Using tabu search to determine the number of kanbans and lotsizes in a generic kanban system

Andrew D. Martin
Department of Political Science, Washington University in St. Louis, St. Louis, MO 63130, USA
E-mail: admartin@artsci.wustl.edu

Te-Min Chang and Yeuhwern Yih
Department of Industrial Engineering, Purdue University, West Lafayette, IN 47907, USA

Rex K. Kincaid
Department of Mathematics, The College of William and Mary, Williamsburg, VA 23185, USA

A generic kanban system designed for non-repetitive manufacturing environments is described. The purpose of this paper is to determine the number of kanbans and lotsizes to maximize system performance. System objectives include minimizing cycle time, operation costs and capital losses. A scalar multi-attribute utility function is constructed and a tabu search algorithm is proposed to search for the optimal utility value. Simulation is used to generate objective function values for each system setup. Four different variations of tabu search are employed. It is shown that a random sampling of the neighborhood provides good results with the shortest computation time. The tabu search algorithm proposed performs much better than a local search. The results are then compared to those from a modified simulated annealing algorithm. Due to the planar nature of the objective function, it is shown that tabu search can provide excellent results, yet a simulated annealing approach provides the same results with better computation time.

1. Introduction

The success of Toyota's production system has drawn increasing attention from researchers and practitioners. Toyota's system is based on the principles of Just-in-Time (JIT) manufacturing [18, 19]. In the JIT approach, kanban systems are employed to promote low inventory levels and short lead times. It is well known that kanban systems work successfully under repetitive environments where market demands are stable; however, under dynamic environments with variable demands kanban systems are inappropriate [10, 14, 15, 17].

Consequently, current research focuses on making kanban systems adaptable to non-repetitive environments. Chang and Yih [7] propose a modified kanban scheme, the generic kanban system, for such a purpose. The generic kanban system is shown to provide more flexibility in controlling system performance compared with push systems and CONWIP [21]. Thus, the feasibility of the generic kanban system to dynamic environments is justified.

The number of kanbans and different lotsizes affect system performance in generic kanban systems. Determination of the number of kanbans and lotsizes that yield the best system performance is thus desired. The purpose of this paper is to suggest an approach which can be used to maximize system performance.

1.1. Problem definition

A generic kanban system is a pull system. In a pull system, demand is placed on the final process stage. The availability of components for this stage is checked. If available, the production of this stage begins; otherwise, a request is issued to the previous stage for the required parts. Upon receiving the request, the previous stage either begins production for the requested components or an upstream request is issued. In this manner, the production of a current stage is "pulled" from the next workstation. Kanbans are pieces of paper attached to containers holding components. Kanbans are used to allow jobs to enter each workstation.

Under dynamic environments, demand on the generic kanban system is unknown. When a demand arrives in the system, kanbans must be issued for all stages since no parts at any stage are made beforehand. Only when the raw materials arrive at the initial station can the actual production of the system start. Moreover, not every kanban at any stage can be issued immediately since the number of kanbans at each stage is limited. A request may be deferred if kanbans at some stage are not available. Production in a generic kanban system thus goes through two phases: the kanban acquisition phase and the actual production phase.

In the actual production phase, as a job is finished at one stage the component is moved to the next downstream stage and the attached kanban of this stage is dropped. This kanban will be acquired by the next demand request. The situation is different from the repetitive environment of the JIT kanban system, where the free kanban will trigger a new production immediately. A detailed description of the generic kanban system can be found in Chang and Yih [7].

Chang and Yih [7] observed that the number of kanbans and lotsizes in the system directly affect system performance (work-in-process and cycle time). Clearly, tradeoffs exist between WIP and cycle time based on the number of kanbans and lotsizes. When minimizing the objectives of both WIP and cycle time, decision making for the best performance is far from trivial. If the number of kanbans at each stage and lotsize for job types could be different, the situation would become more complicated. We strongly suspect that this optimization problem is NP-hard. Hence, we believe that

there does not exist a practical algorithm for determining a globally optimal solution. We propose a heuristic search algorithm, tabu search, to determine near-optimal system performance.

1.2. *Tabu search algorithm*

Tabu search is a heuristic used to solve combinatorial optimization problems which was first proposed by Glover [11]. Tabu search is a hill-climbing algorithm which bans certain moves to escape traps of local optimality. The list of the banned moves is called the tabu list. Memory functions of various lengths are used by the algorithm to intensify and diversify the search in a systematic manner. The tabu list must be small enough to allow the search to carefully scrutinize certain parts of the objective function yet large enough to prevent a return to a previously generated solution.

