Accepted by: J. Computers and Operations Research, special issue on "Neural Networks"

Scheduling with Neural Networks — The Case of Hubble Space Telescope

M.D. Johnston*, H.-M. Adorf†

**Space Telescope Science Institute*
3700 San Martin Drive, Baltimore, MD 21218, USA
SPAN: STSCIC::JOHNSTON — INTERNET: johnston@stsci.edu

†Space Telescope — European Coordinating Facility
European Southern Observatory
Karl-Schwarzschild-Str. 2, D-8046 Garching, F.R. Germany
INTERNET: adorf@eso.org — SPAN: ESO::ADORF

Scope and Purpose: This paper shows how artificial neural networks can be seamlessly merged with unique evidential reasoning techniques to form a flexible and convenient framework for representing and solving complex scheduling problems with many "hard" and "soft" constraints. The techniques described form the core of SPIKE, an operational scheduling environment for long-range scheduling of astronomical observations with the orbiting NASA/ESA Hubble Space Telescope. The methodology of SPIKE, which is currently being adapted to some other space missions, is fairly general and could be brought to bear on a wide range of practical scheduling problems.

Abstract: Creating an optimum long-term schedule for the Hubble Space Telescope is difficult by almost any standard due to the large number of activities, many relative and absolute time constraints, prevailing uncertainties and an unusually wide range of timescales. This problem has motivated research in neural networks for scheduling. The novel concept of continuous suitability functions defined over a continuous time domain has been developed to represent soft temporal relationships between activities. All constraints and preferences are automatically translated into the weights of an appropriately designed artificial neural network. The constraints are subject to propagation and consistency enhancement in order to increase the number of explicitly represented constraints. Equipped with a novel stochastic neuron update rule, the resulting GDS-network, effectively implements a Las Vegas-type algorithm to generate good schedules with an unparalleled efficiency. When provided with feedback from execution the network allows dynamic schedule revision and repair.

Keywords: artificial neural networks — combinatorial optimization — constraint satisfaction problems — evidential/uncertainty reasoning — graph problems: maximum independent set, minimum vertex cover — heuristic search — nonmonotonic reasoning — scheduling — stochastic algorithms

One of the great mysteries in the field of combinatorial algorithmsis the baffling success of many heuristic algorithms.

—R.M. Karp 1975

If there are connectivity structures that are good for particular tasks that the network will have to perform, it is much more efficient to build these in at the start.

—D.H. Ackley, G.E. Hinton & T.J. Sejnowski 1985

1. Introduction

In many domains limited resources have to be optimally utilized and the construction of good schedules is often the principal means of achieving this goal. Efficient scheduling is of great economic importance in the business world, with significant problems arising in manufacturing and factory operations, transportation, and project planning, to name only a few. In other fields scheduling has less of a direct economic impact, but is still of critical importance. The problem we are addressing here is that of optimizing the scientific return of Hubble Space Telescope (HST), a unique multi-billion dollar international space observatory. Similar problems arise for other space- and ground-based observatories, particularly when the coordinated scheduling of these facilities is considered (see, *e.g.*, Johnston 1988a, b, c).

Over the years many scheduling techniques have been studied and developed (see *e.g.* King & Spachis 1980; Bellman *et al.* 1982; French 1986). However, because the general scheduling problem is an NP-hard combinatorial optimization problem (COP) (Ullman 1975; see also Garey & Johnson 1979), large problems still present enormous practical difficulties. The discovery that artificial neural networks could be used to attack complex combinatorial optimization problems (COPs) (Hopfield 1982, 1984; Hopfield & Tank 1985, 1986; Tank & Hopfield 1986; see however Wilson & Pawley 1988) has raised interest in the potential use of these networks for scheduling. Such an approach is appealing because neural networks are intrinsically parallelizable and could in principle be used to solve large problems (for a review see Adorf 1989 and references therein).

In the recent past neural networks have been considered for a variety of scheduling problems, including adaptive control of packet-switched computer communication networks (Mars 1989), integrated scheduling of manufacturing systems (Dagli & Lammers 1989), optimization of parallelizing compilers (Kasahara 1990), planning and scheduling in aerospace projects (Ali 1990), real-time control systems for manufacturing applications (Smith *et al.* 1988) and space mission scheduling (Gaspin 1989).

Artificial neural networks have been applied to delivery truck scheduling (Davis *et al.* 1990), dynamic load balancing (Oglesby & Mason 1989), job sequencing (Fang *et al.* 1990), job-shop scheduling (Foo & Takefuji 1988a, b, c; Zhou *et al.* 1990, 1991), large-scale plant construction scheduling (Kobayashi & Nonaka 1990), load-balancing in message-passing multiprocessor systems (Barhen *et al.* 1987b), non-preemptive, precedence-constrained process scheduling for a single server system (Gulati *et al.* 1987), planning, long-term and real-time scheduling of industrial production processes in a steel plate mill (Li-wei Bao & Yong-zai Lu 1990; Yong-zai Lu *et al.* 1990), precedence-constrained task scheduling on multiprocessors (Price & Salama 1990) real-time flexible manufacturing system (FMS) scheduling including learning (Rabelo & Alptekin 1989a, b), real-time load-balancing of multiprocessors on a mobile robot (Barhen *et al.* 1987a, c), real-time scheduling with specialized hardware (Ae & Aibara 1990), resource allocation in changing environments in the context of aircrew training scheduling (Hutchison *et al.* 1990), routing and load-balancing in large-scale packet-switching computer communication networks (Iida *et al.* 1989), school timetable construction and optimization (Gislen *et al.* 1989; Gianoglio 1990; Yu 1990), shared antenna scheduling of low-altitude satellites (Bourret *et al.* 1989), and student scheduling (Feldman & Golumbic 1989; 1990).

While a number of these investigators have developed artificial neural network *representations* of scheduling problems, there has not emerged any consensus on good network *dynamics* which would permit their application to large-scale problems involving thousands to tens of thousands of activities.

In this paper we describe the results of our work on an artificial neural network approach to complex, large-scale scheduling problems, which has led to the implementation of an operational system for scheduling observations with the NASA/ESA Hubble Space Telescope. Our approach integrates several novel key elements:

- a quantitative representation of "hard" constraints and "soft" preferences which exploits evidential reasoning techniques and which can be automatically translated into the biases and connection weights of a neural network,
- a network topology using multiple asymmetrically-coupled networks with different time constants to represent and enforce certain types of strict scheduling constraints, and
- a network dynamics consisting of discrete, deterministic neurons with a stochastic selection rule.

One of the major advantages of our approach is that no network training is required: instead, the network is designed entirely and automatically from the scheduling constraints, both strict and preference. A second advantage follows from our use of a discrete (in fact binary) stochastic network instead of a continuous deterministic one: this permits the network to run efficiently even on standard serial machines.

The organization of the paper is as follows: The HST scheduling problem is subject to a number of refinements and transformations until it takes the form of a neural network which can be simulated on a serial computer to find feasible solutions for the original problem. We begin with an informal, qualitative overview of the HST scheduling problem (§2) and then distill a formal, quantitative description of constraints using the novel concept of continuously valued *suitability functions* over a continuous time domain (§3). Suitability functions are capable of simultaneously representing "hard" constraints and "soft" degrees of preferences and even uncertainties. Techniques from evidential reasoning are invoked to combine constraints and preferences. Since the neural network requires a discrete time formulation of the scheduling problem, a discussion of suitability function sampling is included here. The formal description of constraints and preferences (§3) allows the generic HST scheduling problem to be cast into a concise mathematical model — a conventional 0-1 programming problem — specified by a number of equalities, inequalities and an objective function to be optimized (§4). These serve as the basis for representing the scheduling problem as a weighted constraint graph (§5) augmented by a special purpose "guard graph", which together form the topology of the neural network. Adding a heuristic, sequential, stochastic neuron update rule (§6) completes the description and provides a dynamic neural network for searching for solutions to the scheduling problem. The resulting neural network forms the core of SPIKE (§7), the operational long-range scheduling system for Hubble Space Telescope. Some experience gained with the system and some considerations on the general applicability of our approach (§8) conclude the paper.

