Title
Solution plans and interactive problem solving

Author
Howden, William E.

Publication Date
1974

Peer reviewed

SOLUTION PLANS AND INTERACTIVE PROBLEM SOLVING

by

William E. Howden

TECHNICAL REPORT #53 - AUGUST 1974

Abstract

Solution Plans and Interactive Problem Solving

William E. Howden
University of California, Irvine

The concept of a "solution plan" is used to characterize the structure of interactive systems in which the user guides the solution process. A formalism for describing and analyzing solution plan structures is presented. The formalism can be used to define the role of the user in an interactive system. The solution plan approach to interaction is particularly applicable to problems which have an obvious graphical representation. An interactive graphics system for solving a class of routing problems is described. Suggestions for the application of the approach to other problems are included.

1. Interactive Problem Solving.

Problem solving activities which involve the use of computers can be thought of as cooperative processes in which a man and a machine combine their problem solving capabilities. In an interactive problem solving system the two agencies influence eath others' behavior during the course of a solution process. The fundamental design problem in the construction of an interactive system is the design of a problem solving structure in which dynamic cooperation can successfully take place.

In most discussions of interactive problem solving structure the user is described as having the responsibility for the overall control or planning of the solution process [1]. He is described as "charting courses", "formulating problems", and "guiding solutions" [2]. The role of the computer is described as that of carrying out defined subtasks of the process. It is described as solving "small" subproblems or carrying out information retrieval and numeric computation tasks. The user supplies the "creative heuristic power" and the computer the "algorithmic power" of the solution process [3].

This paper uses the concept of a "solution plan" to characterize the structure of interactive systems in which the user controls and guides the solution process. In the solution plan approach to interaction the user "plans" a solution and the computer carries out the plan. Both the user and the computer are able to carry out the parts of the problem solving process for which they are best suited. The approach is particularly useful for certain classes of problems for which a graphics display device can be used as the interactive communication medium. A particular interactive graphics system is described whose design is based on the solution plan approach.

The notion of a solution plan can be formalized for an important class of problems. The formalism can be used to classify and analyze different problems and problem solving structures. The paper includes a description of the solution plan formalism and its application to a particular class of problems.

2. Solution Representations.

The term "problem solving" usually refers to the artificial intelligence approach to the solution of a number of classical problems. This class of problems includes both toy problems and games as well as problems from operations research. Several examples of the more important problems are:

(i) Sofa Problem [4]. In the Sofa Problem it is necessary to determine if an irregular three-dimensional object (sofa) can be moved from one point to another inside a three-dimensional structure (house). The problem is important in the design and assembly of structures such as aeroplanes and submarines.

(ii) Travelling Salesman Problem [5]. In this problem a number of "cities" $c_1, c_2, \ldots, c_n$ must be arranged in a sequence $c_{i_1}, c_{i_2}, \ldots, c_{i_n}$ so that the sum of the distances between adjacent cities is minimal or "nearly minimal". This is one of several important routing problems.

(iii) Machine Shop Scheduling Problem [6]. In the classical Machine Shop Scheduling Problem a set of tasks must be scheduled on a set of machines. Each task is assigned to one machine and no machine may process more than one task at the same time. The problem is to construct a minimal makespan schedule. The makespan of a schedule is the finishing time of the last task to be completed. The assignment is usually constrained by a number of precedence relations between tasks. A precedence relation between two tasks prohibits the scheduling of one of the tasks before the scheduled completion time of the other task.

A characteristic feature of these problems is the easy representation of their solutions as paths in graphs, permutations, arrays or some other combinatorial structure. Solutions to the Sofa Problem can be represented as paths in a graph. The nodes in the graph are "positionings" of the sofa in the house. The arcs are "unit" transformations of the sofa from one positioning to an adjacent positioning. Solutions to the Travelling Salesman Problem (TSP) can be represented as permutations of city names. Solutions to the Machine Shop Scheduling Problem can be represented as arrays in which each column is associated with a machine and each row a time slice in some time scale. An entry in the array corresponds to the assignment of a task to a particular machine for a particular unit of time.

