UTOPIA COMPUTER

The »New« in Architecture?

Nathalie Bredella, Chris Dähne,
Frederike Lausch (Eds.)

Forum Architekturwissenschaft
Band 6

Universitätsverlag
der TU Berlin

NETZWERK
ARCHITEKTUR
WISSENSCHAFT

UTOPIA COMPUTER
The “New” in Architecture?

Nathalie Bredella, Chris Dähne,
Frederike Lausch (Eds.)

The scientific series *Forum Architekturwissenschaft* is edited by the Netzwerk Architekturwissenschaft, represented by Sabine Ammon, Eva Maria Froschauer, Julia Gill and Christiane Salge.

The critical concern of the book "Utopia Computer" is the euphoria, expectation and hope inspired by the introduction of computers within architecture in the early digital age. With the advent of the personal computer and the launch of the Internet in the 1990s, utopian ideals found in architectural discourse from the 1960s were revisited and adjusted to the specific characteristics of digital media. Taking the 1990s discourse on computation as a starting point, the contributions of this book grapple with the utopian promises associated with topics such as participation, self-organization, and non-standard architecture. By placing these topics in a historical framework, the book offers perspectives for the future role computation might play within architecture and society.

NETZWERK
ARCHITEKTUR
WISSENSCHAFT

Forum Architekturwissenschaft, Volume 6

UTOPIA COMPUTER

The “New” in Architecture?

Nathalie Bredella, Chris Dähne,
Frederike Lausch (Eds.)

Universitätsverlag
der TU Berlin

→

PABLO MIRANDA CARRANZA

Making Sense without Meaning

Christopher Alexander and the Automation of Design

In his contribution to the influential "Architecture and the Computer" conference in 1964, Christopher Alexander summarised the reorganisation of intellectual labour that, beginning in the 19th century, became finally concretised in the technologies of the computer. In his opinion, computers should be regarded as nothing else than huge armies of clerks, stupid and without initiative, but able to follow to the letter millions of precisely written instructions. This essay examines how architectural design began being digitally transcribed so it would conform to the logics of these armies of clerks, through a close reading of the programs and computer code written by Alexander.

An outlandish machine

This is a story of how the capacities that once defined what an architect was became dislodged and incorporated into computers. Rather than spoken and discussed or historically recorded as discourse, this story was mostly written in computer code. Stories often begin in anecdotal places and with irrelevant events. The one told here started in the library of the now burned-down architecture school in Delft, sometime in the mid–1990s, when I first encountered Serge Chermayeff and Christopher Alexander's *Community and Privacy: Toward a New Architecture of Humanism* and the outlandish machine for generating designs that it described.[1] To use this machine, first one had to reduce a

1 Serge Chermayeff and Christopher Alexander, Community and Privacy: Toward a New Architecture of Humanism (Garden City/NY: Doubleday, 1963).

←

design to a list of requirements and their interdependencies. This description would then be fed into an IBM 704 computer running a program which would separate and organise the requirements into independent sets easily translatable into a design.[2] Chermayeff and Alexander's inspiration for this process was the Taoist butcher described in the *Chuang Tzu*, who could effortlessly cut an ox into distinct parts by sliding a knife between the interstices separating them. While their book eschewed any details about how the methods of this mystical butcher had been translated into a computer program, Alexander described them in some detail in his PhD thesis "Notes on the Synthesis of Form," defended in 1962 and published as a book two years later.[3] However, the actual code for Alexander's program was only available in the MIT research report R62-2, co-authored with engineer Marvin Manheim and published in July 1962 with the less ornate title *HIDECS 2: A Computer Program for the Hierarchical Decomposition of a Set Which Has an Associated Linear Graph*.[4] Alexander and Manheim's report included flowcharts, diagrams, explanations and code listings written in the FORTRAN Assembly Program, or FAP, for the IBM 709 computer (fig. 1). Materials such as these are usually excluded from traditional architectural historiographies, since they call for literacies foreign to their established discourses. To begin with, the addressees of computer code are both humans and machines, and to complicate things further, in contrast to historical writing tools, programming is a form of inscription which is able to write and read by itself.[5] The analysis of programs also needs, then, to include the objects of their reading, their inputs, as well as the products of their writing, their outputs.

2 Ibid., 149–163.

3 Christopher Alexander, Notes on the Synthesis of Form (Cambridge/MA: Harvard University Press, 1964).

4 Christopher Alexander and Marvin L. Manheim, HIDECS 2: A Computer Program for the Hierarchical Decomposition of a Set Which Has an Associated Linear Graph (Cambridge/MA: School of Engineering, Massachusetts Institute of Technology, 1962).

5 Friedrich A. Kittler, "There Is No Software," Stanford Literature Review 9, no. 1 (Spring 1992): 81–90.

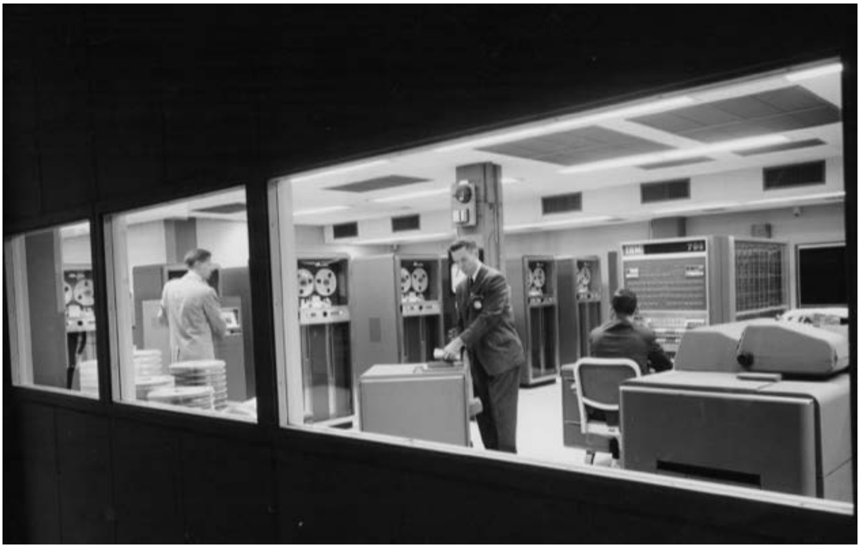


Fig. 1: IBM 709. Image courtesy of International Business Machines Corporation, Dt. UrhR: International Business Machines Corporation

FAP, the programming language used in HIDECS 2, was a form of assembly language, and as such, it was tied to the particularities of the hardware of the IBM 709 it was written for. Architectural historian Alise Upitis has shown how closely linked Alexander's propositions were to the particular characteristics of the hardware he used.[6] Certainly the hardware and the expressive limitations of the FAP language must have influenced the concepts Alexander developed through them. It would still be possible to analyse this code today using, for example, an IBM 709 emulator running in a contemporary computer. In my examination and critique of Alexander's machine, I have chosen instead to translate the HIDECS 2 programs into the commonly used Python programming language.[7] This translation de-emphasizes the

6 Alise Upitis, "Alexander's Choice: How Architecture Avoided Computer-Aided Design c. 1962," in A second Modernism: MIT, architecture, and the 'techno-social' moment, ed. Arindam Dutta (Cambridge/MA: MIT Press, 2013), 474–505.

