Increasing the efficiency of declarative modelling. Constraint evaluation for the hierarchical decomposition approach

Dimitri Plemenos, Karim Tamine
Email : plemenos@unilim.fr, tamine@jabba.unilim.fr
Laboratoire MSI, University of Limoges
123, Av. Albert Thomas, F - 87060 Limoges, France.

Abstract

Declarative modelling in computer graphics allows property-based scene descriptions. The properties of a scene are often used to get a set of constraints which must be resolved. In declarative modelling, scene properties can be imprecise ; thus, the generated system of constraints can admit several solutions. A special form of declarative modelling, declarative modelling by hierarchical decomposition, generates a system of constraints, a part of which has hierarchical dependence structure. In this paper, we present constraint evaluation methods as an alternative to constraint resolution. These methods have several advantages and also some drawbacks. In order to attenuate these drawbacks, new constraint organisation techniques together with changes to the used tree search techniques are presented.

Keywords : Geometric modelling, declarative modelling, computer graphics, constraint resolution, constraint evaluation.

1. Introduction

Modelling is a very important step in computer graphics because it allows designing of scenes, i.e. abstract descriptions of objects of real world, which will be used by display algorithms to obtain more or less realistic images. Well known geometric models allow scene design by using faces or surface patches and solid representation sub-models like octrees, CSG trees or Brep. Two papers by Lee and Requicha ([1], [2]) resume the main geometric models used in solid modelling. However, geometric modelling is badly adapted to scene conception ; a new modelling technique appears at the end of the years 1980. This modelling, so-called declarative modelling, uses high level properties to design scenes. Such a modelling technique presents many advantages and also some drawbacks, especially in the solution generation phase which is generally time consuming. The purpose of this work is to present some improvements in the generation process of a special form of declarative modelling, declarative modelling by hierarchical decomposition. Combination of these improvements allows important acceleration of solution space exploration.

After a brief definition of declarative modelling, we will introduce, in section 2, the concept of declarative modelling by hierarchical decomposition. In section 3, we will present a technique of implementation for this kind of modelling. Some drawbacks of declarative techniques will be discussed in section 4. These drawbacks justify the use of new optimisation techniques in the solution search process, which will be presented in chapter 5. Finally, section 6 will permit to present some conclusions concerning implementation of the new techniques exposed in this paper. New improvement techniques will be presented in section 7.

2. Declarative modelling by hierarchical decomposition

The purpose of declarative modelling is to attenuate drawbacks of classical geometric modelling by offering the possibility of scene description using properties, which can be either precise or imprecise([3], [5]).

More precisely, declarative modelling allows the user to tell which properties must verify a scene, without indicating the manner to obtain a scene with these properties. Because the scene conceiver has no necessarily complete knowledge of all details of the scene he (she) wants to obtain, it seems natural to allow him (her) to use imprecise properties to describe the scene.

In order to allow descriptions at various detail levels, we have introduced a new declarative modelling technique, so-called declarative modelling by hierarchical decomposition (DMHD)

[5]. This modelling technique uses top-down hierarchical description and works as follows :

If a scene is easy to describe, it is described by a small number of properties which can actually be **size** (inter-dimensions) **properties** (higher than large, as high as deep, etc.) or **form properties** (elongated, very rounded, etc.). Otherwise the scene is partially described with properties easy to describe and is then decomposed into a number of sub-scenes and the same description process is applied to each sub-scene. In order to express relationships within sub-scenes of a scene, **placement properties** (put on, pasted on the left, etc.) and **size** (inter-scenes) **properties** (higher than, etc.) are used.

Thus, let's consider the description of a scene composed of three objects, Garage, Walls and Roof and having the following properties :

- the object Walls and the object Roof are pasted on the left of the object Garage and the object Roof is put on the object Walls,
- the object Garage has its top 80% rounded and the object Roof has its top 70% rounded.
- the whole (Walls, Roof) is higher than large and higher than deep.

