Optimised Search Heuristic Combining Valid Inequalities and Tabu Search

ISSN: 0302-9743.

Optimised Search Heuristic Combining Valid Inequalities and Tabu Search

Susana Fernandes[1,⋆] and Helena R. Lourenço[2,⋆⋆]

[1] Universidade do Algarve, Faro, Portugal
sfer@ualg.pt
[2] Univertitat Pompeu Fabra, Barcelona, Spain
helena.ramalhinho@upf.edu

Abstract. This paper presents an Optimised Search Heuristic that combines a tabu search method with the verification of violated valid inequalities. The solution delivered by the tabu search is partially destroyed by a randomised greedy procedure, and then the valid inequalities are used to guide the reconstruction of a complete solution. An application of the new method to the Job-Shop Scheduling problem is presented.

Keywords: Optimised Search Heuristic, Tabu Search, GRASP, Valid Inequalities, Job-shop Scheduling.

1 Introduction

Recently a new class of hybrid procedures, that combine local search based (meta) heuristics and exact algorithms of the operations research field, have been designed to find solutions for combinatorial optimisation problems. Fernandes and Lourenço [1] designated these methods by Optimised Search Heuristics (OSH). Different combinations of different procedures are present in the literature, and there are several applications of the OSH methods to different problems (see the web page of Fernandes and Lourenço (2007))[1].

We present an OSH procedure that uses valid inequalities to reconstruct a local optimal solution that has been partially destroyed. We first build a feasible solution with a GRASP procedure and perform a tabu search to get a "good" local optimum. To continue searching the solution space we perturb the current solution partially destroying it and then rebuilding it. A greedy randomised method is used to delete some elements from the local optimal solution. We then test the existence of violated valid inequalities by the partial solution. These allow us to establish a new search path for rebuilding a complete feasible solution, and hopefully lead us to an attractive unexplored region of the solution space. We named this procedure Tabu_VVI from **Tabu** with **V**iolated **V**alid **I**nequalities.

⋆ The work of Susana Fernandes is supported by the the program POCI2010 of the Portuguese Fundação para a Ciência e Tecnologia.

⋆⋆ The work of Helena R. Lourenço is supported by Ministerio de Educacion y Ciencia, Spain, MEC-SEJ2006-12291.

[1] http://www.econ.upf.edu/~ramalhin/OSHwebpage/index.html

The idea of this new method is to mimic the cuts in integer programming, letting the violated valid inequalities cut off regions of the solution space where the objective function would have a value not better than the one of the current solution. This way the search is guided from a local optimal solution to a higher-quality region of the search space.

The procedure is illustrated with an application to the job-shop scheduling problem.

This paper is organized as follows: we start by presenting a literature review and motivation, proceed introducing the job-shop scheduling problem and continue describing the application of the Tabu_VVI to it. Computational results are presented along with comparisons to other OSH methods applied to the job shop problem and also to the state of the art tabu search algorithm of Nowicki and Smutnicki [2].

2 Literature Review and Motivation

In the literature we can find a few works combining metaheuristics with exact algorithms applied to the job-shop scheduling problem, designated as Optimised Search Heuristics (OSH) by Fernandes and Lourenço [1].

Chen, Talukdar and Sadeh [3] and Denzinger and Offermann [4] design parallel algorithms that use asynchronous agents information to build solutions; some of these agents are genetic algorithms, others are branch-and-bound algorithms.

Tamura, Hirahara, Hatono and Umano [5] design a genetic algorithm where the fitness of each individual, whose chromosomes represent each variable of the integer programming formulation, is the bound obtained solving lagrangean relaxations.

The works of Adams, Balas and Zawack [6], Applegate and Cook [7], Caseau and Laburthe [8], Balas and Vazacopoulos [9] and Pezzella and Merelli [10] all use an exact algorithm to solve a sub problem within a local search heuristic for the job-shop scheduling. Caseau and Laburthe [8] build a local search where the neighbourhood structure is defined by a subproblem that is exactly solved using constraint programming. Applegate and Cook [7] develop the shuffle heuristic. At each step of the local search the processing orders of the jobs on a small number of machines is fixed, and a branch-and-bound algorithm completes the schedule. The shifting bottleneck heuristic, due to Adams, Balas and Zawack [6], is an iterated local search with a construction heuristic that uses a branch-and-bound to solve the subproblems of one machine with release and due dates. Balas and Vazacopoulos [9] work with the shifting bottleneck heuristic and design a guided local search, over a tree search structure, that reconstructs partially destroyed solutions. The procedure of Pezzella and Merelli [10] is a tabu search that uses a branch-and-bound to solve one-machine subproblems; both at the construction of the initial solution and at a re-optimisation phase of the algorithm.

