# Integrating Case Based Reasoning and Tabu Search for Solving Optimisation Problems

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

Tabu search is an established heuristic optimisation technique for problems where exact algorithms are not available. It belongs to the same family as simulated annealing or genetic algorithms. It extends the basic iterative improvement scheme by adding control learning. A technique of this kind, intensification, captures experience established on a frequency-based analysis of past search. Experience is reused while the same optimisation process is going on in order to guide search to better solutions.

In this paper, we introduce a case-based reasoning approach for control learning in tabu search. Search experience concerns operator selection and is represented by cases. The aim of case reuse is to improve conflict resolution. While the proposed method is domain independent, we present its application to the NP-hard uncapacitated facility location problem. Experimental results show that adding our approach to a basic tabu search optimisation significantly improves solution quality on the evaluated benchmark problems. It reduces the gap to the optimal solution by a factor of nearly 2.

## Résumé

La Recherche Tabou est une heuristique établie permettant d'optimiser des problèmes pour lesquels des algorithmes exacts ne sont pas utilisables. Elle appartient à la même famille de techniques que le recuit simulé et les algorithmes génétiques. Cette méthode étend le cycle basic d'amélioration itérative en recourant à des techniques d'apprentissage de contrôle de recherche. Une parmi ces dernières, l'intensification, tente d'acquérir de l'expérience de contrôle à partir d'une analyse fréquentielle de la recherche antérieure. Cette expérience est ensuite réutilisée afin de mieux guider la recherche ultérieure sur la même instance de problèmes.

Dans cet article, nous introduisons une technique de raisonnement à partir de cas pour l'apprentissage de contrôle dans la méthode tabou. Les cas représentent l'expérience relative à la sélection des opérateurs. Ils sont réutilisés afin d'améliorer la résolution de conflits. La méthode étant indépendante du domaine, nous l'illustrons par son application au problème NP-dur de placement et d'allocation d'entités. Les résultats expérimentaux montrent que l'ajout de la méthode proposée, à une technique de recherche tabou simple, améliore la qualité des solutions identifiées pour des problèmes de références. L'écart entre les solutions produites et la solution optimale se trouve réduit peu ou prou d'un facteur 2.

This paper has been published in: *First International Conference on Case-based Reasoning*, 1995, in: Aamodt A., Veloso M. (eds.), *Case-Based Reasoning Research and Developpement*, Lecture Notes in Artificial Intelligence 1010, Springer Verlag, Berlin, 1995, p. 451-60

# Integrating Case Based Reasoning and Tabu Search for Solving Optimisation Problems

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**

Tabu search is an established heuristic optimisation technique for problems where exact algorithms are not available. It belongs to the same family as simulated annealing or genetic algorithms. It extends the basic iterative improvement scheme by adding control learning. A technique of this kind, intensification, captures experience established on a frequency-based analysis of past search. Experience is reused while the same optimisation process is going on in order to guide search to better solutions.

In this paper, we introduce a case-based reasoning approach for control learning in tabu search. Search experience concerns operator selection and is represented by cases. The aim of case reuse is to improve conflict resolution. While the proposed method is domain independent, we present its application to the NP-hard uncapacitated facility location problem. Experimental results show that adding our approach to a basic tabu search optimisation significantly improves solution quality on the evaluated benchmark problems. It reduces the gap to the optimal solution by a factor of nearly 2.

## 1. Introduction

There are important NP-hard optimisation problems which cannot be solved efficiently by exact optimisation from mathematical programming [15]. A local search iterative improvement technique, *tabu search* [7, 8] has regularly shown to be apt in producing high-quality solutions in such domains. Its success is based on the ability to learn about optimising a particular problem instance while optimisation is in progress, and to reuse this experience to control future search. This learning process, called intensification, is strongly domain dependent and essentially of statistical or frequency-based nature [23].

Case-based reasoning research recognised the importance of solving optimisation problems early, and various approaches have been investigated. For example EASe [20], demonstrated in the domain of logic synthesis, learns on simple problem instances how it can improve them, and reuses these acquired cases while optimising more complex ones. The system CABINS [21], designed for the repair problem in job shop scheduling, maintains the case library as an extensional definition of the optimisation criteria, providing the criteria for selecting repair actions.

