Case-Based Reasoning and Tabu Search for the Single Machine Scheduling Problem: first Results

Stephan Grolimund, Jean-Gabriel Ganascia

Université Pierre et Marie Curie - Paris VI, LAFORIA - IBP - CNRS
4, place Jussieu, BP 169, F-75252 PARIS Cedex 5, France

E-mail: {grolimund, ganascia}@laforia.ibp.fr

Abstract

Combinatorial optimisation differs from general problem solving in the nature of goals to be achieved. The difficulty of achieving optimisation goals leads to a general lack of confidence concerning the knowledge that can be learned during optimisation. We point out that this makes analytical learning techniques, like EBL or derivational analogy, inappropriate to the resolution of optimisation problems. Operations research methods cope with combinatorial optimisation, however without using symbolic learning.

Integrated in the Tabu search framework, we propose a system that learns control knowledge in optimisation domains while the resolution is in progress. A case-based learning technique is used to store and reuse control knowledge within the same instance of an optimisation problem. The approach, called ALOIS, does not attempt to transfer the acquired case library to any other instance of the same problem. ALOIS is designed to be domain-independent, using a general rule-based reasoner to explore the solution space. Its application to the single machine scheduling problem with variable set-up times is illustrated and discussed.

Résumé

L'optimisation combinatoire se distingue de la résolution de problèmes de par la nature des buts qu'elle poursuit. La difficulté qu'il y a à résoudre les problèmes d'optimisation combinatoire engendre une méfiance à l'égard des connaissances qui peuvent être apprises durant l'optimisation. De ce point de vue, les techniques d'apprentissage analytique classiques ne sont guère adaptées à la résolution de problèmes d'optimisation, alors qu'en revanche, les méthodes issues de la recherche opérationnelle s'y prêtent. Ces dernières n'intègrent cependant pas de techniques symboliques d'apprentissage.

Nous proposons une approche, intégrée dans la méthode Tabou, permettant l'apprentissage de connaissances de contrôle dans des problèmes d'optimisation en cours de résolution. Une technique d'apprentissage à partir de cas acquiert et réutilise des connaissances de contrôle à l'intérieur de la même instance de problèmes. L'approche, appelée ALOIS, ne transfert pas de connaissance d'une optimisation à l'autre. En outre, elle est indépendante du domaine grâce à l'utilisation d'un système à base de règles pour l'exploration de l'espace de recherche. Nous illustrons et commentons l'application d'ALOIS à un problème d'ordonnancement de base.

Case-Based Reasoning and Tabu Search for the Single Machine Scheduling Problem: first Results

Stephan Grolimund **Jean-Gabriel Ganascia**

Université Pierre et Marie Curie - Paris VI
Laforia - IBP
4, place Jussieu, BP 169
F-75252 PARIS Cedex 5, France

E-mail: {grolimund, ganascia}@laforia.ibp.fr

Tel: ++ 33 1 44 27 71 19

Fax: ++ 33 1 44 27 70 00

Abstract

Combinatorial optimisation differs from general problem solving in the nature of goals to be achieved. The difficulty of achieving optimisation goals leads to a general lack of confidence concerning the knowledge that can be learned during optimisation. We point out that this makes analytical learning techniques, like EBL or derivational analogy, inappropriate to the resolution of optimisation problems. Operations research methods cope with combinatorial optimisation, however without using symbolic learning. Integrated in the Tabu search framework, we propose a system that learns control knowledge in optimisation domains while the resolution is in progress. A case-based learning technique is used to store and reuse control knowledge within the same instance of an optimisation problem. The approach, called ALOIS, does not attempt to transfer the acquired case library to any other instance of the same problem. ALOIS is designed to be domain-independent, using a general rule-based reasoner to explore the solution space. Its application to the single machine scheduling problem with variable set-up times is illustrated and discussed.

Keywords: case-based learning, control learning, learning and optimisation, combinatorial optimisation, scheduling, tabu search

1. Introduction

Analytical learning allows problem solvers to adapt their general resolution mechanism to the treated problem in order to improve performance. Common to such methods (e.g. [Minton, 1989], [Veloso, 1992], [Laird et al, 1986]) is their focus on problem solving or plan generation problems. However, optimisation problems do only partially fit into this framework. There, the overall goal is not (only) to the construction of plans, or operator sequences, that lead to solutions, but to find those solution states which optimise a numeric quality measure, the objective function, defined over the whole solution state space.

