Scheduling a Single Robot in a Job-Shop Environment through Precedence Constraint Posting

D. Díaz[1], M.D. R-Moreno[1],
A. Cesta[2], A. Oddi[2], and R. Rasconi[2]

[1] Universidad de Alcala, Alcala de Henares, Madrid, Spain
{daniel.diaz, mdolores}@aut.uah.es
[2] ISTC-CNR, Italian National Research Council, Rome, Italy
{amedeo.cesta, angelo.oddi, riccardo.rasconi}@istc.cnr.it

Abstract. The paper presents recent work on using robust state-of-the-art AI Planning and Scheduling (P&S) techniques to provide autonomous capabilities in a space robotic domain. We have defined a simple robotic scenario, reduced it to a known scheduling problem which is addressed here with a constraint-based, resource-driven reasoner. We present an initial experimentation that compares different meta-heuristic algorithms.

1 Introduction

Our current work is mainly aimed at injecting constraint-based, resource reasoning techniques applied to space robotics, especially for applications in planetary exploration. Promoting autonomy for future space missions certainly involves enormous benefits such as by reducing operational costs, enabling opportunistic science, or incrementing mission quality in terms of safety, science return, reliability and flexibility. Very often, the implementation of autonomous high-level competences *on-board* (specially to deep space missions) becomes essential since real time command dispatching is not possible due to the astronomic distances involved [7], or to the scarce time windows for establishing communication.

The following futuristic mission scenario is an hypothetical application example that aims at illustrating how autonomy could be applied in deep space missions.

Mission Scenario. *The complete scenario describes a future ESA deep space mission where groups of autonomous rovers are in charge of collaborating for transporting supplies between different martian sites. Typical rover activities include loading, transporting and unloading the supplies, as well as performing basic self-maintenance tasks. Rovers are able to autonomously synthesize efficient action plans by optimizing energy management and plan completion time. Furthermore, advanced on-board re-planning capabilities are necessary in order to hedge against environmental uncertainty (i.e., rugged terrain, harsh weather conditions, etc.).*

We start our analysis from a simplified version of the previous scenario. In particular, problem complexity is reduced by considering: (1) a unique robot and (2) a

completely deterministic environment. We refer to this simplified problem formulation as *scheduling a single robot in a job-shop environment*. This problem can succinctly be described as the problem of synchronizing the use of a set of machines to perform a set of jobs over time. Each job consists of a chain of activities that are to be sequentially processed on different machines for their entire execution. Each activity entails the processing of one item on behalf of machine whose utilization is exclusive for the whole duration of the activity. Each machine can only process one activity at a time, while all activity durations are fixed. Additionally, the rover is in charge of transporting the items among all present machines in order to allow their processing along all the jobs; at the beginning of the process all the items are kept in an initial storage area, while at the end all the items will have to be unloaded in a final storage area. Transportation times between different machines depend upon the traveling distance, and the rover can simultaneously carry a maximum number of items. The goal is to synthesize a schedule where both machine and rover activities are synchronized towards the successful execution of all jobs while minimizing total completion time (*makespan*).

The solution that we propose in this paper is based on the last reported results in constraint-based P&S techniques, with particular attention to the "Precedence Constraint Posting" (PCP) approach as described in [4, 9, 3].

The remainder of the paper is structured as follows: section 2 describes in detail the problem of *scheduling a single robot in a job-shop environment*; section 3 introduces our constraint-based solution method as well as a meta-heuristic strategy for solution optimization; in section 4 we report performance results on a set of reference benchmark problems; finally, a conclusions section closes the paper.

2 Problem formulation

In this section we define the tackled problem as well as the constraint-based representation we use as a reference for problem solving.