This paper introduces a domain-independent case-based learning technique that is directly designed for experience learning at the intensification level inside tabu search. As function-replacing hybrid system, it combines the strength of the renowned, general optimisation technique with CBR's capabilities of capturing experience in domains where only weak knowledge is available. The paper is structured as follows: Section 2 presents the test domain that will serve as example, uncapacitated facility location, Section 3 outlines the tabu search concepts necessary for understanding our method, Section 4 introduces the learning technique, and Section 5 presents a preliminary experimental evaluation.

## 2. Facility Location Problems

Facility location problems [17] deal with the optimal placing of an unknown number of facilities (e.g. plants or warehouses), within a finite set of possible locations, in order to minimise the cost of delivering demand to a given set of clients (Fig. 1). More precisely, for $m$ possible facility locations, the binary variables $y_i$ design the opening of a facility at location $i$ by $y_i = 1$. For $n$ clients with demand $D_j$, the real variables $x_{ij}$ hold the flow from facility $i$ to client $j$. Each opened facility $i$ charges a fixed cost $F_i$. Further, unitary costs $c_{ij}$ are charged for each flow $x_{ij}$. They are typically related to the distance between facility and client. The optimisation goal, expressed by the objective function $f(\cdot)$, is to find the facility configuration that minimises the total cost. The most simple and most studied variant of facility location is the Uncapacitated Facility Location Problem (UFLP), where the facilities' capacities are unlimited. It has important practical applications in transportation and other industry sectors [17]. Using the given notation, its mathematical programming formulation is :

(UFLP) minimise: $f = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} F_i y_i$

subject to:

$$\sum_{i=1}^{m} x_{ij} = D_j, \qquad j = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij} \le Q_{\max} \cdot y_i, \qquad i = 1,\ldots,m; \qquad with: Q_{\max} \ge \sum_{j=1}^{n} D_j$$

$$x_{ij} \in \Re^{+}, \qquad i = 1,\ldots,m; j = 1,\ldots,n$$

$$y_i \in (0,1), \qquad i = 1,\ldots,m$$

UFLP is NP-hard. Even though, its mathematical properties provide a means to find global optimal solutions efficiently, even for large problem instances[1] [3]. As soon as additional constraints are introduced, exact optimisation techniques are not

[1] We distinguish *problem* from *problem instance*, a usual separation in operations research. The problem is "uncapacitated facility location", and a problem instance may be defined by giving let's say 21 clients and a set of 50 possible warehouse locations.

available anymore and heuristic optimisation like tabu search has to be used. UFLP is interesting for an experimental evaluation because of benchmark problem instances. Results of heuristic approaches can thus be compared to the global optimal solutions.

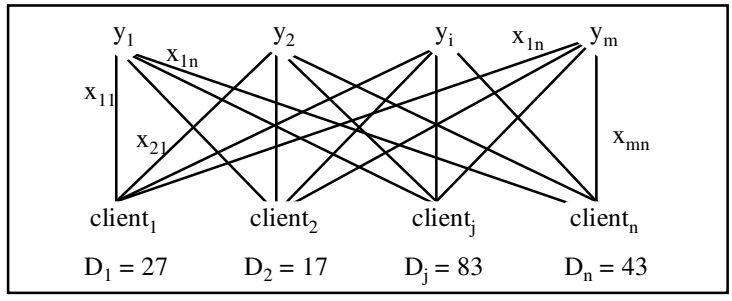


**Fig. 1.** General m × n facility location

## 3. Tabu Search

Local search (or hill-climbing) heuristics explore a search space by stepwise moving from one solution state *S* to another one, applying operators *o*, in order to improve the objective function *f(S)* iteratively. Three variants of this classic deepest descent search scheme produce high quality solutions (optimal or near optimal ones) in a variety of different domains [19]: simulated annealing, genetic algorithms and tabu search. *Tabu search* [7, 8] is a general, domain-independent meta-heuristic that tackles search control. Its application to facility location problems has produced high-quality solutions [6, 12].