7 All code is available at gitlab.com/Zenbagailu/hidecs-2-python.

material and technical conditions behind the programs, but at the same time foregrounds the abstractions and concepts they implement, making it easier to clarify their relationships to coeval discourses and techniques. The code used in this text is mostly based on the flow charts of appendix C and descriptions in appendices D to F of the research report (fig. 2). Functions in the Python code have the same names as Alexander and Manheim's original subprograms, when they implement the same functionality. Like most software, HIDECS 2 was also a work in progress. Many of the descriptions of features in the document were not implemented as of December 1961 (when it ran on the MIT IBM 709) and are not described in detail in the report, even though some of them appear in the FAP listings of appendix G. A few of these features were completed in HIDECS 3 and were presented in a subsequent research report *Hidecs 3: Four Computer Programs for the Hierarchical Decomposition of Systems Which Have an Associated Linear Graph.*[8]

HIDECS 2 implemented the design method that Alexander would explain at length in *Notes on the Synthesis of Form*, published as a book in 1964. The objective of the method was to find a "good fit" between form, "a part of the world over which we have control," and a context, "that part of the world which puts demands on this form."[9] According to Alexander this good fit was typical of "primitive," "folk," "closed," or "anonymous" cultures, the result of an "unselfconscious" [sic] and slow process of adaptation and dynamic equilibrium between the complex demands of the context and forms, a process resembling the cybernetic principle of homeostasis. In contrast, modern "selfconscious" [sic] design was a response to the rapidly changing contexts of industrial societies. Whereas slow and "unselfconscious" adaptations needed no representation of the context, the "selfconscious"

8 Christopher Alexander, HIDECS 3: Four Computer Programs for the Hierarchical Decomposition of Systems which have an Associated Linear Graph (Cambridge/MA: School of Engineering, Massachusetts Institute of Technology, 1963).

9 Alexander, Notes on the Synthesis of Form, 15–27.

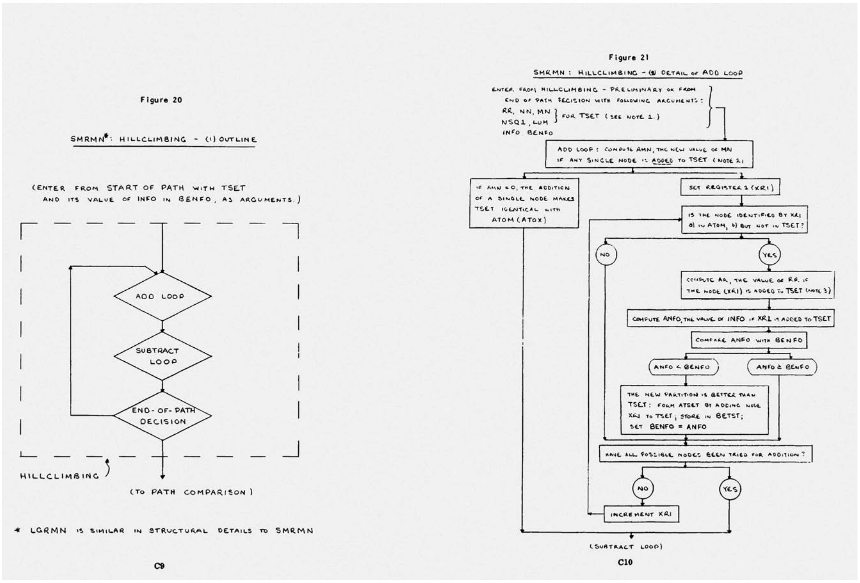


Figure 20
SMRMN*: HILLCLIMBING - (1) OUTLINE
(ENTER FROM START OF PATH WITH TSET AND ITS VALUE OF INFO IN BENFO, AS ARGUMENTS.)
ADD LOOP
SUBTRACT LOOP
END-OF-PATH DECISION
HILLCLIMBING
(TO PATH COMPARISON)
* LGRMN IS SIMILAR IN STRUCTURAL DETAILS TO SMRMN
C9
Figure 21
SMRMN: HILLCLIMBING - (2) DETAIL OF ADD LOOP
ENTER FROM HILLCLIMBING - PRELIMINARY OR FROM END OF PATH DECISION WITH FOLLOWING ARGUMENTS:
RR, NN, MN
NSQ1, LUM
FOR TSET (SEE NOTE 1.)
INFO BENFO
ADD LOOP: COMPUTE AMN, THE NEW VALUE OF MN IF ANY SINGLE NODE IS ADDED TO TSET (NOTE 2)
IF AMN ≤ 0, THE ADDITION OF A SINGLE NODE MAKES TSET IDENTICAL WITH ATOM (ATOX)
SET REGISTER 1 (XR1)
IS THE NODE IDENTIFIED BY XR1 a) IN ATOM, b) BUT NOT IN TSET?
NO
YES
COMPUTE AR, THE VALUE OF RR IF THE NODE (XR1) IS ADDED TO TSET (NOTE 3)
COMPUTE ANFO, THE VALUE OF INFO IF XR1 IS ADDED TO TSET
COMPARE ANFO WITH BENFO
ANFO < BENFO
ANFO ≥ BENFO
THE NEW PARTITION IS BETTER THAN TSET: FORM ATSET BY ADDING NODE XR1 TO TSET; STORE IN BETST; SET BENFO = ANFO
HAVE ALL POSSIBLE NODES BEEN TRIED FOR ADDITION?
NO
YES
INCREMENT XR1
(SUBTRACT LOOP)
C10

Fig. 2: Flow charts of the Hill Climbing procedure and detail of the add loop in *HIDECS* 2, 1962

designer worked from a mental picture of that context, a picture that, because of the changing and complex conditions, Alexander claimed, was almost always wrong.[10] Alexander's solution to this incongruity was to use the computer to calculate the picture, since only the computer could represent and resolve the complex interrelations of evolving contextual demands. This process would begin by a designer describing the context as a finite set of design requirements and their interdependencies, represented in the computer using a mathematical structure known as a graph. This graph would be instrumental in sorting requirements into a hierarchical description of interdependent subsets, which would correspond to a clear mental picture of the context. The report explained this process succinctly: "The input to the program is a graph; the output is a tree, a hierarchical ordering of the graph's vertex set and its partitioned subsets. Because of the correspondence between the graph and the problem, the tree which is

10 Ibid., 73–83.

obtained by the program provides an orderly scheme for dealing with the requirements posed by a particular problem."[11]
Alexander called this scheme "the program" since, as Manheim explained later, it defined a sequence of actions to solve the problem of finding a good fit.[12] The subsets of problems would be sufficiently simple to suggest their resolution, similar to how "diagrams of forces" dictated the biological form of radiolaria or molluscs, in D'Arcy Thompson's influential analyses of their morphology.[13] These independent design responses would then be composed according to the hierarchical order of the programme. While this mental picture of the context could be calculated, its resolution into a design, Alexander insisted, required invention, which was impossible to implement using the computer.[14]