2. Scheduling the Hubble Space Telescope

2.1. HST and scheduling overview

Hubble Space Telescope (HST) is a large satellite observatory launched by the Space Shuttle in April 1990. It consists of a reflecting telescope with a primary mirror of 2.4m diameter which focuses light onto an array of five scientific instruments: two imagers, two spectrographs, and a photometer. As a result of lack of interference by the Earth's atmosphere, the resolution, sensitivity, and ultraviolet wavelength coverage of HST are considerably greater than those obtainable with ground-based telescopes. During its nominal mission lifetime of 15 years HST is expected to significantly increase our understanding of a wide range of astronomical objects and phenomena, ranging from bodies in our own solar system to the most distant galaxies. HST science operations are conducted at the Space Telescope Science Institute (STScI) on the campus of Johns Hopkins University in Baltimore, with command and data interfaces to the spacecraft through NASA's Goddard Space Flight Center.

Shortly after launch it was discovered that the main mirror of the telescope had been incorrectly figured, thus reducing the sensitivity of HST from its original design goals. In spite of this, the resolution of HST remains considerably better than that of any ground-based observatory, and plans are being made now to install corrective optics during a 1993 Space Shuttle service mission. Although the mirror problems have delayed the start of full-scale operation of the observatory, they have not changed the nature or importance of optimum scheduling of telescope operations.

HST is operated as a guest observatory. Astronomers from around the world submit proposals to conduct scientific investigations with HST. Following a peer review and selection process, successful proposers prepare and furnish to STScI the details of their scientific programs, including the specific exposures desired and any scientific constraints on how they are taken. These proposals are submitted in machine-readable form over computer networks, and, following an automatic error- and feasibility-checking step, are stored in a central proposal database (Jackson *et al.* 1988). The first general proposal solicitation was completed in October 1989, and the process will be repeated annu-

ally throughout the HST mission. In order to provide scheduling flexibility and to guarantee maximum observatory usage, approximately 20% more proposals are accepted than can reasonably be expected to be executed.

2.2. HST scheduling constraints

Proposers can specify a wide variety of constraints on their exposures in order to ensure that their scientific goals are achieved. The most common constraints are relative timing requirements, *e.g.* exposure ordering, minimum or maximum time separations, interruptability, and repetitions. These types of constraints are common to other scheduling problems as well. Most are strict in the sense that they must be satisfied by every schedule. Others may be treated as preferences, *i.e.* they can be relaxed if necessary to obtain a feasible schedule.

Proposers may also constrain their exposures by specifying the state of the spacecraft and instruments (in absolute terms, or relative to other observations) and on the environmental conditions that must obtain when the exposures are taken. For example, it may be desirable to orient the telescope in a particular way in order to place a target into a spectrograph slit. It is also common to define "contingent" (*i.e.* conditional) exposures, to be taken only if precursor observations reveal features of sufficient interest. At HST's resolution, targets are often not precisely identifiable from the ground and in these cases target acquisition exposures must be scheduled. Most of the instruments also require various calibration exposures to be taken before or after science observations.

In addition to proposer-specified constraints, there are a large number of implied constraints on the HST schedule: these arise from operational requirements on the spacecraft and instruments, many of which are derived from the low orbital altitude of HST and consequent short (approximately 95 minute) orbital period. Because of the frequent earth blockage, only about 40 minutes of each orbit on average are available for data collection. There are also high-radiation regions over the South Atlantic where the instruments cannot be operated. Other constraints apply particularly to faint targets, where interference from stray and background light (*e.g.* scattered sunlight from the Earth's atmosphere, the Moon, and from interplanetary dust) must be minimized.

Policy and resource constraints significantly complicate the scheduling problem. Policies refer to conditions that must be satisfied globally by the schedule to ensure that the overall distribution of observing time constitutes a balanced program. Resource constraints are of several types: in addition to the obvious limitation on how much observing time is available, other resources are limited and must be allocated during the scheduling process. For example:

- The HST ground system is limited in how much data it can handle in any given 24-hour period.
- There is limited onboard computer storage for commands.
- The tape recorders have limited capacities for storing data.
- Only about 20 minutes per orbit of high-speed communications with the ground is permitted, due to limited access to the Tracking and Data Relay Satellites (TDRSs) through which all communications are routed.
- Only about 20% of all observations can be scheduled with real-time observer interaction.

Scheduling is further complicated by the fact that not all constraints are accurately predictable. For example, the motion of HST in its orbit is perturbed by atmospheric drag to the extent that the precise in-track position of the telescope is not predictable more than a few months into the future (in contrast, the plane of the orbit is predictable to within a few degrees for many months). Guide stars for pointing control are selected on the ground, but stars that appear single on the ground may in fact be multiple and thus be unusable by the interferometric detectors used to maintain pointing stability.

HST is operated almost entirely in a preplanned mode. Communication contacts through the TDRS network must be requested a few weeks before observations are taken and cannot easily be changed. There is very limited onboard command memory and almost no onboard decision-making capability: command sequences must be fully pre-specified. The process of generating the final command loads for uplink is time-consuming, since every precaution must be taken that the commands are correct and safe. There is a limited ability for astronomers to interact directly with the telescope while their observations are being taken, but this is restricted to small target pointing adjustments and instrument configuration changes (*e.g.* selection of the proper optical filter).

In spite of the preplanned operation of HST, the scheduling process must be able to react to unplanned changes. New observing programs ("targets of opportunity") may arrive within the scheduling period, requiring a schedule revision. The execution of a scheduled observation may fail, *e.g.* due to a failed guide star acquisition, or a loss of guide star lock, or an instrument or spacecraft anomaly. Also the availability of communication links is not assured

ahead of time with certainty, and may require an observation to be rescheduled at the last minute. These factors make HST scheduling both dynamic and stochastic: the set of tasks to schedule changes in an unpredictable way.

Although the detailed schedule of observations must (at least nominally) be complete about two months before execution, in fact it is necessary to schedule much further in advance: this is because, for scientific reasons, constraints among observations often extend over many months (in extreme cases up to several years) and it is necessary to ensure that all components of an extended sequence are in fact schedulable. It is also necessary to give notice sufficiently in advance to those astronomers who need to plan to visit the STScI when their observations are being executed.

Typical long-range HST schedules will cover approximately a one- to two-year interval at a time resolution of about one week. The schedule is refined to a greater and greater level of detail as execution approaches. Such a schedule includes some 10,000-30,000 exposures, each participating in a few to several tens of constraints. Exposures on the same targets are grouped to the maximum extent possible, thus reducing the number of individual activities to schedule by a factor of 5-10. Many of the absolute-time constraints are periodic with different periods and phases: *e.g.* occultation by the Earth, proximity of the dark or sunlit Earth to the telescope field-of-view, passage through high radiation regions, regression of the orbital plane, and lunar and solar interference with observations of a given target (see Fig. 2-1). As a consequence, there are generally several opportunities to make any particular observation, and part of the scheduling problem is to make an optimal choice among these opportunities for as many observations as possible. The interaction of many constraints on varying timescales makes it impossible to identify any single dominant scheduling factor.

