Sociedad de Estadística e Investigación Operativa
Top (2004) Vol. 12, No. 1, pp. 169–191

Branch and Win: OR Tree Search Algorithms for Solving Combinatorial Optimisation Problems

Rafael Pastor and Albert Corominas
Department of Business Administration. Research Institute IOC
School of Industrial Engineering of Barcelona
Technology University of Catalonia
Avda. Diagonal, 647, 7ª planta
08028 Barcelona, Spain
E-mail: rafael.pastor@upc.es *albert.corominas@upc.es*

Abstract

Currently, most combinatorial optimisation problems have to be solved, if the optimum solution is sought, using general techniques to explore the space of feasible solutions and, more specifically, through exploratory enumerative procedures in trees and search graphs. We propose Branch and Win, a general formulation for understanding and synthesising the different tree search procedures that have been presented in the literature of operations research as well as in that of artificial intelligence. Several general ideas are also presented, whose application allows designing new hybrid search algorithms, in order to implement the procedure.

Key Words: Combinatorial optimization, branch and bound, artificial intelligence.

AMS subject classification: 90C27, 90C57, 68T20.

1 Introduction

As it is known, it is very difficult to solve optimally most of the combinatorial optimisation problems due to their intrinsic complexity. At present, though specific procedures exist to solve some specific combinatorial optimisation problems, most of them have to be solved employing general techniques to explore the space of feasible solutions, and, more specifically, through exploratory enumerative procedures in trees and graphs of states (in the enumerative procedures, all solutions are generated and enumerated, explicitly or implicitly).

We analyse the exploratory enumerative procedures in OR tree representations -which have been exposed both in the operations research and

Manuscript received: February 2002. Final version accepted: January 2004.

artificial intelligence (AI) literatures- and we propose a new meta-algorithm that we have called *Branch and Win*: a new general formulation that includes all of them. *Branch and Win* allows for a better understanding of tree search algorithms for combinatorial optimization problems, synthesizes the different tree search procedures that have been presented in the literature, assembles the different elements that are part of these search procedures and uses them dynamically in the tree and provides a means for developing new search strategies and procedures such as hybridizations of search algorithms with local search and metaheuristics.

The rest of the paper continues as follow. In Section 2 the resolution enumerative procedures and the existent general formulations are analysed. *Branch and Win* is formulated and described in Section 3. In Section 4 we claim the integrating character of *Branch and Win*. Several general ideas that allow designing new hybrid search strategies are enumerated in Section 5. Finally, some conclusions are discussed in Section 6.

2 Enumerative Solution procedures

A great number of enumerative search procedures for solving combinatorial optimisation problems have appeared in the operations research literature as well as in the AI literature over the last few decades. Most of these search approaches have been described by different authors and sometimes the definitions presented are not specific enough to know whether they refer to a selection strategy or to a particular algorithm and, in this case, what other characteristics are included in the search algorithm. For example, the depth first search, which always explores a child node of the node most recently separated, exists. But an approach like the previous one has also been given the name of depth first search which, moreover, incorporates a procedure for grouping equivalent states. With the aim of unifying these techniques, general formulations have also been presented.

But sometimes there are aspects that have not been sufficiently taken into account from a practical point of view. Among others: completeness of the corresponding partial solution, the state of the exploration graph (total number of nodes generated, amount of memory used, etc.), the environment conditions in which the problem is solved (maximum allowed computing time, maximum available memory, access time to the medium -RAM, hard disk, etc.- on which a node can be stored, etc.).

2.1 Exploratory enumerative procedures

We present a brief introduction to the relationships between diverse exploratory enumerative procedures for solving combinatorial optimisation problems.

Most of the enumerative procedures have been increasingly employed both in operations research and in AI to solve real problems. Sometimes a procedure has been given different names being the same technique (for example, the depth first procedure has been called several different ways: depth first search, DFS, linear search, single branch search, LIFO search and vertical search). And some procedures present so many similarities that they could be considered redundant: in Barr and Feigenbaum (1981) and Kumar and Kanal (1983b), branch and bound procedures (B&B), dynamic programming techniques (DP) -in this work, DP is used to refer to finite-deterministic dynamic programming techniques- and the heuristic search procedure A* -introduced by Hart et al. (1968)- are said to be very similar, although B&B and DP have been used in operations research and A* in AI (e.g., according to Barr and Feigenbaum (1981), A* is a *best first search* with an evaluation function formed by the sum of the minimal cost associated with reaching the node n from the root, plus a heuristic estimate of the minimal cost of reaching an objective node from the node n). Furthermore the relationships between these classes of procedures have been rather controversial.

For example, Nilsson (1980) defines the algorithm A as a procedure A* that cannot guarantee the optimality of the solution. While many other authors (Barr and Feigenbaum (1981), Korf (1990), Rich and Knight (1991)) consider that there only exists a procedure (A*) and it may or may not guarantee the optimality of the solution depending on the fulfillment or not of the admissibility property. In this case there also exist particular definitions of A*: Rich and Knight (1991) incorporate a procedure for grouping equivalent states and consider that the cost of a path is equal to its number of stages. Pearl (1984) considers A* as a particular Z* (a special best first search described in Pearl (1984)). As it is discussed by several authors (Pearl (1984) and Greenberg (1996), among others), if the selection strategy of the next node to be explored consists in minimising the depth of the node, the procedure A* is equivalent to that of the breadth first search; and, if it consists in minimising minus the depth of the node, the procedure A* is equivalent to that of the depth first search (Pearl (1984)).