Lourenço [11] and Lourenço and Zwijnenburg [12] use branch-and-bound algorithms to strategically guide an iterated local search and a tabu search algorithm. The diversification of the search is achieved by applying a branch-and-bound method to solve a one-machine scheduling subproblem obtained from the incumbent solution.

In the work of Schaal, Fadil, Silti and Tolla [13] an interior point method generates initial solutions of the linear relaxation. A genetic algorithm finds integer solutions. A cut is generated based on the integer solutions found and the interior point method is applied again to diversify the search. This procedure is defined for the generalized job-shop problem.

The interesting work of Danna, Rothberg and Le Pape [14] “applies the spirit of metaheuristics” in an exact algorithm. Within each node of a branch-and-cut tree, the solution of the linear relaxation is used to define the neighbourhood of the current best feasible solution. The local search consists in solving the restricted MIP problem defined by the neighbourhood.

We are especially interested in combinations of exact and heuristic methods where the exact procedures can be used to strategically guide the heuristic ones. In this paper we mimic the cutting plane algorithms using the verification of the existence of violated valid inequalities to guide the search in the solution space. We are not aware of any other works using this methodology.

We chose to apply this new method to the job-shop scheduling problem because it is considered a particularly hard combinatorial optimisation problem of the NP-hard class, and so a few methods that combine exact and heuristic procedures have already been design to handle it.

3 The Job Shop Scheduling Problem

The job-shop scheduling problem (JSSP) has been known to the operations research community since the early 50’s [15]. It is considered a particularly hard combinatorial optimisation problem of the NP-hard class [16] and it has numerous practical applications; which makes it an excellent test problem for the quality of new scheduling algorithms. These are main reasons for the vast literature on both exact and heuristic procedures applied to this scheduling problem.

The job-shop scheduling problem considers a set of jobs to be processed on a set of machines. Each job is defined by an ordered set of operations and each operation is assigned to a machine with a predefined constant processing time (pre-emption is not allowed). The order of the operations within the jobs and its correspondent machines are fixed a priori and independent from job to job. To solve the problem we need to find a sequence of operations on each machine respecting some constraints and optimising some objective function. It is assumed that two consecutive operations of the same job are assigned to different machines, that each machine can only process one operation at a time and that different machines cannot process the same job simultaneously. We will adopt the maximum of the completion time of all jobs – the makespan – as the objective function.

A common representation for the job-shop problem is the disjunctive graph $G = (O, A, E)$ [17]; where O is the node set, corresponding to the set of operations with two dummy operations; 0 representing the source node and $o + 1$ the sink node; A is the set of arcs between consecutive operations of the same job, and E is the set of edges between operations processed by the same machine. For every node j of $O\setminus\{0, o + 1\}$ there are unique nodes i and l such that arcs

(i, j) and (j, l) are elements of A. Node i is called the job predecessor of node j - $jp(j)$ and l is the job successor of j - $js(j)$. Finding a solution to the job shop scheduling problem means replacing every edge of E with a directed arc, constructing an acyclic directed graph $D_S = (O, A \bigcup S)$ where $S = \bigcup_k S_k$ corresponds to an acyclic union of sequences of operations for each machine k. The optimal solution is the one represented by the graph D_S having the critical path from 0 to $o + 1$ with the smallest length or makespan.

4 Tabu_VVI Applied to the JSSP

The algorithm Tabu_VVI has two main stages. The first stage consists of building a feasible solution, and executing the tabu search procedure starting from it. The second stage consists of a large step followed by the tabu search, and it is repeated for a predefined number of iterations. The large step partially destroys the solution delivered by the tabu search, looks for violated valid inequalities that enforce some order between unscheduled operations, and then rebuilds a complete solution respecting those established orders. The information about the algebraic structure of the problem within the valid inequalities is used to guide the search. The idea is to perturb the current complete solution achieving diversification and leading the search method to new unexplored regions of the solution space.

The main loop of the algorithm is stopped either when the lower bound of the instance is achieved (LB), or a predefined maximum number of iterations are executed without improving the upper bound (UB). Figure 1 shows a not detailed and simplified pseudo-code of algorithm Tabu_VVI.