For problems of this size and complexity it is more important to devise computationally efficient "satisficing" algorithms than to have algorithms which may be guaranteed to find optimal solutions but which exhibit poor average-case time behavior. Practical limitations on computational resources remain a major factor influencing algorithm development. These limits can not be significantly raised just by applying faster computers (see Garey & Johnson 1979, pp. 7–8).

2.3. Scheduling goals

The most important general scheduling goal for HST is observatory *efficiency*: as many observations as possible should be scheduled within the interval under consideration. However, schedule efficiency must be balanced against several other important factors:

- **schedule quality**: scheduling to maximize quality is concerned with placing observations at times which maximize the quality of the data obtained. While it may be possible to schedule a given observation at some particular time, such a time may in fact be a poor choice given the specific nature of the observation. For example, an observation sensitive to background light can be scheduled when the background light level is high, at the expense of either more noise in the data or an increased exposure time.
- **schedule risk**: scheduling to avoid risk attempts to minimize the impact of unpredictable factors in the schedule. An example of this might be scheduling observations when there are multiple pairs of guide stars available, so that the impact of any single pair being unusable (which could cause the observation to fail) is minimized.
- **observation priority**: observations are divided into different priority levels, based on anticipated scientific return. All else being equal, it is more important to schedule higher priority observations.

The goals described above refer primarily to predictive scheduling. In reactive scheduling, which modifies an adopted schedule based on feedback from execution, there is another goal to consider: minimizing the disruption to the ongoing schedule. This is particularly important for observations which require observers to make travel plans in advance, or for those on which numerous future observations depend (Sponsler & Johnston 1990).

In order to compare and rank different schedules it is of course necessary to quantify these general scheduling goals. In SPIKE these factors are modeled in terms of *suitability functions* which describe the absolute and relative time dependence of scheduling constraints and preferences and provide the basis for schedule optimization.

2.4. Summary

The outline above provides a flavor of the complexities arising in the HST scheduling problem, succinctly characterized by the task of scheduling over an extremely large time span a very large number of prioritized activities of potentially variable duration, which are restricted by a multitude of absolute time restrictions and are related to each other by a significant number of relative time relationships, both strict and preference. The problem of scheduling

the operations of HST's several non-identical science instruments ("processors") with their individual set-up times is further aggravated by the existence of time-variable resource requirements and resource bounds. In its general form, the HST scheduling problem belongs to the most difficult class IV (King & Spachis 1980) of *dynamic, stochastic* scheduling problems. Further details about HST scheduling are provided in Johnston (1987, 1989a, 1990); Miller *et al.* (1987, 1988); Johnston & Miller (1988, 1990); Miller & Johnston (1991). A description of the HST proposal submission and scheduling process is given in Adorf (1990).

3. Scheduling constraints and preferences

The scheduling constraint framework described in this paper was designed not just to schedule HST observations but to handle a general class of scheduling problems. We focus on the problem of scheduling a set of activities A_i ($i=1,\ldots,N$) of fixed durations d_i over the time interval $[t_A,t_B]$ subject to a set of constraints $\{C_\alpha\}$. The constraints, which will be discussed more fully below, convey two major types of information to the scheduler:

- **feasibility** constraints specify conditions or times when activities may or may not be scheduled. We interchangeably use the term "strict constraint" for this type, as they may not be violated under any circumstances. A few examples in the HST scheduling context are:
  - Never schedule an observation when the Sun is within 35° of the target
  - Make sure that instrument calibration occurs within 24 hours of the science observation
  - Never schedule simultaneous activities that would require more than the total power available
- **preference** constraints specify quality judgements (based on objective or subjective factors) on conditions which are preferred to obtain in the final schedule. In the context of HST scheduling, preference constraints occur in a wide variety of forms, for example:
  - Schedule observations preferably when the background noise is minimized
  - Schedule two particular observations as close together as possible

It is important that both feasibility and preference information be considered *simultaneously* during schedule construction: ignoring feasibility constraints can obviously lead to unimplementable schedules, but ignoring preference constraints (in order to simplify the problem) can lead to grossly suboptimal schedules. For this reason we have developed the concept of *suitability functions* (Johnston 1989b) as an amalgam of two well-studied frameworks, namely *constraint satisfaction problems* (CSPs) for satisfying feasibility constraints, and *evidential reasoning* techniques as a means to combine preference constraints.

3.1. Constraint satisfaction, weight of evidence, and suitability functions

If the role of preferences is ignored, the basic scheduling problem can be cast into the form of the following CSP:

> Given a set of N activities $A_1,\ldots, A_N$ to be scheduled over the interval $[t_A,t_B]$, and a set of constraints $C_\alpha(A_i, A_j, \ldots)$; find an assignment of all activities A_i to times $t_i \in [t_A,t_B]$, such that all constraints are satisfied.

In this formulation a constraint $C_\alpha(A_i, A_j, \ldots)$ is simply a subset of the Cartesian product $[t_A,t_B] \times [t_A,t_B] \times \ldots$ which specifies combinations of values (in this case times) which are compatible or incompatible with each other. CSPs on discrete domains arise in a variety of applications and methods for solving them have been widely studied: for a recent review see Meseguer (1989).

There are two aspects of the scheduling problem that limit the direct applicability of discrete CSP methods: the time domain is continuous (cf. Rit 1986, Sadeh & Fox 1988, Dechter *et al.* 1989), and preferences are ignored, *i.e.* only strict constraints are considered. The latter point is especially important, since in the HST domain, as in many other scheduling problems, it is not enough to find feasible schedules: preference constraints must be satisfied to the greatest extent possible.

Consider the scheduling of an activity A_i, given that activities $A_{j\neq i}$ are already scheduled at times t_j. A human scheduler would assess the opportunities for scheduling A_i at various times by considering the implications of the constraints on A_i. These constraints might take a variety of forms, but can be generally be cast into statements of the following general type:

Given that activities A_j... are scheduled at times t_j..., the *degree of preference* for scheduling activity A_i at time t due to constraint C_α is $W_{i\alpha}(t_1,\ldots,t_{i-1},t_i=t,t_{i+1},\ldots,t_N) \equiv W_{i\alpha}(t;t_{j\neq i})$.

Here the dependence on $t_{j\neq i}$ can be dropped for constraints depending only on t, *i.e.* which are independent of when other activities are scheduled.

The degrees of preference represented by $W_{i\alpha}(t;t_{j\neq i})$ can be assigned over some numerical range based on the value judgment of the importance of the constraint, with larger values of $W_{i\alpha}$ corresponding to greater preference. $W_{i\alpha}$ can represent deterministic constraints, or some classes of intrinsically unpredictable constraints, *e.g.* $W_{i\alpha}$ can also be formulated in terms of (a function of) the probability that some desirable condition will hold. The latter is particularly applicable to preference constraints concerned with minimizing risk.

To properly assess the possible scheduling times for an activity A_i it is necessary to *combine* in some manner the degrees of preference $W_{i\alpha}$ derived from all applicable constraints. This combination process is formally similar to that employed in a number of rule-based expert systems which assess evidence for and against various conclusions. While this approach to uncertainty reasoning is known to have its limitations — in particular, the knowledge base should form a tree so that no evidence is counted twice via alternative paths of reasoning (Pearl 1988) — it is presumably adequate for most scheduling problems and has the advantage of being computationally feasible.

