A new local search algorithm with shortlisting for plan repair in Flexible Job-Shop Problems: the 15 puzzle algorithm with potential

Hélène Soubaras

Thales Research & Technology
Decision & Optimization Laboratory, Campus Polytechnique, Palaiseau, France

Abstract : On-line rescheduling is a domain of temporal planning which is still very open to algorithmic studies since it arises practical difficulties: operational constraints on the differences between the initial and the repaired schedule, constraints on the processing means (the solver that was used to compute the initial schedule is no longer available, and there are limited computational resources), and CPU constraints (one must react in real time). In Flexible Job-Shop Problems (FJSPs), rescheduling is needed when a change occurs in the problem during the execution of the schedule: a machine disruption, or a new job to include in the schedule, or delays,... Local search algorithms are suited to adapt and thus to repair plans since their principle is to perform iteratively little modifications from an initial schedule. This is the case of the Mastoerlli & Gambardella's tabu search (Gambardella & Mastrolilli, 1996), a reference in the FJSP literature. We derived from it the new algorithm which is described in this paper, the 15 puzzle algorithm with potential. The CPU performance is improved by the use of a surrogate function – the potential – and a preselection. A surrogate function is a function which replaces anopther. Theoretical results and computational performance evaluation are shown.

1 Introduction

Rescheduling during a plan execution in a Flexible Job-Shop Problem (FJSPs), which is known to be a NP-hard problem, imposes additional practical difficulties:

- for operationel purpose, it can be necessary to take into account a stability criterion, i.e. to minimize the differences between the initial and the repaired schedule (e.g. for human reasons, to avoid disurbing a team),
- the computational resources to replan may be limited (e.g. it is performed on a remote platform), and in particular the solver which computed the initial schedule is not available,
- one must react in real time and thus the CPU must be limited.

This paper is a complete presentation of a new algorithm, the 15 puzzle algorithm with potential, that was initially developed and patented for rescheduling (Soubaras, 2012, 2015, 2016). But it can also perform more generally schedule adaptation (Soubaras, 2015) and even problem solving. It is a local search approach using the classical Mastorelli & Gambardella's tabu search neighborhood (Gambardella & Mastrolilli, 1996).

We will present the particular scheduling problem we address, the FJSP, and its associated rescheduling problem. Our original algorithm, the 15 puzzle algorithm with potential, will be explained after a detailed presentation of Masoterlli & Gambardella's tabu search, from which it is derived. The potential function it employs will be exposed in theory and computational results will compare the performance of the proposed algorithm and the original tabu search on bencnmarks.

2 The problem to solve

2.1 Model of the FJSP scheduling problem

The Flexible Job-Shop Problem (FJSP) is a generalization of the Job-Shop Problem (JSP), which is the classical scheduling problem consisting of jobs that have to be executed on a set of machines (or resources). A job is a sequence of tasks (or operations). Each task v has to be processed by one given machine $\mu(v)$ with a given processing time (or duration) p_v. Each machine can perform only one task at a time. To solve the JSP one must determine a start time for each task by optimizing a criterion – which is generally the makespan C_{max} (overall processing time). The JSP is considered as one of the most difficult problems in combinatorial optimization.

In the FJSP, for some tasks v there is a set $M(v)$ of several possible machines that can perform v. The FJSP can be decomposed into two sub-problems: first an assignment problem, to assign a machine $\mu(v)$ to all the tasks, and secondly a JSP. The FJSP is known as being a NP hard problem.

To develop our algorithm, we used an additional hypothesis, which is generally true in practice: the machines are fully exchangeable. This means that if a given task v can be performed with a machine r belonging to a set $M(v)$ of machines, if another task v' can be performed by a given machine $r' \in M(v)$, then it can be performed by any machine of $M(v)$. So $M(v') = M(v)$.

In this paper, we will also suppose that the processing times of the tasks does not depend on the machine they are assigned to. But our algorithm has also been extended to the case where the processing times depend on the machine (see in (Soubaras, 2016) where we take into account the moving times in a problem of FJSP with transportation).

So, the FJSP model we used can be precised as follows:

- a finite set of N_j jobs, each one being an ordered sequence of tasks,
- a finite set of N_m machines,
- a partition of the set of machines into N_{cm} classes of exchangeable machines,
- availability time constraints on machines,
- a set of N_p tasks v,
- the job j to which belongs each task,
- the precedence constraints between the tasks within the jobs,
- $M(v)$ the class of machine needed to execute each task v,
- p_v the processing time of each task v; it does not depend on the assigned machine $\mu(v)$.