Tabu search is designed to solve the following type of problem:

$$\begin{aligned} &\text{Minimize} && U(x) \\ &\text{subject to} && x \in X. \end{aligned}$$

$U(x)$ is the utility function to be defined in section 3. The condition $x \in X$ describes the constraints on the vector x. In practice, it is assumed that the search will begin at a good, feasible solution. A neighborhood $N(x) \subset X$ is then constructed to define adjacent moves, or those moves accessible from vector x. The search proceeds by searching $n \in N(x)$ and selecting the best non-tabu move in the neighborhood. An aspiration criterion is also used to define a condition where the tabu status of a certain move can be over-ridden. Short-term memory functions are employed to intensify and diversify the search. A comprehensive description of tabu search can be found in Glover and Laguna [13].

For the purposes of this optimization problem, the heuristic used in the tabu search algorithm is a local improvement scheme, beginning with a good, feasible solution. Various members of the neighborhood will be tested and a move will be made based on the value of the objective function. In fact, the neighborhood will be searched for a move which incrementally improves the value of the objective function or the move which degrades it least. The tabu list will be employed to direct the search away from any local optima the search has already encountered. A basic outline for the tabu search algorithm is taken from Skorin-Kapov [20].

Step 1. *Construction phase*

- generate a feasible solution

Step 2. *Improvement phase*

Perform improvement scheme MAXIT times and then do one of the following:

- *Intensify* the search by restarting the improvement phase at the current best solution.
- *Diversify* the search (explore a new area of the feasible region) and repeat step 2.
- Stop and display the best solution found. The specific algorithm employed to determine optimal kanban numbers and lotsizes for this problem will be discussed in section 4.

Tabu search has been extensively employed in the production literature, yet it has rarely been applied to kanban systems. Tabu search has been applied to many production scheduling problems. Barnes and Laguna [4] provide a survey of the production literature. They discuss how tabu search has been applied to single-machine scheduling problems. Tabu search has provided near-optimal solutions for many of the problems for which optimal or near-optimal solutions were never found. Also, the application of tabu search to the vehicle routing problem is discussed. It is shown that tabu search gives a 15–19% reduction in distribution costs as compared to other algorithms.

Extensive work has been done in the m-machine, n-job flow shop sequencing problem [2,22,23]. In this problem, the objective is to minimize maximum flow time (or makespan). Tabu search has been applied to this historically difficult problem, and for 80% of the problems tested, tabu search provides the most efficient algorithm [23]. Taillard [22] extended this work and implemented a parallel tabu search to solve this sequencing problem. He found drastic improvements in performance times with respect to Widmer and Hertz's results, as well as when compared to other algorithms. Adenso-Díaz [2] further refined the search by looking at the weighted tardiness of jobs in the flow shop. He used a restricted neighborhood structure which yielded drastically reduced CPU time while returning solutions of equal quality.

Tabu search has also been employed to solve the traditional job shop scheduling problem [9]. In this NP-hard problem, the makespan is once again minimized. Dell'Amico and Trubian propose a new heuristic by employing a different neighborhood structure. This extremely robust algorithm was tested by looking at 53 benchmark problems and the optimal solution or a near-optimal solution was found for each problem. They concluded that tabu search was far superior to older local search based algorithms or exact branch and bound methods.

Brandimarte [5] looks at routing and scheduling in the flexible job shop. He uses a hierarchical meta-heuristic for this optimization problem. He shows that tabu search is superior to truncated exponentiation schemes, greedy algorithms, and local search algorithms. It is shown how the tabu search meta-heuristic can be used to produce many different, robust architectures which provide excellent results for this type of problem. He explains how many different heuristics can be exploited by tabu search and shows that the algorithm is quite practical to common applications.

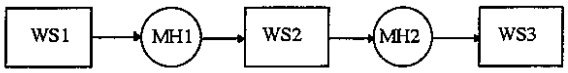
The flexible-resource flow shop scheduling has also been attempted by the tabu search algorithm [8]. This is a multi-objective optimization problem which requires a

nested search approach. Daniels and Mazzola show how tabu search is quite effective in providing near-optimal solutions and in fact generates optimal solutions for nearly 70% of the tested problems. These results were quite good for the FRFS problem.

In the next section, the system configuration under study is described and the associated parameters are defined. In section 3, the multiple objective problem is formulated as a scalar utility function. Next, a tabu search algorithm is proposed to determine optimal kanban numbers and lotsizes for this system. In section 5, four tabu search variations are described and the experiment is designed. Finally, in section 6, the results are reported and appropriateness of tabu search algorithms for generic kanban systems is discussed.