4.1 Building a Feasible Solution

We first build a feasible solution using a GRASP_B&B algorithm [18]. It is a simple heuristic that includes a branch-and-bound method at the building phase of a GRASP procedure. A GRASP [19] is an iterative process where each iteration consists of two steps: a randomised building step of a greedy nature and a local search step. The branch-and-bound is used in the building step to solve subproblems of single machine scheduling problems. The neighbourhood of the local search uses the notions of blocks of critical operations, defining critical pairs of operations belonging to the same block, and performing forward and backward moves on them. A block of critical operations is a maximal ordered set of consecutive operations of a critical path (in the disjunctive graph that represents the solution), sharing the same machine. Let $L(i, j)$ denote the length of the critical path from node i to node j.

Two operations u and v form a forward critical pair (u, v) if:

a) they both belong to the same block;
b) v is the last operation of the block;
c) operation $js(v)$ also belongs to the same critical path or v is the last operation of the job;

d) the length of the critical path from v to $o+1$ is not less than the length of the critical path from $js(u)$ to $o+1$ ($L(v, o+1) \geq L(js(u), o+1)$).

Two operations u and v form a backward critical pair (u, v) if:

a) they both belong to the same block;
b) u is the first operation of the block;
c) operation $jp(u)$ also belongs to the same critical path or u is the first operation of the job;
d) the length of the critical path from 0 to u, including the processing time of u, is not less than the length of the critical path from 0 to $jp(v)$, including the processing time of $jp(v)$ ($L(0, u) + p_u \geq L(0, jp(v)) + p_{jp(v)}$).

Conditions d) are included to guarantee that all moves lead to feasible solutions [9]. A forward move is executed by moving operation u to be processed immediately after operation v. A backward move is executed by moving operation v to be processed immediately before operation u.

For a detailed description of the GRASP_B&B algorithm please refer to [18].

4.2 Tabu Search

A tabu search procedure [20,21] is a local search procedure that inspects the whole neighbourhood of a current solution x and executes the move that produces the best neighbour *ybest*. The value of *ybest* may be worse than the one of x, so the move that goes back from *ybest* to x becomes forbbiden, named tabu moves. The set of tabu moves is updated in every iteration of the method, so

Tabu_VVI

```
xi = GRASP_B&B(runs)
x = TabuSearch(xi)
UB =makespan(x)
xb = x
while((UB > LB) and (#iterations without improvement < max #iterations))
    xd = Destroy(x)
    xd = FindValidInequalities(xd)
    x = Rebuild(xd)
    x = TabuSearch(x)
    if(makespan(x) < UB)
        update UB
        xb = x
    endif
endwhile
return(xb)
```

Fig. 1. Outline of Tabu_VVI: (xi) - initial feasible solution, (x) - current complete solution, (xd) - partially destroyed solution, (xb) - best solution, (LB) - lower bound derived from the makespan of the first bottleneck machine

the neighbourhood definition is dynamically updated. The procedure stops after a predefined number of iterations have been performed without improving the best solution found.

In order to implement a simple tabu search procedure we need to define the neighbourhood structure, the tabu length that defines how long will a move remain tabu, and an aspiration criterion, to be able to execute moves abusively considered tabu. (this abuse happens because we do not keep track of the pair of solutions before and after a move, but only of some features of the move).

The neighbourhood structure of the tabu search implemented is the same used in the local search of the GRASP_B&B [18]. But this time we keep track of those moves rejected by conditions d) because they could produce a cycle in the disjunctive graph, thus leading to an infeasible solution. When the neighbourhood is empty, we look in these rejected moves for feasibility and execute the one that generates the best feasible solution. If none of the rejected moves produces a feasible solution we then execute the tabu move that would remain tabu for the shortest number of iterations.

The number of iterations a move (performed on solution x) stays tabu – the tabu length – is defined so it depends on the size of the neighbourhood of solution x. If a solution x has many neighbours, the reverse move of the one executed to leave from it stays tabu for a longer number of iterations than the reverse move of the one executed to leave from a solution y with a smaller neighbourhood. This way we state that the possibility of returning to a previously visited solution is not equal for every solution but depends on the number of neighbours it has.

The aspiration criterion allows a tabu move to be executed if the value of the resulting solution is better than the best one found so far.

Every time the tabu search improves the best known solution we apply an intensification scheme that consists in repeating the tabu search, this time duplicating the number of allowed iterations without improvement.

4.3 Large Step

Partially destroying a solution. The tabu search module of the algorithm provides a local optimal solution and its makespan is an upper bound for the optimal value. This solution is then perturbed using a greedy randomised method to eliminate the sequences of processing operations of some machines. Considering the acyclic directed graph that represents the solution, arcs connecting operations processed by the same machine are deleted. This method is biased toward machines that, when their sequence of processing operations is deleted, lead to a bigger reduction on the makespan of the solution. We keep "deleting" machines (destroying the sequence for processing the operations) until the makespan of the resulting partial solution is less than the upper bound.