To solve this FJSP, one must find for each task v a machine $\mu(v) = r \in M(v)$ and a start time $t_{start}(v)$.

2.2 The repair problem

Plan repair is needed when a change in the planning problem occurs during the plan execution. The plan which is being executed is no longer relevant. The tasks that has not been executed yet have to be rescheduled.

The difficulty in practical is that one must perform on-line planning, and thus provide a result in a short computation time, with possibly limited computation resources; in particular the initial solver which generated the initial schedule may be unavailable.

To solve the plan repair problem, we have the following additional parameters:

- a time of schedule interruption: t_0,
- the parameters describing the FJSP problem modification (jobs or machines added or suppressed, new availability time constraints on machines, new durations of tasks...).

If $t_0 = 0$, the execution has not started yet: this is the particular case of complete solving.

3 Literature review

Many methods have been proposed since decades for FJSP solving. But the problem of rescheduling has less been studied.

3.1 FJSP solving

3.1.1 Extension of JSP solving

As we said, a FJSP where the machines have been assigned to the tasks becomes a JSP. This allows to decompose the FJSP problem and to derive many solutions from the JSP to the FJSP (Xia & Wu, 2005). For solving the realistic case with more than two jobs, two types of approaches have been used: hierarchical approaches, where the two sub-problems of assignment of tasks to machines and sequencing are treated separately, and integrated approaches. Hierarchical approaches are based on the idea of decomposing the original problem in order to reduce its complexity. Brandimarte (Brandimarte, 1993) was the first one to apply a decomposition approach into the FJSP. He solved the assignment sub-problem by using some existing dispatching rules and then focused on the scheduling sub-problem, which was solved by using a tabu search heuristic. After that Paulli applied a hierarchical approach (Paulli, 1995). Zribi *et al.*focused on the assignment problem itself with a hierarchical approach (Zribi *et al.*, 2007). Saidi *et al.*presented a mathematical model and a tabu search algorithm to solve the FJSP with sequence-dependent setups (Saidi-Mehrabad & Fattahi, 2007). They used a hierarchical approach with two heuristics, the first one for assigning each task to one machine out of two possible machines and the second one for sequencing. Fattahi *et al.*(Fattahi *et al.*, 2007) presented a mathematical model to achieve optimal solution for small sized problems and developed two heuristic approaches, with tabu search, to solve the real sized problems; they concluded that the hierarchical algorithms have better performance than the integrated ones.

Several algorithms using the shifting bottleneck heuristic, which is one of the classical algorithms to solve the JSP, have been extended to the FJSP. Q. Zhang studied one approach in his PhD thesis (Zhang, 2012) and other publications (Zhang *et al.*, 2011, 2012b,a) for the FJSP with transportation problem. J. Gao *et al.*(Gao *et al.*, 2006a) proposed for the FJSP a hybrid of a genetic algorithm and the shifting bottleneck. J. Gao *et al.*also proposed the same kind of approach for multiobjective FJSPs (Gao *et al.*, 2007).

3.1.2 Local search techniques

Mastorelli & Gambardella (Gambardella & Mastrolilli, 1996) proposed a local search approach for FJSP scheduling. It is based on an intuitive heuristic where the idea is to move some tasks from one machine queue to another. Their approach has been widely reused in the literature. We will also apply this heuristic to plan repair and plan adaptation in this paper (see Section 4). Liouane *et al.*proposed a solution with ant colony and local search (Liouane *et al.*, 2007).

3.1.3 Genetic approaches

Genetic approaches are quite promising to solve such a hard problem, and the literature proposes many studies. The proposed algorithms are often hybrid. Indeed, it appears that the genetic algorithms converge more rapidly if they are associated with a local search optimization.

J. Gao *et al.*proposed a hybrid genetic algorithms for the FJSP (Gao *et al.*, 2006b), one of them involving variable neighborhoods (Gao *et al.*, 2008). They also proposed a method called *multistage based genetic algorithm* to solve the FJSP problem (Gen *et al.*, 2009). There are also many multiobjective approaches.

Mastorelli & Gambardella's neighborhood for tabu search (Gambardella & Mastrolilli, 1996) has also been associated by G. Zhang *et al.*to genetic approaches ((Zhang *et al.*, 2008b). G. Zhang *et al.*also proposed a genetic algorithm with variable neighborhood for the FJSP (Zhang *et al.*, 2008a).