Input: The snare of semiotics

"[A]s a rule, concepts are not generated or defined in extension; they are generated in intension. That is, we fit new concepts into the pattern of everyday language by relating their meanings to those of other words at present available in English."[15]
In these few lines Alexander laid out what was the concern of his method and algorithms: not the specific words defining a problem, but their interrelations. Alexander made explicit reference to philosopher Rudolf Carnap's distinction between "intension" and "extension," between conditions of signification and conditions of truth, that is, between the semiotic mechanisms that give rise to signification and the reality of the objects that are signified. Through his use of graphs (fig. 3) Alexander chose to identify the structure of the problem as the genuine source of its meaning,

11 Alexander and Manheim, HIDECS 2, 7.

12 Marvin L. Manheim, "Problem Solving Processes in Planning and Design," Design Quarterly 66/67 (1966): 31–39.

13 D'Arcy W. Thompson, On Growth and Form, new ed. (Cambridge: Cambridge University Press, 1942). Alexander, Notes on the Synthesis of Form, 21.

14 Alexander, Notes on the Synthesis of Form, 84–94.

15 Ibid., 67.

rather than looking for it in the actual requirements that described the problem.
This was in effect a transposition into design of the dualism between form and substance typical of structuralist semiotics.[16] However, rather than equating this difference with that between architectural form and its meaning, the path followed by formalist analysis and postmodernist architecture, Alexander envisaged design instead as the strictly linguistic problem of correctly representing a design program. Seventy years before, Gottlob Frege, Carnap's teacher and mathematics professor at the University of Jena, had first characterized this dualism which underpins Alexander's approach in terms of sense and reference: sense pertained to the relationships between objects, names or signs in their capacity to produce meaning (this would correspond to Carnap's concept of intension, as used by Alexander); reference would denote the reality or truth of a sign or a word.[17] By focusing on the structure of the problem represented as a graph, HIDECS 2 exemplifies the capacities of the computer to produce and make sense through its codes and logical forms, its diagrams and structures. But this is a "sense without meaning," one for which referent or substance are irrelevant, typical, in philosopher Maurizio Lazzarato's view, of the a-signifying semiotics of current technical and economic apparatuses.[18]
Alexander's use of graphs, as much as the rest of his sense-making toolbox, had its provenance in operations research and management science, where graphs had been extensively researched for their ability to analyze and optimize distribution networks and

16 Other examples of this form/content dualism are Ferdinand de Sassure's distinction between signifier and signified or Louis Hjelmslev's distinction between the two separate planes of expression and signification in linguistics. For an overview of the main theories of semiotics see Umberto Eco, A Theory of Semiotics, Advances in Semiotics (Bloomington/IN: Indiana University Press, 1976). For a critique of the basis of semiotics and structuralism: Jacques Derrida, Of Grammatology (Baltimore/MD: Johns Hopkins University Press, 1976).

17 Gottlob Frege, "Über Sinn und Bedeutung," Zeitschrift für Philosophie und philosophische Kritik 100 (1892): 25–50.

18 Maurizio Lazzarato, Signs and Machines: Capitalism and the Production of Subjectivity, trans. Joshua David Jordan (Los Angeles/CA: Semiotext(e), 2014).

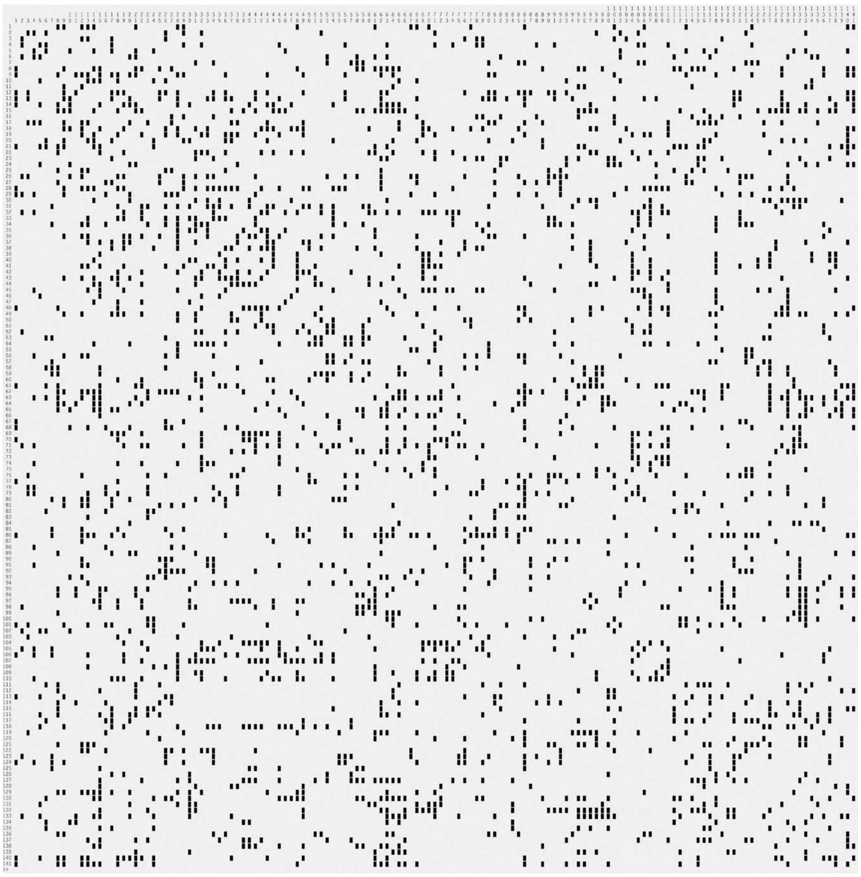


Fig. 3: Matrix of the graph of requirements for an Indian Village in "Notes on the Synthesis of Form," output of the Python implementation of HIDECS 2. Code by the author

logistic infrastructures.[19] In its intensional description of a design context, HIDECS 2 took as its input any design requirements and represented the structure of their interdependencies as a graph,

19 Examples of this research during the period led to well-known algorithms, such as the Ford-Fulkerson algorithm for calculating maximum flow: Lester Randolph Ford and Delbert R Fulkerson, "A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem," Canadian Journal of Mathematics 9 (1957). Another related example is the Hungarian method to solve the allocation problem: Harold W Kuhn, "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly 2, no. 1/2 (1955).

