Constraint satisfaction problem solved by minimal changes method

Simek Pavel, Czech Technical University, Faculty of Electrical Engineering, Dept. of Computer Science and Engineering, Karlovo namesti 13, 12135 Prague 2, Czech Republic
email:simekp@cslab.felk.cvut.cz
Slavik Pavel, Czech Technical University, Faculty of Electrical Engineering, Dept. of Computer Science and Engineering, Karlovo namesti 13, 12135 Prague 2, Czech Republic
email:slavik@cs.felk.cvut.cz

Abstract

Direct manipulation has become a very popular interaction technique in GUI's in the last few years. It allows the user to manipulate graphical objects in natural way. However, in complicated cases the user must perform complex manipulations that burden the user with many details. Therefore it is necessary to develop new techniques that are more user-friendly. One of many possible approaches is the use of constraints. By means of constraints it is possible to reduce the extent of interaction from the user as the constraints introduced express relations among objects that can be manipulated.

This paper describes a new technique that allows for interacting with graphical objects in an effective way. Moreover the implemented system contains features that visualize constraints. This supports the user during interaction as he/she can clearly see the mutual links between objects.

Keywords:

Constraints, Geometric modeling, Constraint Solving, Graphical Interaction, Graphical User Interfaces

1. Problem description

The constraints usually mirror some physical aspects and some relations that exist between corresponding real objects. In such a way they describe some implicit knowledge that can be used during interaction. Because the constraints (relations among objects) should be satisfied at any moment they are permanently evaluated during interaction. For example that means: when one object has been moved the dependent object (the dependency is described by constraints) moves "automatically" in order to satisfy the constraints. In such a way the user interaction is limited to the manipulation with the first object only [Sla94].

There exist several ways to express constraints. In our case we will express the constraints by means of equations.

Example:

There are two objects: O1 and O2, with reference points [x1,y1] and [x2, y2] respectively. The constraint defining the constant distance D can have the following form:

x2=x1+xD; y2=y1+yD;

where xD and yD are projections of the distance D on both axes. These constraints allow us to define unambiguously the relative position between both objects.

The constraints can be of a much more general nature than described above. They can express relations such as parallelity, perpendicularity, immediate contact of two objects and constraints that are not of a geometric nature (like Ohm's law satisfied in the displayed circuit) etc.

A good example of constraints that express relations between graphical objects on the screen is the example given in the editor ThingLab [BorDui86]. By means of graphical objects we can define through direct manipulation constraints that link two or more graphical objects together. The following figure represents an example where conversion between Celsius and Fahrenheit degrees is described.

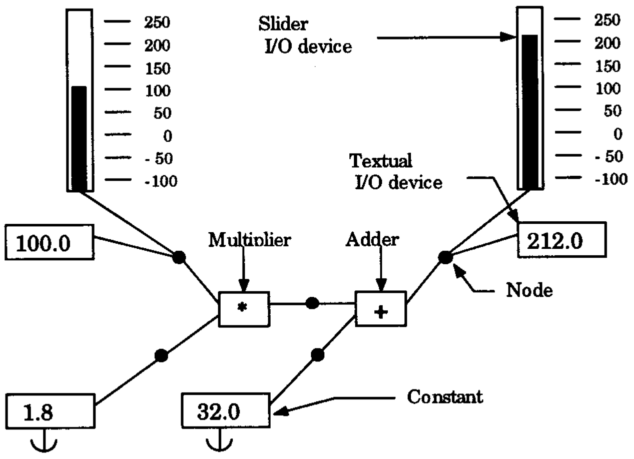


Figure 1. A Celsius - Fahrenheit converter.

Another application of constraints is the object design in parametric CAD systems. The dimensions that define the general shape of an object are considered parameters. The "less important" dimensions can be derived from these parameters. This results in the situation where a slight change of one or two parameters can lead to a substantial change of the whole object. Here we are faced again with the situation when not very extensive user interaction leads in many cases to a very extensive change of the object appearance.