2. The system configuration

A single production line which includes three workstations for production processes and two material handling stations for transportation is proposed (see figure 1). A job will go from the first workstation to the final workstation and then leave the system as a finished product. Two types of jobs (A and B) are produced in the system. The parameters used in this model are listed in table 1.



WS: Work Station MH: Material handling station

Figure 1. A single production line under simulation study.

Table 1

Parameters used in simulation study.

Station	Number of servers	Type A service time (min)[a]	Type B service time (min)[a]	Setup time (min)[b]
WS1	2	U(1.6, 5.6)	U(0,4)	EXP(3)
MH1	1	U(0, 12)	U(0, 12)	N/A
WS2	3	U(2, 4)	U(1, 3)	EXP(2)
MH2	1	U(0, 12)	U(0, 12)	N/A
WS3	2	U(1, 3)	U(2, 4)	EXP(6)

[a] At WS, the service time is for 10 units. At MH, it is for one lot.
[b] Setup time is for one lot.

It may seem that the choices of system parameters are quite arbitrary. However, using a generic kanban system is a new proposal in the production literature. In fact, previous work has shown that generic kanban systems are quite flexible, yet are only

suitable for certain types of applications [7]. It is impossible to simulate a generic kanban system with all possible system operating conditions. The parameters chosen for this experiment are based on two important considerations: system congestion and bottleneck location. The system configuration posited in table 1 is not congested and the bottleneck location is at workstation one. Generic kanban systems have been shown to work effectively in dealing with a bottleneck, as long as the system is not congested. Therefore, the parameters where chosen to represent a common configuration encountered in industry. It is also important to note that the methodology used to determine kanban numbers and lotsizes is generic to all system configurations. Thus, the work in this paper can be applied to other configurations. Nonetheless, the configuration chosen is representative and can be used to effectively judge heuristics for determining kanban numbers and lotsizes.

In this model, the service time for each server is assumed to be uniformly distributed. The setup time for one lot is assumed to be exponentially distributed and setups only occur when the previous type of job is different from the current one. The interarrival time of dynamic demands is assumed to be exponentially distributed. The demand rate is 0.75 per hour for type A; 0.25 per hour for type B. The quantity of a demand ranges from 120, 240, 360, 480, to 600 units, which is assumed to be uniformly distributed. A demand can be partitioned into several lots on which kanbans are attached. The rule for assigning lots to servers is always first-come-first-served (FCFS).

3. Formulation of multi-objective optimization

3.1. Objectives in the system

Given a generic kanban system, the first step to optimizing system performance is to define the system objectives. Here, the three objectives considered are minimizing cycle time, operation costs and capital losses.

The cycle time (CT) is defined as the time between two consecutive outputs. By reducing the cycle time, a higher throughput rate is achieved. With dynamic demands, the cycle time is bounded below by the demand rate and above by the congested system capacity by definition.

The operation cost (OC) consists of several components. First, the work-in-process (WIP) is considered a cost to the operation. The WIP cost is further divided into two parts: interest on investment (carrying cost) and storage charges. The computation of the carrying cost is based on the average WIP flowing through the system. The storage cost is calculated proportionally to the maximum storage space needed during the time horizon.

The setup and lot transfer frequency is affected by different lotsizes. The smaller the lotsize, the greater the frequency will be. Therefore, the associated costs for setup and transportation are involved. The structure of the operation cost thus consists of carrying costs, storage costs, setup costs and transportation costs.

Finally, the intangible risk of building up high inventories is considered by looking at capital loss (CL). High inventory levels indicate high capital investment in the system and diminished cash flow which directly influences a company's survival chance. A loss function on the WIP capital is constructed to capture such possible loss.

3.2. *Utility function approach*

Based on the objectives defined above, we would face a multi-objective optimization problem if we were to simultaneously optimize those objectives. The utility function approach [16] is adopted to simplify this problem. A utility function is defined as a scalar function of objectives. The utility function is created by a decision maker to represent an a priori preference. The benefit of using the utility function is that it can reflect the decision maker's tradeoffs (linear or nonlinear) among the objectives defined.