regardless of the actual requirements and the specific design problem they may have described: a kettle, a building, or an urban plan. These operations are symptomatic of the depoliticizing and depersonalizing effect that Lazzarato assigns to this "sense without meaning," which, by making heterogeneous spheres formally equivalent, integrates them into rationalized schemes of production.[20] They exemplify the abstraction and alienation that became the target of post-structuralist critiques of linguistic models from the late 1960s onwards, and to which Lazzarato's more recent analysis belongs. In what became a seminal contribution to post-structuralism, philosopher Jacques Derrida found it in the supposedly arbitrary relationship instituted between systems of signs and their objects, between their sense and their reference, a logic of alienation infused with western ethnocentrism, the result of understanding signs and representations exclusively through the principles of western phonetic writing.[21] The reduction of the design of an Indian village to a graph (fig. 3), used by Alexander in *Notes*, reads almost like a caricature of this ethnocentric depoliticization and depersonalization, of the techno-scientific objectivities of this "sense without meaning" that, according to Donna Haraway, are nothing else than the white man's gaze.[22] Requirements such as "Harijans regarded as ritually impure, untouchable, etc.," "Cattle treated as sacred, and vegetarian attitude," or "Need for elaborate weddings," are treated the same way as the specifications for a highway interchange,[23] a fuel pump, a jet engine or a tea kettle, all transformed into data by the digitalized techno-scientific gaze. In all its absurdity and

20 Lazzarato, Signs and Machines, 40–41.

21 Derrida, Of Grammatology, 50–51.

22 Donna Haraway, "Situated knowledges: The Science Question in Feminism and the Privilege of Partial Perspective," Feminist Studies 14, no. 3 (1988).

23 Christopher Alexander, Marvin L. Manheim and Massachusetts Institute of Technology, Department of Civil Engineering, The Design of Highway Interchanges: An Example of a General Method for Analysing Engineering Design Problems, research report (Cambridge/MA: Dept. of Civil Engineering, Massachusetts Institute of Technology, 1962).

colonial mentality, the process recalls the exhilarating irrationality of a Raymond Roussel novel.[24]

This feeling of absurdity is not just the product taking the distinction between intension and extension, between sense and reference, to the extreme, but also the consequence of stating and considering the problem in terms of its mathematical form. At a technical level, the method chosen by Alexander for finding optimal subsets of requirements was so computationally complex from the outset that the problem he posed was intractable in practice. The reason for this is that the total number of potential partitions in a set grows exponentially with the number of its elements, given by its power set—the amount of all possible sets—divided by two, that is 2^{n-1} (where n is the number of elements, in this case design requirements). The example of the Indian village in *Notes* had 141 elements, with results in 2^{140}, or 1,393,796,574, 908,163,946,345,982,392,040,522,594,123,776 potential subdivisions, all of which would need to be checked in order to find an optimal partition. These large numbers are unmanageable for humans, and in practice also for machines. A program running millions of times faster than the Python version written for this paper, taking one picosecond to do the same calculation that now takes 40 microseconds, would still need 44,196,999,458 billion years to consider each possible subdivision. These numbers clearly represent inhuman amounts of work, a burden of a cosmic order even for a machine. The reason for Alexander to propose such a convoluted method can only be justified because it would enable him to use another instrument in his sense-making toolbox: heuristics. Problems of this complexity can only be computationally approached using methods that provide good enough solutions, rather than optimal ones.

24 I am thinking specially of “Impressions of Africa.” Raymond Roussel, Impressions d’Afrique (Paris: A. Lemerre, 1910). Half a century later, partly inspired by Roussel’s work, the OuLiPo group used computational methods similar to those of Alexander to generate a literature that embraced their mechanical irrationality as one of its defining qualities: Warren F. Motte and OuLiPo, OuLiPo: A Primer of Potential Literature (Lincoln/NE: University of Nebraska Press, 1986). For a description of the role of modern mathematics in work associated with OuLiPo: Alice Bamford, “Mathematics and Modern Literature: Passages from ‘Chalk and the Architrave,’” New Left Review, no. 124 (2020).

Program: clerks that climb hills

“A DIGITAL COMPUTER is, essentially, the same as a huge army of clerks equipped with rule books, pencil and paper, all stupid and entirely without initiative, but able to follow exactly millions of precisely defined operations. There is nothing a computer can do which such an army of clerks could not do, if given time.”[25]
Heuristics, and for that matter, much of Alexander’s approach to design, where influenced by the then new fields of artificial intelligence and cognitive science. These in turn belonged to a history of automation of human thought that already begun more than a century earlier, as it was explained by economist Herbert Simon, an initiator of both fields and a main reference in Alexander’s work. In his introduction of the concept of heuristics at to the Operations Research Society Of America (ORSA) in 1957, Simon recapitulated the common history of the computer and operations research. When mathematician Gaspard de Prony was faced with the enormous task of calculating logarithmic and trigonometric tables for the French cadastre immediately after the French Revolution, he decided to apply Adam Smith’s principle of the division of labour—illustrated in *The Wealth of Nations* through the manufacturing of pins—to the menial work involved in mathematical reckoning. A few decades later Charles Babbage, the putative father of the computer, took the next step by replacing this readily fragmented and disciplined clerical labour with machinery. In addition to describing the history of the computer as that of the mechanization of intellectual work, Simon also outlined the future of operations research (OR) in expanding its scope from the ‘well structured’ management problems with which the field had been concerned since its beginnings in the Second World War, to the ‘ill structured’ ones that made up the majority of (and most important) top-level management and executive problems. His proposal was “to handle

25 Christopher Alexander, “The Question of Computers in Design,” Landscape 14, no. 3 (1965).

with appropriate analytic tools those problems that we now tackle with judgement and guess" through a "theory of heuristic (as contrasted to algorithmic) problem solving." This could be used then "both to understand human heuristic processes and to simulate such processes with digital computers." Through such a theory, intuition, insight, and learning would no longer be the exclusive domain of humans, as any large high-speed computer could also be programmed to exhibit them.[26]

The translatability of human thought into computation was only possible if the mind was also considered a kind of computing device, executing thinking processes as its programs. This was precisely the central premise of the new field of cognitive psychology. Alexander explicitly referred to the work of psychologist George Miller, founder of the Center for Cognitive Studies at Harvard with which Alexander collaborated. His seminal paper "The Magical Number Seven, Plus or Minus Two" empirically investigated the storage capacity of people's short term memory, and the general capacity of the human mind to store and transmit data. Miller proposed an equivalent to the bit, what he called the "chunk": the information unit of the mind, and described the limit of human processing as about "seven plus or minus two" "chunks" at a time, with a transmission speed of about five seconds.[27] These defined the computational constraints of the human mind in terms of CPU registers and speed. It was with such a limited hardware that heuristic processes were originally run, that is, with the algorithms of the mind.[28] The economic significance of understanding human cognition as a series of computational processes earned Herbert Simon the Nobel Prize in Economics in 1978. His idea of "bounded rationality" supplemented the conceptual limitations of rational choice, a central idea in the classical economics of Adam Smith or David Ricardo, as it considered

26 Herbert A. Simon and Allen Newell, "Heuristic Problem Solving: The Next Advance in Operations Research," Operations Research 6, no. 1 (1958).