3.2 Existing techniques for plan repair

In the literature, there are several plan repair strategies that do not use the initial solver; in (Nebel & Koehler, 1995) the dilemma of repairing a plan or replanning from scratch is discussed; in (Alterman, 1988; Kambhampati & Hendler, 1992), a theoretical formalism of causality is studied. Some plan adaptation with search techniques use the LPG (Local search for Planning Graphs) algorithm (Gerevini & Serina, 2000; Fox *et al.*, 2006), which is a local search based on a graph. Many other approaches use partial plans, such as *refinement planning* (van der Krogt & de Weerdt, 2005), which proceeds in two phases: first removing the tasks that prevent the plan from reaching its goal, then planning to extend the partial plans.

Local search is well suited to plan repair since it proposes a new solution close to a given initial one. Huang (Huang *et al.*, 2010) proposed a local search repair method for FJSP by formulating as a constraint satisfaction problem.

Y. M. Wang *et al.*proposed a genetic algorithm for the FJSP with machine disruptions (Wang *et al.*, 2013).

Apart from the plan repair issue, the problem of handling uncertainties on task processing times has also been studied in the literature (Beaudry *et al.*, 2010; Coles *et al.*, 2011). These approaches are stochastic (they are interested in the probability of satisfying a goal). In these approaches time is a constraint in models involving concurrency between tasks.

We also studied another approach to solve our plan repair problem in a crisis management application, which is based on an automatic generation of the description of the updated planning problem in PDDL language Soubaras *et al.* (2013), in order to compute an exact solution with CPT or a suboptimal but faster one with YAHSP (Vidal & Geffner, 2006). But those tools can solve only small problems.

4 The reference tabu search algorithm

Before describing our proposed solution, we will describe here the classical Mastorelli & Gamberdella tabu search approach that was proposed for FJSP solving (Gambardella & Mastrolilli, 1996). Our algorithm will be derived from it.

The principle of any tabu search heuristic is to start from one given solution and search iteratively new solutions, each one being in a neighborhood of the previous one, and chosen in this neighborhood by optimizing a given criterion. To avoid a cyclic behavior of the algorithm, it is forbidden to choose a solution which has already been choosen. This is why one stores a tabu list. Tabu search is not an exact algorithm, since it is not sure to reach the optimum, but it is quite efficient. In this particular tabu search algorithm, the criterion to optimize is the makespan. The neighborhood and the tabu list are defined hereafter.

4.1 Moving a task

The local search consists in applying so-called *elementary modifications* in the schedule, and choosing good candidates iteratively. The elementary modifications are movings of tasks and they define the neighborhood, i.e. the set of possible solutions to examine at one step. One moves a task in the schedule from one machine queue to another, like in a sliding puzzle, by replacing the machine r of a given task by another one r' belonging to the same class of machines.

4.2 The tabu list

The tabu list is defined as follows ((Gambardella & Mastrolilli, 1996)): when moving a task v from its original queue onto the queue of machine r' behind task v_{prev}, one adds the triple =(v_{prev}, v, r') to the tabu list. If the tabu list is too long its size is limited by forgetting the oldest triples.

5 The proposed 15 puzzle algorithm with potential

We will describe here the 15 puzzle algorithm with potential, which is derived from the Mastorelli & Gamberdalla tabu search approach for FJSP solving (Gambardella & Mastrolilli, 1996), described in Section 4. The difference is that here the computation of the makespan to optimize is replaced by a surrogate function named *potential*, and an associated preselection.

We also investigated a multiple criteria version of the 15 puzzle algorithm with potential. We detailed it in (Soubaras, 2015): the makespan is replaced by several criteria that are applied in lexicographic order, even if the makespan remains the main criterion. The other criteria are distances that will be reminded in Section 6.1.

5.1 The shortlist principle

The idea is to use an approximate criterion – the potential – which is faster to compute than the actual makespan, in order to reduce the computational time (CPU). Some solutions are shortlisted on the basis of the potentials, and then the computation of the actual makespan is performed on the shortlist only (and not on all the neighborhood).

The algorithm examines successively a growing number N_{modif} of elementary modifications of the initial plan. These modifications are defined exactly as in the reference tabu search described in Section 4.1. At each iteration, we have a selected schedule containing $N_{modif} - 1$ modifications and we examine how to add one more modification.