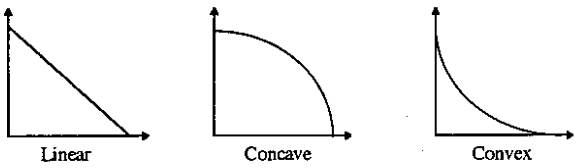
In our case, the utility function $U(x_i)$ is defined as

$$U(x_i) = U(CT, OC, CL),$$

where CT is cycle time, OC is operation cost, and CL is capital loss. Under the assumption of the additive property of objectives, the function $U(x_i)$ can be further expressed as

$$U(x_i) = W_1 * U_1(CT) + W_2 * U_2(OC) + W_3 * U_3(CL),$$

where U_i is the individual utility function for each objective and W_i is the decision maker's weighting factor associated with each utility function. For each individual utility function, linear, concave, or convex models can be used (see figure 2). In the



OV = objective value; U = utility value

Figure 2. Typical utility function models.

linear model, the utility value has a constant decremental rate compared to a decrease in the objective value (OV). A concave model is used if the decremental rate in the smaller OV region is less than that in the larger OV region. The convex case is the converse of the concave case.

To derive mathematical equations for the utility functions, we have to know the upper bound of the objective value as well as the upper bound of the utility value. In our case, the maximal utility value is set to be 10 for each utility function. The upper

bound of the objective value is set based on the simulation results of extreme cases (i.e., many or very few kanbans in the system). In other words, simulations are run with nothing in the system and with the system congested. These values, along with definitional bounds, are used as upper or lower bounds for the objective function.

The individual utility function is formulated as follows. For the cycle time, a linear model is assumed and the upper bound is set to be 5 hours. The mathematical form of the cycle time utility function is

$$U_1(CT) = 2(5 - CT),$$

where CT is the cycle time.

A linear model for the operation cost utility function is also assumed. The upper bound is set to be \$2400. The resultant utility function for the operation cost is

$$U_2(OC) = (2400 - OC)/240,$$

where OC is the operation cost.

Finally, the utility function for the *WIP* capital loss is defined. The typical shape of a loss function is exponential (see figure 3); the higher the *WIP* level in the system, the more capital a company will lose. From such a loss function, a concave utility

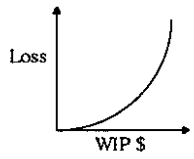


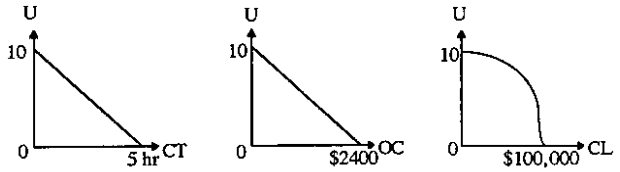
Figure 3. Loss function of *WIP* capital.

function can be obtained by reflecting the curve along the x-coordinate. The upper bound for *WIP* capital is set to be \$100,000. The utility function is obtained by fitting the exponential form as follows:

$$U_3(CL) = 11 - \exp(2.4 \times 10^{-5} CL),$$

where CL is the capital loss.

The overall utility functions are shown in figure 4.



CT: cycle time; OC: operation cost; CL: capital loss

Figure 4. Individual utility functions in the generic kanban problem.

Finally, the utility function $U(x_i)$ becomes

$$U(CT, OC, CL)$$
$$= 1 * [2(5 - CT)] + 1 * [(2400 - OC)/240] + 1 * [11 - \exp(2.4 \times 10^{-5} CL)],$$

where the weighting factors for each utility are set to be 1. Weighting each of the factors in this manner provides greater flexibility for the practitioner, since she can weight the utility function to best suit her production needs. Similarly, this weighting is chosen so objective function values can be compared to previous work. In fact, different weights on the factors only change the scale of the objective function. It makes no difference when trying to compare different heuristic methods. Therefore, the weighting factors for each utility are set to be 1.

From this development, our optimization problem can be stated in combinatorial terms. Given the discrete system parameters n (n = number of workstations) and m (m = number of possible lotsizes) as well as the distributions over service and setup times, this problem can be stated as follows:

Maximize	$U(CT, OC, CL)$,	
where	$CT = f(k_i, l_j)$,	
	$OC = g(k_i, l_j)$,	
	$CL = h(k_i, l_j)$,	
subject to	$k_i \in \{0, 1, 2, \ldots\}$	$\forall k_i, i = 1, 2, \ldots, n$,
	$l_j \in \{0, 1, 2, \ldots\}$	$\forall l_j, j = 1, 2, \ldots, m$,
where	k_i = integral number of kanbans,	
and	l_i = discrete number of possible lotsizes.	

In the following section, the tabu search algorithm is developed with respect to this application.