This description gives all known properties of the scene but it is not easy to understand and possible contradictions are difficult to detect. Instead of this one, we propose a description based on a functional hierarchical decomposition, by introducing new virtual objects grouping other, real or virtual objects. Thus, in description of figure 1, we introduce the new objects Residence and House.

The scene named *Residence* is difficult to describe and is decomposed into two sub-scenes, *House* and *Garage*. House is also difficult to describe and is decomposed into *Walls* and *Roof*.

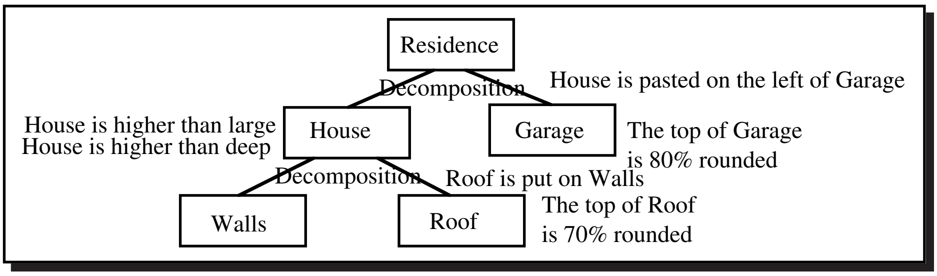

Figure 1 : hierarchical description of a scene

This user description is translated into an internal model, made up of a set of Prolog-like rules and facts.

Thus, the above top-down description will produce the following internal model :

```
Residence (x,p) ->  CreateFact (x,p)
                    House (y,x)
                    Garage (z,x)
                    PastedLeft (y,z,x);
House (x,p)    ->   CreateFact (x,p)
                    HigherThanLarge (x)
                    HigherThanDeep (x)
                    Walls (y,x)
                    Roof (z,x)
                    PutOn (z,y,x);
Garage (x,p) ->     CreateFact (x,p)
                    TopRounded (x,80);
Walls (x,p) -> CreateFact (x,p);
Roof (x,p) ->  CreateFact (x,p)
               TopRounded (x,70);
```

In this set of rules appear properties like *TopRounded*, *PutOn*, etc. which are already defined into the modeller. Additional properties can be added to the knowledge base of the modeller by the expert user. The predicate *CreateFact* is automaticaly generated by the modeller and is used to manage hierarchical dependencies within elements of the scene.

The main advantages of such a hierarchical top-down description are :

- description can be made at various detail levels and thus enables to describe scenes in progressive manner.
- the properties of the scene to design can be described locally, without having to think about properties of other parts of the scene.
- hierarchical decomposition enables property factoring ; thus, if *Walls* and *Roof* must be on the left of *Garage*, it is enough to impose this property to their parent *House*.

The scene generation uses the **hierarchical multi-level generation** technique, which can be described as follows :

Starting with an approximative sketch of the scene, the modeller applies description rules in order to refine progressively this approximative representation and to get, finally, a set of solutions (scenes) verifying the properties wanted by the user.

The scenes generated are actually solids or a union of solids. Generation of other kinds of scenes by top-down multi-level description an hierarchical generation is possible but not yet experimented.

In order always to get a finite number of solutions, even with unprecise properties, we use a discrete workspace, defined during the description step. The use of a discrete workspace is a tool of limitation of the number of generated scenes, which also permits, in many cases, to insure unicity of the proposed solutions.

The main advantages of the hierarchical multi-level generation technique are :

- generation can be made at various detail levels ; thus, one can stop the scene generation at a given detail level, even if the scene description includes additional detail levels.
- generation insures inheritence of properties ; thus, if one specifies that *House* is pasted on the left of *Garage*, this property will be true for all sub-scenes of *House*, at any level. On the other hand, if *Residence* has the property *Rounded*, this property is inherited by the sub-scenes of *Residence* in order to form a scene with rounded aspect.