2. Constraint satisfaction

To describe constraint problems in general, we discuss a finite set of variables, finite set of constraints and finite or infinite domain for variables. Satisfaction means to find assignment for variables from their domain with respect to constraints.

The known methods for the constraint satisfaction can be divided according to various criteria [Doh94]. One of the frequently used criteria is the type of values that the variables in constraints can be assigned to. Either we work with a *finite* set of values (e.g. set of colors that can be used as values in expressions) or an *infinite* one (e.g. set of real numbers).

In the first case (*finite* domain of values) several methods have been developed:

- reduction techniques ... The given problem is transformed to a simpler problem that is equivalent to the former one.

- search techniques ... Subsequent assignment of values to variables to satisfy constraints given.
- chronological backtracking ... One value is assigned to one variable at a time. The assignment is checked against previously assigned values and if it is rejected then another value is taken. If there are no available values for the variable then the previously assigned variable becomes the new value
- forward backtracking ... Value assigned to variable is propagated by constraint to other variables to ensure that all yet undetermined variables have at least one value in their domain compatible with assigned values .
- synthetic techniques ... All possible solutions to the problem should be found.

In the second case (*infinite* domain of values) several methods are used:

- numeric techniques ... An approximate solution is found. Initial conditions are defined and iteration process is performed until the iteration criteria are not satisfied. Examples of these techniques are relaxation and the Newton-Raphson iteration (Here the Jacobian matrix is utilized to compute new values for the variable). The Jacobian contains the partial derivatives of the equation functions with respect to the variables in the equations.
- approximate methods ... Toleration value sets are defined for variables used.

Both given methods (numeric techniques and approximate methods) should converge. Unfortunately this is not always the case. In many cases (providing that the convergence is guaranteed) the computation is very time consuming.

In practice we often use constraint-based techniques for the solution of geometric constraints. The geometric nature of the problem allows us to apply methods based on the use of special geometric features. These features can considerably speed up the process of the constraints satisfaction. This approach is used in practice mainly in the field of CAD where parametric design and the use of variational models is applied. Variational designed refers to using simultaneous equations to define constraints on the model variables.

The use of the geometric based approach is not a new one. The first application of this method dates back to the beginning of the 60's [Sut63]. The extension of the approach used resulted in the design and implementation of the ThingLab system [BorDui86]. There are several possible approaches to satisfy a set of constraints (using the geometry based methods). A set of constraints can be solved either serially (one by one) or in parallel .

3. Our approach

One of the crucial problems when solving a set of constraints is the situation when there are more solutions available. There are different methods on how to select one "proper" solution out of a set of solutions. The strategies used are often based on some specific topics that are characteristic for the application investigated. In our case we tried to follow the laws of physics that are applicable in many applications of this kind (where constraints can be applied). The key point is the fact that in the general case when more objects (and constraints among them) are involved we can express the mutual relations (constraints) among objects by means of a network. The change of an input variable is distributed along this network in accordance with the constraints specified. We can control the distribution of the changes in such a way that the change of the state of an object should be minimal.

The minimal change method has two main rules:

1. *Change of initial variable should produce as little as possible other changes through the network (changes should have local character).*
2. *When it's necessary, the change of variable is minimal.*

This approach requires redefining the solving process. We are solving a given problem with respect to an already known correct state and with respect to the change of one variable at a time. As an example we can give the typical change of a graphical object using a graphical cursor.