4. Proposed tabu search

After the utility function $U(x_i)$ is formulated, a tabu search algorithm is used to search for the kanban numbers associated with each workstation and lotsizes for each job to render a maximal utility function. In the tabu search algorithm, memory parameters as well as neighborhood structures must be defined. A feasible initial solution is chosen arbitrarily to start the algorithm. The following is an outline for an ACCEPT(x_i) subroutine in a tabu search algorithm for move x_i with associated utility value $U(x_i)$.

ACCEPT(x_i)
{
IF $U(x_i)$ > global_best THEN accept x_i; /* Aspiration criterion */
ELSE IF $U(x_i)$ > local_best AND x_i NOT tabu THEN accept x_i;
ELSE reject x_i;
}

A move is defined with respect to the neighborhood structure. The heuristic can progress from its original point to any other point in the neighborhood as long as it meets the criterion in the following subroutine. The neighborhood is constructed such that the algorithm cannot progress to an infeasible region by banning such moves. For this algorithm, the aspiration criterion is set such that tabu moves are accepted only if that move is the best seen so far. A circular array structure of length short_mem is employed to keep track of tabu moves. The array holds the move proper as well as the objective function value associated with that move. A TABUSTATUS(x_i) subroutine is used to determine if a move is acceptable. The search records which moves are the best so far as well as where the search has gone.

The comparison of the value of the utility function to other values originates from dealing with deterministic combinatorial problems where $U(x_i)$ is constant; however, under dynamic environments, $U(x_i)$ values are estimated based on four replications from simulation. A more meaningful comparison between utility function values is by looking at not only the mean of the four values, but also the variance. The acceptance formula is thus modified to guarantee an improving move has a 90% or better chance of being indifferent or greater than previous moves. If $U(x_i)$ falls in the indifference region and the mean is greater, the move is accepted. The paired-t test is used to reconstruct the acceptance rule by analogizing the comparison with 90% confidence interval. The ACCEPT(x_i) sub-routine is thus modified as follows:

ACCEPT(x_i)
{
IF $U(x_i)$ > global_best AND global_t_score > – 1.64 THEN accept x_i;
/* Aspiration criterion */
ELSE IF $U(x_i)$ > local_best AND x_i NOT tabu AND local_t_score > –1.64 THEN
accept x_i;
ELSE reject x_i;
}

By varying the length of the tabu list and testing different variations of the algorithm, tabu search has been shown to escape local optimality and provide good solutions to various combinatorial optimization problems [12]. Glover explains that the length of the tabu list is application dependent. Thus, experimentation is required to find a tabu list long enough to prevent cycling while short enough to not hamper the search. In the following section four different algorithms tested will be described and the experimental design will be posited.

5. Experimental design

Four different algorithms are used to optimize kanban numbers and lotsizes. A tabu length of 10 was arbitrarily chosen to test each of the four neighborhood structures. This list was used to ban certain solutions within the neighborhood. If a move was the best vector we had seen so far, the aspiration criterion is met and the move is accepted regardless of the tabu status of the vector. To compare our results with previous results [6], two starting solutions were tested for each variation of tabu search. These configurations are Seed I (5, 5, 5, 5, 5, 60, 60), and Seed II (15, 15, 15, 15, 15, 60, 60), where the first five elements are the kanban numbers for each of the five stations and the last two numbers represent the lotsize for job types A and B. Note that is this case, Seed I is better than Seed II since the associated objective function value is larger. The number of trial solutions to be evaluated is limited to 5000 since computation time for simulation is quite large. This algorithm was coded in the C programming language. Since the search may end at a point which is not the global best, the search needs to keep track of the best vector we have seen so far.

The neighborhood structure is defined as follows. For kanbans, the number in each station can increase by one, decrease by one, or remain the same (kanban numbers were required to remain greater than zero and less than forty). The possible lotsizes are restricted to 10, 20, 30, 40, 60, and 120. The lot size could stay the same, move to the next higher lotsize, or move to the next lower lotsize. For lotsize 10 (or 120), the lotsize could remain the same or go to the next higher (or lower) lotsize. In the first variation (RANDOM SAMPLING), 25 points were randomly sampled from the neighborhood. Twenty-five simulations would take place and moves would be evaluated by the ACCEPT(x_i) subroutine.

The second algorithm (RANDOM FIRST BEST) also randomly sampled from the entire possible neighborhood, but the ACCEPT(x_i) subroutine was modified to accept the first improving move and restart the search at the accepted point without looking at other parts of the neighborhood. The first improving move has been shown to improve computation times in traveling salesman problems and tends to accelerate convergence [3].