3. Scene generation

3.1 Scene génération using an inference engine

The first version of the MultFormes hierarchical declarative modeller uses a special inference engine to implement the hierarchical multi-level generation technique. This inference engine processes the rules and facts of the internal model. The applied strategy uses backward resolution and breadth-first search.

Starting from a goal to state, the inference engine scans the rules of the internal model and, if one of the rules is applicable, it applies the rule as long as the rule remains applicable.

A rule is applicable if the two following conditions are verified together :

- its conclusion can be unified with the current goal,
- the base of facts generated by this rule is not saturated.

The nature of a rule processed by the inference engine can be one of the following :

- nonterminal rule,
- terminal rule,
- the special rule *CreateFact*.

Only nonterminal rules are applicable many times.

This generation technique presents some drawbacks : the memory space used increases too fast with the scene complexity ; it is not possible to display individual solutions when generated ; it is difficult to establish a general application order for rules in order to avoid unnecessary search.

3.2 Generation using constraint evaluation

Hierarchical multilevel generation technique using only an inference engine and deduction rules is memory consuming because generated facts include several solutions. So, we have developed a variant of this technique, briefly described in the following lines.

The processing of deduction rules by an inference engine has two kinds of results :

- generation of a hierarchical fact, to each node of which a set of constraints (so called **local** constraints) is associated. A set of **global** constraints is also generated and associated to the whole fact (figure 2).
- reduction of variation intervals of variables, so called **attributes**, which describe a scene.

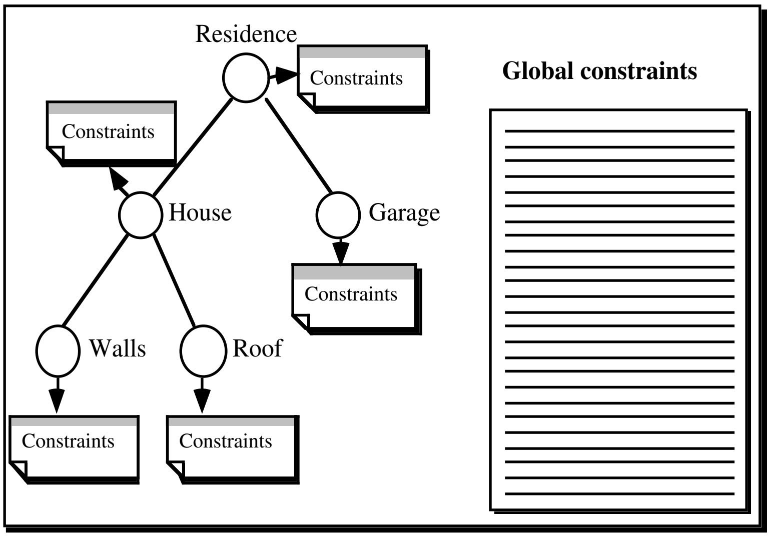

Figure 2 : Hierarchical fact, local and global constraints

Reduction of variation intervals is obtained by the application of rules other than form rules. These rules have a double action : reduction of variation intervals and constraint generation.

Thus, application of the rule *CreateFact* has as consequences :

- Generation of a new partial fact and a link with its parent.
- Generation of constraints indicating that the values of the corresponding sub-scene attributes are bounded by the ones of its parent in the hierarchy.

In the same way, application of size rules permits to reduce variation intervals of some attributes. Thus, application of the rule *HigherThanLarge* produces the following effects :

- Generation of a constraint describing the corresponding property : Breadth < Height.

- Reduction of the variation interval of the attribute *Breadth* :
  If *MaxHeight < MaxBreadth*, do *MaxBreadth=MaxHeight - 1.*

- Reduction of the variation interval of the attribute *Height* :
  If *MinHeight < MinBreadth*, do *MinHeight=MinBreadth + 1.*

The other size rules are processed in the same way.

Unfortunately, this reduction of variation intervals is not very efficient. Reduction of variation intervals occurs only to margins of intervals and the gain is relatively light for a refined sampling of the workspace.