After a predefined maximum number of global iterations are executed without improving the best solution found, the algorithm continues, for the same amount of iterations, this time choosing to "delete" machines that lead to the smallest reduction on the makespan. While the best solution found keeps being updated, we keep running the algorithm, alternating the criteria for “deleting” machines from the solution.

Finding violated valid inequalities. Having a partial solution and an upper bound (UB) for the optimal value, we then test the existence of violated valid inequalities. These allow us to establish some orders between operations of each unscheduled machine.

The procedure looks for violated valid inequalities for every machine whose sequence of operations is not present on the current partial solution. The process cycles through all the “deleted” machines and is repeated until no more orders between operations are set.

We use the same inequalities that were used in the branch-and-bound algorithms of Carlier and Pinson [22] and Applegate and Cook [7].

Let α be a machine of the instance whose sequence of processing the operations was deleted from the solution, and S_α any given sub-set of the operations processed by α. Every operation i has an earliest possible starting time - e_i, a processing time - p_i and a minimum completion time after it is processed - f_i.

If for any given set S_α and any given operation $i \in S_\alpha$, $\min\limits_{j \in S_\alpha \setminus \{i\}} \{ e_j \} + \sum_{j \in S_\alpha} p_j + \min\limits_{j \in S_\alpha} \{ f_j \} \geq UB$ then, to be possible to reduce the upper bound, operation i must be processed on α before any other operation in S_α. The inverse inequality $\min\limits_{j \in S_\alpha} \{ e_j \} + \sum_{j \in S_\alpha} p_j + \min\limits_{j \in S_\alpha \setminus \{i\}} \{ f_j \} \geq UB$ states that operation i must be processed on α after any other operation in S_α.

Let C_α be the set of operations not yet ordered for machine α, $E_\alpha \subseteq C_\alpha$ the sub-set of operations that could be scheduled first, and $F_\alpha \subseteq C_\alpha$ the subset of operations that could be scheduled last. If there is an operation $i \in E_\alpha$ such that $e_i + \sum_{j \in C_\alpha} p_j + \min\limits_{j \in F_\alpha} \{ f_j \} \geq UB$ then i can be removed from E_α. If E_α contains only one operation, then it must be processed on α before any other operation in C_α. The reverse inequality $\min\limits_{j \in E_\alpha} \{ e_j \} + \sum_{j \in C_\alpha} p_j + f_i \geq UB$ states that i cannot be scheduled after all the other operations in C_α, and should be removed from F_α.

Not all the sub-sets S_α are inspected when looking for violated valid inequalities that allow us to fix orders between operations of one machine, as it would be too computationally expensive. A reduced number of sub-sets are formed including operations by its decreasing values of starting and completion times.

If when looking for violated valid inequalities we find none, then we reintroduce a deleted machine in the solution and we look again for violated valid inequalities. The machine to add to the solution is chosen randomly. If the violated valid inequalities lead to incompatible sequences of operations, this means we cannot improve the upper bound (UB) with the set of sequenced machines, and another machine is deleted from the solution. If this happens repeatedly and the solution becomes empty, then the current complete solution is optimal.

Rebuilding a complete solution. The solution is reconstructed including the sequence of operations of one machine at a time. The order of adding the sequences in the machines to the solution is the same as for the elimination. The first machine to be re-included in the solution is the one that was first removed, and so on. The schedule of operations for each machine is determined using a modified version of the Schrage algorithm [23] that considers pre-defined orders between operations. Each time the sequence of operations of a machine is re-included in the solution, a restricted local search is executed, where it is forbidden to change orders fixed by the valid inequalities. When a new sequence of operations is included, we look for new violated valid inequalities in all remaining unscheduled machines, trying to fix more orders between operations.

After the solution is complete, local search is executed.

5 Computational Results

We have tested the algorithm Tabu_VVI on 132 benchmark instances: abz5-9 [6], ft6, ft10, ft20 [24], la01-40 [25], orb01-10 [7], swv01-20 [26], ta01-50 [27] and yn1-4 [28] [2]. The size of the instances is measure by the number of operations (equal to the number of jobs times the number of machines). The instances have different sizes: ft6 is the smaller one with 6×6 operations; la01-05 have 10×5; la06-10 have 15×5; ft20 and la11-15 have 20×5; abz5-6, ft10, la16-20 and orb01-10 have 10×10; la21-25 have 15×10; la26-30 and swv01-05 have 20×10; la36-40 and ta01-10 have 15×15; abz7-9, swv06-10 and ta11-20 have 20×15; ta31-40 and yn1-4 have 20×20; the bigger ones are ta41-50 with 30×20 operations.