Problem description. The problem of scheduling a single robot in a job-shop environment involves synchronizing the use of a set of machines $M = \{m_1, \ldots, m_n\}$ and a rover R to perform a set of jobs $J = \{j_1, \ldots, j_n\}$ over time. Each job consists of a set of operations (or Activities) $A_j = \{l_{0j}, u_{1j}, a_{1j}, l_{1j}, \ldots, u_{ij}, a_{ij}, l_{ij}, \ldots, u_{nj}, a_{nj}, l_{nj}, u_{(n+1)j}\}$ to be sequentially processed, where a_{ij} is the activity belonging to Job j performed by machine $\mu_{ij} \in M$, while u_{ij} and l_{ij} are, respectively, the *Unload* and *Load* activities that the rover must perform at machine μ_{ij}. l_{0j} is the Load activity performed by the rover at the initial storage location, while $u_{(n+1)j}$ is the Unload activity performed by the rover at the final storage location.

The execution of all the activities is subject to the following constraints:

– *Resource availability*: each activity a_{ij} requires the exclusive use of μ_{ij} during its entire execution, i.e., no preemption is allowed. All the activities belonging to the same job demand distinct machines. The activities u_{ij} and l_{ij} require the exclusive use of the rover R.

– *Processing time constraints*: all activities a_{ij}, u_{ij} and l_{ij} have a fixed processing time. Both the machines and the rover can perform one operation at a time.
– *Transportation time constraints*: for each pair $< \mu_{ix}, \mu_{jy} >$ of machines, the rover *R* is in charge of moving all items processed by the machine μ_{ix} to the machine μ_{jy}. This entails performing a Load activity l_{ix} at μ_{ix}, transporting the item at μ_{jy}, and finally performing an Unload activity u_{jy} at μ_{jy}. The time necessary to travel from μ_{ix} to μ_{jy} is directly proportional to the traveling distance, and is modeled in the problem in terms of *sequence dependent setup times* st_{ij} which must be enforced between each $< l_{ix}, u_{jy} >$ activity pair. All setup constraints satisfy the *triangle inequality property*, i.e., given three machines μ_i, μ_j and μ_k, the condition $st_{ij} \leq st_{ik} + st_{kj}$ always holds.

Constraint-based problem representation. The solution here proposed considers the scheduling problem as a special type of a Constraint Satisfaction Problem (CSP) [6]. A general description of a scheduling problem as CSP involves a set of variables with a limited domain each, and a set of constraints that limits possible combinations. Hence, a feasible CSP solution is defined as an assignment of domain values to all variables which is consistent with all imposed constraints. A general CSP solution scheme can be seen as an iterative procedure that interleaves two main steps in each cycle:

– A *"decision making step"* where a variable is chosen to be assigned with a specific domain value. The decision of the assignment to do next could be taken by either systematically following an exhaustive search technique (such as a simple depth-first search), or by using more efficient approaches that use variable and value ordering heuristics to guide the search process. Typical general purpose heuristics generate variable and value orderings by selecting the "most constrained variable (MCV)" and the "least constraining value (LCV)" respectively.
– A *"propagation step"* where a set of "inference rules" prune unfeasible solutions in advance, by removing elements from variable domains when a decision step is performed. Path consistency algorithms such as "all pairs shortest paths" are typically used.

New problem modelling assumptions are considered in order to adapt the initial problem formulation to our constraint-based solution scheme. Hence, the problem is described now as follows (see figure 1): each machine μ_{ij} is considered as a binary resource that process all job operations a_{ij}. The Unload and Load activities are devised to be performed by a single robotic arm able to manage one item at a time (also modelled as a binary resource). We introduce an additional kind of activity c_{ij} that requires the use of a new cumulative resource to be processed by the rover, with the aim of modelling the rover capability of *simultaneously carrying multiple items*. The execution of c_{ij} activities starts when the corresponding Load activity l_{ij} starts, and finishes at the termination of the Unload activity u_{i+1j}.