We examine all possible additional elementary modification and reject it if it is in the tabu list (of Section 4.2), or if it does not satisfy the precedence constraints within the jobs; otherwise, we compute a function called potential. Then we shortlist the $N_{shortlist}$ modifications who have the best potentials; then we compute their associated schedules and their makespan; finally we retain the solution with best makespan. And we update the tabu list.

Note that, as we said in the introduction, the 15 puzzle algorithm with potential can also be used, as other local search algorithms, to solve the problem completely. One must start by generating randomly a first non-optimal solution, and improve it by local search iterations. This can be combined with genetic techniques to select good initial random solutions.

5.2 The potential function

The potential function used in the 15 puzzle algorithm is a surrogate function which replaces the exact calculation of the makespan C_{max} by an approximation which is faster in computational time. It consists in bounding the effect on the makespan C_{max} we will obtain after moving a task.

5.2.1 Expression

Before making an elementary modification, each job j of a schedule has a known completion time $C_{max}(j)$. The known makespan C_{max} is the maximal completion time:

$$C_{max} = \max_j C_{max}(j)$$

Suppose we want to move a given task v of duration p_v in the queues: this will delay some other tasks of at most p_v and put forward other tasks of at most p_v. One does not know the new completion time C'_{max}, but one knows which jobs j will be delayed or put forward by this modification: the jobs that have tasks behind v in the start queue may be put forward, and the jobs that have tasks behind v in its new queue may be delayed.

So, one can compute for each job j the following bounds $\underline{C'_{max}}(j)$ and $\overline{C'_{max}}(j)$ for its new completion time $C'_{max}(j)$ as follows:

For a job j that may be put forward, we compute the following lower bound for $C'_{max}(j)$:

$$\underline{C'_{max}}(j) = C_{max}(j) - p_v$$

otherwise, if the job will not be put forward:

$$\underline{C'_{max}}(j) = C_{max}(j)$$

For a job j that may be delayed, we compute:

$$\overline{C'_{max}}(j) = C_{max}(j) + p_v$$

otherwise, if the job will not be delayed:

$$\overline{C'_{max}}(j) = C_{max}(j)$$

so we can deduce the bounds for C'_{max} as follows:

$$\underline{C'_{max}} = \max_j \underline{C'_{max}}(j) \quad \text{and} \quad \overline{C'_{max}} = \max_j \overline{C'_{max}}(j)$$

The potential function is a trade-off between those two bounds, by averaging with a risk aversion coefficient α. So it is defined by:

$$V = (1 - \alpha) \times \underline{C'_{max}} + \alpha \times \overline{C'_{max}} \tag{1}$$

5.2.2 Justification

We want to show here that the potential V is related to C'_{max}, the resulting makespan after an elementary modification (a moving of a task).

The expressions of $\underline{C'_{max}}$ and $\overline{C'_{max}}$ given in Section 5.2.1 imply that

$$\overline{C'_{max}} - \underline{C'_{max}} \leq 2p_v$$

On the other hand, Equation 1 implies, for a given modification:

$$\underline{C'_{max}} \leq V \leq \overline{C'_{max}}$$

So we can deduce

$$|V - C'_{max}| \leq 2p_v \tag{2}$$

We have thus shown that the potential value is close to the makespan.

5.2.3 Generalization to uncertainty about the durations

One can notice that the potential can be computed as well when there are uncertainties about the durations p_v of some ongoing or future tasks v if we know their bounds $p_v \in I_v = [\underline{p_v}, \overline{p_v}]$. I_v is the uncertainty interval for p_v. Note that if p_v is certain, $I_v = \{p_v\}$.

In a sequence of two tasks v and v' with uncertain durations, the resulting uncertainty interval for the sequence duration will be

$$I_v \otimes I_{v'} = [\underline{p_v} + \underline{p_{v'}}, \overline{p_v} + \overline{p_{v'}}] = \{t + t', /t \in I_v, t' \in I_{v'}\} \tag{3}$$

This operation $\otimes$ on uncertainty intervals is associative and it can be repeated for any finite number of successive tasks and delays. So one can compute with Equation 3 the bounds $\underline{C'_{max}}(j)$ and $\overline{C'_{max}}(j)$ for the jobs completion times, and then deduce the corresponding values for the bounds of the resulting makespan $\underline{C'_{max}}$ and $\overline{C'_{max}}$, and so the potential V. This extension to uncertainty intervals makes the potential function more powerful than the exact computation of the makespan, which cannot be known in that case.

One can note that the uncertainties about the duration of some tasks broadens the difference between the bounds for C'_{max}. This makes the potential more random. But the question is to know when the exact duration of the tasks is known, and what to do then.