The typical problem solving process operates by generating or transforming a solution representation. Howden's Sofa Problem program [7] uses heuristic search to generate a path through a graph. Lin's Travelling Salesman Problem program constructs a sequence of candidate solutions by interchanging the elements of a permutation. Gere's heuristic program [8] generates a sequence of schedule arrays. The approach to interactive problem solving which is described in this paper is applicable to problems whose solution representations are one of several well defined combinatorial structures and to solution processes which can be described in terms of the generation and transformation of these structures.

3. Solution Plans.

Different kinds of plans can be formulated during a solution process. A procedural plan consists of a proposed sequence of problem solving processes to be carried out during problem solution. A solution plan is a sketch of a proposed solution. Most uses of the word "plan" in the problem solving literature refer to solution plans. Newell [9] calls a plan a "goal structure"; Doran [10] a "solution outline which will provide intermediate subgoals"; Good [11] "a sequence of subgoal descriptions"; and Slagle [12] a "rough outline of a possible solution". The more recent problem solving literature also refers to procedural plans. Hewett [13] has designed a language in which it is easy to construct simple procedural plans.

The structure of an interactive problem solving process in which the user plans the generation and transformation of solutions can be characterized as that in which he creates and manipulates solution plans. He directs the computer to carry out a plan by directing it to solve subproblems defined by the plan. Different kinds of interactive problem solving processes correspond to different kinds of solution plans. In the Sofa Problem the solution representation is a path through a graph. The user can impose a plan on a sofa problem solution process by choosing a sequence of intermediate solution path nodes. The intermediate nodes correspond to intermediate positionings of the sofa between the starting and goal positionings. It is easy to construct an automatic sofa problem solver for problems which do not have houses containing tricky passage ways and blind alleys. A plan can be used to replace a complicated sofa problem by a sequence of smaller subproblems which the computer can solve unassisted. In the Travelling Salesman Problem the solution representation is a permutation. The user can construct a travelling salesman problem plan by arranging the cities in subgroups and then imposing an ordering on the subgroups. It is easy to solve travelling salesman problems with less than 10 or 15 cities. A plan can be used to replace a larger problem by a sequence of smaller subproblems. The solutions to the smaller subproblems can be joined together in the order indicated by the plan to form a problem solution.

A solution plan interactive system for the TSP is described in Section 5. In the following section a formalism called a solution grammar is used to formally define the concept of a solution plan.

4. Solution Representation Grammars and Solution Plans

(a) Solution Grammars Solution grammars are structural definitions of classes of solution representations. The solution grammar formalism which is described in this section is derived from Narasimhan's [14] picture grammar formalism.

A solution grammar consists of attributes, primitive objects, compound objects, relations and composition rules. Each object has a specified number of attributes. Relations between objects are defined in terms of these attributes. Attribute and relational variables can be used to stand for undetermined attributes and relations. Composition rules are of the form:

$$t_1(\alpha) \rightarrow r(t_2(\beta), t_3(\gamma)).$$

This rule declares that t_1, with attributes α, is composed of t_2 and t_3. t_2 has the attributes β and t_3 the attributes γ. The relation r holds between the attributes β and γ. t_2 and t_3 are the <u>constituents</u> of t_1. If β and γ contain attribute variables then r is a constraint on the attribute values which these variables may assume. Subscripts are used to distinguish between objects of the same syntactic type. The above rule specifies that a t can be formed from two other t's when a certain condition r holds. The compound objects of a grammar are those objects which can be formed from primitive and compound objects by applications of the composition rules in the grammar.

The set of all TSP solution permutations can be described by the following grammar.

Attributes. Each object has a single attribute. The attribute is a set of ordered pairs (c,i) where c is the name of a city and i a position for the city in a tour.

Primitive Objects. The set of primitive objects includes all single city tours $t(\{(c,i)\})$

Compound Objects. The set of compound objects includes all tours $t(\{(c_1,i_1),\ldots,(c_n,i_n)\})$ which can be formed using the composition rules.

Relations. D (disjoint) is a relation between two objects. It specifies that the cities in one object are distinct from the cities in the other.

Composition Rules.

$$t(\{(a_1,i_1),\ldots,(a_k,i_k),(b_1,j_1+k),\ldots,(b_m,j_m+n)\})$$
$$\rightarrow D(t_a(\{(a_1,i_1),\ldots,(a_k,i_k)\}),t_b(\{(b_1,j_1),\ldots,(b_m,j_m)\})).$$

This particular simple permutation grammar joins together two disjoint city subpermutations by putting all the cities in the second subpermutation "after" those of the first. Other grammars can be defined which describe more complicated ways of joining together subpermutations. Solution grammars can be used, for example, to describe particular patterns of solutions in terms of component subpatterns.