Related work. Research in CSP-based scheduling has been mainly focused on the development of effective heuristic-biased models for efficiently controlling the scheduling

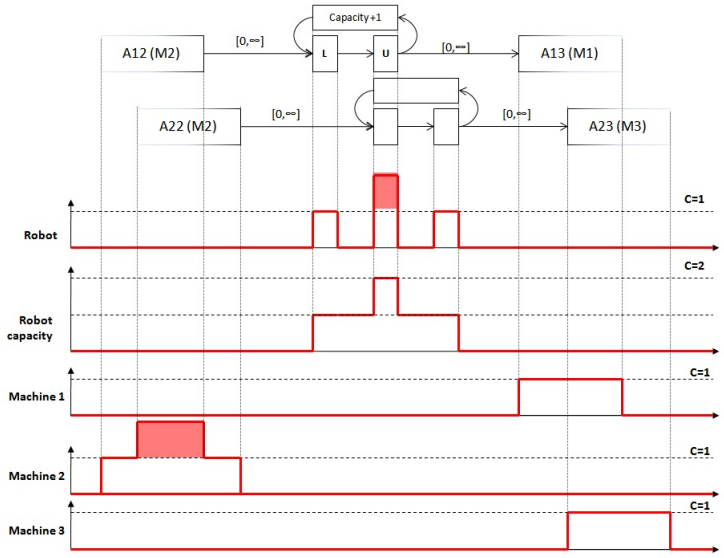

Fig. 1: A cumulative resource is introduced to model the multi-capacity robot usage.

search process (without incurring in the overload of a backtracking search). Two different directions can be mainly distinguished to cope with problems which involve both unary and cumulative or multi-capacity resources respectively. In the first case, some initial heuristic-biased procedures implement deterministic and one-pass solutions such as the precedence constraint posting algorithm proposed by [4]. In the second case, we can mention the resource profile-driven algorithm ESTA [3] as an efficient contraint-based scheduling technique that additionally addresses cumulative resources.

3 The profile-based solution (extended-ESTA)

Since cumulative resources are needed to model the multi-capacity robot usage, we have chosen the ESTA algorithm [3] as reference constraint-based solving procedure to implement our solution. Furthermore, we have studied and adapted the basic principles of a recent extension of the SP-PCP (Shortest Path-based Precedence Constraint Posting) algorithm proposed in [8]. More concretely we have adopted the new set of *dominance conditions* introduced in [8] as a set of four basic rules to decide the conflict resolution strategy by considering sequence-dependent setup times.

3.1 Adapting PCP to our robot scheduling problem

ESTA was initially developed for solving the scheduling problem known in OR literature as (RCPSP/max) [2]. It follows an advance precedence constraint posting schema that uses two different abstraction levels of the CSP formulation (i.e., the robot scheduling problem represented as a temporal constraint network) to deal with temporal and

resource aspects of the problem respectively. Thus, ESTA algorithm basically consists on iteratively interleaving the two following steps until a conflict-free schedule is found:

- *Temporal analysis.* Corresponds to the first step of the algorithm and consist on creating a basic temporal network (ground-CSP) to represent and reason about the temporal constraints of the problem. Thus, temporal constraint network described here corresponds to a Simple Temporal Problem (STP) formulation where time points represent start and end times of activities, and temporal constraints between time points represent both the duration of the activity and the precedence relations between pairs of activities. Temporal propagation (for computing current bounds for all time points after posting a new temporal precedence constraint) and solution extraction[1], are operations directly performed over this STP formulation.
- *Resource analysis.* Roughly speaking, the second step of ESTA basically consist on the following sequence: firstly, a meta-CSP representation is created by identifying a set of capacity violations inferred from previous ground-CSP, where variables correspond to the remaining resource conflicts and values to the set of feasible activity orderings to solve them; secondly, a resource conflict is selected by applying a variable ordering heuristic; and finally, selected conflict is solved by using a value ordering heuristic that imposes a new precedence constraint (over the ground-CSP) between some pair of competing activities that contributes to the conflict.