We also propose hereafter a fuzzy version for the potential, by introducing membership functions to the uncertainty intervals and combining them for the function $\otimes$:

For the two membership functions $f(t)$ and $g(t')$ satisfying $f(t)$ (resp. $g(t')) \in [0;1]$ if $t \in I$ (resp. $t' \in I'$), and is null otherwise, we have the resulting membership function $h(u)$ for $u = t + t' \in I \otimes I'$ given by

$$h(u) = \max_{t \in I} \left(f(t) g(u-t) \right).$$

The fuzzy potential can be the value corresponding to the maximum:

$$V = Argmax(h)$$

or an average weighted by the membership function:

$$V = \frac{\int u.h(u)}{\int h(u)}$$

With thes new expressions of the potential, our proposed 15 puzzle algorithm, as well as the tabu search, keeps its properties to take into account the stability criterion.

In this paper we tested the proposed algorithm in the crisp case only, but this expression for a fuzzy potential shows the possibilities of the potential function principle.

6 Performance evaluation

In this Section, we will compare the performances of the 15 puzzle algorithm with potential to the reference tabu search (Gambardella & Mastrolilli, 1996), on the complete solving of a series of some FJSP problems.

6.1 Performance criteria

Various performance criteria can be found in the literature for FJSP solving (Gambardella & Mastrolilli, 1996; Kacem *et al.*, 2002). The basic ones we will use rely on the makespan, denoted as C_{max} or C, and the computation time (CPU). We will add two distance criteria defined below. So, we will present values for the following performance criteria:

- $Av(C_0)$ is the average initial makespan measured on the 5 random allocations used as initial schedules,
- C_{best} is the minimal makespan over the 8 runs obtained with the algorithm,
- $< C >$ is the average makespan obtained with the algorithm,
- $\sigma(C)$ is the standard deviation of the makespan obtained with the algorithm,
- $< CPU >$ is the average computation time in seconds,
- D_t the temporal distance,

  $$D_t = \sqrt{\sum d_t(v, v')}$$

  where (v, v') are the couples of corresponding tasks in the schedule before and after repair, and

  $$d_t(v, v') = \left(t_{start}(v) - t_{start}(v')\right)^2 + \left(t_{end}(v) - t_{end}(v')\right)^2$$

  is the square distance between the initial times t_{start} and final times t_{end} of the tasks,

- a distance in the neighborhood domain, D_m (which is equal to the number of modifications).

6.2 Computational results

We will present our results on classical benchmarks of FJSP problems that have been tested by Mastorelli & Gambardella (Gambardella & Mastrolilli, 1996) and in many publications related to this reference article.

These are FJSP problems for complete solving. We did not use schedule repair benchmarks because those FJSP benchmarks are very classical, and because if a local search algorithm manages to solve a problem, it is necessarily able to adapt it and then to repair a schedule.

6.2.1 The set of benchmarks

In their famous reference article (Gambardella & Mastrolilli, 1996), Mastorelli & Gambardella published their results on a series of benchmarks, that have been reused by many people in the literature. Four of them are FJSP problems available on the IDSIA (institut de recherche en intelligence artificielle, Manno, Switzerlan) web site: `http://people.idsia.ch/~monaldo/fjsp.html`.

- Brandimarte ((Brandimarte, 1993));
- Dauzere (Dauzère-Peres and Pauli (Dauzère-Pérès & Paulli, 1997));
- Barnes (Chambers ans Barnes (Chambers & Barnes, 1996));
- Hurink ((Hurink *et al.*, 1994)).

We analyzed all of them to see whether they satisfy the hypothesis of fully exchangeable machines. The results are shown in Table 1.

	Fully exchangeable	N_{max} (Max. number of iterations)
Brandimarte	all: No	10^5
Dauzere	01a, 02a, 03a, 07a, 08a, 09a, 13a, 14a, 15a: Yes 04a, 05a, 06a, 10a, 11a, 12a, 16a, 17a, 18a: No	4×10^5
Barnes	all: Yes	4×10^5
Hurink	edata: Yes rdata: Yes vdata: Yes	10^5

Table 1: Analysis of each benchmark, and maximum number of iterations used in (Gambardella & Mastrolilli, 1996).

To evaluate the 15 puzzle algorithm with potential, tests were performed on a set of benchmarks where the machines are fully exchangeable. The problem size was first extracted and they are shown in Table 2.