(b) *Solution Plans*. Suppose S is a set of solution representations for some problem. If S is defined by a solution grammar then each element of S will have one or more *phrase trees*. A phrase tree is a diagram of an object's constituent structure. It describes the object as a hierarchy of compositions of primitive and/or compound objects. Figure 1 contains a phrase tree for the permutation $t(\{(c_4,1),(c_1,2),(c_3,3),(c_2,4)\})$.

Figure 1

Solution grammar phrase trees contain terminal, non-terminal and relational nodes. The non-terminal nodes are compound objects and the terminal nodes primitive objects. The relational nodes denote the composition of compound objects from constituent compound and/or primitive objects.

In a complete phrase tree there are no attribute or relational variables and all paths terminate with primitive objects. An incomplete phrase tree is either part of a phrase tree or contains unbound variables. The variables may be either relational or attribute variables. An incomplete phrase tree is a structured rough outline of a solution representation. Different types of incomplete phrase trees correspond to different types of solution plans.

Definition A solution plan is an incomplete solution grammar phrase tree.

Figure 2 contains a TSP solution plan for the solution permutation in Figure 1. It describes a plan in which the cities have been grouped into two subproblems. The variables x_i in the compound objects in the tree stand for undetermined subtour city positions. The complete phrase tree in Figure 1 can be thought of as an instantiation of the plan in Figure 2.

Figure 2

The solution grammar formalism is general and can be used to define solution plan structures for any class of combinatorial solution representations. It can be used, for example, to define patterns of sofa problem paths in terms of subpatterns of subpaths or patterns of theorem proofs in terms of subproofs.

5. Interactive System for the Travelling Salesman Problem.

The solution plan approach was used to design the interactive travelling salesman problem system which is described in [15]. The system can be used to solve Euclidian travelling salesman problems. The user interacts with the system through the creation and transformation of solution plans. The solution plans constructed by the user are incomplete permutation phrase trees.

The system was implemented using an IMLAC display and a RAND tablet. The user of the system creates solution plans by using the display and the tablet to draw polygons around groups of cities. The graphics routines automatically create internal solution plan data structures. The user can create, transform or delete parts of solution plans by typing in solution plan commands. At any time he can request that the computer solve a subproblem corresponding to a subgroup of cities. The system contains several automatic subproblem solvers. Each of the subproblem solvers is useful for a particular class of subproblems.

Groups of cities can be thought of as "super-cities". In the TSP system the user can request that the computer solve a super-city subproblem. Synthesis commands can be used for joining together subproblem solutions in the order indicated by a supercity solution.

(Howden [15]) describes several experiments with the system.

In one experiment solutions were constructed for the South American Travelling Salesman Problem (Figure 3). This problem contains 68 cities and is very expensive to solve using an automatic travelling salesman problem program. Two different kinds of solution plans were constructed during the interactive solution of this problem. The first kind of plan consisted of a single circular tour of subproblems (Figure 3). Figure 3 contains the solutions to several of the subproblems and the solution to the "super-city subproblem". The second kind of plan consisted of an inner circular tour and an outer circular tour (Figure 4). In Figure 4 the subproblems have been solved. In the experiment different plans, subproblem solutions and subproblem syntheses were created and transformed. Figure 5 contains the best solution that was discovered.

Figure 3

Figure 4

Figure 5

6. Man-Machine Tradeoffs

There are several possible goals in the design of an interactive problem solving system. A primary goal is to build a system that can be used to solve larger problems or to solve problems more efficiently than is possible with a completely automatic system. Another goal is to provide a problem solving tool which allows a user to construct solutions which, although suboptimal or incomplete, are still "semantically meaningful." In a well designed interactive system the user should be able to direct the computer to optimize or solve those parts of a problem which he thinks are most important.