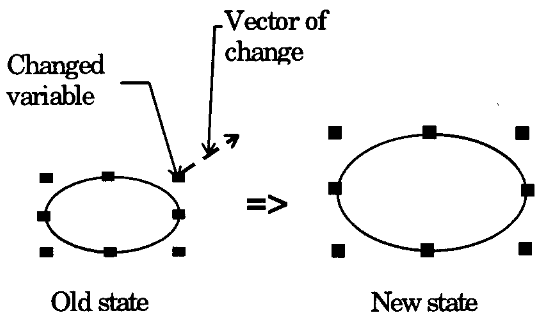
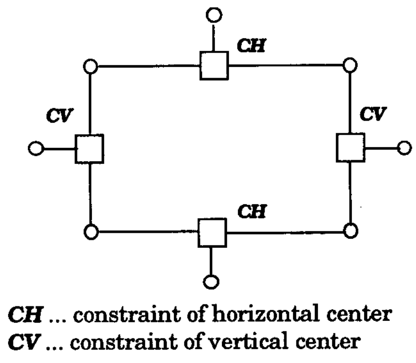


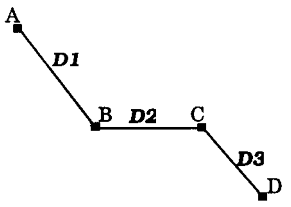
Figure 2. Graphic modification.


CH
CV
CV
CH
CH ... constraint of horizontal center
CV ... constraint of vertical center

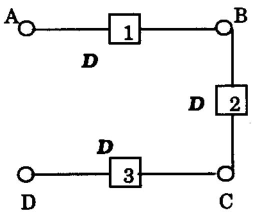
Points are represented by circles and constraints are represented by squares

One very important feature when working with constraints is the possibility to define "control points" that are mutually linked with constraints. Manipulation with one control point results in the change of the global appearance of the object. In the previous example (ellipse) eight control points are defined. It is possible to express the constraints used in the following way: Every three points are constrained together by a central constraint in the vertical direction and horizontal direction. Together all 8 control points define the shape of a geometric object. In general the control points control the appearance of objects (either a ellipse or a rectangle or anything else).

A general case when more objects than two are involved can be seen in the following figure. Let us be given three line segments D1, D2 and D3. They are linked by means of the length constraints, every line segment has an assigned fixed length. The endpoints of two line segments are linked together through the same position constraint (they are identical).



3 line segments with control points A..D



Graph based representation .

Figure 3. Preview example .

Let us change the position of point A to a new location A' and we describe how our algorithm solves this situation. A change of variable A (coordinates of point) is propagated through constraint D1 to point B. From our first rule we try to stop propagating at this point. We get equations for variable B using all constraints connected with point B. These equations describe the set of possible solutions. The above situation is described in figure 4, where equations for point B are represented by circles C1 and C2. Points B'1 and B'2 are in the set of available solutions.

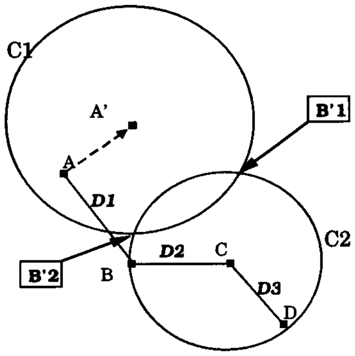


Figure 4. First step in solving change.

From figure 4 it is obvious that there exist two solutions to the given problem that satisfy the constraints (B'1 and B'2). According to our strategy (the minimal change) the position B'2 is chosen. It is obvious that the changes caused by the input change have local character that speed up considerably the constraint evaluation.

We have described the idea of our algorithm for a non-cycled constraint network. In the following example we show how to handle cycles in a constraint network.Cycles are treated in the case when e g. position of a point can be determinated by two different sequences of constraints that are sequentially evaluated.

Which sequence (path in the network) will be selected is dependent on the way how constraints are defined. In general the sequence used consists of constraints that have higher priority than the constraints in other constraint sequences. The priority can be either assigned by the user explicity or in specific cases can be assigned on the first come first served base (in accordance with order the constraints were constructed).

In general we can say that starting point of path is always the variable to which the initial change would be performed. The initial change is produced by graphical cursor or equal graphical pointing device. When the new value is assigned to the point all connected constraints are checked and only when they are not satisfied then the applied change is recursively propagated to connected points. The order in which connected constraints are propagated is given by already described pritority system.