27 George A. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," Psychological Review 63, no. 2 (1956).

28 Herbert A. Simon, The Sciences of the Artificial, Karl Taylor Compton Lectures (Cambridge/MA: MIT Press, 1969).

the limited computational and informational capabilities for making rational inferences by individual actors.[29]
If the argument was that the mind was a computer, and a bad one at that, it is reasonable to conclude that the remedy to its limitations must be the use of actual computers with higher processing capacities. In order to contend that any activity is worthy of computerization, it must be presented as complex enough to exceed human capacities for cognitive labour. It thus became imperative for Alexander to present design as a computational task of such complexity that it would necessarily exceed the “seven plus or minus two chunks” of human memory (and its five second of transmission speed). Also, by assigning it a high order of complexity, the solution of a design problem would require the application of heuristic methods belonging to the new field of artificial intelligence. But despite any Promethean claims about AI, the computational clerk that would take over design could operate the very simple procedure described in the following lines in Python:

```
def addLoop(graph, vSet, vSub):
    selected = min( {el for el in vSet if el not in vSub}, key = lambda el : calculateGraphInfo(graph,vSet,vSub | {el}))
    return selected if calculateGraphInfo(graph,vSet,vSub | {selected}) < calculateGraphInfo(graph,vSet,vSub) else None

def subtractLoop(graph, vSet, vSub):
    selected = min(vSub, key = lambda el : calculateGraphInfo(graph,vSet,vSub - {el}))
    return selected if calculateGraphInfo(graph,vSet,vSub-{selected}) < calculateGraphInfo(graph,vSet,vSub) else None

def hillClimb(graph, vSet, vSub):

    while True:
        toAdd = addLoop(graph,vSet,vSub) if len(vSub) < len(vSet) - 1 else None
        if toAdd:
            vSub.add(toAdd)

        toSubtract = subtractLoop(graph,vSet,vSub) if len(vSub)>1 else None
        if toSubtract:
            vSub.remove(toSubtract)

        if not toAdd and not toSubtract: #if it could not improve
            break

    return vSub
```

Fig. 4: The hill-climbing algorithm. Code by the author

The code above (fig. 4) describes a *hill climber*, a type of algorithm used extensively in mathematical optimization methods. Its name refers to the heuristic it employs, which can be compared to finding a summit in a terrain by examining points immediately

29 Incidentally, Simon’s idea were partly inspired by neoliberal economist Friedrich Hayek, who had also argued that the price system would overcome the limitations of individual rational choice. Herbert A. Simon, Models of Man: Social and Rational. Mathematical Essays on Rational Human Behavior in a Social Setting (New York/NY: Wiley, 1957).

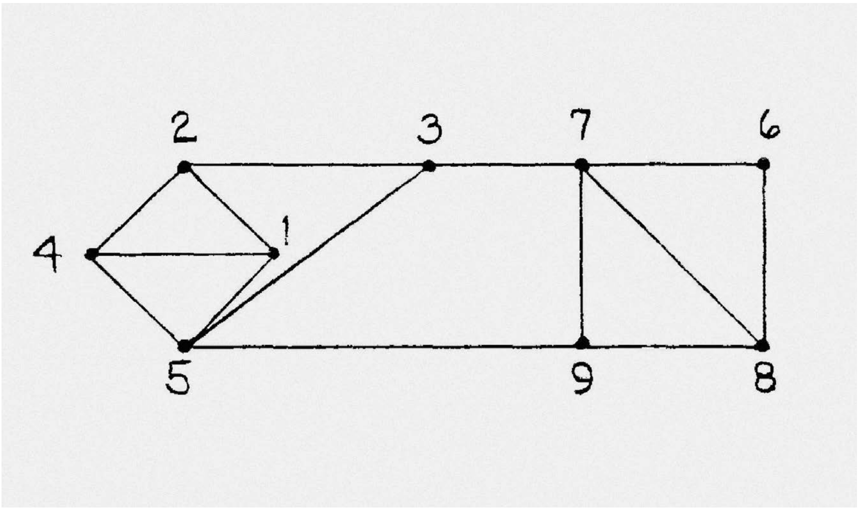


Fig. 5: Example diagram from the HIDECS 2 report, Christopher Alexander and Marvin Manheim

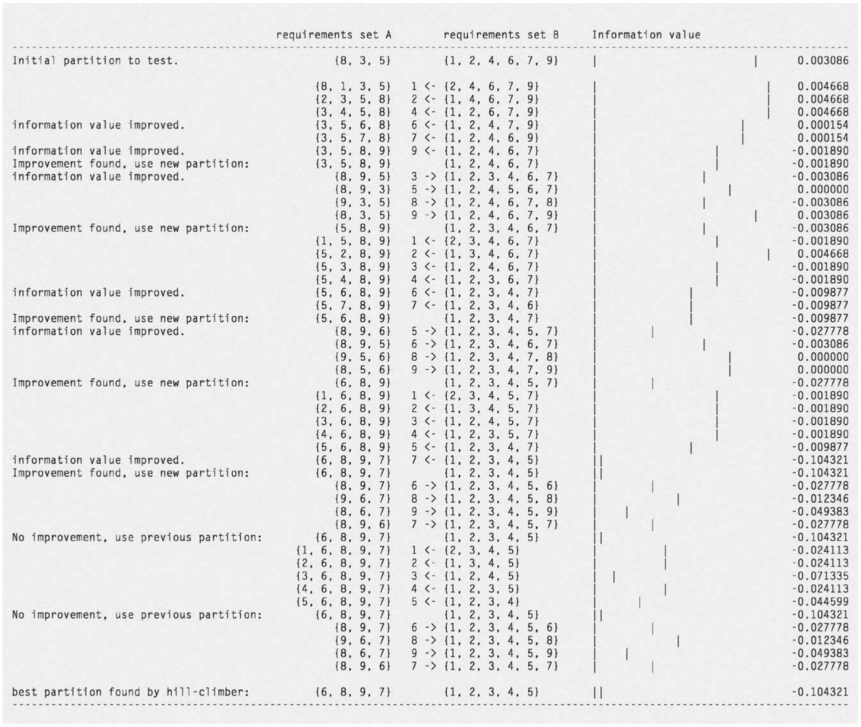
close to the present location, moving to the highest one, and repeating the operation until no higher points can be found. The hill top reached will be higher than its surroundings (a local maximum), but not necessarily the highest (a global maximum). Hill climbing belongs to a class of optimization techniques known as greedy algorithms, since these cannot consider trade-offs between temporary losses and longer term gains (the *hill-climber* cannot go down a valley to reach a higher peak). The limitations of greed as a strategy are compensated however by the simplicity of its implementation.
In the Python translation of HIDECS 2, this heuristic is implemented as follows: the hill climbing function, declared as *def hillClimb()*, takes three parameters as its input: a *graph*, describing the interdependencies between all the requirements; *vSet*, the actual set of requirements to be partitioned; and *vSub*, a subset of requirements, representing one half of a tentative partition. The function iteratively calls *addLoop*, which discovers whether the partition can be improved by moving any requirement to its *vSub* half; subsequently, *subtractLoop*, tests for improvements in moving some requirement from the *vSub* set to the other half. If no improvements can be made, the algorithm has found

a relatively optimal partition and returns it to the main process, which will continue trying to further subdivide it. Otherwise, if improvements are still possible, it repeats the same process until no improvements are found. The scores for each partition, calculated through the *calculateInfo* function, describe how interdependent the two halves are, while the goal of the hill-climber is to find a cut in which the two halves are as interdependent as possible.[30] The testing of partitions against a value (equivalent to the height of a terrain), and its incremental improvement, is analogous to the described hill-climbing procedure. Below (fig. 6) is the result of applying the Python code to one of the graphs (fig. 5) from the research report.[31] That the graph did not correspond to a real problem or program is just another example of the split between sense and reference at work. The numeric meanderings of the hill climbing—or rather "hill descending," as the program tries to minimize a value—start from an arbitrary partition with indices {8, 3, 5} on one side and {1, 2, 4, 6, 7, 9} on the other. The printout shows the different tests and incremental improvement of the values (fig. 6).