For Nilsson (1971), Barr and Feigenbaum (1981) and Pearl (1984) A* is the smallest cost first search procedure if there is not heuristic information. On the other hand, Rich and Knight (1991) discuss a larger number of possibilities. According to Greenberg (1996), branch and bound for integer programming is a particular case of A* in operations research, where the evaluation function takes the value of the objective function of the linear relaxation of the node (without commenting on the main characteristic of B&B: the pruning by a bounding test).

Barr and Feigenbaum (1981) name ordered state space search to the best first search and specify that the breadth first, the uniform cost and the depth first are specific cases of this. On the other hand, Nilsson (1971) and Pearl (1984) specifically define the smallest cost first search (uniform cost search or cheapest first strategy), while most of the authors consider it a special case of the best first search. And Pearl (1984) differences between whether a feasible solution is only looked for (best first search strategy) and whether an optimal solution is looked for (algorithm that is named best first search* by Pearl).

The relationships between B&B and DP have also been rather controversial. While some authors claim that DP is a more general technique than B&B, others consider that it is exactly the opposite. There are also those who claim that some DP techniques can be stated in a framework of B&B, although B&B approaches can also be presented as DP procedures with a bounding test -see, e.g., Marsten and Morin (1978), Kumar and Kanal (1983b) and Ibaraki (1988).

It is also necessary to emphasise the existence of authors that seem to have the relationships between these classes of procedures clearer: "*A class of algorithms similar to A* is used in operations research under the name of branch-and-bound algorithms*", Barr and Feigenbaum (1981), p. 64.

The situation previously described shows that the scene of search procedures for solving combinatorial optimisation problems is rather unstructured, dispersed and insufficiently formalized. On the other hand, in the last years new search procedures have been defined -branch & price (e.g., Savelsbergh (1997)), branch & cut (e.g., Hoffman and Padberg (1993)), branch & reduce (e.g., Ryoo and Sahinidis (1996)), branch & peg (Goldengorin et al. (2004)), etc.- that can be integrated in well-known frameworks, as well as other search techniques as constraints propagation (see, e.g., Haralick and Elliott (1980) and Brailsford et al. (1999)). This can speed up

the development of new procedures, new names or new hybrid techniques, still complicating the current scene more.

In Pastor (1999) and Pastor and Corominas (2000) a wide survey of enumerative search procedures for solving combinatorial optimisation problems is presented. In these works, the relationships, deficiencies and similarities between the enumerative procedures related in the operations research literature as well as in AI literature are studied and analysed; and the following conclusions are obtained:

- There are important elements that have not been sufficiently taken into account from a practical point of view. For instance: total number of generated nodes, amount of memory used, maximum allowed computing time, maximum available memory.
- Dispersion, confusion and even duplicity exist in the definitions of the procedures and in the terminology used. Partly this is due to the fact that these techniques have been used both in operations research and in AI.
- The enumerative search procedures have got diverse common elements whose specification allows to obtain them.
- As some authors present, some search procedures derive from others; but these derivations could be more general.
- In the extreme case, an enumerative procedure would generate the whole state-search, being very inefficient. To reduce the space of states three general reduction tactics exist: elimination of dominated states, pruning of nodes and detection of empty nodes.
- The consistency techniques coming from AI can be embedded, as it is done in some procedures, in branch and bound frameworks -see, e.g., Bockmayr and Kasper (1998), who propose a common framework for constraint programming and integer programming called branch and infer. This connects the search procedures coming from both investigation areas.

2.2 Generalized search procedures

A general formulation whereby all of these procedures can be viewed in a unified manner seems advisable. This necessity has already been detected and diverse general search schemas have already been presented. Among all of these, we consider that some schemas are more general and unifying than others, and even in some formalization, as in Corrêa (1995), some search procedures are shown as specific cases of the proposed general schema. We have studied and analysed the following references: Corominas and Companys (1977), Kumar and Kanal (1983a), Kumar and Kanal (1983b), Nau et al. (1984), Ibaraki (1988), Tuy and Horst (1988) -who present a general B&B scheme for global optimisation-, Helman (1989) and Corrêa (1995). As a result, the following deficiencies (which affect, in a greater or smaller degree, all of them) can be enumerated (Pastor (1999)):

- On some occasions the terminology that is used is not very common in operations research (especially in references coming from AI). This has a negative impact on its understanding and practical applicability.
- The used elements are usually presented with scarce precision.
- Many search procedures are not considered, particularly the most recent.
- The selection strategy of the next node to explore is usually static.
- The following exploration elements are not generally used: Reformulation and pre-processing, in the original problem as well as in the partial problems. Use of local optimisation techniques and local search procedures, in the partial problems and when a new feasible solution is obtained. Calculation of new superior bounds (if we are minimising) in the partial problems. Use of diverse bounding, reformulation and reduction procedures. The environment conditions in which the problem is solved (maximum allowed computing time, maximum available memory, etc.) and the state of the exploration graph.