	N_j	N_m	N_p	jobs length
Hurink/rdata la01, la02, la03, la04 and la05	10	5	50	5
Barnes mt10x	10	11	100	10
Barnes mt10xx	10	12	100	10
Barnes mt10xxx	10	46	100	10

Table 2: Size of the tested benchmarks. In each one of these problems, all the jobs have the same length.

Problem	$Av(C_0)$	C_{best}	$< C >$	$\sigma(C)$	$< CPU >$	D_t	D_m
Hurink/rdata							
la01	2177.4	581	599.6	13.71	127.78	1.61	90
la02	1956.4	585	601	23.85	143.12	1.5	96.2
la03	1490.4	497	505.4	6.62	126.38	1.63	93.2
la04	1777.8	521	538.6	11.36	129.31	1.45	100
la05	1681.6	466	508	26.08	131.85	1.98	101.8
Barnes							
mt10x	3214.6	1066	1140.2	53.87	147.94	2.75	93.4
mt10xx	3274.4	1112	1163	29.21	152.67	2.72	91.8
mt10xxx	3214.6	1116	1152.6	29.67	162.07	2.72	89.8

Table 3: Performance on the test benchmarks obtained with the tabu search. For each benchmark we generated 5 initial schedules randomly.

Problem	C_{best}	$< C >$	$\sigma(C)$	$< CPU >$	D_t	D_m
Hurink/rdata						
la01	809	850.8	34.96	79.26	1.41	30.4
la02	700	767.8	48.95	85.21	1.42	36.6
la03	642	774.2	137.48	115.15	1.13	23.6
la04	841	1027.8	110.51	92.71	0.83	17.4
la05	740	890.6	127.11	84.92	1.31	23.8
Barnes						
mt10x	1513	1713.4	190.67	42.51	1.46	20
mt10xx	1585	1645.2	50.83	35.44	1.62	26
mt10xxx	1573	1672.2	107.23	47.99	1.61	27.6

Table 4: Performance on the test benchmarks obtained with the 15 puzzle algorithm with potential. The shortlist size was $N_{shortlist} = 7$ and the risk aversion coefficient was $\alpha = 0.5$. We used the same random initial schedules as in Table 3.

6.2.2 Performance comparison between the two approaches

In Tables 3 and 4, we provide the results obtained with the reference tabu search and the 15 puzzle algorithm with potential. The 5 random schedules that we generated as initial solutions were the same as in the tabu search implementation.

The comparison of those results is summarized in Table 5.

We can notice clearly that, for a loss in the makespan, the 15 puzzle algorithm is faster and performs better temporal distance.

Problem	C_{best}	$< C >$	$< CPU >$	D_t	D_m
Hurink/rdata	+42%	+58%	-30%	-28%	-72%
Barnes	+42%	+46%	-75%	-33%	-73%

Table 5: Average gains on the performance values obtained with the 15 puzzle algorithm with potential in comparison to the tabu search. Note that the makespan is less good (since it is higher for the 15 puzzle), but the CPU and the temporal distance are better (inferior).

Note that we must consider the performance comparison between the tabu search and the 15 puzzle algorithm with care, since it is highly dependent on the way we compute the potential comparatively to the exact makespan. In particular, we did not include techniques to eliminates cycles before computing them, and this is a loss of time because they are eliminated only during the computing of the makespan.

6.3 Complexity

The 15 puzzle algorithm with potential is interesting in comparison to the tabu search – from which it is derived – because it is faster. We will estimate here the difference of complexity.

Let τ_{15} be the time needed for the computation of the potential for one modification, and let $\tau_C$be the time necessary to compute the associated makespan C_{max}. There are in average N_p/N_m tasks per machine queue, so the same number of possible places in a queue. And there are N_m/N_{cm} exchangeable machines that can perform a given task. So there are in average

$$N_p/N_m \times N_m/N_{cm} = N_p/N_{cm}$$

possible places to put each one of the N_p tasks. When one of these places is chosen, one obtains a new modification. So there are N_p^2/N_{cm} possible modifications to explore at each iteration.