Local constraints are introduced in order to improve generation. The purpose of this improvement is to avoid exploration of some, more or less important, sets of values if it is clear that this exploration is useless.

Thus, it is possible that the embedding box of a node verifies a size property, whereas this property is not verified by a particular scene generated into this embedding box (figure 3). In this case, it is useless to explore all of the sub-scenes of the current scene occurrence. So, in figure 3, all scenes must verify Breadth < Height but this property is not verified by a particular scene. Local constraints allow to avoid immediately this scene and its sub-scenes.

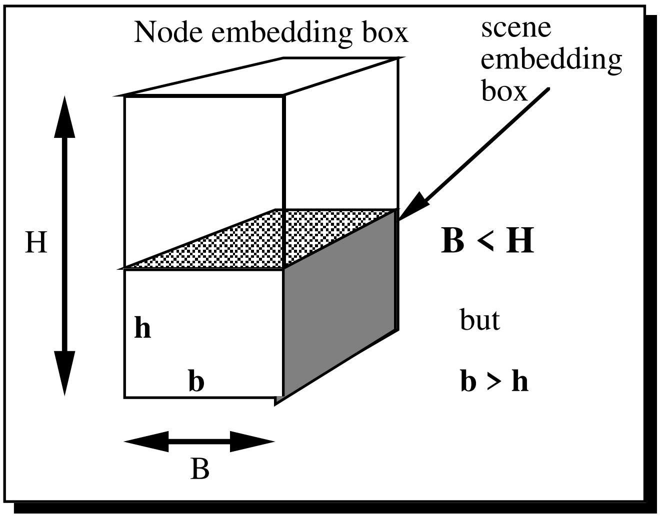

Figure 3 : Current node and scene embedding boxes

The only way to know whether attribute values of a node occurrence verify a property is to evaluate the constraints generated by this property. Unfortunately, with only global constraints, this evaluation occurs too late, when all of child nodes of the current node have been generated. On the other hand, it is no possible to evaluate all constraints for each occurrence of a node. Introduction of local constraints, i.e. constraints concerning only this node, permits to resolve this problem.

Thus, a separated module traverses the hierarchical fact and, for each set of values of attributes,evaluates sets of constraints. For each new set of values of attributes of a node occurrence, local constraints of the node occurrence are evaluated. If all local constraints for this occurrence are verified, the generation process is pursued with descending nodes of the current node occurrence. Otherwise, a new occurrence of the current node is generated and so exploration of all descending nodes of the current node occurrence is avoided.

Global constraints are evaluated only if a complete occurrence of the root node has supported a successful evaluation of local constraints of all node occurrences of the tree depending on this root node occurrence.

A solution is accepted if local and global constraints are satisfied. If a solution is found, form properties are applied and internal representation of the corresponding scene is created.

Scene generation using constraint evaluation offers better performances than the one using an inference engine, particularly low memory occupation and fast generation of solutions.

4. Advantages and drawbacks of generation using constraint evaluation.

Several recent modellers use constraints to express properties of scenes [4]. The main advantage of generation using constraint evaluation in DMHD is its generality. Indeed, it allows to process constraints of any kind and degree, in opposition with various methods of constraint resolution.

Another advantage of this technique is its low cost in memory : a single hierarchical fact is generated and memory occupation does not increases with the number of solutions.

The main drawback of the method is its cost in terms of useless attempts, very important because of the hierarchical nature of variation intervals. Indeed, reduction of variation intervals occurs only to margins of these intervals and gain is relatively light for a refined sampling of the workspace.

The problem of evaluation of a set of constraints, often called the constraint consistence problem, is treated in several papers and books [9, 10, 11, 12, 13, 14]. Some of the improvements proposed in this paper are inspired, directly or indirectly, from these works.

In the following section, we will study methods to improve generation by reducing the number of attempts to explore useless branches.

5. Improvements of constraint evaluation

In order to improve performances of generation by constraint evaluation, we have studied a number of techniques which will be presented in this section.