The main process begins dividing all the requirements in half, using the hill climbing heuristic just described, and further applying the same procedure to each resulting partition, subdividing the requirements until no more subdivisions are possible. Code (fig. 4) and printouts (fig. 6) demonstrate how these prosaic "armies of clerks" reorganized what were once human capacities by: first, treating thought as labour that can be mechanized; second, pointing out the effective limits of human cognition in comparison with machines; and third, by arguing how the increasing complexity of intellectual tasks exceeds human cognitive capabilities, and thus requires the use of mechanized computational labour. In this redefinition of intelligence, the technocratic discourses of operations research, cognitive science, and artificial intelligence present both a problem—bounded rationality—and its solution, through the redistribution and incorporation

30 Alexander, Notes on the Synthesis of Form, 190.

31 Alexander and Manheim, HIDECS2, 6.



```
                                         requirements set A         requirements set B     Information value
------------------------------------------------------------------------------------------------------------------------------------
Initial partition to test.                        {8, 3, 5}         {1, 2, 4, 6, 7, 9}     |                        |       0.003086

                                               {8, 1, 3, 5}    1 <- {2, 4, 6, 7, 9}        |                          |     0.004668
                                               {2, 3, 5, 8}    2 <- {1, 4, 6, 7, 9}        |                          |     0.004668
                                               {3, 4, 5, 8}    4 <- {1, 2, 6, 7, 9}        |                          |     0.004668
information value improved.                    {3, 5, 6, 8}    6 <- {1, 2, 4, 7, 9}        |                      |         0.000154
                                               {3, 5, 7, 8}    7 <- {1, 2, 4, 6, 9}        |                      |         0.000154
information value improved.                    {3, 5, 8, 9}    9 <- {1, 2, 4, 6, 7}        |                  |            -0.001890
Improvement found, use new partition:          {3, 5, 8, 9}         {1, 2, 4, 6, 7}        |                  |            -0.001890
information value improved.                       {8, 9, 5}    3 -> {1, 2, 3, 4, 6, 7}     |                |              -0.003086
                                                  {8, 9, 3}    5 -> {1, 2, 4, 5, 6, 7}     |                    |           0.000000
                                                  {9, 3, 5}    8 -> {1, 2, 4, 6, 7, 8}     |                |              -0.003086
                                                  {8, 3, 5}    9 -> {1, 2, 4, 6, 7, 9}     |                        |       0.003086
Improvement found, use new partition:             {5, 8, 9}         {1, 2, 3, 4, 6, 7}     |                |              -0.003086
                                               {1, 5, 8, 9}    1 <- {2, 3, 4, 6, 7}        |                  |            -0.001890
                                               {5, 2, 8, 9}    2 <- {1, 3, 4, 6, 7}        |                          |     0.004668
                                               {5, 3, 8, 9}    3 <- {1, 2, 4, 6, 7}        |                  |            -0.001890
                                               {5, 4, 8, 9}    4 <- {1, 2, 3, 6, 7}        |                  |            -0.001890
information value improved.                    {5, 6, 8, 9}    6 <- {1, 2, 3, 4, 7}        |              |                -0.009877
                                               {5, 7, 8, 9}    7 <- {1, 2, 3, 4, 6}        |              |                -0.009877
Improvement found, use new partition:          {5, 6, 8, 9}         {1, 2, 3, 4, 7}        |              |                -0.009877
information value improved.                       {8, 9, 6}    5 -> {1, 2, 3, 4, 5, 7}     |        |                      -0.027778
                                                  {8, 9, 5}    6 -> {1, 2, 3, 4, 6, 7}     |                |              -0.003086
                                                  {9, 5, 6}    8 -> {1, 2, 3, 4, 7, 8}     |                    |           0.000000
                                                  {8, 5, 6}    9 -> {1, 2, 3, 4, 7, 9}     |                    |           0.000000
Improvement found, use new partition:             {6, 8, 9}         {1, 2, 3, 4, 5, 7}     |        |                      -0.027778
                                               {1, 6, 8, 9}    1 <- {2, 3, 4, 5, 7}        |                  |            -0.001890
                                               {2, 6, 8, 9}    2 <- {1, 3, 4, 5, 7}        |                  |            -0.001890
                                               {3, 6, 8, 9}    3 <- {1, 2, 4, 5, 7}        |                  |            -0.001890
                                               {4, 6, 8, 9}    4 <- {1, 2, 3, 5, 7}        |                  |            -0.001890
                                               {5, 6, 8, 9}    5 <- {1, 2, 3, 4, 7}        |              |                -0.009877
information value improved.                    {6, 8, 9, 7}    7 <- {1, 2, 3, 4, 5}        ||                              -0.104321
Improvement found, use new partition:          {6, 8, 9, 7}         {1, 2, 3, 4, 5}        ||                              -0.104321
                                                  {8, 9, 7}    6 -> {1, 2, 3, 4, 5, 6}     |        |                      -0.027778
                                                  {9, 6, 7}    8 -> {1, 2, 3, 4, 5, 8}     |            |                  -0.012346
                                                  {8, 6, 7}    9 -> {1, 2, 3, 4, 5, 9}     |    |                          -0.049383
                                                  {8, 9, 6}    7 -> {1, 2, 3, 4, 5, 7}     |        |                      -0.027778
No improvement, use previous partition:        {6, 8, 9, 7}         {1, 2, 3, 4, 5}        ||                              -0.104321
                                            {1, 6, 8, 9, 7}    1 <- {2, 3, 4, 5}           |          |                    -0.024113
                                            {2, 6, 8, 9, 7}    2 <- {1, 3, 4, 5}           |          |                    -0.024113
                                            {3, 6, 8, 9, 7}    3 <- {1, 2, 4, 5}           |  |                            -0.071335
                                            {4, 6, 8, 9, 7}    4 <- {1, 2, 3, 5}           |          |                    -0.024113
                                            {5, 6, 8, 9, 7}    5 <- {1, 2, 3, 4}           |      |                        -0.044599
No improvement, use previous partition:        {6, 8, 9, 7}         {1, 2, 3, 4, 5}        ||                              -0.104321
                                                  {8, 9, 7}    6 -> {1, 2, 3, 4, 5, 6}     |        |                      -0.027778
                                                  {9, 6, 7}    8 -> {1, 2, 3, 4, 5, 8}     |            |                  -0.012346
                                                  {8, 6, 7}    9 -> {1, 2, 3, 4, 5, 9}     |    |                          -0.049383
                                                  {8, 9, 6}    7 -> {1, 2, 3, 4, 5, 7}     |        |                      -0.027778

best partition found by hill-climber:          {6, 8, 9, 7}         {1, 2, 3, 4, 5}        ||                              -0.104321
------------------------------------------------------------------------------------------------------------------------------------
```

Fig. 6: Optimising a partition of a graph (fig. 5) by the Python version of HIDECS 2. Code by the author

of previous inalienable human constituents within their computational systems and assemblies. The new place of design in these apparatuses is apparent in the diagram that is the output of HIDECS 2 (fig. 7). It has become the fragmented task of resolving singular problems (in this case the conflicts between sets of requirements, grouped together in the diagram into columns) according to the dictates of the technical systems effectively distributing design as intellectual labour.