There are N_{modifs} iterations, and for each one retains the best solution. Thus, the total computation time to perform N_{modifs} modifications with the tabu search

$$T_{tabu} = \mathcal{O}(\tau_C \times N_{modifs} \times N_p^2/N_{cm})$$

In the 15 puzzle algorithm with potential, at each one of the N_{modifs} iterations, the potential is computed for all the possible modifications, but the complete calculus of C_{max} will be performed only on the $N_{shortlist}$ shortlisted possible solutions. The overall computational time of the algorithm will then be

$$T_{15} = N_{modifs} \times \mathcal{O}(\tau_{15} \times N_p^2/N_{cm} + \tau_C \times N_{shortlist})$$

As $\tau_C = \mathcal{O}(N_p)$ and $\tau_{15} = \mathcal{O}(N_j)$, the ratio of the CPUs of the two algorithms will be around

$$T_{15}/T_{tabu} = \mathcal{O}(N_j/N_p + N_{shortlist} \times N_{cm}/N_p^2)$$

The second term will be negligeable and so

$$T_{15}/T_{tabu} = \mathcal{O}(N_j/N_p)$$

So, if one takes a small value for the shortlist length, the 15 puzzle algorithm with potential will be the less complex and thus faster. This theoretical evaluation of the complexity confirms the computational results we showed in Section 6.2.2.

7 Conclusion

We have proposed a new approach, the 15 puzzle algorithm with potential, for schedule solving, repair and adaptation in FJSP problems. It is based on a local search of moving tasks from one machine to another. It is an original approach derived from a classical tabu search (Gambardella & Mastrolilli, 1996)], where the optimization is based on a shortlisting with calculus of potentials. It works from a given initial schedule without using any initial external solver. One operational advantage is that this algorithm handles a criterion to maintain little distance to the initial schedule. The 15 puzzle is less optimal in makespan than the tabu search but it often obtains a solution which is close to the optimum in a shorter time and with less distance to the initial schedule. The 15 puzzle can also handle a larger class of problems since it can address uncertainties about the processing times of the tasks (thanks to the fuzzy version of the potential function).

Implementation and validation tests show that this new algorithm works, providing repaired schedules that are shorter in time duration than the single compression of a disrupted schedule. For small problems, it often finds an optimal solution with few modifications. And if is well suited to manage a flow of demands.

This capability to involve uncertainty about tasks processing time can be exploited more in further works. We just showed theoretically in this paper that an uncertainty about task processing times can be taken into account in the potential function. But one could do more testing with the fuzzy version of the potential.

The benefit of the computation time saved by the shortlisting performed in the algorithm can be particularly interesting in genetic approaches, where it is often useful to introduce local search. This is probably one of the most promising ways to go on with this research work.

References

ALTERMAN R. (1988). Adaptive planning. *Cognitive Science*, **12**(3), 393–421.

BEAUDRY E., KABANZA F. & MICHAUD F. (2010). Planning for concurrent action executions under action duration uncertainty using dynamically generated bayesian networks. In *ICAPS*, p. 10–17.

BRANDIMARTE P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, **41**(3), 157–183.

CHAMBERS J. B. & BARNES J. W. (1996). Tabu search for the flexible-routing job shop problem. *Graduate program in Operations Research and Industrial Engineering, The University of Texas at Austin, Technical Report Series, ORP96-10*.

COLES A. J., COLES A. I., CLARK A. & GILMORE S. T. (2011). Cost-sensitive concurrent planning under duration uncertainty for service level agreements. In *Proceedings of the Twenty First International Conference on Automated Planning and Scheduling (ICAPS-11)*.

DAUZÈRE-PÉRÈS S. & PAULLI J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, **70**, 281–306.

FATTAHI P., MEHRABAD M. S. & JOLAI F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, **18**(3), 331–342.

FOX M., GEREVINI A., LONG D. & SERINA I. (2006). Plan stability: Replanning versus plan repair. In *In Proc. ICAPS*, p. 212–221: AAAI Press.

GAMBARDELLA L. & MASTROLILLI M. (1996). Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling*, **3**(3).

GAO J., GEN M. & SUN L. (2006a). A hybrid of genetic algorithm and bottleneck shifting for flexible job shop scheduling problema hybrid of genetic algorithm and bottleneck shifting for flexible job shop scheduling problem. In *Proceedings of the 8th annual conference on genetic and evolutionary computation*, p. 1157–1164: ACM.

GAO J., GEN M. & SUN L. (2006b). Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. *Journal of Intelligent Manufacturing*, **17**(4), 493–507.

GAO J., GEN M., SUN L. & ZHAO X. (2007). A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering*, **53**(1), 149–162.

GAO J., SUN L. & GEN M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, **35**(9), 2892–2907.

GEN M., GAO J. & LIN L. (2009). Multistage-based genetic algorithm for flexible job-shop scheduling problem. In *Intelligent and evolutionary systems*, p. 183–196. Springer.