It is necessary to emphasise that these general schemas have represented an important advance at that moment, for structuring and unifying diverse particular search procedures.

3 *Branch and Win*: OR TREE SEARCH algorithms for Solving Combinatorial Optimisation Problems

The conclusions shown in the previous section, provide evidence of the necessity for designing a new general realistic procedure with the usual terminology in operations research in order to solve combinatorial optimisation problems. To meet this necessity we propose a new meta-algorithm that we have called *Branch and Win*. It is necessary to comment that the nearest antecedents of *Branch and Win*, which we have taken as starting point, are the general procedures presented in Companys (1975) and Corominas and Companys (1977).

3.1 Definitions

*** Problem to be solved:**

We propose a general definition of a combinatorial optimisation problem, in which, after defining the value of the integer variables, a procedure exists to find an optimal solution to the remaining problem:

$$[OPT]\tilde{Z} = \tilde{f}(X,T)$$
$$(X,T) \in S,$$

where: S is the feasible space inside the total space of solutions; X is a vector of integer bounded variables; T is a vector of continuous variables; and given X' is possible to obtain T', which provides the optimal solution of the problem for X'. So one of the more important combinatorial optimisation problems is also included: mixed integer linear programming.

The original problem to be solved can be reduced immediately to the following mathematical program:

$$[MIN]Z = f(X)$$
$$X \in E \subseteq F,$$

where: F is the finite set of possible solutions (usually the Cartesian product of the domains of the variables); E is the finite set of feasible solutions; X is a solution; $f : E \to \kappa$, with κ an ordered set (usually the integer or the continuous numbers).

*** Node:**

Let: $\wp(F)$ the set of subsets of F; N the set of natural numbers; and S_i a subset of solutions. We define a node V_i as follows: $V_i \in \wp(F) \times N$, therefore, $V_i = (S_i, i)$.

The natural number is necessary to identify the node, since the subsets of solutions belonging to two nodes may coincide.

*** Empty node and terminal node:**

V_i is an empty node $\iff S_i \cap E = \emptyset$. V_i is a terminal node $\iff |S_i| = 1$.

*** Branching procedure:**

Let M_0 be a primitive branching procedure. M_0 makes the elements of $\wp(F)$ with cardinality ≥ 2 correspond with two or more subsets of F. M_0 separates the subsets of F without eliminating any element from $\wp(F)$. And the following properties are true: (1) $Y \in M_0(S) \Rightarrow Y \neq S$; (2) $\cup_{\{\forall Y \in M_0(S)\}} Y = S$; and (3) $\forall Y, Z \in M_0(S) \Rightarrow Y \nsubseteq Z$.

Defined M_0, let M be a branching procedure. M is a correspondence that associates a subset of (terminal or non-terminal) nodes to a non-terminal node: M consists in applying a M_0 procedure to S_i, and in assigning a natural number to the S_{ij} generated from S_i, so that a natural number is only assigned once. The branching procedure begins in the node $V_1 = (F, 1)$, the node root, and it continues in the other nodes providing an arborescence. The following properties are true: (1) M does not produce a loss of solutions, either feasible or not feasible (this result comes out of the second property of M_0); and (2) the generated arborescence is finite: in a finite set, M carried out separations that generate subsets of non coincident solutions (this result immediately comes out of the definition of M_0 and of its first property).

Moreover, a terminal node is assumed to be recognisable and to be immediate to ascertain if it is empty or not. So, if V_i is a terminal node or has been detected to be empty, then this node does not have successors (term introduced in the following section).

*** Successor or predecessor node and descendent or ancestor node:**

V_j is a successor node of V_i and V_i is a predecessor node of $V_j \iff V_j \in M(V_i)$. V_j is a descendent node of $V_i \iff V_j \in M(V_i) \cup M^2(V_i) \cup M^3(V_i) \cup \ldots$. V_j is an ancestor node of $V_i \iff V_j \in M^{-1}(V_i) \cup M^{-2}(V_i) \cup M^{-3}(V_i) \cup \ldots$

When M is applied and a successor node V_j is obtained, V_j is said to have been generated in the branching process.

*** Separated and completely separated node:**

V_i is separated $\iff$ one or some of its successors have been generated. V_i is completely separated $\iff$ all of its successors have been generated.

*** Bounding procedure:**

p_e is a bounding procedure, if p_e makes every node V_i correspond with $C_e(V_i) \in \kappa$, where $C_e(V_i) \leq f(X)\ \forall X \in S_i \cap E$ and $\forall p_e \in P$ (being P a finite set of procedures p_e). Thus, $C_e(V_i)$ is a lower bound of the value of the objective function of the feasible solutions contained in the node V_i, obtained through the bounding procedure p_e. To work with several bounding procedures allows to have a succession of bounding procedures, which can be ordered in increasing order of calculation cost, which usually provides an increasing succession of the quality of the gap.