The solution plan approach to interaction can be used to produce a more efficient problem solving tool in those problem solving situations in which the added cost of the user's "planning time" is compensated for by a decrease in CPU computation time or by an increase in the range of problems which can be solved. In some cases it is clear that the solution plan approach allows the solution of problems whose solution by automatic procedures would be impossible or prohibitively expensive. It is easy to construct examples of travelling salesman or sofa problems which are too large or too complicated to solve automatically yet which can be solved using solution plan interaction. In other cases the added efficiency is dependent on the tradeoff between the costs of user and computer time. In the TSP experiment described above, the user was able to replace a 68 city problem with 6 or 7 smaller subproblems. The solutions to the subproblems required 85 minutes "user time" and 373 computer CPU seconds. The application of the Lin procedure [5] to the whole problem produced a solution which was less optimal than the interactive solution and required 1296 seconds CPU time. If one hour user time is equated with 100 CPU seconds then the interactive solver produced a better solution at less than half the cost. Similar results were achieved in other experiments, one of them with a problem having no obvious solution plan structure.

The solution plan approach emphasizes the construction of "meaningful" solutions. The user in a solution plan system is expected to guide the solution process by recognizing problem patterns and proposing subproblem solution structures corresponding to those patterns. Automatic problem solvers have a tendency when faced with an unsolvable problem to fail in a fixed and relatively uncontrollable way. In the solution plan approach the user determines which parts of a problem solution will be incomplete or less optimal. He remains in control of the solution process.

Different attempts have been made to incorporate solution planning in automatic heuristic problem solvers. It has proved to be very difficult to mechanize the process of recognizing problem patterns and of constructing and manipulating solution plans. The solution plan approach to interaction trades off the incorporation of these processes into a problem solving system against the cost of the time that the user must spend interacting with the computer.

7. Solution Plan Interaction and Graphics

The most important feature of solution plan interaction is the ability of the user in a solution plan system to describe a solution structure or solution idea for a particular problem without having to construct a general definition of the solution idea or the situations in which it is appropriate. In the South American TSP experiment, for example, the user was able to try out "double tour" and "single tour" solutions by grouping cities together into single and double tour solution plans. It was not necessary for him to define single and double tour types of solution plans or when these types of plans should be applied. There are similar examples for other travelling salesman problems and for sofa problems.

In order to apply solution plan interaction to a problem it is necessary to externally represent the problem to the user in such a way that he can recognize its subproblem structure and solution patterns. For some problems the external representation will be essentially the same as the problem's combinatorial solution representation. For other problems it will be different, although there will usually be a well defined correspondence between the subproblem features of the external representation and a solution plan decomposition of the solution representation. The TSP and sofa problems have natural geometric external problem representations consisting of points in a plane and objects in three-space. Their solution representations are permutations and paths through graphs. The external representations allow a user to recognize subproblem patterns. The corresponding solution representation plans allow subproblems to be precisely defined and acted upon by general purpose automatic problem solving routines.

Solution plan interaction can be applied to problems having both one and two dimensional external representations. It is likely that it will be most useful for solving problems with two-dimensional external representations and will require the use of a graphics display device.

Both the TSP and sofa problems are of this type. It is occasionally possible to construct effective one-dimensional pattern definitions and pattern recognition algorithms but it has proved to be excessively difficult to design two-dimensional pattern recognition methods that can be compared to the pattern recognition capabilities of humans. The use of graphics display devices, and of solution plan interaction, permits the introduction of powerful problem solving capabilities into a problem solving system without requiring the mechanization of two-dimensional pattern recognition processes.

Graphics display devices can be used in several ways in interactive problem solving systems. Their most important use is in displaying an external representation of the problem to the user. They can also be used to externally represent the current solution plan and to assist in the definition of new solution plans. If the user gets his solution plan ideas by looking at the displayed external representation of a problem it is likely that he will be able to define his solution plans in terms of and that they can be displayed along with the representation. In the TSP system solution plans are externally represented as polygons around groups of cities. The user can carry out part of the process required to create a new solution plan by drawing new polygons. The system automatically relates external solution plan representations to internal solution plan phrase trees. In a sofa problem system, particular solution plans could be represented by paths through the displayed house. The user could carry out part of the process of constructing new plans by pointing at locations inside the house.

8. Conclusions

(a) Solution Plans. Investigation of the solution plan idea indicates that it can be used to describe the structure of interactive problem solving processes in which the user guides the construction of the solution. It can also be used to describe and to relate automatic to interactive problem solving processes. Experience with the TSP system and speculation about a proposed sofa problem system indicates that solution plan interaction can be used to produce acceptable solutions to problems which are too large to solve using automatic problem solvers.