GEREVINI A. & SERINA I. (2000). Fast plan adaptation through planning graphs: Local and systematic search techniques. In *Proc. of 5st Int. Conf. on AI Planning and Scheduling (AIPS 2000)*, p. 112–121, Menlo Park, CA, USA: AAAI Press.

HUANG Y., ZHENG L., WILLIAMS B. C., TANG L. & YANG H. (2010). Incremental temporal reasoning in job shop scheduling repair. In *Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on*, p. 1276–1280: IEEE.

HURINK J., JURISCH B. & THOLE M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, **15**(4), 205–215.

KACEM I., HAMMADI S. & BORNE P. (2002). *Lower Bounds for Evaluating Schedule Performance in Flexible Job Shops*. Rapport interne, DTIC Document.

KAMBHAMPATI S. & HENDLER J. A. (1992). A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence*, **55**(2-3), 193–258.

LIOUANE N., SAAD I., HAMMADI S. & BORNE P. (2007). Ant systems & local search optimization for flexible job shop scheduling production. *International Journal of Computers, Communications & Control*, **2**(2), 174–184.

NEBEL B. & KOEHLER J. (1995). Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence*, **76**, 427–454.

PAULLI J. (1995). A hierarchical approach for the fms scheduling problem. *European Journal of Operational Research*, **86**(1), 32–42.

SAIDI-MEHRABAD M. & FATTAHI P. (2007). Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, **32**(5-6), 563–570.

SOUBARAS H. (2012). Patent no fr1202686 – procédé de réordonnancement d'une liste initiale ordonnée de tâches en une liste finale ordonnée de tâches.

SOUBARAS H. (2015). Reactive multiobjective local search schedule adaptation and repair in flexible job-shop problems. In *Proc. of MCO (Int. Conf. on Modelling, Computation and Optimization of Information Systems and Management Science)*, Metz, France.

SOUBARAS H. (2016). Méthodes locales pour la réparation en ordonnancement de fjsp avec transport. In *Proc. of ROADEF*, Compiègne, France.

SOUBARAS H., ALIGNE F. & SAVÉANT P. (2013). Omar : un outil d'aide à la décision pour optimiser, suivre, alerter et réparer en gestion de crise. In *Proc. of JFPDA*, Lille.

VAN DER KROGT R. & DE WEERDT M. (2005). Plan Repair as an Extension of Planning. In 15^{th} *International Conference on Automated Planning and Scheduling (ICAPS-2005)*, p. 161–170: AAAI Press.

VIDAL V. & GEFFNER H. (2006). Branching and Pruning: An Optimal Temporal POCL Planner based on Constraint Programming. *Artificial Intelligence*, **170**(3), 298–335.

WANG Y. M., YIN H. L. & QIN K. D. (2013). A novel genetic algorithm for flexible job shop scheduling problems with machine disruptions. *Int. J. of Advanced Manufacturing Technology*, **68**, 1317–1326.

XIA W. & WU Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, **48**(2), 409–425.

ZHANG G., GAO L., LI X. & LI P. (2008a). Variable neighborhood genetic algorithm for the flexible job shop scheduling problems. In *Intelligent Robotics and Applications*, p. 503–512. Springer.

ZHANG G., SHI Y. & GAO L. (2008b). A genetic algorithm and tabu search for solving flexible job shop schedules. In *Computational Intelligence and Design, 2008. ISCID'08. International Symposium on*, volume 1, p. 369–372: IEEE.

ZHANG Q. (2012). *Contribution à l'ordonnancement d'ateliers avec ressources de transports*. PhD thesis, Université de Technologie de Belfort-Montbeliard.

ZHANG Q., MANIER H. & MANIER M. (2012a). A hybrid metaheuristic algorithm for flexible job-shop scheduling problems with transportation constraints. In *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012*, p. 441–448.

ZHANG Q., MANIER H. & MANIER M.-A. (2011). A disjunctive graph and shifting bottleneck heuristics for multi hoists scheduling problem. In *18th World Congress of the International Federation of Automatic Control (IFAC)*, volume 18, p. 6963–6968.

ZHANG Q., MANIER H. & MANIER M.-A. (2012b). A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Comput. Oper. Res.*, **39**(7), 1713–1723.

ZRIBI N., KACEM I., EL KAMEL A. & BORNE P. (2007). Assignment and scheduling in flexible job-shops by hierarchical optimization. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, **37**(4), 652–661.