Since the size of the neighborhood is so large, an exhaustive search of the entire neighborhood is infeasible. However, Chang and Yih [6] developed a heuristic called the distance effect which restricts possible moves in the neighborhood. Only vectors with non-decreasing kanban numbers demonstrate the distance effect. Thus, the third variation (TOTAL DISTANCE) of tabu search simulates all vectors in the neighborhood which demonstrate this distance effect and accepts the best move found.

The final variation tested (TOTAL FIRST BEST) looked at all vectors which demonstrated the distance effect and used a modified ACCEPT(x_i) subroutine to restart the search at the first improving move. This was applied in an attempt to speed up the search.

Seed I and Seed II were used as initial solutions for each of these algorithms and the results are reported below. Subsequently, the best algorithm was looked at more

carefully by implementing tabu lists of varying lengths and comparing those results to a local search. In the following section, the results are summarized and compared to previous results which used a modified simulated annealing algorithm [6].

6. Results and analysis

Each of the variations was coded in the C programming language and were run using each initial solution. The results are summarized in table 2, as well as results from a local search. In all cases, the tabu criterion was invoked to prohibit certain moves from being made. CPU times are given in the number of simulations run (ITE).

Table 2

Results from each variation of tabu search.

Variation	Seed I results	Seed II results
RANDOM SAMPLING	24.867 on ITE 793	24.795 on ITE 3371
RANDOM FIRST BEST	24.858 on ITE 3355	24.489 on ITE 80
TOTAL DISTANCE	24.867 on ITE 2435	24.869 on ITE 4167
TOTAL FIRST BEST	24.826 on ITE 2591	24.827 on ITE 4452
LOCAL SEARCH	24.867 on ITE 2876	24.690 on ITE 2869

ITE represents the iteration number.

This is used as a measure because the programs were not run on a dedicated server. This also allows comparison across different platforms, and allows practitioners to estimate computation times using their system configuration.

Both of the structures which relied on using the distance effect (TOTAL DISTANCE and TOTAL FIRST BEST) did provide quality solutions. However, the computation time involved for generating these solutions was too great. For an application where computation time is not limiting, both of these more systematic approaches are feasible. However, for this problem, computation time is an important factor. Thus, both of these structures could be quite promising for some problems, but they were overly limiting for this application.

The results from RANDOM FIRST BEST were not as competitive as those from the RANDOM SAMPLING structure. The RANDOM FIRST BEST did accelerate very quickly to a local optimum, yet it made rash moves which kept the search from finding a better optimum by merely accepting the first improving moves. For problems where computation times are of utmost importance and slightly lower objective function values are acceptable, the first-improving refinement would be quite useful. However, with the kanban number and lotsize problem, the tradeoff in algorithm speed costs too much. Thus, the RANDOM SAMPLING algorithm is judged to be best, not only because it provides the best objective function values, but also because it achieved

those values the quickest. The associated kanban numbers for the optimal solution for Seed I were (1, 2, 4, 5, 10) with lot sizes (40, 40). For Seed II, the random sampling modification gave (14, 2, 3, 28, 6) with lotsizes (60,60) as the optimal solution.

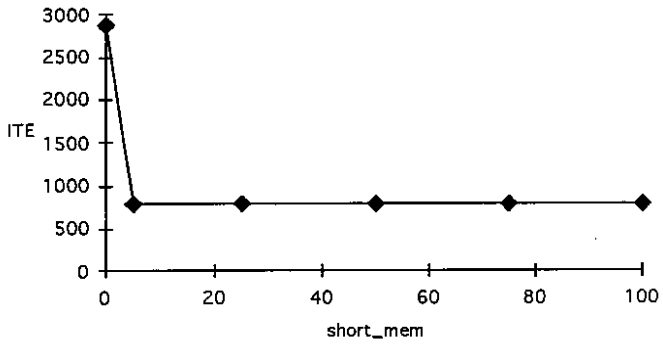
After the RANDOM SAMPLING algorithm was judged to be superior, the length of the tabu list (short_mem) was altered to try and determine a tabu length which adequately diversified the search without overly restricting it. Various tabu lengths were tested. In table 3 these results are summarized, along with those of a local search (short_mem = 0).

Table 3

Results from using RANDOM SAMPLING with differing tabu lengths.