In other words, ESTA algorithm implements a (one-pass) greedy resource-driven scheduler that uses an earliest start-time resource profile projection (ground-CSP) to later perform a resource analysis and iteratively select and level "resource contention peaks" (i.e., over-commitments). More concretely, resource analysis consist on synthesizing the meta-CSP by computing (sampling) the Minimal Critical Sets (MCSs), i.e., sets of activities that overlaps in time and demand same resource causing over-commitments, such that whatever subset does not cause a resource conflict (see figure 2).

The search strategy for selecting and solving resource contention peaks is biased by the following variable and value ordering heuristics:

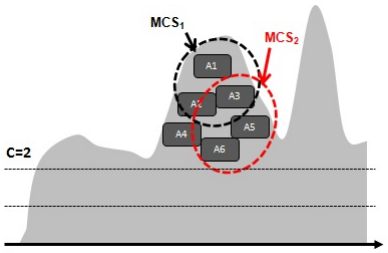

Fig. 2: Meta-CSP generation example (MCSs computation).

- Once the decision variables are computed (candidate MCSs), a most constrained variable ordering heuristic chooses the MCS with the smallest temporal flexibility (free temporal space).
- Conflict resolution is performed by a least constrained value ordering heuristic that levels the contention peak by posting a simple precedence constraint

[1] Solution extraction provides a conflict-free schedule in the form of Earliest Start Schedule (ESS): a consistent temporal assignment of all time points with their lower bound values that is also resource consistent.

Algorithm 1: Conflict selection and resolution process.

```
Conflict ← SelectConflict (MetaCSP)
Precedence ← SelectPrecedence (Conflict)
GroundCSP ← PostConstraint (GroundCSP, Precedence)
```

between two activities that belong to the related MCS according to the following criteria: the greater the flexibility is retained after posting a precedence ordering constraint, the more desirable it is to post that constraint. This kind of search heuristics that use the temporal flexibility retained between each pair of activities to bias the variable and value selection decisions are typically known as *slack-based heuristics* [11].

Algorithm 1 depicts in detail the basic steps corresponding to the conflict resolution process previously explained: within the first step, a collection of candidate MCSs is computed; the second step selects the most constrained MCS and the ordering choice to solve it; last step imposes the new leveling constraint within the ground CSP.

Some enhancements have been introduced to basic ESTA algorithm in order to extend it to specificities of the robot scheduling problem. Let us recall that two types of resources are managed in this case: on one hand, binary resources are used to model the machines and the robot usage; and on the other hand, a cumulative resource is used to model the robot multi-capacity aspect. We essentially introduce the following modifications: the `SelectPrecedence()` and `PostConstraint()` functions that implement the variable and value ordering decisions are able to detect different types of resource contention peaks and impose the corresponding (simple or setup time-bounded) precedence constraint. More concretely, the profile projection analysis now synthesizes two different sets of MCSs separately:

- First MCS set correspond to the resource profiles of the machines and robot multi-capacity usage, and it is synthesized in the same way than the original ESTA does.
- Second MCS set is the result of the contention peaks analysis performed over the robot usage resource. In this case, setup times associated to load/unload activities are also taken into account by introducing the underlying rationale of the extended dominance conditions: distances between each pair of activities are analysed such that a conflict is found if the separation between them is less than the corresponding setup time.

Figure 3 illustrates the different kind of activities related to both binary and cumulative resources as well as some precedence constraints between them.

Once both sets of candidate MCSs are computed, they are merged to select the most constrained MCS. Similarly, the solution of the conflict consists in imposing a *"simple precedence constraint"* if the selected pair of activities belonged to the first MCS set, or a *"precedence constraint with a setup time"* otherwise with the following exception: if a *"crossed conflict situation"* is solved (see figure 4), a specific profile projection analysis is performed with the aim of avoiding possible dead-ends. If the cumulative-related activity attached to the target activity constrained by the new ordering is involved in a peak, the opposite precedence ordering constraint is imposed.