Operator selection is guided by an operator evaluation function *p(o)* which reflects the change of *f(·)* before and after applying the operator, i.e. $p(o) = f(S) - f(o(S))$. Tabu search adds a three-level control structure to this general operator selection criteria. The base level of control consists of a *tabu list L* and copes with local minima where greedy local search often gets trapped in. During search, each visited solution *S* is put in *L*, which is managed by a first-in first-out strategy. As long as $S \in L$, operators that would lead back to *S* are declared *tabu* and, for this reason, are forbidden to apply. The idea is to force the search to move away from *S* and, in case *S* is a local minimum, to push it as far away for that it will not get attracted by *S* again. Tabu lists may either be of fixed size (length 7 seems to be well adapted to various problems) or, often a better alternative, of variable size [1].

Tabu search's two upper levels of control, *intensification* and *diversification*, build the so-called mid-term and long-term memories of the search process. They are meant to improve the future search, benefiting from learning by experience. This experience comes from search on the same problem instance. 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 not rigorously defined and are problem dependent. Frequency-based criteria are used often in this case [23]. For example in facility location, typical

intensification consists of fixing the number of opened facilities for some iterations, forcing the exploration of different similar configurations, while diversification may be achieved by compelling the opening (or closing) of a facility that did not change status for a given number of iterations [6].

## 4. Combining Tabu Search and Case-Based Reasoning

We claim that case-based reasoning is particularly well suited for control learning in tabu search, especially for intensification. First, intensification is a form of learning based on experience [7, 8]. Second, only shallow knowledge, resumed by "promising", is available to qualify experience. Consider any solution $S$. For $S$ being of "high quality", $f(S)$ has to be close to $f(S^*)$, $S^*$ the optimal solution. Of course, neither $S^*$ nor $f(S^*)$ are known during optimisation. In practice, "high quality" and "promising" may not refer to $S^*$, but only to the best solution(s) found during past search. Case-based reasoning is well adapted to problems where experience extensively represents domain knowledge and where only a weak domain theory is available [13].

Control learning for planning has been investigated in machine learning as well as in case-based reasoning [11, 16, 22]. Optimisation differs from planning in nature: First, planning focuses on *how* to achieve solutions, while in local search optimisation, where solutions are usually constructed easily, the task is to find *good* ones [10, 23]. As consequence, memorising operator chainings, i.e. receipts for constructing solutions, is not necessary the best choice. We abandon it completely, focusing learning on single operators [18]. Second, there is no evidence that *good* solutions for one problem instance share any characteristics with *good* solutions for another problem instance [23]. We thus do not transfer acquired experience from one problem instance to another one, a practice that is of major concern in plan learning but completely uncommon in tabu search.

The remaining part of this paper introduces ALOIS ("Analogical Learning for Optimisation to Improve operator Selection"). Its local search technique is detailed below. A straightforward case-based learning technique is integrated: cases in ALOIS capture experimental, approximate qualifications, called *reward*, of operator firing outcomes. The hypothesis for case reuse is that, in a similar state, firing the operator again will result in similar qualifications. Cases are relevant to the operator schema they have been created from, and are composed of a description of the state the operator was fired in, together with the reward attributed to this firing. Case retrieval occurs for each operator becoming applicable again. The state's description is used to identify the most similar case between the relevant available cases, and the retrieved case's reward modifies the operators firing priority.