An optimal solution has already been found for 83 of these instances; namely abz5-7, ft6, ft10, ft20, la01-40, orb01-10, swv01-02, swv05, swv13-14, swv16-20, ta01-10, ta14, ta17, ta31, ta35-36 and ta38-39.

We have tested a few slightly different versions of the method Tabu_VVI. Within the tabu search module, different values of the tabu length parameter were tested: equal to the number of neighbours; half of it and the double of it. Also inside the tabu search module, we have tested not to look for those moves rejected by conditions d), so when a neighbourhood is empty the eligible tabu move is always the one executed. The number of tabu iterations allowed without improving the best solution was set to the number of operations of each instance. Within the rebuild module, we have also tested to build the sequence of processing operations in one machine using a branch-and-bound method instead of just the priority rule of the Schrage algorithm. The orders between operations that were fixed by the find violated valid inequalities module are always respected.

At the first stage of the method Tabu_VVI, the GRASP_B&B algorithm was run for 10 iterations to generate the initial feasible solution and tabu search was run for 100 iterations without improvement.

The algorithm has been run on a Pentium 4 CPU 2.80 GHz and coded in C.

[2] These instances can be found in http://people.brunel.ac.uk/ mastjjb/jeb/orlib/files/ files jobshop1.txt and jobshop2.txt

In order to measure the performance of the algorithm we use the percentage of relative error to the lower bound - RE_{LB} (or to the optimum if the problem is closed). $f(x)$ stands for the makespan of the best solution found.

$$RE_{LB}(x) = 100\% \times \frac{f(x) - LB}{LB}$$

The next table 1 presents the performance of two variants of the algorithm that we found most successful, considering the sum of the RE_{LB} for all the instances tested, and also a column with the best results over all the 15 variants tested. The two chosen variants are tabu_mv_inf, the variant described earlier, and tabu_mv_bb, where moves rejected by conditions d) are not considered and branch-and-bound is used in the rebuilding module. The tabu tenure is set to be equal to the number of neighbours in both variants.

The table shows the average values (over a group of instances) of the RE_{LB} and the time in seconds to the best solution found.

Table 1. Results by Tabu_VVI: variants tabu_mv_inf and tabu_mv_bb, and the best of all variants, for all groups of instances, in average percentage of the relative error to the lower bound, and the average time to the best, in seconds

instances	tabu_mv_inf		tabu_mv_bb		best_all_variants	
	$avg(RE_{LB})$	$avg(time)$	$avg(RE_{LB})$	$avg(time)$	$avg(RE_{LB})$	$avg(time)$
abz	2.11	63.77	1.93	61.02	1.71	81.46
ft	0	11.72	0	0.58	0	0.15
la01-05	0	0.12	0	0.12	0	0.03
la06-10	0	0.02	0	0.03	0	0.02
la11-15	0	0.04	0	0.05	0	0.04
la16-20	0	1.79	0	1.67	0	0.44
la21-25	0.11	23.13	0.06	14.80	0	7.94
la26-30	0.29	54.12	0.26	40.88	0.17	83.39
la31-35	0	0.38	0	0.39	0	0.27
la36-40	0.47	22.68	0.22	33.50	0.05	57.08
orb	0.23	7	0.09	14.13	0	4.30
swv01-05	2.89	88.05	2.93	120.43	2.33	127.91
swv06-10	8.89	336.94	9.51	204.27	8.06	281.64
swv11-15	1.78	1734.51	2.03	825.21	1.41	1854.58
swv16-20	0	1.58	0	1.64	0	1.58
yn	7.49	339.33	7.91	73.61	7	163.95
ta01-10	0.63	77.72	0.81	67	0.24	49.52
ta11-20	3.47	54.20	3.70	86.60	3.12	177.24
ta21-30	6.51	319.27	6.60	269.97	5.96	319.02
ta31-40	1.79	230.90	1.60	258.49	1.26	220.62
ta41-50	6.04	650.87	5.88	559.71	5.47	1016.21

Table 2. Results by variants tabu_mv_inf and tabu_mv_bb of Tabu_VVI, and the algorithm of Caseau and Laburthe, in average percentage of the relative error to the lower bound, and the average time to the best, in seconds

instances	Tabu_VVI				CL	
	tabu_mv_inf		tabu_mv_bb			
	$avg(RE_{LB})$	$avg(time)$	$avg(RE_{LB})$	$avg(time)$	$avg(RE_{LB})$	$avg(time)$
abz	2.11	63.77	**1.93**	61.02	2.57	112.67
ft	0	11.72	0	0.58	0	112
la01-05	0	0.12	0	0.12	0	3.80
la06-10	0	0.02	0	0.03	0	0.75
la11-15	0	0.04	0	0.05	0	27
la16-20	0	1.79	0	1.67	0	25.08
la21-25	0.11	23.13	**0.06**	14.80	0.11	551.40
la26-30	0.29	54.12	**0.26**	40.88	0.47	4322.25
la31-35	0	0.38	0	0.39	0	2108.40
la36-40	0.47	22.68	**0.22**	33.50	0.37	2476.40
orb	0.23	7	**0.09**	14.13	1.66	111.11

We have found a new upper bound, 1765, for instance swv10 in 101 seconds.