Output: not seeing the forest for the trees

“Design today has reached the stage where sheer inventiveness can no longer sustain it. To make adequate forms, one must be able to explore the relations between circumstances more fully

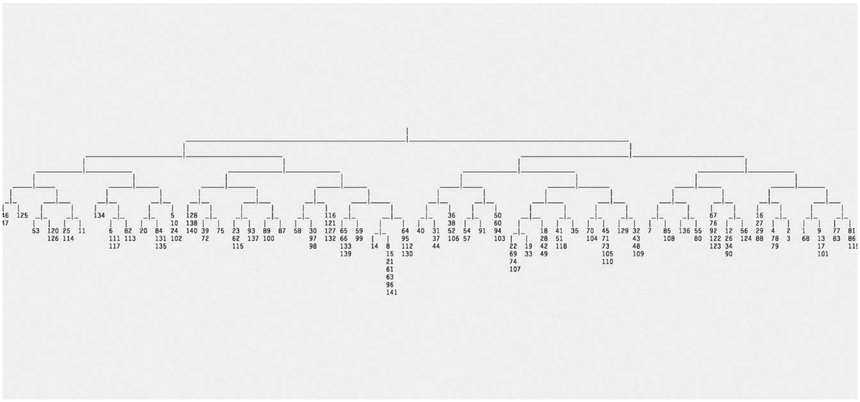


Fig. 7: Output of Python HIDECS 2: Hierarchical decomposition into subsets of the requirements for an Indian Village

than is done at present, so that the decision as to just where to apply precious and limited inventive power can be made. Fortunately, large computers and the techniques for data-processing have become generally available in the past decades."[32] Alexander defined the output of the hill-climbing procedure as "the program": a hierarchical decomposition of design requirements that could be tackled in a predetermined order, and which could be visualized as a tree. This program distributed design as cognitive labour so that intellectual resources could be used more effectively. Herbert Simon and his collaborator at Carnegie Mellon, political scientist James March had already applied a similar factorization to the whole organizational complex of bureaucracies and corporations, so that a number of nearly independent parts would make them "sufficiently simple to be encompassed by a human mind."[33] According to Simon, hierarchical structures could be found in biological form, in society and even in the structure of the universe. They defined an "architecture of complexity" that was either inherent in nature, or simply the result of human cognitive limits, placing non-hierarchical systems "beyond our

32 Chermayeff and Alexander, Community and Privacy, 161.

33 James G. March and Herbert A. Simon, Organizations (New York/NY: Wiley, 1958), 152.

capacities of memory or computation."[34] Organizational structures could become naturalized this way, either as an inherent property of the cosmos or as a category of cognition. This naturalization also made them prescriptive: corporations, urban proposals, and design practice should all follow this natural ordering principle. But it only took Alexander a couple of years to dismiss the hierarchical diagrams he had so vehemently defended and to present, with the same conviction, a "semi-lattice" as an abstract ordering principle in "A City is Not a Tree."[35] These semi-lattices were, unsurprisingly, nothing other than the output of the SIMPX and EQCLA programs of HIDECS 3, the successor to HIDECS 2.[36] In 1967, three years after the publication of *Notes* and on the other side of the Atlantic, Gilles Deleuze dissected what was then the dominant movement in continental philosophy in "How Do We Recognize Structuralism?" Deleuze proposed that what was specific to this movement was the introduction of the symbolic as a third order or regime, which mediated between the differentiation of the real and imaginary and became the substratum of both at the same time. In this symbolic regime, structural objects were defined by "elements which claim to account both for the formation of wholes and for the variation of their part." These elements have "neither extrinsic designation, nor intrinsic signification … they have nothing other than a sense: a sense which is necessarily and uniquely 'positional.'"[37] Deleuze's succinct description of structuralism explains the reason for Alexander and Simon's commitment to hierarchies and trees. The symbolic of mathematical formulas, programs, and software became the substratum of the real—the architecture of complexity—and of its incarnation into the imaginary in the form of projects and proposals. It also explains how easy it was to shift this commitment to new diagrams that, operating within the order of the symbolic, could

34 Herbert Simon, "The Architecture of Complexity," Proceedings of the American Philosophical Society 106, no. 6 (1962).

35 Christopher Alexander, "A City is Not a Tree" (paper presented at the Architectural Forum, 1965).

36 Alexander, HIDECS 3: Four Computer Programs.

37 Gilles Deleuze, Desert Islands and Other Texts, 1953–1974 (Los Angeles/CA: Semiotext(e), 2004).

restructure both the real and the imaginary, since their allegiance was not towards specific structures, but to the order of the symbolic manifest in the computer. Lazzarato has explained how sign machines like diagrams and programs "do not speak but function."[38] They make the symbolic operative, not as models or representations, but by producing and enacting the positional sense that Deleuze gave to structures. In the trees that are the output of HIDECS 2 (or the semi-lattices of HIDECS 3), this new sense of design is in its given position as a capacity irreducible to computation, but bounded and organized within a structure of control that distributes it as cognitive labour. Architects today do not use a design method that even Alexander renounced already in the preface to the 1971 edition of *Notes*. But schemes not too different from the diagrams of HIDECS 2 and HIDECS 3 now integrate architectural capacities into the workflows of building information modelling (BIM). According to its proponents in the software industry and architectural practice, BIM's unified digital models facilitate coordination between the increasingly large number of experts involved in building production, enabling "higher quality work, greater speed, and improved cost effectiveness for the design, construction, and operation of buildings."[39] Parametric design, another current digital design technology, has made complexity not just a problem to solve, but a sought-after effect; the generation of varying geometries implies impossible amounts of work if hand drawn, and their physical realization through robotic processes unachievable with traditional methods of mass production or by any artisanal means. Either as an unavoidable condition or as a desired characteristic, this complexity is today mediated through the same analytic and methodical factorization identified in HIDECS 2. In the case of BIM, through strategies for the management, distribution, and allocation of specialized tasks

38 Lazzarato, Signs and Machines, 115.

39 BIS Autodesk, "Building Information Modelling," Autodesk Inc. White Paper, San Rafael, CA (2002).

within the design and construction of a building; in parametric design, through the composition of procedures and algorithms (proprietary or often freely contributed by members of online communities) into digital workflows. Behind these current trends we can identify arguments similar to those of Alexander: that the use of computers leads to efficient ways of distributing design work, which is a necessary response to architectural production processes that are too complex to tackle without their mediation. The capacities that defined the figure of the humanist architect, disciplined through the different institutions that regulate the instruction of these capacities and their application in practice, have been factorized and redistributed into new positions by the computer, positions that are often literally those in front of a CAD terminal. These technical systems and structures distribute and integrate increasingly specialized competences to resolve what is presented as the inhumanly complex task of design. The insertion of this symbolic order of the computer also leaves architecture at the mercy of any future reorganizations, which tend to follow a logic of technological progress and obsolescence. Oblivious to the capacities of software as an operational ideology, we submit to the repositions and distributions they perform on us, and which are their "sense without meaning."