In general, there are two requirements for the application of solution plan interaction to a problem. The first is that the problem have some external representation from which it is possible for a user to discover useful subproblem patterns. The second is that the subproblems can be defined in terms of some combinatorial solution representation. The TSP and sofa problems have these properties. It appears as though the Machine Shop Scheduling Problem does not. There does not appear to be any subproblem patterns for this problem which can be used to decompose large MSSP's into smaller MSSP's. No successful class of solution plans have been discovered for the problem.

Perhaps the most important ability which the human can contribute to a man-machine system is his ability to recognize geometric and abstract patterns. The important ability of the computer is its ability to quickly carry out the solution of straightforward subproblems. In a solution plan interactive problem solving system the man and the machine play the roles in which they are most effective.

(b) Formalism. The solution grammar formalism can be used to formally define the notion of a solution plan. The formalism can be used to define what it means to "chart a course" or to create a "goal structure" for an important class of problems and interactive solution processes.

Solution grammars can be used to characterize the structure of automatic as well as interactive solution processes. Solution grammar composition rules can be thought of as production rules. The application of a production rule to a compound object containing variables "replaces" the subproblem specified by that object with a structure of subproblems. All of the better known automatic general problem solvers (e.g. Newell, [16], [10], [17]) are production solvers. They generate solutions by applying sequences of production rules.

The solution grammar formalism can be used to classify different kinds of interactive and automatic problem solving processes. Automatic problem solvers which generate solutions by applying production rules generate solution plan phrase trees. Different kinds of solution plans correspond to different kinds of solution grammar composition rules. The formalism can be used to prove that certain kinds of plans cannot be generated for certain kinds of solution representations. [18] contains an extensive investigation of the solution grammar formalism and its application to the classification and analysis of solution plans.

(c) Future Research. Continuation of the research described in this paper could include an investigation of the applicability of solution plan interaction to different classes of problems. It could also include an investigation into problem solving in general. Solution grammars can be used to formalize one aspect of the general problem solving process: the notion of a solution plan. In order to create a unified general theory it will be necessary to formalize and integrate other aspects of problem solving.

References

[1] J. McCarthy: Time Sharing Computer Systems. In: Conversational Computers (Edited by W.D. Orr). John Wiley and Sons, New York (1968).

[2] J.C.R. Licklider: Introduction to the Session on Man-Machine Communication. In: Communication Processes (Edited by F.A. Geldard). Pergamon Press, Oxford (1965).

[3] J.E. Doran: An Approach to Automatic Problem Solving. In: Machine Intelligence 1 (Edited by N.I. Collins and D. Michie). Oliver and Boyd, Edinburgh (1967).

[4] W.E. Howden: The Sofa Problem. Computer Journal, 1968, 11, 3.

[5] S. Lin: Computer Solutions of the Travelling Salesman Problem. The Bell System Technical Journal, 1965, 18, 1.

[6] S.M. Johnson: Optimal Two and Three Stage Production Schedules with Set-up Times Included. Naval Research Logistics Quarterly, 1954, 1, 1.

[7] W.E. Howden: SOFA-3 -- A program for the Three-Dimensional Sofa Problem, Technical Report 8, Department of ICS, University of California at Irvine (1971).

[8] W.S. Gere: Heuristics in Job Shop Scheduling. Management Science, 1966, 13, 2.

References (Con't)

[9] A Newell: The Possibility of Planning Languages in Man-Computer Interaction. In: Communication Processes (Edited by F.A. Geldard). Pergamon Press, Oxford (1965).

[10] J.E. Doran and D. Michie: Experiments with the Graph Traverser Program. Proceedings of the Royal Society, 1966, 29A.

[11] I.J. Good: A Five Year Plan for Automatic Chess. In: Machine Intelligence 2 (Edited by E. Dale and D. Michie). Oliver and Boyd, Edinburgh (1968).

[12] J.R. Slagle: Artificial Intelligence -- The Heuristic Programming Approach. McGraw Hill, New York (1971).

[13] C. Hewitt: Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot, Ph.D. Thesis. Massachusetts Institute of Technology (1972).