In this example (Figure 5) we can choose two paths to go through the constraint network. We have two different priority ordered connected constraints. Constraint network consists of just one single cycle. When we initially change position of one point we can continue propagating in this example either clockwise or anti-clockwise. These two paths are described in table 1 (the change is first distributed to the point B) and table 2 (the change is first distributed to the point C). The result for table 1 resp. table 2 is shown on figure 5b) resp. 5c).

The table should be read in the following way (see the next page):

The line 1 is the first step in iteration process. We are now dealing with point A. The position of the point A is not correct, because there is the initial change (we must proceed with the initial change). The initial change has been added to the point A position. After changing the point A position, we propagate this change to points B and then to C. The position change of the point A propagated through constraint fl_1 is added to point B position (the same for point C). We changed positions of points B, C so we must check them. When we perform back step (in column *Next points*) we are investigating last changed point in constraint network and in this point we continue with the set of connected constraitns ordered by priority.

Point has the correct position in the case when all constraints connected with point are satisfied.

Explanation to tables:

Notation:

In ... Initial change

fl ... Fixed length constraint.

Δ(point) ... Change of point.

cons1 ∩ cons2 ... result from solving equations given by constraint1 and constraint2

Table 1 (order is B C) :

Step	*Point*	*Is correct ?*	*Change*	*Connected changes*	*Next points*
1	A	No	A=A+In	B=B+fl_1(Δ(A)), C=C+fl_2(Δ(A))	B,C
2	B	No	B= fl_1∩fl_3		back
3	C	Yes			back
4					back

Table 2(order is C B) :

Step	*Point*	*Is correct ?*	*Change*	*Connected changes*	*Next points*
1	A	No	A=A+In	B=B+fl_1(Δ(A)), C=C+fl_2(Δ(A))	C,B
2	C	No	C= fl_2∩fl_3		back
3	B	Yes			back
4					back

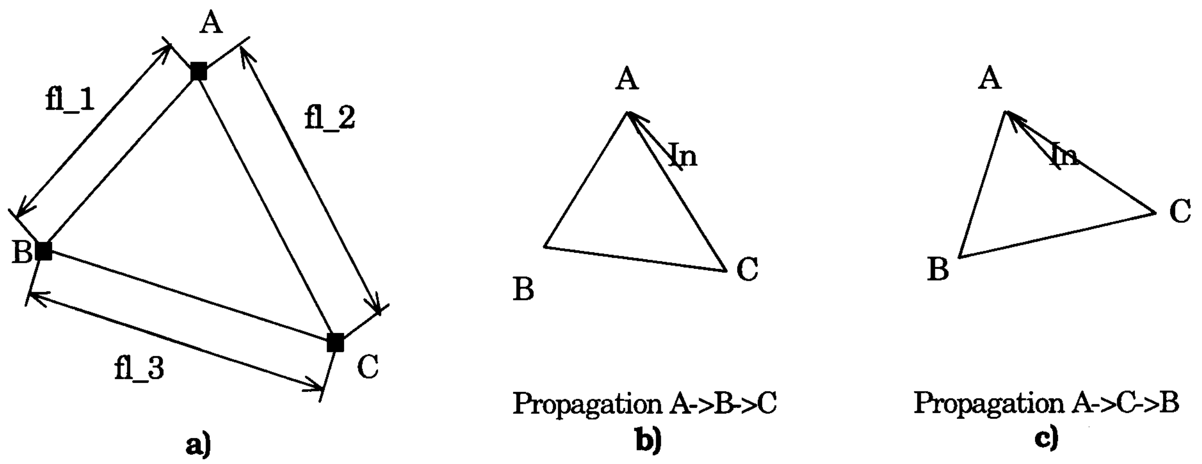


Figure 5. Example for solving cycles.