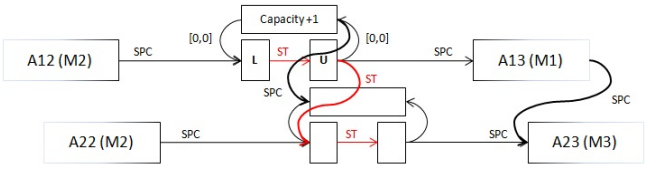

Fig. 3: Some examples of simple precedence constraints (SPC) and precedence constraints with setup times (ST).

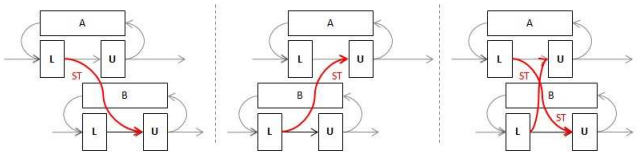

Fig. 4: The possible crossed orderings that may lead to a dead-end.

3.2 Providing better solutions

Since the previous one-pass, greedy solution does not guarantee optimality, we used an efficient optimization framework (in contrast to the costly backtracking-based schemes) with the aim of providing better results. We adopted the advanced IFLAT [10] iterative sampling optimization schema that allows us to find lower solution's makespan and overcome situations where unsolved conflicts are encountered. The underlying idea is to iteratively run the extended-ESTA algorithm such that different paths are randomly explored within the search space, by using a "meta-heuristic [1] strategy" to bias the process.

The algorithm 2 illustrates the IFLAT process. The procedure takes two parameters as input: an initial solution *S* and the amount of backtracking (*MaxFail*) that delimits the number of failed attempts at improving the solution. The exploration of different solutions is broaden by a meta-heuristic strategy that basically interleaves the following two steps on each cycle: firstly, a *retraction step* removes an arbitrary number of solving constraints (with a specific retracting probability) from the "critical path" of the last solution; and secondly, a *flattening step* attempts at repairing the "partially destroyed solution" by running extended-ESTA with it. If a better makespan solution is found, the best solution (S_{best}) is updated and the *counter* is reset to 0. Otherwise, if no improvements have been found within the *MaxFail* iterations, the algorithm returns the best solution encountered.

4 Preliminary Results Analysis

In this section we provide a first evaluation of the efficiency of both the deterministic extended-ESTA algorithm, and the iterative sampling framework for solution optimization. The test input data was obtained from previous work of [5] which aimed at

Algorithm 2: IFLAT optimization algorithm

```
Input: S, MaxFail
Output: S_best

S_best ← S
counter ← 0
while (counter ≤ MaxFail) do
    // Retracting step
    RELAX (S)
    // Repairing step
    FLATTEN (S)
    if (Mk (S)<Mk (S_best)) then
        S_best ← S
        counter ← 0
    else
        counter ← counter + 1
```

providing robust and efficient solutions for a large variety of scheduling problems. In particular we are interested in the "trolley problem" instances [12], as a well studied problem in OR that essentially contains the same elements than our robot scheduling problem[2].

The considered benchmarks consist on problems of size *n x m*, where *n* is the number of jobs and *m* corresponds to the number of machines. The number of operations or activities per job is equal to the number of machines. We used two different sets of solvable instances of size *10x5* where the capacity of the robot is *2* and *3* respectively. The equivalent benchmark sizes on our model is given by the following formula: $n \cdot (m + (m + 1) \cdot 2 + (m + 1) + 1)$. Hence, the real size of the scheduled problem instances is *10x24*.

We implemented the solving procedures under evaluation in Java and were tested them on a PC with 3Gb of RAM and an AMD Athlon(tm) XP 2400+ of CPU.