[14] R. Narasimhan et al: Bubble Talk: The Structure of a Program for On-Line Conversation with ILLIAC III, Report COO-1018-1003, File 604, University of Illinois (1964).

[15] W.E. Howden: The Planning Approach to Interactive Problem Solving and the Travelling Salesman Problem, Technical Report 20, Department of ICS, University of California at Irvine (1973).

References (Con't)

[16] A. Newell et al: Report on a General Problem Solving System. In: Information Processing: Proceedings of the International Conference on Information Processing, UNESCO, Paris. Butterworths, London (1959).

[17] J.R. Slagle and P. Bursky: Experiments with a Multipurpose, Theorem Proving Heuristic Program. JACM, 1968, 15, 1.

[18] W.E. Howden: Representation Plans and Problem Solving, Ph.D. Thesis. University of California at Irvine (1973).

Figure Captions

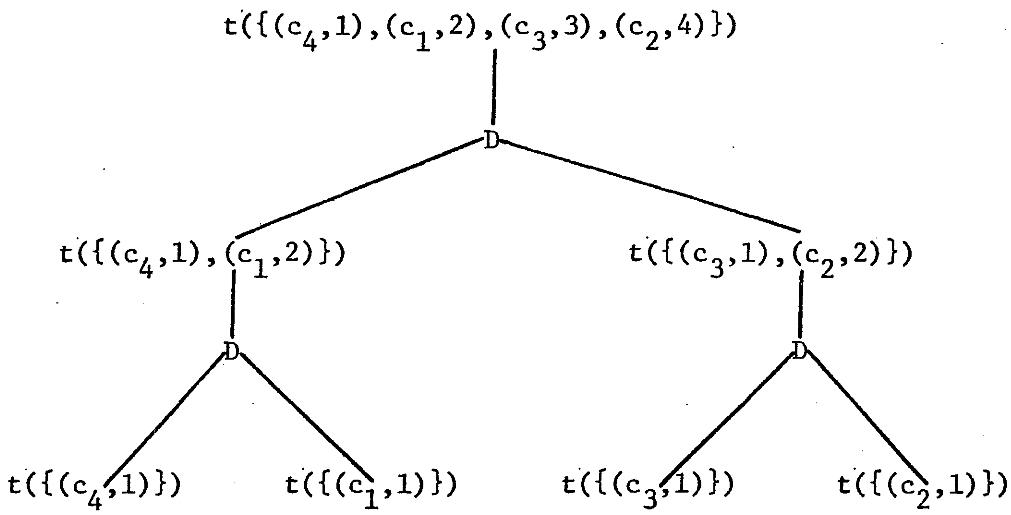
Figure 1. Solution Grammar Phrase Tree

Figure 2. Permutation Solution Plan.

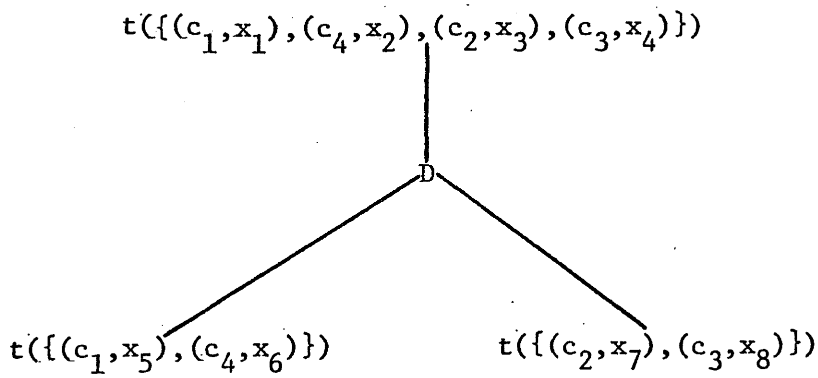
Figure 3. Single Tour Plan.

Figure 4. Double Tour Plan.

Figure 5. Minimal Solution.



t({(c4,1),(c1,2),(c3,3),(c2,4)})
D
t({(c4,1),(c1,2)})
t({(c3,1),(c2,2)})
D
D
t({(c4,1)})
t({(c1,1)})
t({(c3,1)})
t({(c2,1)})


t({(c1,x1),(c4,x2),(c2,x3),(c3,x4)})
D
t({(c1,x5),(c4,x6)})
t({(c2,x7),(c3,x8)})