Bibliography

Alexander, Christopher. "A City is Not a Tree." Paper presented at the Architectural Forum, 1965.

——. *HIDECS 3: Four Computer Programs for the Hierarchical Decomposition of Systems Which Have an Associated Linear Graph*. Cambridge/MA: School of Engineering, Massachusetts Institute of Technology, 1963.

——. *Notes on the Synthesis of Form.* Cambridge/MA: Harvard University Press, 1964.

——. "The Question of Computers in Design." *Landscape* 14, no. 3 (1965): 6–8.

Alexander, Christopher, and Marvin L. Manheim. *HIDECS 2: A Computer Program for the Hierarchial Decomposition of a Set Which Has an Associated Linear Graph.* Massachusetts Institute of Technology Civil Engineering Systems Laboratory Research Report. Cambridge/MA: School of Engineering, Massachusetts Institute of Technology, 1962.

Alexander, Christopher, Marvin L. Manheim and Massachusetts Institute of Technology. Department of Civil Engineering. *The Design of Highway Interchanges: An Example of a General Method for Analysing Engineering Design Problems,* research report. Cambridge/MA: Dept. of Civil Engineering, Massachusetts Institute of Technology, 1962.

Autodesk, BIS. "Building Information Modelling." *Autodesk Inc. White Paper, San Rafael, CA* (2002).

Bamford, Alice. "Mathematics and Modern Literature: Passages from 'Chalk and the Architrave.'" *New Left Review*, no. 124 (2020): 107–123.

Chermayeff, Serge, and Christopher Alexander. *Community and Privacy; Toward a New Architecture of Humanism*. Garden City/NY: Doubleday, 1963.

Deleuze, Gilles. *Desert Islands and Other Texts, 1953–1974.* Los Angeles/CA: Semiotext(e), 2004.

Derrida, Jacques. *Of Grammatology.* Baltimore/MD: Johns Hopkins University Press, 1976.

Eco, Umberto. *A Theory of Semiotics. Advances in Semiotics.* Bloomington/IN: Indiana University Press, 1976.

Ford, Lester Randolph, and Delbert R. Fulkerson. "A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem." *Canadian Journal of Mathematics* 9 (1957): 210–218.

Frege, Gottlob. "Über Sinn und Bedeutung." *Zeitschrift für Philosophie und philosophische Kritik* 100 (1892): 25–50.

Haraway, Donna. "Situated Knowledges: The Science Question in Feminism and the Privilege of Partial Perspective." *Feminist Studies* 14, no. 3 (1988): 575–599.

Herbert, Simon. "The Architecture of Complexity." *Proceedings of the American Philosophical Society* 106, no. 6 (1962): 467–482.

Kittler, Friedrich A. “There Is No Software.” *Stanford Literature Review* 9, no. 1 (Spring 1992): 81–90.

Kuhn, Harold W. “The Hungarian Method for the Assignment Problem.” *Naval Research Logistics Quarterly* 2, no. 1/2 (1955): 83–97.

Lazzarato, M. *Signs and Machines: Capitalism and the Production of Subjectivity.* Translated by Joshua David Jordan, Los Angeles/CA: Semiotext(e), 2014.

Manheim, Marvin L. “Problem Solving Processes in Planning and Design.” *Design Quarterly*, no. 66/67 (1966): 31–39.

March, James G. and Herbert A. Simon. *Organizations*. New York/NY: Wiley, 1958.

Miller, George A. “The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information.” *Psychological Review* 63, no. 2 (1956): 81.

Motte, Warren F. and Oulipo. *OuLiPo: A Primer of Potential Literature.* Lincoln/NE: University of Nebraska Press, 1986.

Roussel, Raymond. *Impressions d’Afrique*. Paris: A. Lemerre, 1910.

Simon, Herbert A., and Allen Newell. “Heuristic Problem Solving: The Next Advance in Operations Research.” *Operations Research* 6, no. 1 (1958): 1–10.

Simon, Herbert A. *Models of Man: Social and Rational. Mathematical Essays on Rational Human Behavior in a Social Setting*, New York/NY: Wiley, 1957.

———. *The Sciences of the Artificial. Karl Taylor Compton Lectures*. Cambridge/MA: MIT Press, 1969.

Thompson, D’Arcy W. *On Growth and Form*. New ed. Cambridge: Cambridge University Press, 1942.

Upitis, Alise. “Alexander’s Choice: How Architecture Avoided Computer-Aided Design C. 1962.” In *A Second Modernism: MIT, Architecture, and the ‘Techno-Social’ Moment*, edited by Arindam Dutta, 474–505. Cambridge/MA: MIT Press, 2013.

→ CONTENTS

Bibliographic information published by the Deutsche Nationalbibliothek

The German National Library lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.dnb.de.

Universitätsverlag der TU Berlin, 2023

https://verlag.tu-berlin.de
Fasanenstr. 88, 10623 Berlin
Tel.: +49 (0)30 314 76131
E-Mail: publikationen@ub.tu-berlin.de

Editing: Daniela Petrosino
Proofreading: Clara Dick
Translation: Ben Liebelt
Picture Editing: Jürgen Schreiter, Darmstadt
Layout: Stahl R, www.stahl-r.de
Typesetting: Julia Gill, Stahl R
Print: docupoint GmbH

ISBN 978-3-7983-3270-6 (print)
ISBN 978-3-7983-3271-3 (online)

ISSN 2566-9648 (print)
ISSN 2566-9656 (online)

Published online on the institutional repository of the Technische Universität Berlin:
DOI 10.14279/depositonce-15964
http://dx.doi.org/10.14279/depositonce-15964

The critical concern of the book “Utopia Computer” is the euphoria, expectation and hope inspired by the introduction of computers within architecture in the early digital age. With the advent of the personal computer and the launch of the Internet in the 1990s, utopian ideals found in architectural discourse from the 1960s were revisited and adjusted to the specific characteristics of digital media. Taking the 1990s discourse on computation as a starting point, the contributions of this book grapple with the utopian promises associated with topics such as participation, self-organization, and non-standard architecture. By placing these topics in a historical framework, the book offers perspectives for the future role computation might play within architecture and society.

Universitätsverlag der TU Berlin
ISBN 978-3-7983-3270-6 (print)
ISBN 978-3-7983-3271-3 (online)