However, the techniques used in rule-based systems for evaluating evidence for or against discrete conclusions cannot be applied directly to scheduling, since a *continuum* of scheduling conclusions must be considered (*e.g.*, schedule A_i at t_i and A_j at t_j, etc.). What is required in this case is a "continuum" version of uncertainty reasoning, formulated in a way which efficiently expresses the variety of constraints that typically appear in these problems and which retains information about choices that affect schedule optimality. Such a formulation, called *suitability functions*, has been developed and is motivated and described in detail in Appendix A. For the purposes of the following discussion, the key point is that we can define a *suitability function* $S_i(t)$ whose non-negative value represents approximately the degree of preference of scheduling activity A_i at time t. $S_i(t)$ is defined by multiplicatively combining suitabilities $S_{i\alpha}(t)$ determined by each constraint C_α which acts on A_i, along with any scheduling decisions represented by a restriction operator $R_i(t)=0$ for excluded times, 1 otherwise. A suitability value of zero at time t has the special meaning that scheduling A_i at t would violate a strict constraint. Larger positive values indicate a greater degree of scheduling preference.

All of the conventional binary temporal interval relationships (before, after, during, etc.: see Allen 1983) are easily represented by appropriate suitability functions, along with a large class of far more general *temporal couplings* (Shapiro 1980). Both temporal constraints based on metric time and general relative time preference constraints can be represented and combined in a straightforward manner.

The suitability function framework is also capable of modeling *imprecision*, as is encountered *e.g.* in the context of airline crew scheduling in the minimum time a crew needs to change planes or in the statement of work rules. The concept of real-valued suitability functions over a continuous time-domain introduces enough flexibility to adequately model many real-world scheduling situations, thus helping to avoid manual intervention (see Gianessi & Nicoletti 1979, p. 391).

3.2. Time discretization and suitability function sampling

The neural network approach to be described below requires a *time-discretized* representation of the continuous scheduling problem considered so far where a *scheduling decision* denotes the assignment of activity A_i to some subinterval I_m of $[t_A,t_B]$, or possibly to some specific time within $[t_A,t_B]$. It is therefore necessary to consider how to discretize the representation of time (unless there exists some natural time discretization in terms of which the constraints can be defined). This requires a choice of how to treat the problem of sampling an intrinsically continuous quantity. As a general rule, the sampling-interval must be less than the timescale for significant changes in the scheduling constraints. If this condition is satisfied, then one has to decide upon a suitable *sampling procedure* defining how to treat those strict constraints that would prevent the scheduling of an activity over some, but not all, of a given interval I_n. The basic choice is whether to exclude the entire interval or not:

(a) If the entire interval is excluded, then there is a risk that feasible solutions may be missed.

(b) If the interval is *not* excluded, then the scheduler may find what appears to be a feasible configuration, but which turns out not to be feasible when the timing is examined in detail.

In HST scheduling we have generally chosen option (b) for sampling suitability functions, but the choice must be determined for each problem type based on the characteristics of the constraints and the difficulty of dealing with the consequences. Often, there will be a natural time unit in terms of which constraints are defined, so that no sampling error will occur.

In the following we denote by $E_m[f(t)]$ the sampled value of the quantity f(t) over the mth time interval I_m, and $E_{mn}[f(t_1,t_2)]$ to be the sampled value of a function $f(t_1,t_2)$ where t_1 ranges over I_m and t_2 over I_n. The appropriate definitions of the sampling operators E_m and E_{mn} depend on the choice of interval exclusion above, and on an averaging approach for variations of f(t) over the interval. In practice we have defined E_m to be simply the mean value of f(t) over the interval, and $E_{mn}[f(t_1,t_2)]$ to be:

$$E_{mn}[f(t_1,t_2)] = \max_{\{t_1 \in I_m\}} \left\{ f(t_1,t_2)\Big|_{t_2=\min\{I_n\}} \ , f(t_1,t_2)\Big|_{t_2=\max\{I_n\}} \right\} \tag{3-1}$$

where $\min\{I_n\}$ and $\max\{I_n\}$ are the earliest and latest times, respectively, in the interval I_n. Alternative definitions of the sampling operators E_m and E_{mn} could be, *e.g.*, to take the values of their arguments at the midpoints of the intervals.

Unary constraints

Unary constraints apply to single activities only, *i.e.* they are independent of other activities and the times at which they are scheduled. They therefore depend on absolute time only and determine the "bias" values b_{im} for an activity A_i. Consider the contribution of all unary constraints to the total suitability for activity A_i in the absence of any scheduling restrictions (*i.e.* $R_i(t)\equiv 1$):

$$S_i^{unary}(t) = \prod_{\alpha} S_{i\alpha}^{unary}(t) \tag{3-2}$$

In order to later (§5.2) combine the constraint and preference weights in a simple additive neuron, we must both sample the suitability and convert it back to an additive form. The conversion is straightforward provided that a sampling operator has been chosen and that small suitability values are treated appropriately:

$$b_{im} = \begin{cases} \ln \dfrac{E_m[S_i^{unary}(t)]}{S_o} & \text{if } E_m[S_i^{unary}(t)] \geq S_o, \\ -b_o & \text{otherwise} \end{cases} \tag{3-3}$$

where $E_m[S_i^{unary}(t)]$ denotes the appropriate sampled value of $S_i^{unary}(t)$ over the schedule subinterval I_m, S_o is the minimum suitability value to be considered as schedulable, and b_o is the negative of the bias value to be associated with an unschedulable interval.

Note that the bias is non-negative only if the sampled suitability is $\geq S_o$. This has the straightforward interpretation that negative biases indicate unschedulable subintervals, and that larger positive bias values represent higher suitability values, *i.e.* intervals with higher preference. The threshold of schedulability can be modified by adjusting S_o: higher values cause exclusion of low suitability intervals, at the possible expense of excluding feasible solutions.

Binary constraints

Binary activity constraints are derived from suitability functions which specify the impact of scheduling one activity on another. Let $S_{i\alpha}(t_i;t_j)$ be the suitability of scheduling activity A_i at time t_i given that A_j is scheduled at t_j, and $E_{mn}[S_{i\alpha}(t_i;t_j)]$ be the sampled form of this expression, where t_i and t_j are taken to range over times in the intervals I_m and I_n, respectively. Then, for each such constraint C_α, the following term is computed and collected additively in a "weight" matrix:

$$W_{im,jn} = W_{im,jn} + \begin{cases} \ln E_{mn}[S_{i\alpha}(t_i;t_j)] & \text{if } E_{mn}[S_{i\alpha}(t_i;t_j)] \geq S_o \text{ and } \neq 1, \\ -\omega_o & \text{if } E_{mn}[S_{i\alpha}(t_i;t_j)] < S_o \end{cases} \tag{3-4}$$

Note that E_{mn} should be defined so that the weights become symmetric: $W_{im,jn} = W_{jn,im}$.

Higher-arity constraints

While higher order constraints (involving three or more activities) can be represented by suitability functions, they are not explicitly converted to biases or connection weights in our representation. Instead, a general mechanism for handling constraints of this type has been developed based on the introduction of appropriate "hidden variables". This technique is described below in §5.

4. Mathematical model of the HST scheduling problem

We will now state an approximate abstraction of the HST scheduling problem. The formulation we derive will take the form of a nonlinear 0-1 integer programming problem with a set of linear equality and inequality constraints and a quadratic objective function. This form is particularly well-suited for comparisons with other abstract combinatorial optimization problems (COPs) in the literature. As we will see (§5), this description is also directly translatable into the topology (*i.e.* the static structure) of the desired neural network.