5.1 Dynamic reduction of variation intervals

Reduction of variation intervals, introduced in section 3 is a static one and this is the main reason of its lack of efficiency for a refined sampling of the workspace. It would be interesting to complete this static reduction with a dynamic reduction of variation intervals of the node's attributes.

This reduction technique is based on the fact that, for a node occurrence, values of attributes of its descending nodes cannot be outside of intervals defined by minimal and current values of attributes of the current node..

Thus, if for a given node we have :

MinBreadth(Node)=1, MaxBreadth(Node)=10 and CurrentBreadth(Node)=2,

each child node of this node had, until now, to verify :

MinBreadth (ChildNode)=1,
MaxBreadth (ChildNode)=10 and
1≤CurrentBreadth (ChildNode)≤10.

With dynamic interval reduction, each child node must verify :

MinBreadth (ChildNode)=1,
MaxBreadth (ChildNode)=2 and
1≤CurrentBreadth (ChildNode)≤2.

In order to avoid unnecessary evaluation of node occurrences, dimensions of the current node occurrence must be propagated to descending nodes to reduce their variation intervals. In this manner, variation intervals of dimensions of the nodes of the hierarchical fact are computed dynamically, permitting to avoid several useless attempts.

These theoretical results are confirmed by the facts : implementation of this technique allows drastic reduction, with sampling refinement, of the number of useless attempts to evaluate branches. Time is linearly reduced with sampling refinement.

5.2 Evaluation of constraint sub-systems

Constraints generated during the generation process are of two types : **local** and **global** constraints. However, global constraints are in reality the expression of **placement** and **size inter-scene** properties.

We can organise global constraints into several constraint sets, where every set contains a reference to one node of the hierarchical fact and to its child nodes [7]. The main motivation of such an organisation is to be able to localise as quickly as possible the reason of a failure while evaluating global constraints. Indeed, it is useless to evaluate occurrences of descending nodes of a node if the set associated with this node and its child nodes is not verified.

Let us consider the current state of the hierarchical fact, during generation phase, shown in figure 4.

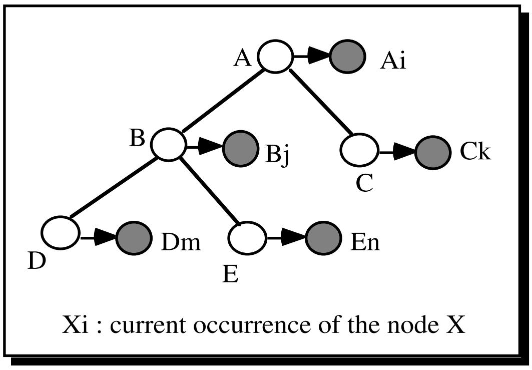

Figure 4 : current state of hierarchical fact

Suppose that, during evaluation, triple (Ai, Bj, Ck) does not verify the set of global constraints GC(A) associated with node A and its child nodes. Then, this triple does not verify the whole set of global constraints.

The new state of the hierarchical fact is obtained by computing the next occurrence of the deeper unsaturated node (D in figure 4). When a new evaluation of global constraints will occur, evaluation will fail again because the set of constraints GC(A) will not be verified as the triple (Ai, Bj, Ck) will remain the same. Thus, it is not necessary to evaluate occurrences of B, C or A for which constraints GC(A) are not verified.

To avoid unnecessary evaluations, we must avoid all occurrences Bj, Bj+1, ... of B for which constraints GC(A) are not verified. If the node B is saturated, i.e. if no occurrence of B after Bj verifies GC(A), we must apply the same process to node C. If all child nodes of the node A are saturated, the same process must be applied to the node A. If A is also saturated, evaluation of the set GC(A) will fail.