The values of best known lower and upper bounds were gathered from the paper of Jain and Meeran [15] and the papers of Nowicki and Smutnicki [2], [29], [30].

5.1 Comparison to Other OSH Methods

The optimised search methods applied to the job-shop scheduling problem, that we know of and have mentioned in the literature review, are only applied to the older and easier instances of the problem, except for the works of Balas and Vazacopoulos [9] and Pezzella and Merelli [10], that will be treated separately.

The method of Danna, Rothberg and Le Pape [14] is applied to instances of the weighted-tardiness version of the problem, and the work of Schaal, Fadil, Silti and Tolla [13] is applied to the generalised scheduling problem.

Our method, Tabu_VVI is better for all the comparable instances (except for one or two exceptions), in quality of the solutions and in computational time, then the works of Chen [3], Denzinger and Offermann [4], Tamura, Hirahara, Hatono and Umano [5], Adams, Balas and Zawack [6], Applegate and Cook [7], Lourenço [11] and Lourenço and Zwijnenburg [12]. In table 2 we show the comparison results to the work of Caseau and Laburthe (named CL), because it is the best of these methods and also because it is the one that presents results for more instances. Their algorithm was run on a SunSparc 10 machine. The running times for their method are not scaled for our PC. Nonetheless we state our algorithm is faster and achieves better quality solutions.

Comparison to Guided Local Search. The guided local search procedure of Balas and Vazacopoulos [9] designs a search procedure based on local improvements and accepting non improving moves, using structures of neighbourhood trees. Each neighbourhood tree corresponds to a cycle of the guided local search procedure. Each node of the tree stores a solution and each edge connects neighbour solutions. Feasible solutions are built solving to optimality by branch-and-bound all one-machine subproblems (like the shifting bottleneck heuristic [6]). After a few cycles of neighbourhood trees, the procedure randomly destroys the best solution found; deleting the sequence of operations for some machines, and then reconstructs the partially destroyed solution repeating the all process.

Here we compare our best results to their best reported version SB-RGSL10, which stands for shifting bottleneck with randomised guided local search. The 10 means the number of times the all process is repeated. We call it BZ. Their algorithm was run on a SunSparc 30 machine. The comparison results between algorithms Tabu_VVI and BZ are shown in table 3. Although we used different computers and their running times are not scaled for our PC, we can still say that our method is always faster then BZ. Quality values that win the comparison are shown in bold.

Comparison to the Tabu Search with Shifting Bottleneck. The procedure of Pezzella and Merelli [10] combines tabu search with the shifting bottleneck heuristic. The later is used to build the initial solution, and also at the

Table 3. Results by the best of all variants of Tabu_VVI and the best variant of the algorithm of Balas and Vazacopoulos; in average percentage of the relative error to the lower bound, and the average time per run to the best, in seconds

instances	Tabu_VVI		BZ	
	$avg(RE_{LB})$	$avg(time)$	$avg(RE_{LB})$	$avg(time)$
la01-05	0	0.03	0	5.9
la16-20	0	0.44	0	47
la21-25	0	7.94	0	139.6
la26-30	**0.17**	83.4	0.19	121.6
la36-40	0.05	57.1	**0.03**	278
orb	**0**	4.30	0.10	80.18
swv01-05	2.33	128	**2.02**	1290
swv06-10	**8.06**	282	9.64	2917
swv11-15	**1.41**	1855	2.12	9173
yn	7	164	**5.96**	5938
ta01-10	**0.24**	49.5	0.25	1182
ta11-20	**3.12**	177	3.34	3383
ta21-30	**5.96**	319	6.57	4377
ta31-40	1.26	221	**1.13**	5069
ta41-50	**5.47**	1016	5.71	10726

re-optimisation phase of the algorithm. Whenever the tabu search cycle improves the best known solution, the procedure deletes the sequence of operations of all critical machines (machines with operations in the critical path). After shifting bottleneck rebuilds the solution, the tabu search is repeated. The tabu search module uses a dynamic management of three different neighbourhood structures and a tabu list of variable size, dependent of how many tabu iterations have been executed. The algorithm, that we name PM, was run on a Pentium 133 MHz. Table 4 shows the comparison results between algorithms Tabu_VVI and PM. Quality values that win the comparison are shown in bold.