A simple example of such a combinatorial optimisation problem is the *single machine scheduling problem* (Fig. 1), where a given set of n jobs has to be scheduled on one machine [Barnes & Laguna, 1993]. Typically, the jobs have different production times, and the set-up times between them vary according to the particular sequence of jobs. As numeric optimisation criteria, minimal total production time (makespan) or minimal total waiting time are often used. The problem is NP-hard. A solution state is a state where all jobs are scheduled on the machine, and the optimisation task consists in finding the solution state for which the objective function is minimised (or maximised).

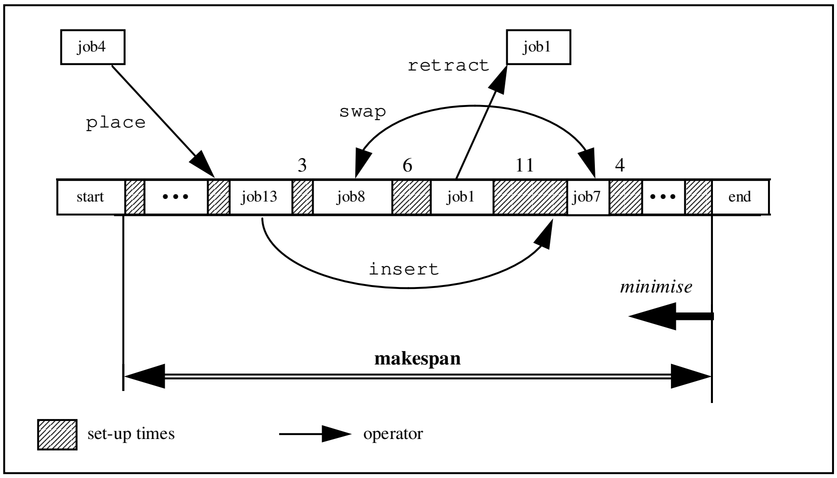


Figure 1: The single machine scheduling problem with variable set-up times

Combinatorial optimisation problems are well-known in operations research [Papadimitriou & Steiglitz, 1982]. One important class of heuristics dealing with them, *local search* (or hill-climbing), explores the search space by performing iteratively improvement transformations on solution states in order to explore the search space as complete as possible. Three variations of this scheme have shown to be extraordinarily successful [Reeves, 1993]: genetic algorithms, simulated annealing and tabu search [Glover, 1989, 1990]. Tabu search explicitly incorporates a transfer of knowledge, acquired from past search episodes, to provide control for future search steps, which is in its function similar to control knowledge in problem solving. However, knowledge is acquired base on a statistical analysis of the search space exploration. As far as we know, no symbolic learning techniques has been studied in combination with tabu search.

Our aim is to learn during optimisation inside a tabu search method. We claim that for an optimisation problem, whose resolution is primary based on an exhaustive search through a solution state space [Johnson et al, 1988], it is beneficial to analyse the search while it is in progress, to extract experiences on the underlying search space, and to reuse them, as the same search goes on, in order to find solutions of higher quality. For this purpose, we introduce ALOIS, a domain independent analytical learner, that acquires control knowledge, using a case-based learning technique, while a tabu-search-like optimisation is under progress. Applying case-based reasoning to optimisation problems is not new. EASe [Ruby & Kibler, 1992], demonstrated in the domain of logic synthesis, learns on easier problem instances[1] how to improve them, and reuses the gained cases while working on more complicate ones. The CBR system CABINS [Sycara & Miyashita, 1994], designed for the repair problem in job shop scheduling, reuses the cases, seen as an extensional definition of the optimisation criteria, for the selection of repair actions. The case library may be acquired directly from the user. Both approaches differ from ALOIS in that they a) do not address optimisation from scratch, but refine solutions previously constructed by other, renowned techniques, b) separate the case acquisition phase from the case reuse phase, and c) in ALOIS, the initially empty case library is acquired and reused during the optimisation of *one and the same* problem instance. Further, our approach does not make any assumptions concerning the treated problem. We claim that it is domain-independent in the same sense as is tabu search.

The paper is structured as follows: Section 2 discusses the relation between different analytical learning techniques and optimisation problems, Section 3 introduces our case based learning approach, while Section 4 shows the results of ALOIS on the single machine scheduling problem, together with a first discussion of the approach.

---

[1] We distinguish the *problem* from the *problem instance*, a usual separation in operations research. The problem description formalises the high level structures of the treated problem, while the problem instance defines the particular entities that compose a concrete problem to be solved. In our example, the problem is "single machine scheduling with variable set-up times", and problem instances are defined by let's say 7 jobs that have to be scheduled on machine IBP753, knowing all relevant characteristics of both the machine and the jobs.