4.1. Formal description

We assume that the scheduling period is discretized into M time intervals. Following the classical CSP-approach we could associate with each activity a multi-valued state variable specifying the start time of the activity. Instead, we associate with each activity A_i a vector of M binary-valued variables y_{im}. For a feasible commitment of A_i at most one of the variables y_{im} can be set to 1; the others have to be 0. (As we will see later (§5.2), the variables y_{im} can be directly identified with the outputs of a rectangular array of binary neurons. We will therefore refer to collections $\{y_{im}: 1 \le m \le M, \text{ i fixed}\}$ and $\{y_{im}: 1 \le i \le N, \text{ m fixed}\}$ as *rows* and a *columns* of neurons, respectively.) The collection of all y_{im} with defined values represent a *configuration*.

The abstract HST scheduling problem now forms the following COP: Given a set $\{A_i\}$ of activities A_i, a set $\{L_\rho\}$ of resource limits L_ρ, a number M of time units I_m, and

(i) for each activity $A_i \in \{A_i\}$ a set of unary temporal constraints b_{im},

(ii) for each resource $L_\rho \in \{L_\rho\}$ a set of temporal resource bounds (or capacity limits) Q^ρ_m,

(iii) for each activity $A_i \in \{A_i\}$ and each resource $L_\rho \in \{L_\rho\}$ a set of temporal resource requirements q^ρ_{im}, and

(iv) for each pair of activities A_i and A_j a set of binary temporal constraints $W_{im,jn}$.

The task is to find a feasible, non-preemptive schedule $\sigma: \{A_i\} \rightarrow \{I_m\}$, specified by a set of assignments of activities to starting times, *i.e.* an assignment of values 0 or 1 to the state variables y_{im} with $m=\sigma(i)$, that:

(1) maximizes the total utility

$$U = \frac{1}{2} \sum_{m,n=1}^{M} \sum_{i,j=1}^{N} W_{im,jn}\, y_{im}\, y_{jn} + \sum_{m=1}^{M} \sum_{i=1}^{N} b_{im}\, y_{im} \tag{4-1}$$

in such a way that the following equalities and inequalities hold:

(2) every activity is scheduled exactly once

$$\sum_{m=1}^{M} y_{im} = 1 \quad \text{(for all activities } A_i) \tag{4-2}$$

(3) no unary temporal (inhibitory) constraint is violated

$$\sum_{m=1}^{M} b^-_{im}\, y_{im} = 0 \quad \text{(for all activities } A_i) \tag{4-3}$$

(4) no binary temporal (inhibitory) constraint is violated

$$\sum_{m,n=1}^{M} \sum_{i,j=1}^{N} W^-_{im,jn}\, y_{im}\, y_{jn} = 0 \tag{4-4}$$

(5) no resource is overused $\sum_{i=1}^{N} q_{im}^{\rho} y_{im} \le Q_m^{\rho}$ (for all resources ρ and times m) (4-5)

In the formulation above we have adopted an *inhibitory* formulation of the CSP ($b_{im}<0$ and $W_{im,jn}<0$ represent incompatibilities), and have separated the general weight terms b_{im} and $W_{im,jn}$ into compatibility (superscript +) and incompatibility (superscript –) constraints:

$$W = W^- + W^+ \quad \text{and} \quad b = b^+ + b^-, \tag{4-6}$$

where componentwise

$$W^- = \min(W, 0)\,,\ W^+ = \max(W, 0)\,,\ b^- = \min(b, 0)\,, \text{and}\ b^+ = \max(b, 0)\,. \tag{4-7}$$

For a feasible solution, incompatibility constraints, which have to be satisfied under all circumstances, do not contribute to the total utility via their bias and weight terms.

4.2. Relation to conventional scheduling problem specifications

The mathematical model stated above provides a fairly general framework allowing the subsumption of various concepts found in formulations of more conventional scheduling problems. For instance, the different HST science instruments ("processors") can be viewed as resources of unit capacity. Activity release times, (time-dependent) temporal activity durations, due dates, overall or individual deadlines *etc.* can all be absorbed into suitably adjusted "hard" or, when relaxed, into "soft" unary constraint terms. Both the simple relative *precedence constraints* (Allen 1983) between activities and the more demanding *temporal couplings* (Shapiro 1980) can be incorporated into "inhibitory" binary constraints $W_{im,jn}$.

We emphasize that the general structure of our scheduling problem is described by a set of linear equalities, a set of linear inequalities and a quadratic objective function to be optimized. As such it is general enough to encompass a wide variety of scheduling problems, *e.g.* the restaurant crew-scheduling problem of Poliac *et al.* (1987), the mean-flow permutation job-shop scheduling problem of French (1986, p. 135), the job-shop scheduling problem of Foo & Takefuji (1988a, b, c), the vehicle routing problem of Christofides *et al.* (1979a) and the nondeadhead/deadhead airline crew scheduling problem of Gianessi & Nicoletti (1979, p. 391). Other COPs subsumed by our model above are the travelling salesman problem (see *e.g.* Gianessi & Nicoletti 1979) and the loading problem of Christofides *et al.* (1979b).

In passing we note that the neural network dynamics introduced below attempts to solve the maximum independent set (or maximum node packing) problem on the constraint graph. A number of real-world scheduling problem such as airline crew, railroad crew and truck delivery scheduling (see Balas & Padberg 1979, p. 153) can be formulated as set partitioning or set packing problems, which in turn can be transformed into equivalent, unique node packing problem on the intersection graph. Thus solving the latter problem solves the former ones.

5. From a set of constraints to the constraint graph

5.1. Constructing the constraint graph

Having cast our general scheduling problem into the form of a nonlinear programming problem, we now address the question of how to convert this representation into a *directed weighted constraint graph* G = (V, A) which will define the topological structure of the neural network. To this end the vertex set V of the graph G is identified with the set of 0-1 variables y_{im} in which the CSP is formulated. The unary "bias" terms b_{im} become vertex weights. For each non-zero component of the binary weight matrix $W_{im,jn}$, we add to the graph a pair of directed communicating arcs, of which the $W_{im,jn}$ become the weights. We can additionally "color" the arcs according to whether they represent compatibility (W^+) or incompatibility (W^-) constraints. Note that a feasible configuration of the original CSP is represented by an independent vertex set in the incompatibility constraint graph (*i.e.* the constraint graph restricted to arcs corresponding to incompatibility constraints). The concept of the constraint graph G = (V, A) is best illustrated by the small example of Appendix B.

5.2. Network biases and connections

The definition of a neural network structure with suitable topology, biases and connection weights is now merely one of nomenclature: We identify each vertex in the constraint graph G = (V, A) with a binary neuron, the state and bias of which are given by the associated 0-1 variable y_{im} and vertex weight b_{im}, respectively. Each arc with associated weight $W_{im,jn}$ in the constraint graph becomes a network connection with $W_{im,jn}$ its connection strength. Then the input of neuron m of row i, denoted by x_{im}, is given by.

$$x_{im} = \sum_{jn} W_{im,jn}\, y_{jn} + b_{im} \,. \tag{5-1}$$

A neuron selected for updating computes its state (= output) from its input via the following ("hard" or "high gain") step transfer function

$$y_{im} \Leftarrow \eta(x_{im}) = \begin{cases} 1 & x_{im} \geq 0 \\ 0 & x_{im} < 0 \end{cases} \,. \tag{5-2}$$

The neural network constructed so far possesses a symmetric connection matrix $W_{im,jn}$ and thus can be viewed as a feedback Hopfield network with an associated Lyapunov ("energy") function (Hopfield 1982; Goles 1987) equal to the negative of the total utility defined above

$$E = -U \,. \tag{5-3}$$

Note that if the auxiliary constraints were absent, the well-known standard dynamics for sequential binary Hopfield networks (Hopfield 1982) could be used to "animate" the constraint graph, effectively implementing an optimization algorithm for the unconstrained COP the network represents. Solutions would correspond to stable fixed points of minimum energy (= maximum utility).