$p_\alpha \in P$ exists such that if V_i is a non-empty terminal node, $C_\alpha(V_i) = f(X)$, $X \in S_i$ (p_α provides the value of the objective function for the only solution, which is feasible, in every non-empty terminal node). Let p_e and p'_e, p'_e is more powerful than p_e ($p_e \leq p'_e$) $\iff C_e(V_i) \leq C'_e(V_i)\ \forall V_i$.

Let P_i be the set of bounding procedures that have been applied in V_i; let C be the best bound associated to V_i: $C(V_i) \max\limits_{\forall p_e \in P_i} \{C_e(V_i)\}$; and let $\overline{C}(V_i)$ be a bound obtained when having information of the predecessor or of the successor nodes. The following properties are true: (1) $V_j \in M(V_i) \Rightarrow \overline{C}(V_j) = \max\{C(V_j), C(V_i)\}$; and (2) $\overline{C}(V_i) = \max\Big\{C(V_i), \min\limits_{\forall V_j \in M(V_i)} \{C(V_j)\}\Big\}$ (provided that V_i be completely separated).

*** Reduction procedure:**

A reduction procedure e, $r_e \in R$ (with R being a finite set of procedures r_e), consists in replacing a node $V_i = (S_i, i)$ by a new node $V_i' = (S_{ie}', i)$, with $S_{ie}' \subset S_i$ and, if V_i is also a non-empty node, $\exists X_{S_i}^* \in S_{ie}'$ if only one optimal solution is looked for (where $X_{S_i}^*$ is an optimal solution in S_i) or $\nexists X_{S_i}^* \in S_i \setminus S_{ie}'$ if all optimal solutions are looked for.

Let r_e and r_e', r_e' is more powerful than r_e ($r_e \leq r_e'$) $\Longleftrightarrow S_{ie'}' \subseteq S_{ie}'$.

Reduction can conclude in fixing variables or in decreasing its ranges of values, although an optimal solution of S_i can also be detected. In this procedure the resulting information can also be transmitted, from successor to predecessor nodes and vice versa: (1) $V_j \in M(V_i) \Rightarrow S_j' = S_j \cap S_i'$; and (2) $S_i' = \cup_{\{\forall V_j \in M(V_i)\}} S_j'$ (provided that V_i be completely separated).

It is easy to check that constraint propagation approaches and consistency techniques (coming from AI) are reduction procedures.

*** Heuristic resolution procedure:**

Let H be a finite set of heuristic procedures. If $h_e \in H$ finds a feasible solution $X \in S_i \cap E$, then $U_e(V_i) = f(X)$ is an upper bound of the value of the objective function of an optimal solution contained in S_i, obtained through the heuristic resolution procedure h_e.

Let h_e and h_e', h_e' is more powerful than h_e ($h_e \leq h_e'$) $\Longleftrightarrow U_e(V_i) \geq U_e'(V_i)$ $\forall V_i$ (h_e' always provides feasible solutions with a gap, relating to an optimal solution, equal to or less than that provided by h_e).

Let H_i be the set of heuristic resolution procedures that have been applied to V_i; let $U(V_i)$ be the value of the best obtained feasible solution associated to V_i: $U(V_i) = \min\limits_{\forall h_e \in H_i} \{U_e(V_i)\}$; and let $\overline{U}(V_i)$ be the value of the objective function of a solution obtained when having information of their successor nodes. So: $\overline{U}(V_i) = \min\{U(V_i), \min\limits_{\forall V_j \in M(V_i)} \{U(V_j)\}\}$.

It is worthwhile emphasizing that neighbourhood exploration procedures/metaheuristics e_e (which will be introduced later) can be defined as a part of a more elaborated procedure h_e.

*** Examination procedure:**

To examine a node V_i consists in applying to it bounding and/or reduction and/or heuristic resolution procedures, with the objective of improving the information that we have about the node and, sometimes, also about others. To have several types of examination procedures allows applying them in an alternative and iterative way; and even these can be ordered in increasing order of difficulty or calculation cost, in the hope that the least expensive procedures allow drawing conclusions briefly: they are applied in increasing order of complexity.

*** Evaluation procedure:**

The evaluation procedure is a correspondence that associates a value $\varphi(V_i) \in \mathbb{R}$ to each node V_i, where φ is the evaluation and selection function of the next node to examine.

In spite of the huge importance of the strategy for selecting the next node to branch, in many papers this is not stated explicitly or is described with few details. On the other hand, the generalization of an evaluation and selection function of nodes is an extremely important matter in the definition of a tree search meta-algorithm, *Branch and Win* in our case. This function, through the specification of the parameters that it incorporates, makes it possible to describe the selection strategies of the main procedures referred in the literature as a particular case of this function.

Most commonly used selection strategies are merely a function of the considered node, and they take neither the state of the exploration graph nor the environment conditions in which the problem is solved into account. From a practical point of view, there are aspects that have not been sufficiently taken into account. In Pastor and Corominas (2000) a general formalization of the selection strategy of the next node to examine is proposed.