2. Analytical Learning and Optimisation Problems

The main difference between the problems "construct a plan to a solution state" and "find the optimal solution state" is the absence of an explicit objective function and reflects the classical difference between decision problems[2] and optimisation problems [Garey & Johnson, 1979]. While for the former *any* solution state is acceptable, the latter asks for a very particular one. Plan generation typically is a decision problem. Nevertheless it could be formulated as optimisation problem, stating the task such as "find the plan with minimal length". However, finding such an optimal plan is usually beyond scope for planning systems [Fox, 1994], with the exception of heuristic search [Nilsson, 1980], [Krovvidy & Wee, 1993].

EBL-like analytical learning typically applies to state-based plan generation, where, given a domain theory, an initial state and some goals, the reasoner has to produce an operator sequence leading to a final state satisfying the goals. Learning thus aims at improving planning performance. Due to the absence of an explicit objective function on solution states, analytical learning techniques do not integrate such quality measures and so only marginally apply to optimisation problems. In addition, as Woodruff (1994) points out, achieving any solution state is not the problem faced in optimisation, and learning how to achieve them is not necessary facilitating the task.

However, some analytical learners implicitly treat optimisation problems. The phenomena called *masking effect*, first described in the context of Prolog/EBG [Clark & Holte, 1992] can be seen as a particular optimisation problem. This because the solution states are evaluated by a quality measure. As it is external to both the problem solver and the learner, learned knowledge fails to capture it produces solution that degrade with regard to the quality measure. Other work on the masking effect [Tambe and Rosenbloom, 1993] points out that operators, which are well suited for planning problems, are overgeneralised for optimisation problems. However, the type of overgeneralisation is not addressed by domain revision techniques. None of this work addresses the masking effect explicitly as optimisation problem: in Prolog/EBG, the problem went away after reordering the clauses, while Tambe and Rosenbloom (1993) introduce and compare different heuristics on how to specialise their operator schemata.

---

[2] Decision problems may be formulated like "Is there a solution for this problem instance?" and require a "Yes/No" answer.

The presence (or absence) of an explicit objective function leads to a radically different nature of goal satisfaction: in problem solving, goal satisfaction is a property of each state on its own, depending on whether the state contains the facts necessary to satisfy the goal description. For optimisation problems goal satisfaction is twofold [e.g. Ruby & Kibler, 1992]. First, a feasible solution has to be constructed, and second between all possible feasible solutions, the one that minimises the objective function has to be identified. The first task straightaway fits into problem solving. Moreover, identifying a solution state is often quite easy (see Section 4). The second task however does not. This because its satisfaction does not depend on one single state, but on the whole, extensionally defined, set of states satisfying the first goal. Furthermore, optimisation problems typically are NP-hard, suggesting strong evidence that complete exploration of this solution space becomes intractable for sufficient large problem instances. The optimality of a solution state cannot be proofed in a polynomial bounded amount of time, and as consequence, the second task can in general not be accomplished.

Based on these observations, there are mainly two reasons against applying classical analytical learning to optimisation problems:

a) In PRODIGY/EBL [Minton, 1989] it is possible to characterise the learned knowledge using logic. Given a proof tree for a target concept and a domain theory, the generalisation phase in EBL constructs the weakest preconditions under which the proof holds. Consequently, in any state satisfying these weakest preconditions the target concept may be derived, as long as the domain theory remains unchanged. The logic framework thus allows to construct formal valid knowledge.

   Goal satisfaction in optimisation problems, depending on a whole *set* of states, however lies outside this logic framework, where a subset of *one* state's descriptors is sufficient to satisfy the goal description.

b) The resolution of the optimisation problem's second task typically would produce recursive proofs, as the same operators intervene during the extensive construction of the solution states (see below). However, negative results concerning worst case performance of EBL-like techniques in domains with recursive proofs [Etzioni, 1993] suggest that performance would degrade after learning, leading to the well-known utility problem [Minton, 1990].

3. Case-based learning inside tabu search

Local search [e.g. Johnson et al., 1988] is an established class of heuristic optimisation techniques based on an iterative improvement of the quality of solutions found during the exploration of a solution state space. Widely applied to combinatorial optimisation problems, these techniques aim to produce good solutions (optimal or almost optimal ones) with regard to the given objective function. Due to the complexity of the treated problems, there is in general no criteria known allowing the identification of a particular solution state as being the optimal one. Tabu search [Glover, 1989] [Glover, 1990] adds a three-level control structure to the general local search framework. In a nutshell, the base level of control, one or several tabu lists, aims to cope with local minima, in which local search gets trapped regularly. The two upper levels, intensification and diversification, build the so-called mid-term and long-term memory of the search process. Intensification aims to focus the search on promising areas of the search space, while diversification tries to direct the search to yet unexplored regions. The interpretations of "promising" and "unexplored" are problem dependent and only intuitively defined. See [Glover et al, 1993] for a collection of for major examples of actual tabu search research.