5.3. Encoding auxiliary constraints

Encoding equality constraints

The conventional procedure for encoding an equality constraint into a network connection matrix consists of adding the equality (suitably rearranged so that it equates to 0) to the unconstrained energy function using a Lagrange multiplier (see *e.g.* Peterson & Söderberg 1989). While admissible in principle, this method has the disadvantage that the network, when equipped with standard Hopfield dynamics, will all too often converge to a stable fixed point of the dynamics which does *not* correspond to a global minimum of the energy function. In this case some of the hard constraints may be violated. No general method is known for adjusting the Lagrangian parameter so that the strict constraints are always fulfilled.

Noting that equality constraints can always be represented by a pair of inequality constraints, we set out to treat the former on the same footing as the more general inequality constraints. For the latter there is a representation method based on the introduction of "hidden variables" which avoids the problems frequently encountered with the Lagrangian method.

Encoding linear inequality constraints

The task consists of encoding linear inequality constraints of the type (Eqns. (4-2), (4-3) and (4-5))

$$\sum_m c_m y_m \leq K \quad \text{and} \quad \sum_m c_m y_m \geq K \tag{5-4}$$

into the neural network, where we have temporarily suppressed one neuron index.

For the generalized "at most K neurons" upper bound inequality constraint two principal encoding architectures are known (Fig. 5-1a, b):

(a) complete, and therefore symmetric, *lateral* ("reciprocal") *inhibition* or

(b) an asymmetrically-coupled, "hidden" *inhibitory guard neuron* enforcing the upper bound constraint in the set of neurons it supervises.

Method (a) is usually preferred for its symmetry-preserving property and is the one implemented in the GDS-network.

It would be desirable to also have a symmetric architecture for the generalized "at least K neurons" lower bound inequality constraint. However, it is unlikely that such a symmetric encoding exists and, following the example above, we therefore use an architecture (Fig. 5-2) with

(b') an asymmetrically-coupled, "hidden" *excitatory guard neuron* enforcing the lower bound constraint in the set of supervised neurons.

The connection weights from the supervised neuron to the supervising guard $W^G_{im,i}$ need only be negative to ensure that the guard is "off" when any neuron in the set is "on". Conversely, when the guard is "on", the input to each neuron should be large enough to overcome the lateral inhibition (if there is any) and other inhibitory input, even from many other neurons. We therefore set the corresponding weight $W^G_{i,im} = g_o$, with a positive constant g_o chosen such that $\omega_o \ll g_o < b_o$. By carefully arranging these network weights, their absolute values will populate three distinct, nonmixing regimes in the space of weights: a low level band for preferences, and medium and high level bands for upper and lower bound constraint enforcement weights, respectively. The appropriate design of the bands also guarantees that strict binary incompatibility constraints (Eqn. (4-4)) are always fulfilled in a feasible solution.

The implementation of all auxiliary inequality constraints in the way described here (Fig. 5-3) modifies the neural network in such a way that *the stable fixed points of the network dynamics correspond (bijectively) to network configurations in which all strict unary and binary constraints are definitely satisfied.*

6. Adding stochastic network dynamics

Having established the static network representation of the scheduling problem in form of a constraint graph augmented by an auxiliary "guard graph", we now have to equip the network with a suitable dynamics capable of delivering at least an approximation to an optimum solution of the COP at hand. Our interest in efficient and potentially parallelizable heuristic search algorithms has yielded the GDS-network described below, which has shown remarkable performance on a variety of CSPs and COPs.

6.1. The GDS-network — what is it?

The guarded discrete stochastic neural network, or GDS-network in short (Adorf & Johnston 1990), is an alternative to the well-known Boltzmann machine (see §6.5). It is a general, parallelizable, randomized, heuristic neural network algorithm suitable for a variety of constraint satisfaction and combinatorial optimization problems.

The static structure of the GDS-network consists of the two major components, the *main* and the *guard* constraint graphs, introduced above. The main network captures the fundamental structure of the COP including the objective function (total utility), whereas the guards enforce the (global) auxiliary constraints.

The GDS-network is a feedback neural network with a stochastic, sequential network update rule. For sequential neural networks, which change their neuron states one at a time, it is useful to introduce a neighborhood structure in the configuration space (Aarts & Korst 1989, p. 131) by calling two configurations *neighbors* if they differ by the activation state of exactly one neuron. The neighborhood structure induces a metric: two configurations are a distance *d* apart if it takes *d* neuron flips to transform one configuration into the other. A *locally optimal configuration* (Aarts & Korst 1989, p. 132) or simply *local optimum* is a network configuration where all neighboring configurations have a higher network energy, *i.e.* the energy cannot be lowered by changing just one neuron state.

As mentioned before, if the main network with its symmetric connection matrix, were equipped with the standard Hopfield dynamics and executed on its own (*i.e.* without any guards), it would settle in a stable fixed point of the network dynamics corresponding to a locally optimal "solution" of the encoded COP. However, local instead of global optimality may mean that a solution comprises one or more neuron rows with no neuron turned "on", which in the context of scheduling means that the corresponding activities are not scheduled at all.

Instead of the standard Hopfield dynamics, the GDS-network is equipped with the following heuristically motivated, sequential stochastic neuron update rule executed once per cycle in a network run (Adorf & Johnston 1990):

1. The set of all rows is determined which contain at least one neuron in an "inconsistent" state. (We call a neuron state inconsistent if its input would lead the neuron to change its state according to Eqn. (5-2).) If no inconsistent row is found, the algorithm stops.
2. A row of neurons is randomly selected from the set of inconsistent rows.

3. For each inconsistent neuron in the selected row the degree of inconsistency $|y_{im} - \eta(x_{im})||x_{im}|$ is determined. Here η denotes the (deterministic) neuron transfer function of Eqn. (5-2).
4. The neuron with maximal inconsistency is selected in that row — if more than one neuron is maximally inconsistent, one is picked at random — and is flipped, thus becoming consistent. This *max-inconsistency* heuristic makes the largest possible change in the network energy due to neuron transitions limited to the selected row.

Note that, in contrast to the Boltzmann-machine (§6.5), where an individual neuron is completely randomly selected for updating and then also reacts nondeterministically to its input, the GDS-network realizes a random automata network (Demongeot 1987) with a controlled stochastic neuron selection but a deterministic neuron update rule.

6.2. How does it work?

Usually the GDS-network is started in an "empty" configuration, *i.e.* all main neurons are switched "off", and consequently all guard neurons are "on". The network proceeds initially by turning neurons "on" and may either proceed directly to a solution or may encounter a row on which all neurons have conflicts with others already "on". In this case the max-inconsistency dynamics will cause the guard to force some neuron to transition from "off" to "on". Such a transition will, however, produce some other conflicts within the main network. The algorithm proceeds to try to resolve them by turning "off" neurons on the conflicting rows, then, in a separate step, turning some other neuron "on". This process proceeds under the control of the max-inconsistency heuristic and the network configuration irregularly oscillates between feasible and infeasible configurations (Adorf & Johnston 1990). In other words, the algorithm switches back and forth between construction work and damage repair. During a run the network spends most of its time attempting to resolve a few remaining consistency problems.