The selection strategy is applied by means of a function of the considered node, the time spent on the computation, the state of the tree and the conditions of the environment in which the problem is solved. It has, therefore, a dynamic nature and it incorporates elements, which had not been traditionally considered but are of great importance (for instance: the position of the node in the tree, the amount of memory available).

*** Dominance relations:**

A node V_i dominates another node V_j, $V_i \succ V_j$, if $f(X^*_{S_i}) \leq f(X^*_{S_j})$ if only one optimal solution is looked for or $f(X^*_{S_i}) < f(X^*_{S_j})$ if all optimal solutions are looked for. Particularly, we can guarantee that this condition is fulfilled if $\forall X_{S_j} \in (S_j \cap E)\ \exists X_{S_i} \in (S_i \cap E) \mid f(X_{S_i}) \leq f(X_{S_j})$ if only one optimal solution is looked for or $f(X_{S_i}) < f(X_{S_j})$ if all optimal solutions are looked for. The objective of checking dominance relations among nodes consists in reducing the explicit enumeration.

*** Properties of the optimal solution:**

Sometimes, the characteristics of the problem to solve allow determining some properties of the optimal solutions. Two situations can be presented: (1) every optimal solution fulfills a property Π; and (2) an optimal solution that fulfills a property Π' always exists.

If it can be determined that a node does not contain feasible solutions with these properties, the node can be pruned. This allows reducing, sometimes in a large way, the size of the search tree to make it explicit.

*** Neighbourhood exploration procedures/metaheuristics:**

In a terminal non-empty node, a neighbourhood exploration procedure $e_e \in EN$ (with EN being a finite set of procedures e_e) explores feasible solutions belonging to the neighbourhood of the current solution (which is, to start with, the solution contained in V_i), until some end condition is fulfilled.

The neighbourhood exploration procedures that are considered here can be both local optimisation techniques (k-exchanges, etc.) and metaheuristics (among others: simulated annealing, tabu search, genetic algorithms, GRASP).

*** States of a node:**

Once the different procedures that can be applied to a node have been defined, now the states that the nodes can have when applying them these procedures are described:

1. Generated or not generated (obtained or not obtained, in the branching process).

2. Pruned or not pruned: a node is pruned if one is completely sure that it does not include any solution of interest; a pruned node is not taken into account and it is enough that any of the following conditions is fulfilled:

   - to be an empty node;
   - to be a terminal non-empty node and to know the value of the solution $f(X)$ if $X \in E$;
   - $\overline{C}(V_i) \geq \overline{Z}$ if only one optimal solution is looked for or $\overline{C}(V_i) > \overline{Z}$ if all optimal solutions are looked for, where $\overline{Z}$ is the objective function value of the incumbent solution -i.e., the best solution obtained-;
   - $\overline{C}(V_i) = \overline{U}(V_i)$;
   - to be dominated by another node V_j: $V_j \succ V_i$;
   - not to contain solutions that fulfill the property Π' if only one optimal solution is looked for, or the property Π if all optimal solutions are looked for;
   - to be completely separated and to have pruned all its successor nodes;
   - to have pruned their predecessor node;
   - not to fulfill a condition of permanency that converts *Branch and Win* in a heuristic procedure: not to be among the best nodes if the memory is finished or when a maximum number of nodes is generated, etc.

3. Virtually examined by a reduction procedure r_e : we have S'_{ie} that cannot be reduced applying r_e.

4. Virtually examined by a bounding procedure p_e applied in S_i: we know a bound of S_i that cannot be improved by applying p_e to S_i.

5. Virtually examined by a heuristic resolution procedure h_e applied in S_i: we have obtained a solution $X \in S_i \cap E$ that cannot be improved with h_e.

6. Completely examined: node virtually examined by every reduction, bounding and heuristic resolution procedures.

7. Closed or not closed: a node is closed if it is pruned or if all of its successor nodes, if it has successors, are completely examined.

8. Evaluated or not evaluated: a node V_i is evaluated if the value of $\varphi(V_i)$ is known. For some functions φ this state is permanent (for example, the depth), but for other it is an ephemeral state and it can only be used to carry out one selection (for example, when φ depends on the remaining allowed computing time).

3.2 The *Branch and Win* meta-algorithm

The objective consists in minimising and only one optimal solution is looked for. We consider obvious the changes that are necessary to carry out if we maximise or if all optimal solutions are looked for.

3.2.1 Definitions

Let: $\overline{X}$, the incumbent solution; $\overline{Z} = f(\overline{X})$, the objective function value of $\overline{x}$; n, a natural number associated with the node generated; $V_1 = (F, 1)$, the root node; L, the list of nodes generated, not pruned and not closed; and, $\hat{V}$, $\hat{V}'$, $\hat{V}''$, nodes.

3.2.2 *Branch and Win* procedure

Below the *Branch and Win* meta-algorithm is described:

Phase 1. BEGIN
Obtain an initial solution $\overline{X}$;
$\overline{Z} = f(\overline{X})$, or, if $\overline{X}$ is unknown, $\overline{Z} = +\infty$
$n = 1$; generate $V_1 = (F, 1)$; $L = \{V_1\}$.