It is this intensification process we are interested in. Intensification qualitatively judges the evolution of past search episodes. This qualification however has to remain approximate (in difference to EBL, see Section 2), as the really desired reference, the *optimal state*, is unknown during optimisation. In this context, where experience only has a weak basement, the use of a (formally) weak case-base reasoning technique seems to be justified, leading us to the introduction of ALOIS ("Analogical Learning for Optimisation to Improve operator Selection"). The local search technique used in ALOIS summarised in Table 1.

ALOIS has two principal components: a) as basic search mechanism, general forward chaining rule-based reasoning is used, allowing a declarative formulation of the employed operator schemata and underlining the domain independence of the approach, and b) a case-based learning technique captures the experience gained during search and reuses it, on the same problem instance, to improve operator selection during conflict resolution. A straightforward case-based learning technique is used: a case resumes the firing of an operator in terms of both the qualitative, a posterior generated, reward attributed to this operator firing together with a concise description of the state the operator has been applied in (Fig.2). For an applicable operator, case retrieval furnishes a set of relevant cases, among which the "most" similar one is selected, using approximate partial matching. The reward contained in the case is used to modify the operators priority to get selected for application.

Given:

- a **problem description** composed of: the *domain theory* (a set of operator schemata), an *evaluation function p(·)* for each operator schema and a numeric *objective function f(·)* to optimise,
- a **problem instance description** with an *initial solution*, and
- a basic **tabu search** control schema (which additionally may be revised by the user).

Do:

- iteratively **explore the search space** by applying the operator with the highest evaluation,
- modify the operator's evaluation function base on past search **experience** manipulated by case-based reasoning.

The goal is to construct *near optimal solutions*.

Table 1: Local search in ALOIS

Cases is stored in a case library using a two-level indexing scheme, branching first on the operator's schema and second on the state's description. When an operator of the same schema becomes applicable, case retrieval, guided by the actual state's description, is performed in order to find cases relevant to this operator.

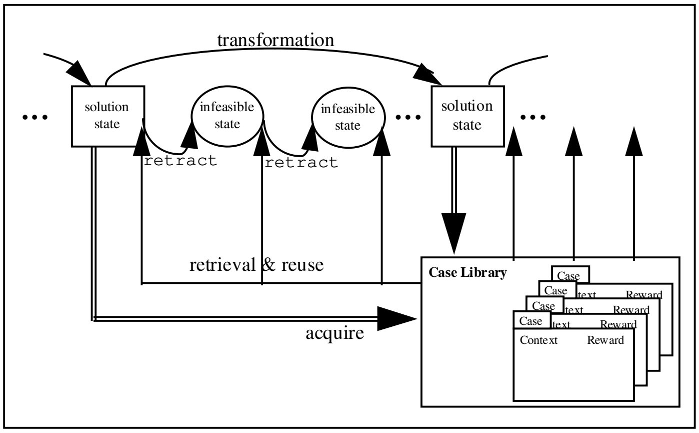

transformation
...
solution state
infeasible state
infeasible state
...
solution state
...
retract
retract
retrieval & reuse
acquire
Case Library
Case
Case
Case
Case
text
Reward
text
Reward
text
Reward
Context
Reward

Figure 2: search space exploration in the scheduling domain

3.1. The rule-based reasoning component

The search space contains two types of states: *solution states* (or solutions) which represent feasible solutions of the treated problem, and *infeasible states* which design situations that do not correspond to acceptable solutions of the problem. As example, in the single machine scheduling domain, a situation where all jobs are scheduled is a solution, while a situation where at least one job is not scheduled corresponds to an infeasible state.

The search space is explored by iteratively applying *transformations* to solution states. A transformation is an operator sequence leading from one solution state to another solution state by passing only through infeasible states (Fig. 2). In local search, each operator typically leads directly from one solution state to the next one, without passing through infeasible states. They strongly resemble STRIPS-style macro-operators [Fikes et al, 1972], and are suffering similar problems concerning their high matching costs. For constructing transformations, we decompose these operators in order to obtain simpler ones that are less expensive in their application. As example, the `insert` operator schema introduced in Section 4 (and Fig. 1), which moves a job A just before a job B, is decomposed into two simpler operator schemata, `retract` and `place`, where `retract` retracts job A from the schedule, leaving it in an infeasible state, and `place` inserts job A back to the schedule, at the position just before job B, producing a solution state again.