In a way the GDS-network's guard neurons can be viewed as external agents temporarily modifying the energy landscape of the main network. This has similarities to the approach taken by Jeffrey & Rosner (1986), who temporarily invert the energy function (*i.e.* every local energy minimum becomes a local maximum and vice versa) when the system configuration is trapped in a local energy minimum.

While the connection matrix of the main network is symmetric, the connection matrix of the total (main + guard) network is not. Consequently there is no guarantee that the network dynamics will ultimately converge to a stable fixed point, although, in practice, the network often fairly rapidly comes to rest in such a configuration. (The probability of non-convergence depends on the difficulty of the underlying CSP.) We have frequently observed that, instead of reaching a stable fixed point, the GDS-network converges to a "stable limit set" of configurations, comparable to the "limit cycle" of deterministic dynamic systems. The network oscillates between configurations within the limit set, but, because of its restricted stochasticity, ultimately cannot escape. Since the dynamics of neural networks with asymmetric connection weights is not yet well researched, not much more can be said at this point about the general behavior of such asymmetric networks.

In order to prevent infinite oscillations, the basic GDS-network has been augmented with a simple *stopping rule:* if the algorithm has not converged after a preset number of cycles, it is started over.

The basic GDS-network without stopping rule can be considered as a stochastic multi-start algorithm of the *Las Vegas* type: it either finds a solution or, with some low probability, announces to have failed to find an answer (cf. Johnson 1984, p. 437). If the GDS-network comes up with an answer, it is always a feasible solution. However if the network does not find a solution, it cannot be said whether the network just failed to find it or whether there exists no solution at all.

When truncated to run in polynomial time, the GDS-network represents a *Monte Carlo* algorithm in the following sense: if it stops *before* truncation, it has converged to a feasible solution. If it is stopped *by* truncation, the "no result" can be interpreted as "there exists no solution" with some probability that this conclusion is erroneous. Therefore using repeated runs (multistart) we can improve the certainty of this result to an arbitrary degree (or conversely refute it altogether). See Appendix C for a discussion of Monte Carlo and Las Vegas algorithms, and the use of the former to solve instances of decision problems via probabilistic classification.

6.3. Why does it work?

The success of a stochastic algorithm is not as easy to explain as that of a deterministic one. We offer the following explanation derived from observing the GDS-network on a variety of CSPs and COPs.

A major innovation of the GDS-network, when compared to the classical deterministic search algorithms is, apart from the built-in stochasticity, that the network frequently operates in the space of infeasible configurations. The

conflicts inform the network of bad decisions made earlier, and the unsystematic, almost "chaotic" way the algorithm proceeds often allows an immediate backjump to a bad assumption. The GDS-algorithm can thus be viewed as a type of constraint-directed search algorithm (Fox 1987).

Allowing the GDS-network to trespass into the domain of infeasible configurations provides it with a means for exploring, in the neighborhood of a given feasible configuration, promising configurations which are infeasible or would be quite a distance apart, had they to be connected through a path of feasible configurations. This effect can be described as *tunneling* from a feasible configuration to another feasible configuration through a forbidden region (Fig. 6-1).

6.4. How does it perform?

Developed in late 1987/early 1988 (see historical note in the Appendix D), the GDS-network showed a surprising performance on the standard N-queens benchmark problem, for which the standard, general network easily constructed solutions for N up to 1024. (This has to be compared with the published record of N=96 for solutions to the N-queens problem found in a performance comparison of general deterministic, backtracking search algorithms (Stone & Stone 1987).) Since then the GDS-algorithm's virtues have been explored on difficult problems, including NP-complete ones such as 3-colorability (Adorf & Johnston 1990). In our experience the GDS-network outperforms many other deterministic or stochastic search algorithms in terms of speed and quality of the solutions produced.

We have found that the performance of the GDS-network scales well with problem size. For instance, on N-queens problems the number of neuron transitions scales as 1.15N when starting with all neurons in their "off" state. For random constraint graphs (Dechter & Pearl 1988) the number of transitions scales linearly with the size of the problem. On HST scheduling the scaling is approximately quadratic in the number of activities to schedule (§7). Of course there are problems for which the network performance is worse than this: on certain types of 3-colorability problems the probability of convergence decreases exponentially with problem size (Adorf & Johnston 1990).

The heuristics embodied in the GDS-network have been analyzed by Minton *et al.* (1990) and successfully applied to other CSPs. They also find that starting with a good initial guess can significantly improve performance. Using a representation specially tailored to the N-queens problem, they have been able to find solutions on a workstation for N as large as 10^6.

6.5. How does it compare with the Boltzmann machine?

For comparison we implemented the standard Boltzmann machine (BM) algorithm within our general neural network representational framework. The BM (Hinton & Sejnowski 1983; Hinton *et al.* 1984; Ackley *et al.* 1985) arises when the concept of simulated annealing — independently proposed by Kirkpatrick *et al.* (1982, 1983) and Cerny (1982, 1985) as a general method for large-scale COPs — is applied to the neuron dynamics of a binary Hopfield network. (Following Aarts & Korst (1989, p. 126) "the basic idea underlying the Boltzmann machine, i.e the implementation of *local constraints as connection strengths in stochastic networks*", had previously been introduced by Moussouris (1974).)

In our BM-implementation all guard connections were disabled, since the BM is guaranteed to asymptotically settle into a global energy minimum by itself. We started our runs with a fairly high temperature, where neuron flips were practically totally random, and used a quasi-stationary cooling schedule, *i.e.* the temperature parameter was reduced at every neuron update cycle by some small amount (typically 1 per mille). The application of the BM to small-sized N-queens problems ($N \leq 16$) has been very discouraging. With neuron transitions occurring randomly all over the main network, the BM would either not settle quickly on a solution, or, particularly when we tried to speed up the cooling, it would all too often converge to a local instead of a global energy minimum.

In view of the apparent popularity of the BM-approach for large-scale COPs, further investigations using supposedly more efficient variants of the classical SA such as fast simulated annealing (FSA; Szu 1986; Szu & Hartley 1987) or mean field annealing (MFA; see *e.g.* Galland & Hinton 1991 and references therein) may be warranted.

6.6. What can it be used for?

Schedule construction for HST and various other types of scheduling problems is of primary concern within this paper. Here the network representation permits an easy updating of all events that actually have happened, thus allowing a convenient reactive scheduling and schedule repair by restart (Sponsler & Johnston 1990).

However, the range of problems that can be cast into the form of a neural network and solved with an appropriate neurodynamics such as the GDS-algorithm seems to be far more general than scheduling. In fact, any CSP-type or

propositional logic problem can be represented. Thus the neural network, when equipped with a suitable interface, forms a general tool for reasoning with "hard" and "soft" constraints. Since assumptions can not only easily be asserted but equally easily be retracted — the built-in constraint propagation mechanism guarantees consistency — the network can be used for nonmonotonic reasoning with uncertainty, quite similar to a general assumption-based truth maintenance system (ATMS, cf. de Kleer 1989). However, the CSP-formulation appears to be conceptually much less clumsy and the network representation to be minimal (in the sense that it is hardly conceivable how a more compact and more efficient representation than the adopted numeric one could be found) for the purpose of nonmonotonic reasoning with uncertainty.