Table 1 shows the performance results of the tests corresponding to the deterministic extended-ESTA and solution optimization framework. The metrics provided are the makespan, the completion time, and the number of attempts that IFLAT performs during an estimated time of $\backsim$1800 seconds. Additionally, it shows the comparative performance value $\Delta LWU\%$[3] (the standard baseline performance metric used in the OR literature), as well as the completion time and the number of the IFLAT attempts in average. The general settings used for the IFLAT tests are the following: the maximum number of the relaxations and the removal probability are respectively set to 4 and 0.2, and the randomization factor is set to 0.2.

[2] Unfortunately, the results described in [5] are not available any more, hence we offer a comparison completely internal to our own work.

[3] Due to the lack of detailed results, the comparative analysis is performed against an estimated lower bound that corresponds to the length of the longest job.

	Extended-ESTA				IFLAT					
	Cap = 2		Cap = 3		Cap = 2			Cap = 3		
Benchmark	MK	CPU_s	MK	CPU_s	MK	CPU_s	$\#Iter$	MK	CPU_s	$\#Iter$
Robot_test (1)	2696	171.25	2680	199.92	2644	2330.65	31	2560	2035.86	22
Robot_test (2)	2817	163.60	2339	185.77	2528	1992.36	31	2566	2123.15	27
Robot_test (3)	2925	162.55	2660	173.86	2671	2131.64	24	2609	2172.49	26
Robot_test (4)	2962	164.55	2656	215.49	2737	2502.52	25	2591	2334.52	27
Robot_test (5)	–	–	2633	191.38	2780	3277.77	23	2384	2035.11	29
Robot_test (6)	–	–	2428	186.77	2512	2366.13	36	2213	2240.06	25
Robot_test (7)	–	–	2492	263.63	2584	2175.12	25	2721	2011.04	24
Robot_test (8)	2652	174.92	2916	251.74	2877	2062.49	26	2474	2054.08	27
Robot_test (9)	2509	173.63	2307	257.82	2291	2089.43	29	2093	2176.26	27
Robot_test (10)	2750	172.36	2463	235.48	2437	2208.64	28	2346	2097.10	28
Robot_test (11)	–	–	3272	253.21	3247	2816.75	21	2874	2052.16	26
Robot_test (12)	3279	125.99	3380	234.37	3343	2636.52	29	3212	2243.99	27
Robot_test (13)	3580	170.30	3179	224.38	3034	1991.38	27	2587	2333.39	28
Robot_test (14)	3064	175.98	3134	223.52	2613	3045.66	25	2558	2211.21	24
Robot_test (15)	2623	164.68	2991	220.45	2609	2457.38	28	2635	2008.09	27
Robot_test (16)	2900	179.57	2890	232.56	2705	2603.94	30	2493	2028.71	29
Robot_test (17)	–	–	3202	241.86	3071	2206.43	24	2858	2133.43	26
Robot_test (18)	3382	216.13	3130	243.87	2739	2200.67	29	2792	2402.53	28
Robot_test (19)	3461	171.28	3071	229.53	2956	2696.34	32	2893	2140.92	28
Robot_test (20)	–	–	2889	224.75	2573	2195.64	32	2521	2036.93	29
	$\Delta LWU\%$	CPU_s^{av}	$\Delta LWU\%$	CPU_s^{av}	$\Delta LWU\%$	CPU_s^{av}	$\#Iter^{av}$	$\Delta LWU\%$	CPU_s^{av}	$\#Iter^{av}$
	307.21	**170.49**	**288.60**	**224.52**	**276.53**	**2399.36**	**27.75**	**256.17**	**2128.55**	**26.70**

Table 1: Preliminary experimental results.