At first glance, transformations strongly resemble to plans in problem solving. However, the employed, decomposed operator schemata lead to an entire absence of failure paths or subgoal interactions during transformation generation due to their decomposition. Therefore, there is no need to acquire control knowledge for improving the transformation construction process. This decomposition has the consequence that the order in which operator schemata have to be applied is known in advance [Grolimund & Ganascia, 1994]. To continue the example, to know that `insert` has been decomposed into `retract` and `place` explicitly guides the reasoner, after having applied a `retract` operator, to apply a `place` operator in order to put the retracted job back to the schedule. Insuring that there are only success paths, no expensive planning effort is necessary to construct transformations.

Similar to local search, each operator schema has a domain-specific prevision function associated. This prevision estimates, for each schema's instance, the objective function value that would be attended at the end of the transformation.

During conflict resolution, operators are ordered and selected according to these previsions. As in tabu search, domain-dependent tabu lists may additionally inhibit operator application.

3.2 The CBL component

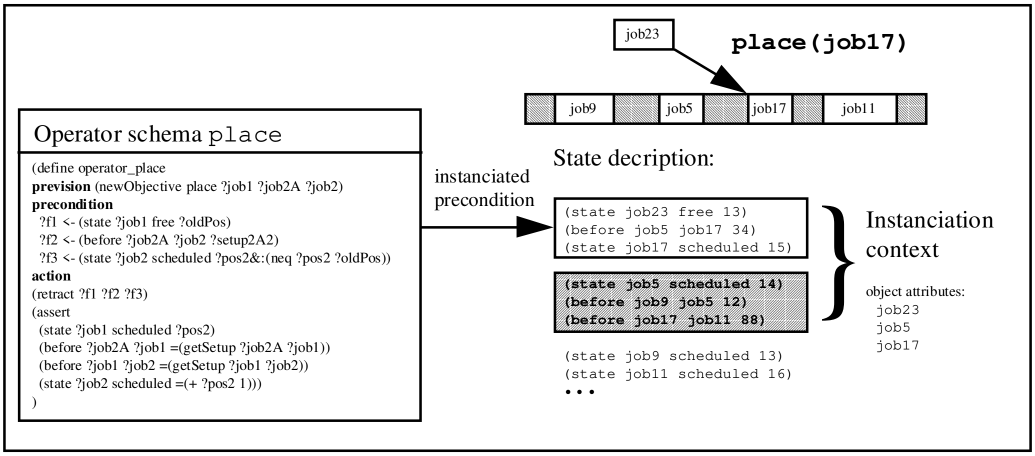
Combinatorial optimisation problems have in general both, symbolic and numeric characteristics. For example in the single machine scheduling domain, placing a job A at a position P may typically be seen as symbolic, whereas the set-up cost between two succeeding jobs A and B is numeric. The state description language, as base for case generation, retrieval and similarity assessment, has to take this into account. An intuitively appealing predicate typing technique is used in KBG [Bisson, 1992] in order to cope with numeric features in the context of inductive learning in first order logic. Extending the state representation language one used in PRODIGY/ANALOGY [Veloso, 1992] by the introduction numeric attributes, we adopt it as description language for case representation in ALOIS.

In Bisson's (1992) representation, the facts of the state description are treated as being literals of the form `(predicateName [attribute]*)`, starting with a predicate name, followed by a variable number of attributes. Attributes have to belong to either of the two classes: objects or values, and have to be typed. We consider the objective attributes to represent the entities of the domain, while value attributes characterise both, these entities and the relations between them. For example, in the machine scheduling domain, we consider the jobs to be object attributes of type `Job`, and the set-up times to be value attributes of type `Integer`. A typical literal in this domain is

```
(before job_A job_B 27)
```

stating that job A is scheduled directly before job B, causing a set-up time of 27 time units.