Phase 2. MAIN
while not $L = \emptyset$ do
SELECT_FROM_$L(\hat{V})$
case $\hat{V}$

```
                case V̂ is a terminal node
                        EXAMINE_TERMINAL (V̂)
                 case V̂ is a completely separated node
                        SELECT_FROM_FAMILY (V̂′)
                        EXAMINE_GENERAL (V̂′)
                else
                        NEW_SUCCESSOR (Nd)
                        if Nd = 1 then
                                GENERATE (V̂″)
                                EXAMINE_GENERAL (V̂″)
                        else
                                SELECT_FROM_FAMILY (V̂′)
                                EXAMINE_GENERAL (V̂′)
                        end if
        end case
    end while
```

Phase 3. DISPLAY
    Show the results

The most important element in the first phase of *Branch and Win* consists in obtaining an initial feasible solution that could be any in principle or, even, that cannot be provided. But the quality of the initial solution usually has a large influence on the necessary time to complete the search: it can allow to begin the pruning process right from the start. In order to obtain an initial feasible solution of certain quality, one or several heuristic resolution procedures can be used, and these can be completed with neighbourhood exploration procedures.

3.2.3 Subroutines

Below the subroutines of *Branch and Win* are described:

* SELECT_FROM_L ($\hat{V}$):
    do $\forall$ node $V_i \in L$
        Calculate $\varphi(V_i)$ if its updated value is unknown.
    end do
    Select a node $\hat{V}$ among those of better value $\varphi(V_i)$.

According to the nature of the elements that make up the dynamic evaluation and selection function $\varphi(V_i)$ -the value of a lower or an upper bound, the depth, etc.-, to calculate it again will be necessary if its updated value is unknown.

* EXAMINE_TERMINAL ($\hat{V}$):
Select an examination procedure and apply it.
if $f(X)$ is available then
if $f(X) < \overline{Z}$ then
$\overline{X} = X$, $\overline{Z} = f(X)$.
end if
if the condition of neighbourhood exploration is fulfilled then
EXPLORE_NEIGHBOURHOOD ($\hat{V}$)
end if
Close the node and, consequently, $L = L \setminus \{\hat{V}\}$
end if
Deduce changes in the state of the node $\hat{V}$ and others.

An examination procedure can contain one or several bounding, reduction and heuristic resolution procedures; moreover, an examination procedure that applies these procedures in an iterative form could be designed (e.g., bounding-reduction-bounding-reduction-...). In this subroutine, the examination procedure only includes bounding procedures. If the node is not closed, its state could change: if a bound that allows to prune it is achieved or if it does not fulfill the properties of the optimal solution. Furthermore, changes in the state of other nodes may occur (e.g., if the value of $\overline{Z}$ is improved); if a node is closed, it is obviously eliminated from L.

* EXPLORE_NEIGHBOURHOOD ($\hat{V}$)
Select a neighbourhood exploration procedure and apply it.

* SELECT_FROM_FAMILY ($\hat{V}'$):
Select a not completely examined node, $\hat{V}', \in \{\hat{V} \cup [M(\hat{V}) \in L]\}$.

* EXAMINE_GENERAL ($\hat{V}'$):
if $\hat{V}'$ is a terminal node then
EXAMINE_TERMINAL ($\hat{V}'$)
else
Select an examination procedure and apply it.
Deduce changes in the state of the node $\hat{V}'$ and others.

end if

In this subroutine, the examination procedure can contain several bounding, reduction and/or heuristic resolution procedures, and, moreover, in an iterative form or not. If the node is closed, all of its descendents are also pruned; moreover, this state can sometimes be propagated towards ancestor nodes. On the other hand, the same changes of states that with EXAMINE_TERMINAL ( ) can occur; but, furthermore, there can also be changes when the dominance relations among nodes are tested. Of course, if it is detected that the node is empty, it is pruned.

* NEW_SUCCESSOR (Nd):
If all node $\in \{\hat{V} \cup [M(\hat{V}) \in L]\}$ is completely examined then
$Nd = 1$
else
Decide the value of Nd (1 to generate a new successor and 0 otherwise)
end if

* GENERATE ($\hat{V}''$):
Select the branching procedure.
Decide the variable or the condition that it is used to branch.
$n = n + 1$.
Generate $\hat{V}'' = (\hat{S}'', n)$.
$L = L \cup \{\hat{V}''\}$.

Usually, the branching procedure provides partitions of F; but it can also provide separations that are not a partition of F (e.g., in the asymmetric travelling salesman problem, to prohibit the arc ij or the arc ji does not provide partitions of the set of solutions). Moreover, for the same problem is possible to define several branching procedures. In the branching process a terminal node is recognisable and is immediate to ascertain if it is empty or not: if it is an empty node, it is closed and not incorporated to L.

4 *Branch and Win*: A general meta-algorithm

Once *Branch and Win* has been formulated, it is not difficult to prove its general and summarising nature. As an example, below we describe, as particular cases of *Branch and Win*, some procedures of the operations research and AI literatures (most of the enumerative procedures not specified

in this Section are not considered difficult to specify as particular cases of *Branch and Win*).