### 4.1 The Optimisation Algorithm

An optimisation problem is defined by a `domain` theory, a problem instance `dataInstance` and a stop condition `stopCondition` (Table 1). As in tabu search, an optimisation starts for each problem instance with an empty case library in an initial

```
function Optimise(domain, dataInstance, stopCondition) : the best solution
1     initialise the engine with domain and dataInstance
2     Current := generate initial solution state
3     Best := Current
4     Caselibrary := empty
5     while stopCondition not satisfied
6           Next := Next_Solution(Current, Caselibrary)
7           O := the set of applied operators to achieve Next
8           if Next is better than Best  then Best := Next
9           for each operator o_i in O do
10                case_i := Generate_case(o_i, Current, Next, Best)
11                Insert(case_i, Caselibrary)
12          Current := Next
13    return Best
```

**Table 1.** Main loop of optimisation in ALOIS

```
function Next_Solution(Current, Caselibrary) : the feasible solution
1     O := empty
2     repeat
3           Agenda := set of non-tabu, applicable operators in Current
4           for each operator o_i in Agenda do
5                 p_0 := priority of applying p(o_i)
6                 p_1 := Retrieve_Reuse(Caselibrary, o_i, p_0)
7                 set p_1 as new priority of o_i
8           reorder Agenda
9           o := select the operator with highest priority in Agenda
10          append o to O
11          Current := o fired in Current
12    until Current is feasible
13    return Current
```

**Table 2.** Solution generation, reusing experience in ALOIS

feasible solution. New solutions are generated iteratively, as long as the stop condition is not satisfied. Typically stop conditions are combinations of: CPU time, iterations or number of iterations without improving the best solution. For each operator, applied to achieve the new solution `Next`, a case is inserted in the case library.

A feasible state is build by iteratively applying operators to the `Current` state (Table 2). The `Agenda` contains the set of applicable operators without those that fall under the tabu conditions imposed by the tabu list. For conflict resolution, the agenda is sorted according to the operator evaluation function *p(o)*, which may be modified during conflict resolution according to relevant experience retrieved from case memory. To guarantee that a feasible state is constructed without any special planning effort (like backtracking), a hand-coded control structure must be provided by the user [9].

UFLP is modelled using three operator schemata: `closeWarehouse` closes one (open) facility and adds it to the tabu list, `separateOneClient` severs one client from the facility that had just been closed, and `affectOneClient` allocates one

client with unsatisfied demand to a facility, opening it if necessary. The hand-coded control strategy imposes that, first, a facility is closed, second, all its clients are separated, and third, each of these clients is allocated to any facility except the one that just has been closed. As long as a facility is member of the tabu list, it is not possible to close it again, while opening it remains allowed. In the initial solution, each client is allocated to its closest facility.

### 4.2 Case Representation and Assessment

Cases are represented in a first order language to facilitate domain modelling. Similarity computation between cases requires therefore partial matching. Constructing the source-target mappings, a subgraph isomorphism problem, is NP-complete [4]. As consequence, worst case matching complexity grows exponentially with the number of descriptors. In case the number of descriptors is proportional to the problem instance size, which may be the case for UFLP as well as for single machine scheduling [10], source-target matching can become as complex as the optimisation problem itself. This is clearly undesirable. Case assessment has therefore to reduce the matching complexity.

For this means, one classic approach is predicate typing [14]. Combinatorial optimisation problems have in general both, symbolic and numeric characteristics. Predicate typing in KBG [5] copes with such numeric features in first order inductive learning. In this representation, literals have the form `(predicateName [attribute]*)`, starting with the predicate name and followed by a variable number of attributes. Attributes have to belong to one of the two classes: object or value, and have to be typed. We consider the object attributes to represent the entities of the domain, while value attributes characterise both, these entities and the relations between them. For example in UFLP, facilities are object attributes of type `Warehouse`, and costs are value attributes of type `Cost`. A typical literal in this domain might be `(clientArc warehouse19 client23 27 450)`, stating that warehouse19 delivers 27 units for a cost of 450 to client23.

However, predicate typing alone may not reduce matching complexity in case there is only one object type, as may happen in simple optimisation domains like single machine scheduling [10]. Another common technique for the complexity reduction of source-target matching, by reducing the length of the retained state description, is footprinting [22]. As in ALOIS a case focuses on only one single operator, footprinting only produces the operator's instanciated precondition. This is a somewhat restrictive description of the situation the operator was applied in. A case should ideally apprehend a description, as complete as possible, of the local configuration of its related solution state. For this task, we define the *instanciation context* as the set of literals, being present in the state the operator applied, describing the domain's entities which were concerned by the operator firing (Table 3).