A case has to capture the state's relevant features. The goal-driven footprinting approach [Veloso, 1992] is however not appropriate in the considered optimisation domains. Based on the above outlined state representation language, we define the *instanciation context*, a subset of the descriptors of the state in which the operator has been applied, as the set of literals in which occur at least one of the object attributes contained in the operator's precondition. Intuitively, the instanciation context contains the complete available description of the domain's entities which are implicated by the applied operator.



```
(define operator_place
prevision (newObjective place ?job1 ?job2A ?job2)
precondition
  ?f1 <- (state ?job1 free ?oldPos)
  ?f2 <- (before ?job2A ?job2 ?setup2A2)
  ?f3 <- (state ?job2 scheduled ?pos2&:(neq ?pos2 ?oldPos))
action
(retract ?f1 ?f2 ?f3)
(assert
  (state ?job1 scheduled ?pos2)
  (before ?job2A ?job1 =(getSetup ?job2A ?job1))
  (before ?job1 ?job2 =(getSetup ?job1 ?job2))
  (state ?job2 scheduled =(+ ?pos2 1)))
)
```

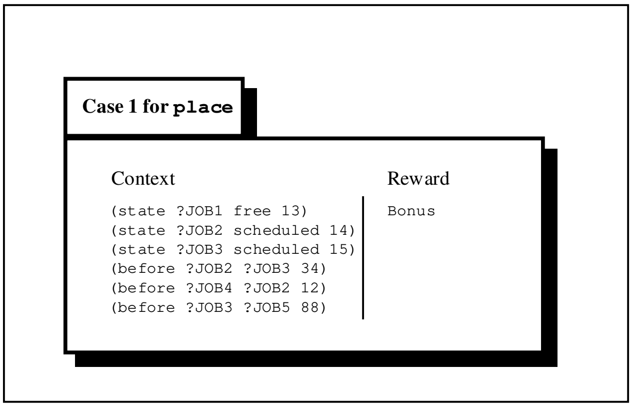
```
(state job23 free 13)
(before job5 job17 34)
(state job17 scheduled 15)

(state job5 scheduled 14)
(before job9 job5 12)
(before job17 job11 88)

(state job9 scheduled 13)
(state job11 scheduled 16)
...
```

Figure 3: instanciation context for an application of `place`

We illustrates the construction of the instanciation context using the machine scheduling operator schema `place` as an example (Fig. 3). On the left, the (slightly simplified) definition of `place` is given, showing its prevision function as well as the precondition and the action. The latter is composed of an add and a delete list. The right side shows `place(job17)`, an instanciation of `place` that puts `job23` between `job5` and `job17`, together with a possible state description. The instanciated preconditions contain the entities `job23`, `job5` and `job17`. All the facts, in which these object attributes occur and which lay outside the instanciated precondition, are grey underlined and constitute, together with the instanciated preconditions, the instanciation context. For performance purposes, the instanciation context is parametrised, replacing object attributes by variables (Fig. 4).



```
(state ?JOB1 free 13)
(state ?JOB2 scheduled 14)
(state ?JOB3 scheduled 15)
(before ?JOB2 ?JOB3 34)
(before ?JOB4 ?JOB2 12)
(before ?JOB3 ?JOB5 88)
```

Figure 4: a case from the machine scheduling domain

Similarity assessment between a source case and a target operator directly depends on the mappings between the source's and the target's attributes. As in first-order languages this matching problem is NP-hard [Bunke & Messmer, 1993], ALOIS relies on a greedy iterative improvement heuristic which produces good-quality mappings without insuring optimality. The heuristic proceeds by creating one mapping after the other, selecting each time the one that maximally improves similarity evaluation. Similarity evaluation of mappings between object attributes is binary, i.e. 1 for a valid mapping and 0 else, while for value attributes, similarity is the ratio, in [0..1], of the attribute values.

These attributes mapping similarities are aggregated over literals using the linear Hamming distance. Overall similarity is then calculated based on Jaccard's similarity measure [Grolimund & Ganascia, 1994]:

$$sim(S,T) = \frac{c}{p+q-c}$$

with: p (q) the number of literals in the instanciation context of the source (target), and c the aggregated similarity evaluation of the complete mapping.

Reward can be attributed once a transformation is accomplished as only in this moment the quality of the transformation's final state is known. Intuitively, an operator's *reward* should depend on the quality of the solution that has been constructed by the transformation the operator was involved in. The reward depends however on the difference between this solution's quality and the optimal solution's quality, which is unknown. We opt therefore for a qualitative, heuristic reward attribution, assigning a fixed, positive amount, or *bonus*, to an operator that contributed in a transformation which actually improved solution quality. Otherwise a fixed, negative reward, or *malus*, is assigned.

4. Results and discussion

ALOIS has been applied to the `n`-jobs, single machine scheduling problem with variable set-up times and minimisation of the makespan, i.e. the total duration of the schedule. This is a very simple combinatorial optimisation problem which, nevertheless, is NP-hard. Tabu search has shown to produce excellent results on such problems [Barnes & Laguna, 1993]. For this reason, we compare the results of ALOIS with those obtained by our re-implementation of this successful tabu search approach[3].