6. Some results

He have compared two generation methods, using respectively static and dynamic reduction of variation intervals of attributes. The following three models have been used for this work :

Model 1

```
Residence (x,p) -> CreateFact (x,p)
                   House (y,x)
                   Garage (z,x)
                   PastedLeft (y,z,x);
House (x,p)     -> CreateFact (x,p)
                   HigherThanLarge (x)
                   HigherThanDeep (x)
                   Walls (y,x)
                   Roof (z,x)
                   PutOn (z,y,x);
Garage (x,p) ->    CreateFact (x,p)
                   TopRounded (x,80);
Walls (x,p) ->     CreateFact (x,p);
Roof (x,p) ->      CreateFact (x,p)
                   TopRounded (x,70);
```

Model 2

```
Houses(x,p) ->  CreateFact (x,p)
                House1(x2,x)
                House2(x5,x)
                PastedLeft(x2,x5,x);
House1(x,p)     -> CreateFact (x,p)
                   HigherThanLarge (x)
                   HigherThanDeep (x)
                   Roof(x3,x)
                   Walls (x4,x)
                   PutOn (x3,x4,x);
House2(x,p)     -> CreateFact (x,p)
                   DeeperThanLarge(x)
                   Part1(x6,x)
                   Part2(x7,x)
                   PastedBehind(x6,x7,x);
Part1(x,p)      -> CreateFact (x,p)
                   AsDeepAsLarge(x)
                   TopRounded(x,80);
Part2(x,p)      -> CreateFact (x,p)
                   AsDeepAsLarge (x)
                   TopRounded (x,80);
Roof(x,p)       -> CreateFact (x,p)
                   Rounded(x,100);
Walls(x,p)      -> CreateFact (x,p);
```

Model 3

The same as model 1 but with a refined sampling of workspace.

The following two tables illustrate obtained results for static and dynamic reduction of variation intervals. For model 1 and model 2, given results are those obtained at the end of the generation process whereas for model 3 generation was put off after 2200 attempts.

Model	Number of candidate solutions	Number of solutions
Model 1	180	6
Model 2	700	6
Model 3	2200	0 (generation off)

Global constraints. Static reduction of variation intervals

Model	Number of candidate solutions	Number of solutions
Model 1	56	6
Model 2	165	6
Model 3	2200	15 (generation off)

Global constraints. Dynamic reduction of variation intervals

Three generated scene examples are shown in figure 5.

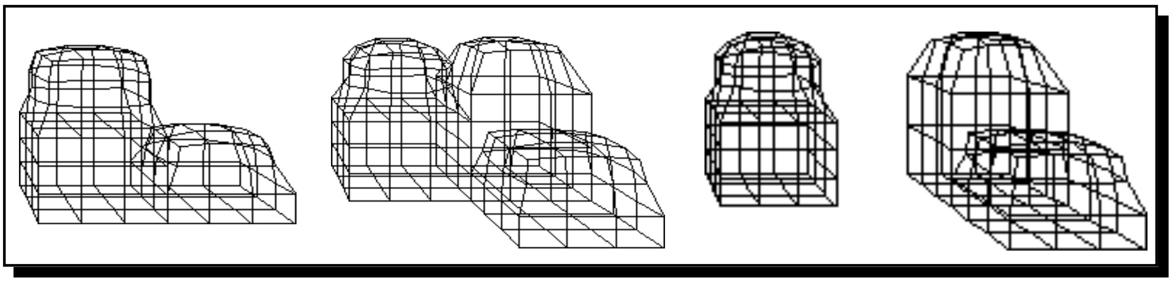

Figure 5 : examples of generated scenes

Comments :

- Generation from model 3 was stopped after a number of candidate solution processing.
- Efficiency of dynamic reduction of variation intervals increases with the complexity of the model. Thus the number of processed candidate scenes is divided by 3 for model 1 and by 4 for model 2.

7. New techniques to improve constraint evaluation

It is possible to improve the techniques studied in section 5 by extending the notion of local constraints.

The problem with global constraints is that they are evaluated too late, in case of failure, when the evaluation tree (hierarchical fact) has been totally traversed and a candidate solution found. Integration of global constraints into local constraints for each node, would permit early evaluation of candidate solutions, before a complete search of the hierarchical fact.

