minimum worker satisfaction constraint. The paper presents various heuristics and evaluates their performance. Performance evaluation is based on an efficiency score given the two maximization objectives are considered in a non-dominated form. The computational experiments consider as factors the number of worker, number of jobs, job type to worker satisfaction scores, and due date tightness. Results show that a few of the heuristics have consistently high efficiency values, although they have limitations in finding feasible solutions. Finally, a prototype was developed in Excel and Visual Basic for Applications to demonstrate the model capabilities to support scheduling in a real world environment.

Future research in scheduling that considers worker satisfaction can take multiple interesting directions. While this paper proposed an aggregate score based on worker satisfaction averages, other approaches to measure this could certainly be developed including non-linear representations and approached that focus solely on maximizing the minimum worker satisfaction (best for the worst off). Given in many cases worker satisfaction may be seen as a secondary measure, problems that maximize worker satisfaction subject to an optimal “regular” measure of performance are also interesting and useful directions. Finally, problems that consider worker satisfaction and their skill at performing the type of task also represent appealing cases for analysis.

REFERENCES

Akbari, M., Zandieh, M., and Dorri, B. (2012) Scheduling part-time and mixed-skilled workers to maximize employee satisfaction, *The International Journal of Advanced Manufacturing Technology*, DOI: 10.1007/s00170-012-4032-4.

Alsheddy, A., and Tsang, E. (2011) Empowerment scheduling for a field workforce, *Journal of Scheduling*, 14(6), pp. 639-654.

Bornstein, C. T., Alcoforado, L. F., and Maculan, N. (2005) A graph-oriented approach for the minimization of the number of late jobs for the parallel machines scheduling problem, *European journal of operational research*, 165, pp. 649-656.

Carlyle, W. M., Fowler, J. W., Gel, E. S., and Kim, B. (2003) Quantitative Comparison of Approximate Solution Sets for Bi-criteria Optimization Problems. *Decision Sciences*, 34(1), pp. 63-82.

Johnson, J.E., Smith, A.L., and Mastro, K.A. (2012) From Toyota to the Bedside: Nurses Can Lead the Lean Way in Health Care Reform, *Nursing Administration Quarterly*, 36(3), pp. 234-242.

Ho, J. C., and Chang Y. L. (1995) Minimizing the number of tardy jobs for m parallel machines, *European Journal of Operational Research*, 84, pp. 343-355.

Leyvand, Y., Shabtay, D., Steiner, G., and Yedidsion, L. (2010) Just-in-time scheduling with controllable processing times on parallel machines, *Journal of Combinatorial Optimization*, 19(3), pp. 347-368.

Lin, B., and Jeng A. (2004) Parallel-machine batch scheduling to minimize the maximum lateness and the number of tardy jobs, *International Journal of Production Economics*, 91, pp. 121-134.

M'Hallah, R., and Bulfin, R. (2005) Minimizing the weighted number of tardy jobs on parallel processors, *European Journal of Operational Research*, 160, pp. 471-484.

Mohan, S. (2008) Scheduling part-time personnel with availability restrictions and preferences to maximize employee satisfaction, *Mathematical and Computer Modelling*, 48(11–12), pp. 1806-1813.

Moore, J. M. (1968) An n job, one machine sequencing algorithm for minimizing the number of late jobs, *Management Science*, 15, pp. 102-109.

Taris, T. W., and Schreurs, P. J. (2009) Well-being and organizational performance: An organizational-level test of the happy-productive worker hypothesis. *Work & Stress*, *23*(2), pp. 120-136.

Ruiz-Torres, A. J., and Barton, R. (2001) Assessment procedure for multiple Pareto solution sets. In *Sixth International Conference of the Decision Sciences Institute, Chihuahua, Mexico, 2001 Proceedings, CD, Paper* (Vol. 61).

Ruiz-Torres, A. J., and Lopez, F. J. (2004) Using the FDH formulation of DEA to evaluate a multi-criteria problem in parallel machine scheduling. *Computers & Industrial Engineering*, *47*(2), pp. 107-121.

Ruiz-Torres, A. J. , Lopez, F. J., and Ho, J. C. (2007) Scheduling uniform parallel machines subject to a secondary resource to minimize the number of tardy jobs, *European journal of operational research*, 179, pp. 302-315.

Ruiz-Torres, A.J., Lopez, F.J., Wojciechowski, P.J., and Ho, J.C. (2010) Parallel machine scheduling problems considering regular measures of performance and machine cost, *Journal of the Operational Research Society*, 61(5), pp. 849-857.

Shahnazari-Shahrezaei, P., Tavakkoli-Moghaddam, R., and Kazemipoor, H. (2012) Solving a new fuzzy multi-objective model for a multi-skilled manpower scheduling problem by particle swarm optimization and elite tabu search, *The International Journal of Advanced Manufacturing Technology*, DOI: 10.1007/s00170-012-4119-y.