Table 4. Results by the best of all variants of Tabu_VVI and the algorithm of Pezzella and Merelli; in average percentage of the relative error to the lower bound, and the average time to the best, in seconds

instances	Tabu_VVI		PM	
	$avg(RE_{LB})$	$avg(time)$	$avg(RE_{LB})$	$avg(time)$
abz	**1.71**	81.5	2.23	151
ft	0	0.15	0	65
la01-05	0	0.03	0	9.8
la06-10	0	0.02	0	-
la11-15	0	0.04	0	-
la16-20	0	0.44	0	61.5
la21-25	**0**	7.94	0.1	115
la26-30	**0.17**	83.4	0.46	105
la31-35	0	0.27	0	-
la36-40	**0.05**	57.1	0.58	141
ta01-10	**0.24**	49.5	0.45	2175
ta11-20	**3.12**	177	3.47	2526
ta21-30	**5.96**	319	6.52	34910
ta31-40	**1.26**	221	1.92	141333
ta41-50	**5.47**	1016	6.04	11512

5.2 Comparison to State of the Art Procedure - Tabu Search with Path-Relinking

Along with the guided local search procedure of Balas and Vazacopoulos [9], and the tabu search with shifting bottleneck of Pezzella and Mirelli [10], one other procedure, due to Nowicki and Smutnicki [2], forms the group of three procedures that are the best up to date methods applied to the job-shop scheduling problem.

The procedure of Nowicki and Smutnicki performs path-relinking between elite solutions found by a tabu search module. The solutions achieved by the path-relinking are then used as starting points for new cycles of the tabu search; the set of elite solutions is updated and the all process is repeated. We can say that the path-relinking works as the diversification strategy of the tabu search.

The algorithm uses a data structure specially designed for the application of this method to the job-shop scheduling problem. The instances of Taillard [27] were used to study the distribution of the local optima solutions in the solution space; and this study supported the design of this method. The algorithm, that we name NS, was run on a Pentium 900 MHz. Unlike all other procedures, the computational times reported by the authors do not include the time needed to build the initial solutions. Table 5 shows the comparison results between algorithms Tabu_VVI and NS. After running for approximately the same amount of time, Tabu_VVI achieves solutions with quality very close to the results of NS.

Table 5. Results by the best of all variants of Tabu_VVI and the algorithm of Nowicki and Smutnicki; in average percentage of the relative error to the lower bound, and the average time to the best, in seconds

instances	Tabu_VVI		NS	
	$avg(RE_{LB})$	$avg(time)$	$avg(RE_{LB})$	$avg(time)$
swv01-05	2.33	128	1.01	462
swv06-10	8.06	282	7.49	514
swv11-15	1.41	1855	0.51	360
yn	7	164	5.18	510
ta01-10	0.24	50	0.11	26
ta11-20	3.12	177	2.81	108
ta21-30	5.96	319	5.68	328
ta31-40	1.26	221	0.78	341
ta41-50	5.47	1016	4.7	975

6 Conclusions

We have developed a powerful, fast and innovative optimised search heuristic to solve combinatorial optimisation problems. It uses an exact technique from the operations research field to guide the search process of a metaheuristic. The procedure, named Tabu_VVI, uses the verification of violated valid inequalities as a diversification strategy of a tabu search procedure. The idea of this new method is to mimic the cuts in integer programming, letting the violated valid inequalities discard the current solution and guide the search from a local optimal solution to a more quality region of the search space.

The procedure was illustrated with an application to the job-shop scheduling problem. We presented some computational results for a large set of benchmark instances, along with comparisons to other similar and successful works. Our new method, Tabu_VVI, always performs better than other methods that combine exact and heuristic procedures. It compares most favourably to two other leading methods for solving the job-shop scheduling problem; the guided local search of Balas and Vazacopoulos [9] and the tabu search with shifting bottleneck of Pezzella and Mirelli [10]. When compared to the state of the art tabu search of

Nowicki and Smutnicki [2], after running for approximately the same amount of time, Tabu_VVI achieves solutions with quality very close to theirs.