Two different problem models are evaluated. Problem modelling differs between the two approaches by the operator schemata employed. In the first approach A, the schema `insert` (Fig. 1). `Insert(job_A, job_B)` moves a job A to the position just before job B. There are O(n^2) different operators applicable to each schedule, with `n` the number of jobs to be scheduled. `Insert` leads directly from one solution state to the next one.

---

[3] Barnes & Laguna (1993) use a slightly different objective function in the work reported. They further notify that, due to the complexity of the problem, exact optimisation techniques are available for problem instance sizes up to 15 to 20 jobs.

In the second approach B, operators pass by infeasible states during a transformation [Grolimund & Ganascia, 1994]. Consequently, the employed operators can be less costly (in terms of number of applicable operators per state). Decomposing the `insert` schema, we obtain the schemata `retract` and `place`, where `retract(job_A)` retracts a job A from the schedule, and `place(job_B)` puts the retracted job back into the schedule at the position just before job B. For any state, O(`n`) operators are applicable, either instanciations of `retract` or of `place`. Reversing the operator decomposition, a transformation is always composed of a `retract` operator followed by a `place`. One can remark that for the selection of a `retract` operator a prevision of the next feasible state's quality has to be used, leading possibly to an imprecision with which the "best" applicable operator is selected.

Both approaches are implemented using a forward chaining inference engine with a RETE-like pattern matcher. This allows a declarative formulation of the operator schemata at the cost of introducing an execution slow-down due to expensive pattern matching. For different problem sizes, in the range of 15 to 60 jobs, 100 randomly generated problem instances have been optimised using both approaches and allowing each time the same number of iterations. On each problem instance, we apply first tabu search alone, and, in a second run, add ALOIS.

Experimental results are promising. Concerning solution quality, approach A produces generally only about 6% better schedules than B on the compared problem instances, due to the disadvantage of using a prediction the `retract` operators compared with the exact solution quality determination used in approach A.

Concerning the execution times of the two approaches, B however performs a considerably better scale-up than A (Fig. 5). This is essentially due to the complexity of the different operator schemata employed. The simpler operator schemata considerably reduce (pattern-) matching costs, at the price of a slight detriment of solution quality.

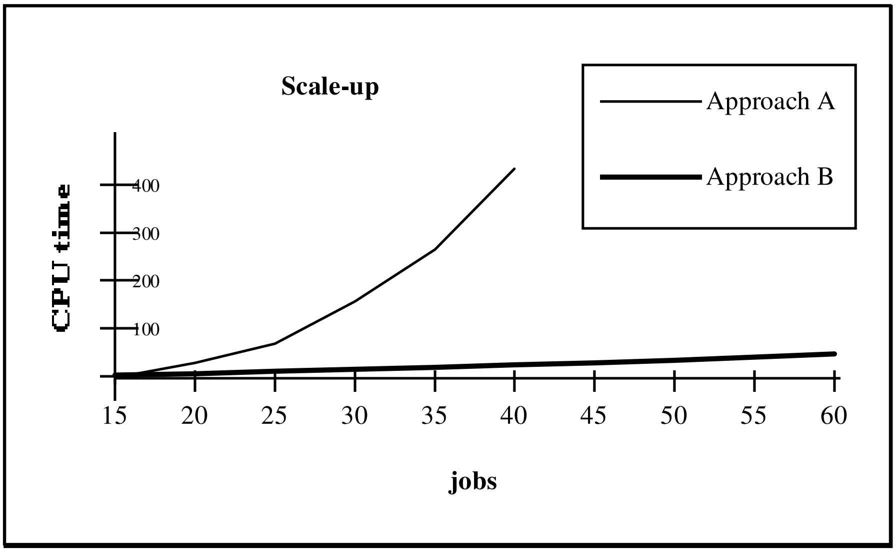


Figure 5: Execution times for a fixed number of iterations

The addition of the ALOIS component to the basic tabu search procedure does, however, not improve solution quality. To explain this, consider the instanciation context in Figure 3. It describes the entities involved in the application of a place operator. However, all place operators have basically this same instanciation context, only the value attributes are varying. This leads to an extreme uniformity of the generated cases, making *all of them* looking like the one presented in Figure 4. This obstacle is particularly due to the simple formulation of this particular scheduling problem. In consequence, cases are not very original and do not contain much information for a particular operator firing. For problems with a more complex structure, this inconvenience will go away.