Tabu length	Seed I results	Seed II results
short_mem = 1	24.867 on ITE 793	Not run
short_mem = 2	24.867 on ITE 793	Not run
short_mem = 3	24.867 on ITE 793	Not run
short_mem = 4	24.867 on ITE 793	Not run
short_mem = 5	24.867 on ITE 793	24.795 on ITE 3371
short_mem = 10	24.867 on ITE 793	24.795 on ITE 3371
short_mem = 25	24.867 on ITE 793	24.795 on ITE 3371
short_mem = 50	24.867 on ITE 793	24.795 on ITE 3371
short_mem = 75	24.867 on ITE 793	24.795 on ITE 3371
short_mem = 100	24.867 on ITE 793	24.795 on ITE 3371
short_mem = 0	24.867 on ITE 2876	24.690 on ITE 2869

ITE represents the iteration number.



ITE represents the iteration number and short_mem represents tabu length

Figure 5. Computation time to reach the final solution for Seed I.

The results for Seed I are represented in figure 5. Note that any tabu list length improved the speed of the search compared to a local search.

For the Seed I problem, the length of the tabu list made no difference to the resultant objective function value. However, the tabu search meta-heuristic performed far better than local search. For the Seed II problem, the tabu list did affect the quality of the solution reached. In fact, implementing the tabu list provided a higher objective function value for Seed II faster than a local search. This leads us to believe that the tabu search restrictions are quite useful in accelerating the search to an optimal solution. Although there is no guarantee our results are optimal, the kanban numbers and lotsizes which render good system performance are successfully determined.

To determine the effectiveness of the tabu-search meta-heuristic to this type of problem, the results need to be compared to previous work. Chang and Yih [6] proposed a modified simulated annealing algorithm to solve this combinatorial optimization problem. For both initial solutions, the tabu search was quite competitive with a traditional simulated annealing algorithm; however, the modified algorithm provided results much more quickly. The objective function values obtained were of equal magnitude, yet the speed of the modified simulated annealing algorithm was over ten times greater. In fact, Chang and Yih indicate that a modified simulated annealing algorithm reached a locally optimal solution with 52 iterations for Seed I and between 355 and 1016 iterations for Seed II. Although the obtained objective

Table 4

Results using simulated annealing and modified simulated annealing.

	Traditional SA	Modified SA
Seed I results	24.864 on ITE 768	24.847 on ITE 52
	24.864 on ITE 879	24.847 on ITE 52
	24.858 on ITE 545	24.847 on ITE 52
	24.858 on ITE 751	24.847 on ITE 52
	24.811 on ITE 135	24.847 on ITE 52
	24.811 on ITE 135	24.847 on ITE 52
Seed II results	24.827 on ITE 1654	24.858 on ITE 1016
	24.847 on ITE 1123	24.826 on ITE 434
	24.858 on ITE 1038	24.864 on ITE 355
	24.847 on ITE 1617	24.864 on ITE 708
	24.826 on ITE 758	24.858 on ITE 731
	24.690 on ITE 2529	24.845 on ITE 467

ITE represents the iteration number. Results are for six independent trials.

function values using tabu search are comparable to those found by the modified simulated anneal algorithm, the modified simulated annealing algorithm arrives at those values much faster. The reader is referred to table 4 for a presentation of the results obtained by Chang and Yih [6].

Ackley [1] compares different bit vector optimization algorithms to determine which is better for certain classes of problems. He tests seven algorithms, including hill-climbing algorithms, variants of genetic algorithms, and simulated annealing algorithms. He compares the performance of each algorithm by looking at several types of functions. By looking at the range of solutions generated by different search heuristics, it is observed that the objective function for the kanban number and lotsize problem is extremely flat and exhibits few local optima. This type of planar function Ackley would classify as having plateaus. His results indicate that for such flat functions, simulated annealing algorithms and genetic algorithms perform better than hill-climbing algorithms, such as tabu search. This is because hill-climbers require extensive computation times to reach optima; however, the other algorithms can get there much quicker. Our comparison of the simulated annealing algorithm and tabu search, our hill-climbing variant, confirms Ackley's conclusion.

7. Conclusion and future work

A generic kanban system that is adaptable to dynamic environments is presented. In this paper, an approach to determine the number of kanbans at each station and lotsizes of job types to optimize system performance is proposed. A representative generic kanban system is formulated to test different heuristic methods for choosing optimal kanban numbers and lotsizes. This approach includes formulating the multi-objective optimization problem by the utility function approach and searching the maximal utility value by using tabu search. A tabu search algorithm is proposed to search for an optimal solution, and it is shown that good results are achieved. A random sampling of the neighborhood is shown to provide good computation times, and the results achieved are quite competitive with previous results. However, the planar nature of our objective function limits the success of our algorithm and better lends itself to simulated annealing and genetic algorithms.