References

1. Fernandes, S., Lourenço, H.R.: Optimized search heuristics. Technical report, Universitat Pompeu Fabra (2007)
2. Nowicki, E., Smutnicki, C.: An advanced tabu search algorithm for the job shop problem. Journal of Scheduling 8, 145–159 (2005)
3. Chen, S., Talukdar, S., Sadeh, N.: Job-shop-scheduling by a team of asynchronous agent. In: IJCAI 1993 Workshop on Knowledge-Based Production, Scheduling and Control, Chambéry, France (1993)
4. Denzinger, J., Offermann, T.: On cooperation between evolutionary algorithms and other search paradigms. In: 1999 Congress on Evolutionary Computation (CEC). IEEE Press, Los Alamitos (1999)
5. Tamura, H., Hirahara, A., Hatono, I., Umano, M.: An approximate solution method for combinatorial optimisation. Transactions of the Society of Instrument and Control Engineers 130, 329–336 (1994)
6. Adams, J., Balas, E., Zawack, D.: The shifting bottleneck procedure for job shop scheduling. Management Science 34(3), 391–401 (1988)
7. Applegate, D., Cook, W.: A computational study of the job-shop scheduling problem. ORSA Journal on Computing 3(2), 149–156 (1991)
8. Caseau, Y., Laburthe, F.: Disjunctive scheduling with task intervals. Technical Report LIENS 95-25, Ecole Normale Superieure Paris (July 1995)
9. Balas, E., Vazacopoulos, A.: Guided local search with shifting bottleneck for job shop scheduling. Management Science 44(2), 262–275 (1998)
10. Pezzella, F., Merelli, E.: A tabu search method guided by shifting bottleneck for the job shop scheduling problem. European Journal of Operational Research 120, 297–310 (2000)
11. Lourenço, H.R.: Job-shop scheduling: Computational study of local search and large-step optimization methods. European Journal of Operational Research 83, 347–367 (1995)
12. Lourenço, H.R., Zwijnenburg, M.: Combining large-step optimization with tabu-search: Application to the job-shop scheduling problem. In: Osman, I.H., Kelly, J.P. (eds.) Meta-heuristics: Theory & Applications. Kluwer Academic Publishers, Dordrecht (1996)
13. Schaal, A., Fadil, A., Silti, H.M., Tolla, P.: Meta heuristics diversification of generalized job shop scheduling based upon mathematical programming techniques. In: CP-AI-OR 1999 (1999)
14. Danna, E., Rothberg, E., Pape, C.L.: Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming, Ser. A 102, 71–90 (2005)
15. Jain, A.S., Meeran, S.: Deterministic job shop scheduling: Past, present and future. European Journal of Operational Research 133, 390–434 (1999)
16. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
17. Roy, B., Sussman, B.: Les problems d'ordonnancement avec constraintes disjonctives. Technical report, Notes DS 9 bis, SEMA, Paris (1964)
18. Fernandes, S., Lourenço, H.R.: A GRASP and branch-and-bound metaheuristic for the job-shop scheduling. In: Cotta, C., van Hemert, J. (eds.) EvoCOP 2007. LNCS, vol. 4446, pp. 60–71. Springer, Heidelberg (2007)

19. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
20. Glover, F.: Tabu search - part i. ORSA Journal on Computing 1(3), 190–206 (1989)
21. Glover, F.: Tabu search - part ii. ORSA Journal on Computing 2(1), 4–32 (1990)
22. Carlier, J., Pinson, E.: An algorithm for solving the job-shop problem. Management Science 35(2), 164–176 (1989)
23. Schrage, L.: Solving resource-constrained network problems by implicit enumeration: Non pre-emptive case. Operations Research 18, 263–278 (1970)
24. Fisher, H., Thompson, G.L.: Probabilistic learning combinations of local job-shop scheduling rules. In: Muth, J.F., Thompson, G.L. (eds.) Industrial Scheduling, pp. 225–251. Prentice-Hall, Englewood Cliffs (1963)
25. Lawrence, S.: Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University (1984)
26. Storer, R.H., Wu, S.D., Vaccari, R.: New search spaces for sequencing problems with application to job shop scheduling. Management Science 38(10), 1495–1509 (1992)
27. Taillard, E.D.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64(2), 278–285 (1993)
28. Yamada, T., Nakano, R.: A genetic algorithm applicable to large-scale job-shop problems. In: Manner, R., Manderick, B. (eds.) Parallel Problem Solving from Nature 2, pp. 281–290. Elsevier Science, Brussels Belgium (1992)
29. Nowicki, E., Smutnicki, C.: Some new tools to solve the job shop problem. Technical Report 60/2002, Institute of Engineering Cybernetics, Wroclaw University of Technology (2002)
30. Nowicki, E., Smutniki, C.: A fast taboo search algorithm for the job shop problem. Management Science 42(6), 797–813 (1996)