Conclusion

In this paper we address control learning in optimisation domains, a subject that is only marginally covered by actual analytical learning techniques. We point out that, albeit well-suited for explanation-based learning, the logic framework does not allow to construct solid knowledge for the resolution of optimisation problems. As an alternative, we use a weak case-based learning technique to store and reuse experiences encountered while optimisation based on a tabu search method. The cases form control knowledge concerning operator selection and are only reused during the optimisation of the problem instance they have been created from.

This approach fully is implemented in the ALOIS system. The use of a declarative operator representation language allows accurate problem modelling, leading ALOIS to a scale up conveniently. This lets us presume that the method is adapted to the resolution of more complex optimisation problems to which classical local search techniques do only troublesome apply. Future work will evaluate this promising approach on facility location/allocation problems [Mirchandani & Francis, 1990], benefiting from the approach's domain independence.

References

Barnes J., Laguna M., A tabu search experience in production scheduling, in: Glover F., Laguna M., Taillard E., de Werra D. (eds.), Tabu Search, *Annals of Operations Research* 42 (1-4), Baltzer Science Publishers, Basel, Switzerland, 1993, p. 141-56

Bisson G., Learning in FOL with a Similarity Measure, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1992, p. 82-7

Bunke H., Messmer B., Similarity Measures for Structured Representations, *European Workshop on Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, Springer Verlag, 1993, p. 106-18

Clark P., Holte R., Lazy Partial Evaluation: An Integration of Explanation-Based Generalization and Partial Evaluation, *Proceedings of the International Conference on Machine Learning*, Aberdeen, Scotland, 1992, p. 82-91

Etzioni O., A structural theory of explanation-based learning, *Artificial Intelligence* 60, 1993, p. 93-139

Fikes R., Hart P., Nilsson N., Learning and executing generalized robot plans, *Artificial Intelligence* 3 (4), 1972

Fox M., ISIS: A Retrospective, in: Zweben M., Fox M. (eds.), *Intelligent Scheduling*, Morgan Kaufmann Publishers, San Francisco, CA, 1994, p. 3-28

Garey M., Johnson D., *Computers and Intractability: A Guide to the Theory on NP-Completeness*, W. H. Freemann and Co., 1979

Glover F., Tabu Search - Part 1, *ORSA Journal on Computing 1* (3), 1989, p. 190-206

Glover F., Tabu Search - Part 2, *ORSA Journal on Computing 2* (1), 1990, p. 4-32

Glover F., Laguna M., Taillard E., de Werra D. (eds.), Tabu Search, *Annals of Operations Research 42* (1-4), Baltzer Science Publishers, Basel, Switzerland, 1993

Grolimund S., Ganascia J.G., *A Case-Based Reasoning approach to knowledge transfer in Tabu Search*, Technical Report 94/06, Université Paris VI, Laforia, Paris, France, 1994

Johnson D., Papadimitrou C., Yannakakis M., How easy is Local Search?, *Journal of Computers and Systems Sciences* 37, 1988, p. 79-100

Krovvidy S., Wee W., Wastewater Treatment Systems from Case-Based Reasoning, *Machine Learning* 10, 1993, p. 341-63

Laird J., Rosenbloom P., Newell A., Chunking in SOAR: The Anatomy of a General Learning Mechanism, *Machine Learning* 1, 1986, p. 11-46

Minton S., *Learning Search Control Knowledge: An Explanation-Based Approach*, Kluwer Academic Press, Boston, 1989

Minton S., Quantitative Results Concerning the Utility of EBL, *Artificial Intelligence* 42, 1990, p. 363-92

Mirchandani P., Francis R. (eds.), *Discrete Location Theory*, John Wiley & Sons, New York, 1990

Nilsson N., *Principles of Artificial Intelligence*, Tioga Pub. Co., 1980

Papadimitriou C., Steiglitz K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, New Jersey, Englewood Cliffs, 1982

Reeves C., *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Pub., London, 1993

Ruby D., Kibler D., Learning Episodes for Optimization, *Proceedings of the International Conference on Machine Learning*, Aberdeen, Scotland, 1992, p. 279-84

Sycara K., Miyashita K., Case-based acquisition of User Preferences for Solution Improvement in Ill-Structured Domains, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1994, p. 44-9

Tambe M., Rosenbloom P., On the Masking Effect, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1993, p. 526-33

Veloso M., *Learning by analogical reasoning in general problem solving*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, 1992

Woodruff D., *Proposal for Chunking and Tabu Search*, Technical Report, University of California Davis, Dec. 1994