**function** Instanciation_context(`operator`) : the instanciation context
1 `precond` := the set of facts that are instanciated preconditions of `operator` in `State`
2 `objects` := extract the object attributes of `precond`
3 **return** `I_context` := the subset of facts of `State` where `objects` occur

**Table 3.** Instanciation context construction

Fig. 2 gives the operator schema `closeWarehouse` from UFLP. The left shows the (simplified) definition of `closeWarehouse` with its evaluation function, the precondition and the action part. The right shows `closeWarehouse(warehouse17)`, an instanciation of `closeWarehouse` where `warehouse17` has the clients `client11`, `client13` and `clients45` allocated. Its instanciated precondition contains the entities `warehouse17`, `client11`, `client13`, and `clients45`. The middle box contains all facts where these object attributes occur and which lay outside the instanciated precondition. They compose, together with the instanciated preconditions, the instanciation context.

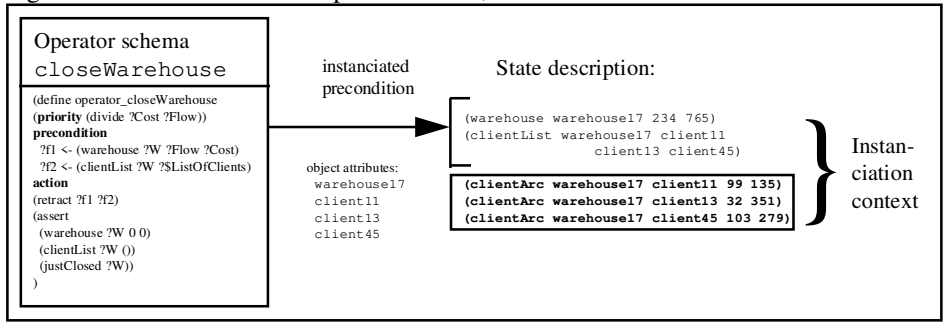


**Fig. 2.** Instanciation context for an application of `closeWarehouse`

The second component of a case, going with the instanciation context, is the reward attributed to the operator firing. An operator's reward depends on the quality of the solution that has been constructed by the transformation the operator was involved in. For this first evaluation of ALOIS, we opt for a simple qualitative and heuristic reward attribution, assigning a fixed, positive amount, or *bonus*, to an operator that contributed to an improvement of solution quality. Otherwise a fixed, negative reward, or *malus*, is assigned. It is constructed once a new feasible solution is obtained, as $f(\cdot)$ may be defined for solutions only. Table 4 summarises case generation.

**function** Generate_case(`operator`, `Current`, `Next`, `Best`) : the case
1 **if** `Current` is better than `Next` **then** `reward` := bonus
2 **else** `reward` = malus
3 `I_context` := Instanciation_context(`operator`)
4 **return** `case` := {`I_context`, `reward`}

**Table 4.** Case generation and reward attribution

### 4.3 Case Reuse

For this experimental evaluation, aimed at clarifying the general interest of the proposed technique, the case library is only rudimentary organised. Cases are added into the base structure, a sequential list, by simply appending them at the end (Table 1). No evaluation concerning the cases originality or pertinence is performed. Case retrieval is as simple as insertion, the target is compared, using partial matching, to each case in the library in order to find the closest one (Table 5).

Source-target matching is done on a literal by literal base, with a one-to-one match between attributes of the same type, using a greedy partial matching heuristic. Literals are mapped in the order they appear in the instanciated precondition, respectively in the order they were added to the instanciation context. Similarity evaluation uses hit counts for mapped entities, and ratios in [0…1] for mappings between the two numeric attributes. The final evaluation of a partial mapping is normalised into [0…1]. The detailed algorithm for matching and similarity evaluation is described in previous work [10].