Is this integration possible ? What are global constraints ? They are the expression of **placement** and **size** inter-scene properties. Thus, we can consider global constraints as local ones, associated to each node, taking into account that these constraints concern also the child nodes of the current node. Now, global constraints disappear and we have only local constraints [8]. For example, consider the following rule :

```
House (x,p)   ->   CreateFact (x,p)
                   HigherThanLarge (x)
                   HigherThanDeep (x)
                   Walls (y,x)
                   Roof (z,x)
                   PutOn (z,y,x);
```

This rule produces two local constraints : Height (x)>Breadth (x), Height (x)>Depth (x) and one global one : Height (x) = Height (y) + Height (z).

With the new improvement technique, global constraints are suppressed and **all three**

constraints become local constraints associated with the node *House*..

This extension of local constraints increases efficiency of generation but it can become more efficient by changing the tree search strategy. Depth-first search must be replaced by a mixed strategy using depth-first and then breadth-first search. Indeed, search of all descending nodes of a node can be useless if its child nodes do not verify constraints which describe inter-scene properties.

The new search strategy is the following : Depth-first search, then breadth-first search for child nodes of a node. If the local constraints are verified, depth-first search is applied again. If breadth-first search fails, tree search is stopped. Figure 6 enumerates all steps of a complete mixed tree search.

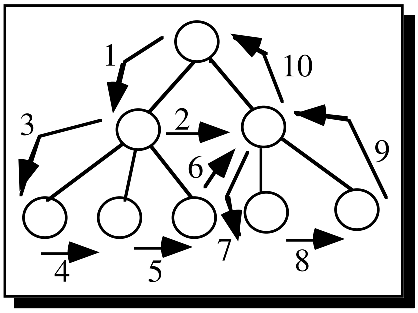

Figure 6 : mixed tree search

Figure 7 illustrates the gain obtained by a mixed search strategy : if the two nodes of level 2 do not verify inter-scene constraints, depth-first search will detect this impossibility only at the end of search, whereas breadth-first search will detect this at the beginning of search and it will cut all descending nodes.

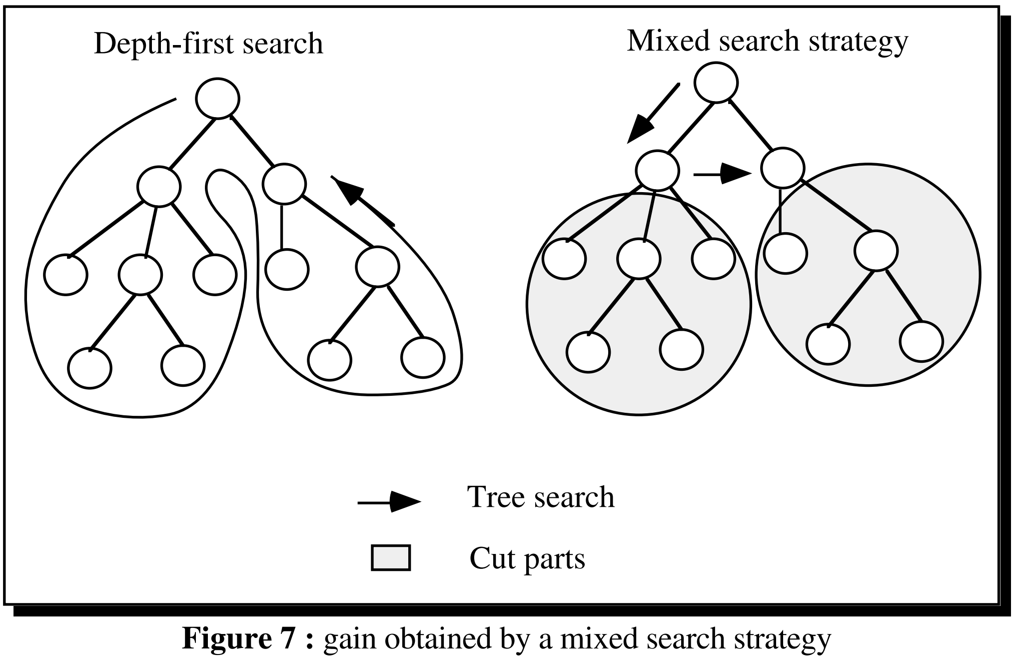

Figure 7 : gain obtained by a mixed search strategy

8. - Conclusion

Constraint evaluation in DMHD is a very interesting technique because it allows a general approach, not depending on degree of constraints.