From the example it is obvious that the result is dependent on which path we propagate changes through the constraint network. The path is controlled by priority assigned to each constraint. Constraints with higher priority are calculated first.

The sequence of constraints is given by the priority of constraints applied. In case when two or more constraints have the same pritority, the first constructed constraint will be used.

Traditional methods solve the problem when the process of constraints satisfaction reached the point where no solution was available by means of a constraint deletion. The reduced constraints set was investigated in order to find a proper solution of this new constraint set. A disadvantage of this approach was that some important links could have been omitted and the solution obtained was not the proper one in relation to the original constraint set. Our approach is to transform the initial change to ensure that this change is acceptable for the whole constraint network. This approach determines, that position of graphic cursor and position of moving control point are not in the same position. In such a case we have to determine how to transform the change of cursor position to change of control point.

The initial change = New_position_graphic_cursor - Old_position_graphic cursor.

Transformation of initial change is made by means of an iteration process. This process is shown in the next example.

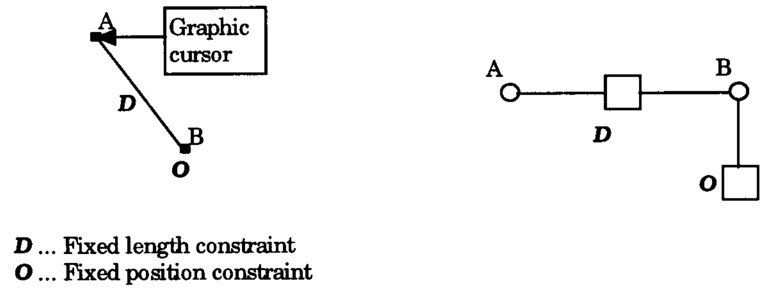

A
Graphic cursor
D
B
O
A
B
D
O
D ... Fixed length constraint
O ... Fixed position constraint

Figure 6. Example for backtracking step.

Let's have the following line segment AB and constraints applied on it to form figure 6.

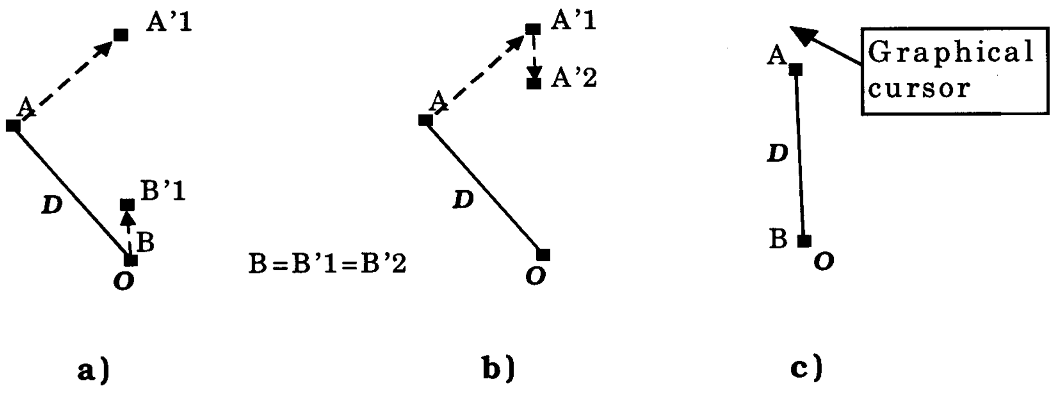


Figure 7. Backtracking process.

Line segment AB has assigned fixed length constraint and fixed position constraint is applied to point B. In the first step we change the position of point A with help of a graphical cursor to new position A'1. This change is propagated to point B and point B temporary changed position to B'1 (Figure 7 a). Now we go to point B and we are trying to solve all constraints connected to point B. It's obvious that it is impossible and position of point B is set to the original position according to fixed position constraint. Then a backtrack is done to the initial point A. From point B the position of point A is computed in order to have the minimal change from actual position (in this case it is A'1). We reached the initial point and we are starting new propagation from point A’2. This propagation will be done well and position A’2 is correct (Figure 7 b). Result is shown on figure 7c, where we can see the difference between the position of the graphic cursor and position of point A.