Yet, there are problems that need to be addressed in future work. First, variants of genetic algorithms need to be considered to determine if they provide better computation times for this optimization class of problems. Also, more sophisticated tabu search algorithms could be looked at. Second, in this paper the entire analysis is based on the linear combination used to define the objective function. A change of those functions will change the best kanban distribution, lotsizes and the associated initial seeds. However, practitioners customarily use such objective functions. The proposed generic kanban system is best tested with the system parameters chosen. In future work, the heuristics developed for selecting a good initial seed [6] should be employed to perhaps better our tabu search results. It has been shown that tabu search does provide high-quality, near-optimal solutions to this optimization problem, yet other meta-heuristics need to be analyzed before we can determine what approach is best to solving this combinatorial optimization problem.

Acknowledgements

The authors thank the Center for Intelligent Manufacturing Systems at Purdue University who funded this study during the summer of 1993. This is a revised version of a paper presented at the Business Applications of Artificial Intelligence Conference, October 1993, Charlottesville, Virginia.

References

[1] D.H. Ackley, An empirical study of bit vector optimization, in: *Genetic Algorithms and Simulated Annealing*, L. Davis, ed., Pittman, London, 1987, pp. 170–204

[2] B. Adenso-Díaz, Restricted neighborhood in the tabu search for the flowshop problem, European Journal of Operational Research 62(1992)27–37.

[3] E.J. Anderson, Mechanisms for local search: Is first-improving best, Working Paper, Judge Institute of Management Studies, University of Cambridge, Cambridge CB2 1RX, UK, 1993.

[4] J.W. Barnes and M. Laguna, A tabu search experience in production scheduling, Annals of Operations Research 41(1993)141–156.

[5] P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, Annals of Operations Research 41(1993)157–183.

[6] T.M. Chang and Y. Yih, Determining the number of kanbans and lotsizes in a generic kanban system: A simulated annealing approach, Working Paper No. 93-3, School of Industrial Engineering, Purdue University, 1993.

[7] T.M. Chang and Y. Yih, Generic kanban systems for dynamic environments, International Journal of Production Research 31(1993).

[8] R.L. Daniels and J.B. Mazzola, A tabu-search heuristic for the flexible-resource flow shop scheduling problem, Annals of Operations Research 41(1993)207–230.

[9] M. Dell'Amico and M. Trubian, Applying tabu search to the job-shop scheduling problem, Annals of Operations Research 41(1993)231–252.

[10] B.J. Finch and J.F. Cox, An examination of just-in-time management for the small manufacturer: With an illustration, International Journal of Production Research 24(1986)329–342.

[11] F. Glover, Heuristics for integer programming using surrogate constraints, Decision Sciences 8 (1977)156–166.

[12] F. Glover, Tabu search: A tutorial, Interfaces 20(1990)74–94.

[13] F. Glover and M. Laguna, Tabu search, in: *Modern Heuristic Techniques for Combinatorial Problems*, C.R. Reeves, ed., Blackwell Scientific Publications, Oxford, 1993, pp. 70–150

[14] W.R. Hall, *Driving the Productivity Machine: Production Planning and Control in Japan*, American Production and Inventory Control Society, Falls Church, VA, 1981.

[15] P.Y. Huang, L.P. Rees and B.W. Taylor, A simulation analysis of the Japanese JIT techniques (with Kanban) for multi-line, multi-stage production systems, Decision Sciences 14(1983)326–344.

[16] R.L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Wiley, New York, 1976.

[17] L.J. Krajewski, B.E. King, L.P. Ritzman and D.S. Wong, Kanban, MRP, and shaping the manufacturing environment, Management Science 33(1987)39–57.

[18] Y. Monden, What makes the Toyota production system really tick?, Industrial Engineering 13(1981) 36–46.

[19] Y. Monden, *Toyota Production System: Practical Approach to Production Management*, Industrial Engineering and Management Press, Atlanta, GA, 1983.

[20] J. Skorin-Kapov, Tabu search applied to the quadratic assignment problem, ORSA Journal on Computation 2(1990)33–45.
[21] M.L. Spearman, D.L. Woodruff and W.J. Hopp, CONWIP: A pull alternative to kanban, International Journal of Production Research 28(1990)879–894.
[22] E. Taillard, Some efficient heuristic methods for the flow show sequencing problem, European Journal of Operational Research 47(1990)65–74.
[23] M. Widmer and A. Hertz, A new heuristic method for the flow shop sequencing problem, European Journal of Operational Research 41(1989)186–193.