Case reuse adapts an operator's priority according to the reward stored in the retrieved case. The bonus-malus reward attribution scheme leads to priority raising in case of `bonus` reward, and priority diminishing in case of `malus` (Table 5.11). Priority adaptation is weighted by the similarity value between the case and the operator.

```
function Retrieve_Reuse(Caselibrary, operator, oldPriority) : the new priority
1     I_context := Instanciation_context(operator)
2     BestCase := empty
3     BestSim := 0
4     for each case_i in Caselibrary do
5          sim := match I_context with case_i
6          if sim > BestSim then
7               BestSim := sim
8               BestCase := case_i
9     reward := the reward of BestCase
10    α := 1 + BestSim
11    if reward is bonus then p := α * oldPriority
12         else p := oldPriority / α
13    return p
```

**Table 5.** Case retrieval and priority adaptation

## 5. Experimental results

The goal of the experimentation is to verify whether or not adding ALOIS to a tabu search optimiser improves the quality of the best solution found[2]. UFLP is chosen, because benchmark problem instances with known optimal solutions are available from the OR-library [2], making it possible to verify how close the different heuristics approach the global optimum. A cross validation scheme, separating training and

[2] Due to the rudimentary case library design, we cannot yet appropriately evaluate the CPU time impact of our approach.

test, is not appropriate here, because similar to tabu search alone, each new optimisation restarts from scratch with an empty case library. In other words, ALOIS is not designed to learn from one optimisation on how to conduct the next one, but for improving an optimisation while it is going on. The system is implemented in C++ and experimentation is conducted on a Sun Sparc workstation using an industrial forward chaining inference engine with Rete-like pattern matching.

The UFLP domain is modelled as introduced in Section 4. Two different tabu list implementations are tested, one with a tabu list of fixed size 7 (T-fix), the other with a list of variable size in the range of 6…10 (T-var), with transition probabilities of 20% for increasing or decreasing the list size after each main iteration. ALOIS has been added to both of these basic tabu methods. Eight medium size problem instances from the OR-library have been optimised (minimised), each one having 50 clients to be affected. Four instances (cap71 - cap74) have 16 possible facility location, the other four (cap101 - cap104) have 25. Optimisation stopped after visiting 30 solutions without finding a new best solution. In practice, no run required more than 150 iterations (i.e. generated solutions).

Results are shown in Table 6. The best solutions produced by each of the 4 methods are given after normalising them such that, for the global optimal solution $S^*$ of each problem instance, $f(S^*) = 1$. The smaller the objective function of the best solution is, the better. The 'improvement' column gives the absolute solution improvement, i.e. the total cost reduction, due to ALOIS. These results show that integrating ALOIS improves solution quality on all but 2 problem instances. On problem instances where tabu search alone does not produce excellent results (e.g. on cap101), adding ALOIS is particularly beneficial. The average gap to the optimal solution is decreased, for T-fix, from 26% to 16%, and, for T-var, from 25% to 13%. Noteworthy, ALOIS improves solution quality on *both* tabu search designs in about the same extent, dividing the gap to optimum by nearly 2.

## 6. Conclusion

This paper introduces a case-based learning technique, ALOIS, to store and reuse operator selection experience encountered while a tabu search optimisation is in

| **problem instance** | **best solution** T-fix | T-fix + ALOIS | **improvement** f(S*) = 100% |
|---|---|---|---|
| cap71 | 1.08 | 1.07 | **0.4%** |
| cap72 | 1.10 | 1.06 | **4.0%** |
| cap73 | 1.10 | 1.06 | **3.7%** |
| cap74 | 1.14 | 1.15 | -0.7% |
| cap101 | 1.41 | 1.05 | **34.7%** |
| cap102 | 1.44 | 1.26 | **14.6%** |
| cap103 | 1.51 | 1.36 | **10.5%** |
| cap104 | 1.57 | 1.24 | **27.0%** |

| **problem instance** | **best solution** T-var | T-var + ALOIS | **improvement** f(S*) = 100% |
|---|---|---|---|
| cap71 | 1.05 | 1.04 | **1.1%** |
| cap72 | 1.05 | 1.06 | -0.9% |
| cap73 | 1.07 | 1.07 | **0.7%** |
| cap74 | 1.14 | 1.07 | **7.1%** |
| cap101 | 1.41 | 1.10 | **28.3%** |
| cap102 | 1.27 | 1.14 | **11.6%** |
| cap103 | 1.45 | 1.37 | **5.4%** |
| cap104 | 1.54 | 1.24 | **24.8%** |