1. Search procedures based on finite-deterministic dynamic programming techniques. The main characteristic of these procedures is to use, as a differential feature, the dominance relations between nodes.

2. Branch and bound algorithms. Search procedures that use the branching (branch) and the bounding (bound) to explore the space of solutions; usually, first a initial feasible solution is calculated by means of heuristic procedures.

3. A*. Branch and bound procedure with particular evaluation and selection strategy of the next node to examine: best first search.

4. Depth-first / Breadth-first / Best-first /... branch and bound. Branch and bound procedures with particular evaluation and selection strategies of the next node to examine: depth search, breadth search, best first search, ...

5. Branch and cut. Branch and bound procedures in which, as a differential feature, the linear programs (results of relaxing the condition of integrity of the variables) are solved by means of cutting plane algorithms; so an iterative bounding procedure is defined, in which the objective consists in obtaining tight bounds. As any procedure derived from *Branch and Win*, it can also include several refinements (among others: reduction procedures, heuristic resolution techniques, neighbourhood exploration metaheuristics).

6. Branch and price. Branch and bound procedures in which, as a distinctive feature, the linear programs (results of relaxing the condition of integrity of the variables) are solved by means of column generation.

7. Hybrid search procedures. Procedures that use several search techniques in an alternative and/or simultaneous form, as well as exact as heuristic techniques; so, once the "pure" procedures are specified as particular cases of *Branch and Win*, these "hybrid" techniques are automatically defined. It should be noted that dynamic programming procedures could be stated in a framework of branch and bound or

vice versa; and neighbourhood exploration techniques could be added in branch and bound procedures.

8. Search procedures based on reduction techniques. The main characteristic of these procedures is to use, as a differential feature, reduction procedures in every node of the arborescence.

9. Constraints propagation procedures. Framework that embeds consistency techniques into an enumerative search algorithm.

10. Heuristic enumerative procedures. It is not difficult to specify some heuristic enumerative procedures as particular cases of *Branch and Win*: when we do without some of the conditions that define them and the search is carried out in a constrained area.

5 Ideas on search based on *Brach and Win*

Once the general and unifying nature of *Branch and Win* has been shown, it is immediately possible to design new search procedures, when we combine the different techniques defined and the parameters of the general evaluation function of the next node to examine -for more details, see Pastor and Corominas (2000)-. Moreover, these can be either exact or heuristic procedures.

Below some general ideas on search are listed; it can be interesting to test these ideas in those combinatorial optimisation problems that are not still solved efficiently. We should emphasize that some of these ideas are already being used in enumerative search procedures:

- To study the influence of investing time trying to find an initial solution of quality to initialize *Branch and Win*: try different heuristic resolution procedures and diverse neighbourhood exploration techniques that these heuristics can include.

- To introduce the concept of dynamism in the different procedures that make up *Branch and Win*, and, especially, in the evaluation procedure -for more details, see Pastor and Corominas (2000)-.

- To include elements that are not sufficiently taken into account, especially to solve industrial problems (allowed computing time, remaining computing time, amount of memory used, etc.).

- To test to use in different orders the elements that are part of the examination procedures (bounding, reduction and heuristic resolution).
- To work with several options in the bounding, reduction, heuristic resolution, neighbourhood exploration procedures and in the properties of the optimal solutions; the objective is to test their use in an increasing order of complexity.
- To transmit properties from successor to predecessor nodes and vice versa, to improve the available information on the nodes.
- To use reduction and heuristic resolution procedures in the intermediate nodes of the arborescence (in all the nodes or only in those that are more promising than the other ones).
- To design heuristic resolution procedures that include neighbourhood exploration techniques.
- To apply neighbourhood search procedures when the value of the incumbent solution is improved or whenever a new feasible solution is obtained.

6 Conclusions

In this work we design *Branch and Win*: an OR tree search meta-algorithm for solving combinatorial optimisation problems. *Branch and Win* allows to specify, as particular cases of it, the different tree search procedures that have been presented in the literature of operations research as well as in that of artificial intelligence.

The conclusions are the following:

- Currently, in the field of search procedures for solving combinatorial optimisation problems, the scene is rather unstructured and somewhat confusing: some authors define and use elements or procedures in one way and others in another way which, although similar, is not equivalent. Moreover, the general formulations that have been presented in the literature have a set of deficiencies that affect, in a greater or smaller degree, all of them.

- With respect to the selection functions and to the exploration strategies that are proposed in the usual procedures, there are aspects that have not been sufficiently taken into account from a practical point of view.
- *Branch and Win* has been formulated, with the following features:
  a) It is general enough to include, as particular cases, all these enumerative techniques.
  b) It takes aspects, which are important from an industrial point of view, into account: maximum allowed computing time, amount of memory used, remaining computing time.
  c) It uses a clear and unifying terminology.
  d) It combines and specifies the diverse common elements that have all these procedures to obtain them.
  e) It allows to use bounding, reduction, heuristic resolution and neighbourhood exploration procedures, in all the partial problems generated.
  f) The concept of dynamism is introduced in the different procedures that make up the meta-algorithm: bounding, reduction, heuristic resolution, ...
  g) It uses a general evaluation and selection function of the next node to examine, including many new elements which had not been traditionally considered but are of great importance: the state of the exploration graph and the environment conditions in which the problem is solved.
  h) It allows to design new search procedures, according to how the elements of *Branch and Win* are combined.