The number of backtrack steps is limited. If there exists more than one path to go through the constraint graph, then these paths are cycled in an iteration process. This gives a better chance for convergence.

This way of interaction means that the cursor is placed in a position that roughly determines the new position.

4. Implementation

The implementation of the given approach was the system VCE (Visual Constraint Editor). This system is designed for use under Microsoft Windows 3.xx. There are some built-in constraints and moreover there are means available to define new constraints.

The visualized constraints allow the user to understand better the relations between single objects.

5. Constraint visualization

When working with graphical objects that are linked with constraints the user can lose the orientation in the complex relations and thus his/her interaction can be rather unsystematic. The user should be given full support from the system in order to perform deterministic actions only. It is advisable to visualize either each single constraint or a selected set of constraints. In such a case the user sees the relations among objects and the behavior of the whole system is much clearer to the user. It was necessary to develop a formalism that allows us to express single types of constraints used to link objects together.

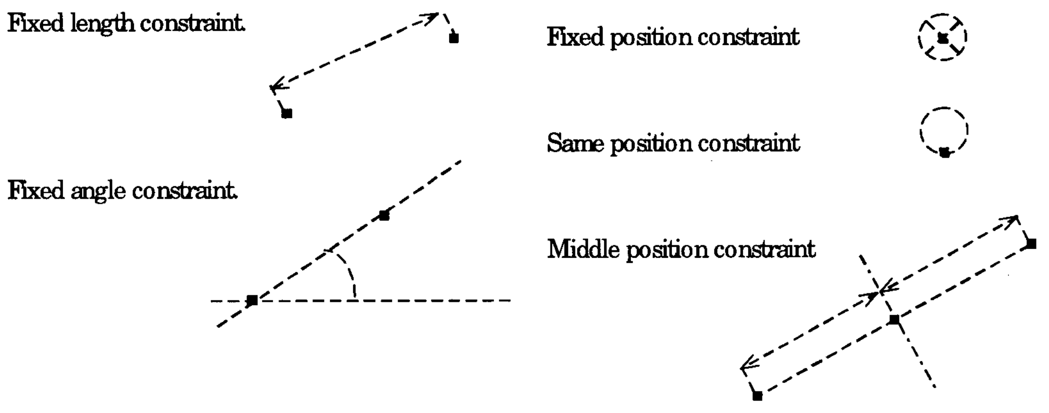


Figure 8. Visualization symbols of constraints.

During object manipulation the user sees the constraints used and it is clearer to him/her why objects behave in that way. As the figure can become rather clumsy when a large number of constraints are used, it is possible to “turn off“ (or “turn on“) the display of constraints.

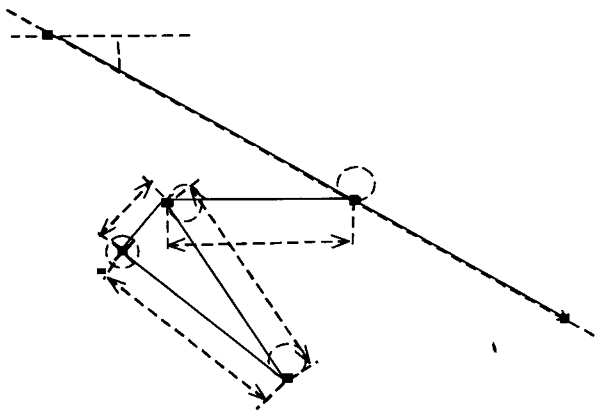


Figure 9. Complex example for visualization constraints.

6. Experience gained from the work with the VCE system

Experimenting with the VCE system results in the next characteristics of the described algorithm. An algorithm can easily handle all situations without cycles in the constraint network. The calculated result after a small practice is predictable and the user can easily change objects to get correct results. However in some situations, when the number of constraints are applied to a small number of variables, the algorithm doesn't give any result at all because the iteration process fails.