From previous results we can mainly extract the following conclusions: the makespan of the solutions is significantly improved, in average, with the IFLAT optimization framework on both instance sets. Furthermore, we can see that the number of solutions found is larger (all instances are solved) with IFLAT. Solution's makespans are also improved if we compare the results of both instance sets each other corresponding to the one-pass experiments. On the contrary, the computation time was increased since this is the price to pay in expenses of getting better quality results. It is worth saying that we are actually studying different ways to enhance the one-pass extended-ESTA solution such that by improving the performance of the profile analysis performed when a crossed constraint is found. This would allow us not only getting better completion times, but also increasing the possibility of optimizing the solution quality with IFLAT, since the number of iterations performed (within the same time) would be also increased.

5 Conclusions

In this paper we have studied the use of existing constraint-based, heuristic-biased techniques to solve the problem of "scheduling a robot in a job-shop environment". In particular we have chosen two different PCP-based algorithms as basis for building our

extended-ESTA solution: an enhanced SP-PCP for addressing problems that involve sequence-dependent setup times, and the advanced profile-based ESTA for dealing with (multi-capacity) cumulative resources. Extended-ESTA basically creates a meta-representation of the temporal network where resource aspects are explicitly addressed by performing a resource contention peaks analysis. Furthermore, setup times are considered when contention peaks are solved since transportation constraints have to be satisfied. A deterministic and greedy implementation has been presented, to later embed it within a larger iterative-sampling search framework. This allowed us to promote solution optimization by broaden search space coverage. Then, we have performed a preliminary empirical experiment over a set of reference benchmarking problems to get initial solution quality evaluation.

Acknowledgments. Daniel Diaz is supported by the European Space Agency (ESA) under the Partnering program Networking and Partnering Initiative (NPI) "Autonomy for Interplanetary Missions" (ESTEC-No. 2169/08/NI/PA). CNR authors are partially supported by MIUR under the PRIN project 20089M932N (funds 2008). The second UAH author is supported by Castilla-La Mancha project PEII09-0266-6640. Special thanks to Michele Van Winnendael for his support throughout the whole course of the study.

References

1. Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.
2. P. Brucker, A. Drexl, R. Mohring, K. Neumann, and E. Pesch. Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. *European Journal of Operations Research*, 112(1):3–41, 1999.
3. Amedeo Cesta, Angelo Oddi, and Stephen F. Smith. A constraint-based method for project scheduling with time windows. *J. Heuristics*, 8(1):109–136, 2002.
4. C. Cheng and S.F. Smith. Generating Feasible Schedules under Complex Metric Constraints. In *Proceedings 12th National Conference on AI (AAAI-94)*, 1994.
5. P. Laborie and D. Godard. Self-Adapting Large Neighborhood Search: Application to Single-Mode Scheduling Problems. In *MISTA-07. Proc. of the Multidisciplinary International Scheduling Conference: Theory & Applications*, 2007.
6. U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences*, 7:95–132, 1974.
7. N. Muscettola, P. Nayak, B. Pell, and B.C. Williams. Remote Agents: To Boldly Go Where No AI Systems Has Gone Before. *Artificial Intelligence*, 103(1-2):5–48, 1998.
8. A. Oddi, R. Rasconi, A. Cesta, and S.F. Smith. Iterative-Sampling Search for Job Shop Scheduling with Setup Times. In *COPLAS-09. Proc. of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems at ICAPS*, 2009.
9. A. Oddi and S.F. Smith. Stochastic Procedures for Generating Feasible Schedules. In *Proceedings 14th National Conference on AI (AAAI-97)*, pages 308–314, 1997.
10. Angelo Oddi, Amedeo Cesta, Nicola Policella, and Stephen F. Smith. Combining Variants of Iterative Flattening Search. *Journal of Engineering Applications of Artificial Intelligence*, 21:683–690, 2008.
11. S.F. Smith and C. Cheng. Slack-Based Heuristics for Constraint Satisfaction Scheduling. In *Proceedings 11th National Conference on AI (AAAI-93)*, 1993.
12. Pascal Van Hentenryck. *The OPL Optimization Programming Language*. MIT Press, 1999.