We have presented, in this paper, improvements increasing the efficiency of constraint evaluation in DMHD.

The major part of these improvements has been implemented and the obtained results are very promising. The obtained gain increases with the complexity of the scene. The only improvement not yet implemented is the last one, based on the extension of local constraints. However, we do think this approach is very promising ; thus it will be integrated into the MultiFormes modeller of the MSI laboratory of the University of Limoges (France) [6].

Improvements presented in this paper permit to increase strongly constraint evaluation efficiency. Thus, even if some constraint resolution techniques remain more efficient than constraint evaluation, this technique has the additional advantage to be applicable to any kind of constraints.

References

[1] LEE Y. T., REQUICHA A. A. G., Algorithms for computing the volume and other integral properties of solids. I.Known methods and open issues. CACM'82 vol. 25, no 9.

[2] LEE Y. T., REQUICHA A. A. G., Algorithms for computing the volume and other integral properties of solids. II.A family of algorithms based on representation conversion and cellular approximation. CACM, vol. 25, no 9, september 1982.

[3] LUCAS M., MARTIN D., MARTIN P., PLEMENOS D., The ExploFormes project : a few steps to declarative modelling of forms. Conference AFCET-GROPLAN Informatique Géométrique et Graphique, Strasbourg, from 29th November to 1st December 1989. Published in BIGRE no 67, January 1990 (in French).

[4] MACULET R. Manhattan algebra, constraint satisfaction and constrained object placement. Conference GROSPLAN 92, Nantes, November 1992 (in French).

[5] PLEMENOS D., A contribution to study and development of scene modelling, generation and display techniques - The MultiFormes project. Professorial dissertation, Nantes, November 1991 (in French).

[6] PLEMENOS D., Declarative modelling by hierarchical decomposition. The actual state of the MultiFormes project. Conférence internationale GraphiCon'95, St Petersbourg (Russie), 3-7 July 1995.

[7] PLEMENOS D. TAMINE K. Generation by inference engine and constraint evaluation in a hierarchical modeller of three-dimensional scenes. AFIG'95, Marseille, 22-24 November 1995 (in French).

[8] PLEMENOS D. TAMINE K. Techniques to optimise constraint evaluation in declarative modelling by hierarchical decomposition. 3IA'96 conference, Limoges (France), 3-4 April 1996 (in French).

[9] FRULWIRTH T., HEROLD A, KUCHENHOFF V., LE PROVOST T., LIM P., MONFROY E., WALLACE M., Constraint Logic Programming. An Informal Introduction. Technical report ECRC-93-5, München, Germany, 1993.

[10] MACKWORTH A. K., Consistency in networks of relations. Artificial Intelligence, vol. 8, no 1, 1977.

[11] MOHR R., HENDERSON T. C., Arc and path consistency revisited. Artificial Intelligence,1986.

[12] VAN HENTENRYK P., DEVILLE Y., TENG C.M., A generic arc-consistency algorithm and its specializations. Artificial Intelligence, vol. 57, 1992.

[13] FREUDER E.C., WALLACE R.J., Partial constraint satisfaction. Artificial Intelligence, vol. 58, 1992.

[14] WALLACE R.J., FREUDER E.C., Comparing constraint satisfaction and Davis-Putman algorithms for the maximal satisfability problem. D.S. Johnson and M.A. Trick editors : Second DIMACS Implementation Challenge, American Mathematical Society, 1995.