The visual presentation is natural and can be used for simulating non-complicated mechanisms.

The result is also dependent on the size of the initial change. When the initial change is large then the connection between old state (before the constraint network was passed through) and

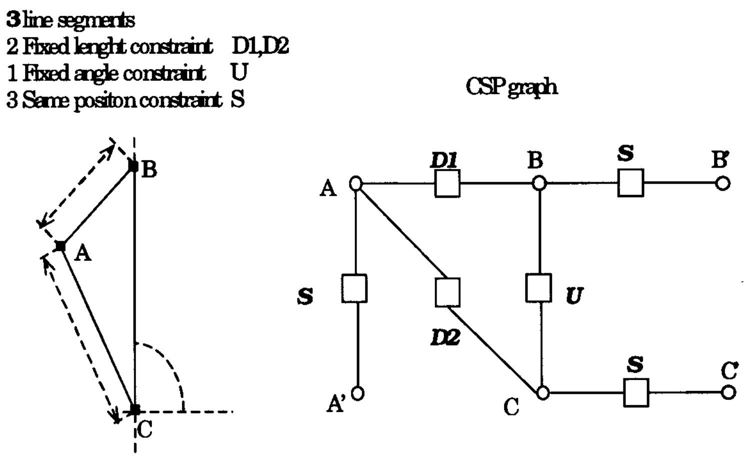

3 line segments
2 Fixed lenght constraint D1,D2
1 Fixed angle constraint U
3 Same positon constraint S
CSP graph
A
B
C
D1
B
S
B'
A
S
U
D2
S
C
A'
C

Figure 10. Example for large initial change.

new state (after satisfying all constraints in constraint network) can be rather weak and that produces unexpected results. In the Windows system where mouse messages are generated dependent on system speed, the speed of the mouse moved by the user can generate different results. This experience we show on the next example. Let's have three line segments (Figure 10). There are two fixed length constraints and one fixed angle constraint. When we change the graphic cursor to a new position together with moving point A, then the user can expect results like figure 11A. This result is really reached only if the given change is applied to small changes and these small changes are applied continuously. Now we try to proceed with one large change. The change of B is propagated to points A and C, then in point A we calculate a new position (figure 11B, circles C1 and C2 displays equations from fixed length constraints). In this point there are two available solutions A'1 and A'2. Our minimal strategy chooses point A'2 because it's closer to original point A. The given result (points A'2, B', C') obviously differs from the expected result.