Thus, the new meta-algorithm that we have called *Branch and Win* allows for a better understanding of tree search algorithms for combinatorial optimization problems and it synthesizes the different tree search procedures that have been presented both in operations research and in artificial intelligence literatures. This meta-algorithm assembles the different elements that are part of these search procedures and uses them dynamically in the tree. And furthermore, *Branch and Win* provides a means for developing new search strategies and procedures.

References

Barr A. and Feigenbaum E.A. (eds) (1981). *The Handbook of Artificial Intelligence (Volume 1).* Kaufmann.

Bockmayr A. and Kasper T. (1998). Branch and Infer: A Unifying Framework for Integer and Finite Domain Constraint Programming. *INFORMS Journal on Computing* 10, 287-300.

Brailsford S.C., Potts C.N. and Smith B.M. (1999). Constraint Satisfaction Problems: Algorithms and Applications. *European Journal of Operational Research* 119, 557-581.

Companys R. (1975). Programación Combinatoria: Aplicación a la Ordenación de Trabajos en un Ordenador con Sistema Operativo en Discos. *Symposia Mathematica* XV, 83-107.

Corominas A. and Companys R. (1977). Procedimiento Generalizado de Branch and Bound. *Qüestiió* 1, 49-62.

Corrêa R. (1995). A Parallel Formulation for General Branch-and-Bound Algorithms. *Lecture Notes in Computer Science* 980, 395-409.

Goldengorin B., Ghosh D. and Sierksma G. (2004). Branch and Peg Algorithms for the Simple Plant Location Problem. *Computers and Operations Research* 31, 241-255.

Greenberg H.J. (1996). Artificial Intelligence. In: Gass S.I. and Harris C.M. (eds), *The Encyclopedia of Operations Research and Management Science.* Kluwer Academic Publishers, 25-28.

Haralick R. and Elliott G. (1980). Increasing Tree Search Efficiency for Constraint Satisfaction Problems. *Artificial Intelligence* 14, 263-313.

Hart P.E., Nilsson N.J. and Raphael B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on System Science and Cybernetics* SSC-4, 100-107.

Helman P. (1989). A Common Schema for Dynamic Programming and Branch and Bound Algorithms. *Journal of the Association for Computing Machinery* 36, 97-128.

Hoffman K.L. and Padberg M. (1993). Solving Airline Crew Scheduling Problems by Branch-and-Cut. *Management Science* 39, 657-682.

Ibaraki T. (1988). Enumerative Approaches to Combinatorial Optimization, part I and II. *Annals of Operations Research* 10 and 11.

Korf R.E. (1990). Search. In: Shapiro S.C. (ed), *Encyclopedia of Artificial Intelligence.* John Wiley, 994-998.

Kumar V. and Kanal L. (1983a). A General Branch and Bound Formulation for Understanding and Synthesising And/Or Tree Search Procedures. *Artificial Intelligence* 21, 179-198.

Kumar V. and Kanal L. (1983b). The Composite Decision Process: a Unifying Formulation for Heuristic Search, Dynamic Programming and Branch and Bound Procedures. *Proceedings of the Third National Conference on Artificial Intelligence*, 220-224.

Marsten R.E. and Morin T.L. (1978). A Hybrid Approach to Discrete Mathematical Programming. *Mathematical Programming* 14, 21-40.

Nau D.S., Kumar V. and Kanal L. (1984). General Branch and Bound, and its Relation to A* and AO*. *Artificial Intelligence* 23, 29-58.

Nilsson N.J. (1971). *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill.

Nilsson N.J. (1980). *Principles of Artificial Intelligence.* Kaufmann.

Pastor R. (1999). *Metalgoritmo de Optimización Combinatoria Mediante la Exploración de Grafos.* Ph.D. Dissertation, Technology University of Catalonia.

Pastor R. and Corominas A. (2000). Strategies of Node Selection in Search Procedures for Solving Combinatorial Optimization Problems: A Survey and a General Formalization programming. *Top* 8, 111-134.

Pearl J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley.

Rich E. and Knight K. (1991). *Artificial Intelligence.* McGraw-Hill.

Ryoo H.S. and Sahinidis N.V. (1996). A Branch-and-Reduce Approach to Global Optimization. *Journal of Global Optimization* 8, 107-139.

Savelsbergh M. (1997). A Branch-and-Price Algorithm for the Generalized Assignment Problem. *Operations Research* 45, 831-841.

Tuy H. and Horst R. (1988). Convergence and Restart in Branch-and-Bound Algorithms for Global Optimization. Application to Concave Minimization and D.C. Optimization Problems. *Mathematical Programming* 41, 161-183.