**Table 6.** Solution quality improvement for UFLP due to ALOIS

progress. Evaluation of the approach is conducted on the uncapacitated facility location problem. It shows that adding ALOIS to the tabu search framework is promising, as solution quality in the test domain is significantly improved for two different tabu search techniques. Further evaluations on other problem domains as well as on other tabu search implementations need to be done to comfort the general interest of the approach.

In order to gain a deeper understanding of the way by which adding a case-based reasoning technique to tabu search improves performance, the actually rudimentary case base organisation has to evolve, introducing an analysis of new cases for novelty, pertinence, relevance and contradictions compared to the one in the case library. Feedback from case reuse should allow the identification of high value cases. We are confident that a sophisticated case library organisation will further improve performance of the approach. Actual work addresses these topics.

**Acknowledgement**

We wish to thank Alain David and Geber Ramalho for fruitful discussions on this work.

## References

[1] Battiti R., Tecchiolli G., The Reactive Tabu Search, *ORSA Journal on Computing* 6, 1994, p. 126-40

[2] Beasley J.E., OR-Library: Distributing Test Problems by Electronic Mail, *Journal of the Operations Research Society* 41, 1990, p. 1069-72

[3] Beasley J.E., Lagrangien heuristics for location problems, *European Journal of Operational Research* 65, 1993, p. 383-99

[4] Bunke H., Messmer B., Similarity Measures for Structured Representations, *First European Workshop on Case-Based Reasoning*, 1993

[5] Bisson G., Learning in FOL with a Similarity Measure, *AAAI*, 1992, p. 82-7

[6] Crainic T., Gendreau M., Soriano P., Toulouse M., A tabu search procedure for multicommodity location/allocation with balancing requirements, *Annals of Operations Research* 42 (1-4), 1993, p. 359-83

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

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

[9] Grolimund S., Ganascia J.G., *A Case-Based Reasoning Approach to Knowledge Transfer in Tabu Search*, Tech. Rep. 94/06, University Paris 6, Laforia - IBP, France, 1994

[10] Grolimund S, Ganascia J.G., *Case Based Reasoning and Tabu Search for the Single Machine Scheduling Problem: First Results*, Tech. Rep., University Paris 6, Laforia - IBP, 1995

[11] Hammond K., CHEF: A model of case-based planning, *AAAI*, 1986

[12] Hansen P., Pedrosa E., Ribeiro C., Location and sizing of offshore platforms for oil exploration, *European Journal of Operational Research* 58, 1992, p. 202-14

[13] Kolodner J., *Case-Based Reasoning*, Morgan Kaufmann, 1993

[14] Michalski R.S., A Theory and Methodology of Inductive Learning, in: *Machine Learning: An Artificial Intelligence Approach*, Tioga, Palo Alto, 1983

[15] Minoux M., *Mathematical Programming*, Morgan Kaufmann, 1989
[16] Minton S., *Learning Search Control Knowledge: An Explanation-Based Approach*, Kluwer Academic Press, Boston, 1989
[17] Mirchandani P., Francis R. (eds.), *Discrete Location Theory*, Wiley, 1990
[18] Mitchell T., Learning and problem solving, *IJCAI*, 1983, p. 1140-51
[19] Reeves C., *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell, 1993
[20] Ruby D., Kibler D., Learning Episodes for Optimization, *ICML*, 1992, p. 279-84
[21] Sycara K., Miyashita K., Case-based acquisition of User Preferences for Solution Improvement in Ill-Structured Domains, *AAAI*, 1994, p. 44-9
[22] Veloso M., *Learning by analogical reasoning in general problem solving*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, 1992
[23] Woodruff D., *Proposal for Chunking and Tabu Sea*rch, Tech. Rep., University of California Davis, Dec. 1994