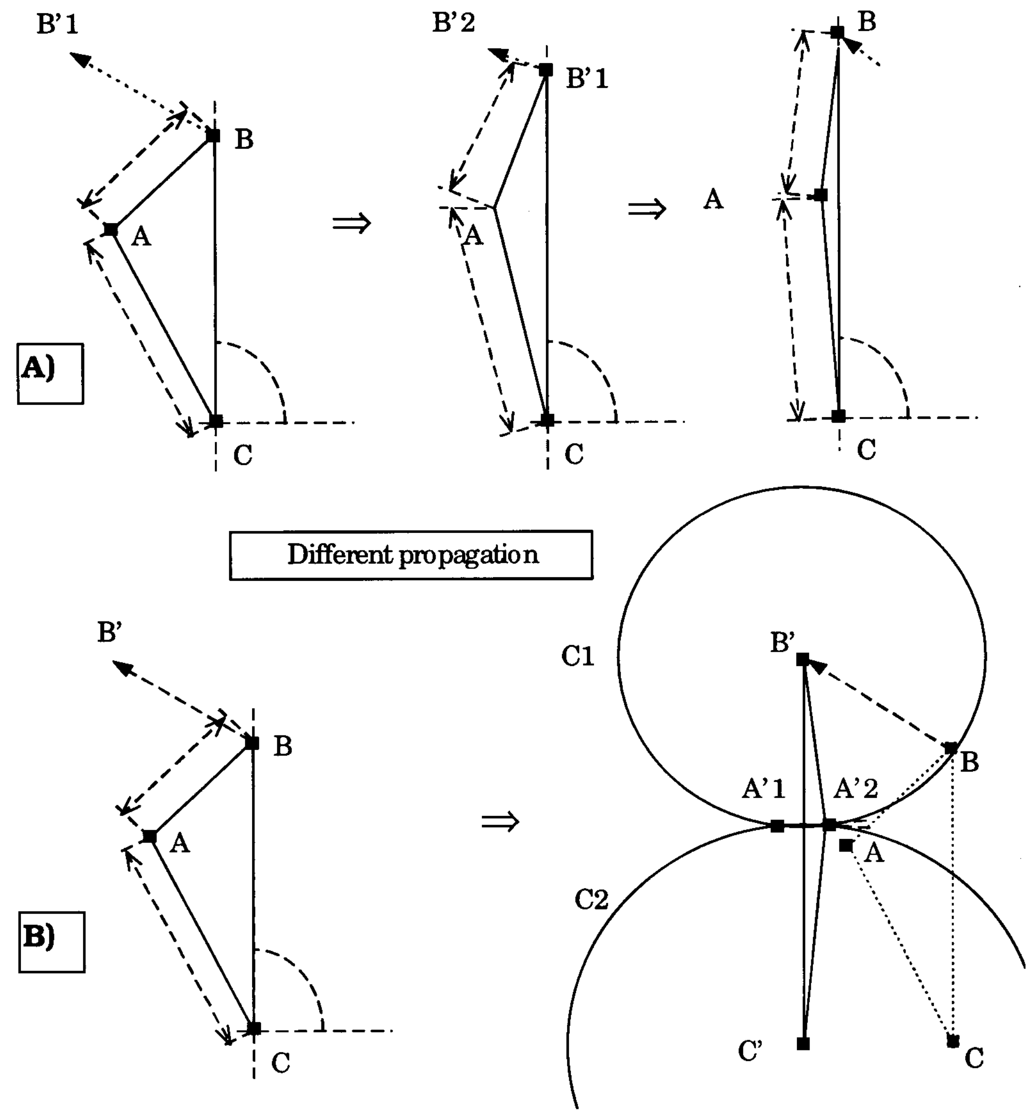


Figure 11. Results after small and large initial change.

The above example is an interesting contribution to the ergonomy of human computer interaction. This situation has not been present in the case of simple interaction of common interaction systems. It is possible to see that complex interaction methods bring problems that have not been met before.

7. Conclusion and future work

The idea of solving a constraint network by minimal changes has been proven to be an acceptable way, of how to handle constraints in 2D space. It's necessary to increase the set of available constraints. There are other techniques that can be useful with this algorithm. For example we can do pre-processing of the network graph to find the best way for walking through the constraint graph to avoid problems with divergency in the iteration process. Also it's necessary to develop rules for adding constraints together.

The described system allows the visualization of constraints that allows on-line "debugging" of the manipulation with the graphical object. The visualization process allows better understanding of the nature of manipulation.

Bibliography

[BorDui86] Borning Alan, Duisberg Robert: Constraint-Based Tools for Building user interfaces, ACM Transaction on Graphics (1986), Volume 5, number 4, pp 345-373

[Doh94] Dohmen Maurice: Constraint techniques in interactive feature modelling, TU Delft REPORT 94-16, Delft 1994

[Sla94] Slavik Pavel: Accelerating Interaction in 3D Space by means of Constraints, U. Dietrich, B. Kehrer, G. Vatterrott: CA-Integration in Theorie und Praxis, Springer 1995, pp. 137-151

[Sut63] Sutherland: Sketchpad - a man-machine graphical communication system, Proceedings Spring Joint Computer Conference 1963, 329-346