7. SPIKE — an integrated part of the HST ground system

The framework described in this paper has been implemented in the workstation-based SPIKE scheduling system for long-range scheduling of Hubble Space Telescope observations. A brief description of the operation of SPIKE is described here, highlighting the use of the neural network scheduler in a practical application.

7.1. The flow of observing programs

As described in §2, HST observing programs (the "jobs") are prepared by astronomers and sent electronically to Space Telescope Science Institute where they are checked for errors and stored in a database (Jackson *et al.* 1988, Adorf 1990). When scheduling begins, programs are retrieved from the database and converted into a form useable by SPIKE. This process is called TRANSFORMATION (Rosenthal 1986; Rosenthal *et al.* 1986; Gerb 1991) and, for the purposes of SPIKE, consists of the following major steps:

- Exposures are aggregated where possible into *scheduling units* consisting of observations which should be done as a contiguous group. These usually observe the same target with the same instrument and could be scheduled separately only at a significant cost in observing efficiency. It is these scheduling units which correspond to the activities scheduled by SPIKE.
- Unary constraints on exposures such as those described in §2 are computed as suitability functions and combined for scheduling units as described in §3 and Appendix A.
- Temporal constraints on exposures are propagated to derive a path-consistent form for precedence and time separation constraints. Temporal constraints on scheduling units are derived from those on exposures.

Aggregated exposures, path-consistent relative constraints, and other orbital and astronomical constraints are recorded in files which are later input for SPIKE scheduling. This pre-processing is valuable not only because is saves time later during scheduling search, but it also identifies unschedulable activities (because of infeasible constraints) as early as possible in the scheduling process. The total computer time devoted to preprocessing a year-long observing program will be approximately one week, which is much longer than the time needed to schedule the results.

Schedule construction proceeds by first specifying the overall scheduling time interval, the choice of subintervals, the resource and capacity constraints, and other runtime parameters, then by loading the pre-processed files defining the activities to schedule and their initial suitability functions. Several scheduling search algorithms including procedural ones are available in SPIKE through a graphical window-based user interface (Fig. 7-1). The neural network is, however, by far the fastest method and provides the most extensive exploration of the search space. All of the search methods are uniformly based on the same suitability functions for representing and propagating constraints and preferences.

In its present mode of operation SPIKE is intended to construct schedules at a resolution of one week over periods of one year of more. Once the contents of a week is defined, the scheduling units contained in it are sent to the Science Planning and Scheduling System (SPSS) about two months before execution for final detailed scheduling. SPSS orders the scheduling units in a week by considering constraints on a more detailed level than SPIKE, then expands the exposures into detailed command requests for the week. The command requests are translated by a system at Goddard Space Flight Center into the onboard computer instructions transmitted to the spacecraft in orbit.

7.2. Performance

As described in §2, the discovery of spherical aberration has delayed the onset of normal HST operations by many months. Up to this time SPIKE has been used either for scheduling a few months into the future (rather than years as

planned), or on large-scale test problems. The full multi-year scheduling concept will be exercised operationally once the scientific program of the telescope is redefined in mid-1991.

The performance of the neural network based on tests with real HST observing programs has been extremely promising. The scheduling of 2,600 scheduling units consisting of about 12,000 exposures takes less than one hour on an Texas Instruments Explorer II workstation (Fig. 7-2). A problem of this size represents about six months of HST observing scheduled at one-week time resolution. Scaling with problem size is empirically approximately linear in the number of subintervals considered, and quadratic in the number of activities to schedule. This is fast enough to permit the exploration of many potential schedules before adopting one as a baseline schedule.

8. Conclusion

We have used the challenging problem of scheduling astronomical observations with the orbiting Hubble Space Telescope (HST) as a motivating example for a large and complex real-world scheduling problem.

"Soft" preferences and uncertainties have been seamlessly integrated along with "hard" incompatibility constraints into the same representational scheme of "suitability functions" defined over a continuous time domain. This novel concept captures objective reasons and subjective human expertise in the form of "evidence" for or against certain scheduling decisions. Techniques from uncertainty reasoning are used to combine disparate pieces of scheduling evidence.

We have shown how the declarative problem description can be translated into a 0-1 integer programming problem formulation, which encompasses, in addition to the HST operations scheduling problem, a variety of other problems such as airline crew scheduling, restaurant crew-scheduling, various forms of job-shop scheduling and vehicle routing. The integer programming problem formulation lends itself to an automatic synthesis of a constraint graph, which forms the static structure of a discrete neural network for constraint propagation and problem solving. The scheme allows the incorporation of general temporal couplings between pairs of activities. The important linear equality and inequality auxiliary constraints, frequently occurring in 0-1 programming problems, are enforceable quite generally and conveniently via complete lateral inhibition and networks of hidden "guard" neurons asymmetrically coupled to the main network.

This representational system, suitably encoded into computer programs, can be used as a simple bookkeeping device for manual constraint satisfaction problem (CSP) solving such as schedule construction or constraint violation checking. Its built-in constraint propagation mechanism — effectively implementing a nonmonotonic truth-maintenance system — allows for quick and easy assertion and retraction of scheduling decisions during predictive and reactive scheduling by means of an "active" timetable as part of a mouse-sensitive, graphical user interface.

The paradigm of feedback neural networks has been used to equip the graph-based representational scheme with a suitable dynamics, thus allowing the network to efficiently solve a variety of CSPs, including scheduling. The approach is particularly appealing because of its apparent *generality*, syntactic *simplicity* and its computational *minimality* (no parser, no pattern matcher etc.). The novel GDS-network with its heuristically controlled stochastic neuron selection rule, effectively implementing a probabilistic "Las Vegas" type algorithm, often finds "good" solutions quickly — to our surprise even on serial machines. In contrast to many other schemes, the GDS-network has no free parameters and no 'cooling schedule' to adjust. Both the representational scheme and the search algorithm show a satisfying scaling behavior with the size of the problem, indicating that the built-in stochasticity of the method efficiently probes the statistical properties of the underlying problem (cf. Karp 1986).

In contrast to most search methods employed so far for CSP-solving, the GDS-network is a hybrid of *construction* and *repair*. It admits inconsistent configurations, thus permitting it to work from infeasible initial guesses. In the context of scheduling the network can therefore equally well be used for extending partial schedules previously generated by other means, or for repairing damaged schedules. The latter is achieved by asserting the successful execution and failures of past activities and generating a new schedule from the updated fact base. In a way the network can thus be likened to a rule-based expert system, except that it executes several orders of magnitudes faster.

The concepts described above form the core of HST's operational long-term scheduler SPIKE, which is an integrated part of the HST ground system. SPIKE has successfully passed serious efficiency tests on both formal combinatorial optimization problems and on realistic collections of HST observing programs.

In conclusion we mention a few open problems subject to future research: Can the behavior of the GDS-network and the related min-conflict heuristic algorithm (Minton *et al.* 1990) be satisfactorily understood by a model based

on the theory of Markov chains? Can the behavior of neural networks with asymmetric connection matrices be better understood theoretically? What are the characteristics of the constraint graphs occurring in the context of HST and other scheduling problems? What are the best techniques for generating good initial guesses, and how do these depend on the structure of the constraint graphs? Can convergence be improved by permitting less strictly controlled stochasticity?

Appendix A: Suitability functions

Suppose that we have defined, as in §3.1, a function $W_{i\alpha}(t;t_{j\neq i})$ for each constraint C_α affecting activity A_i which specifies the preference associated with the hypothesis "schedule A_i at t given that A_1 is scheduled at t_1, A_2 at t_2,...". How should the preferences from different constraints C_α and C_β be combined? Denoting the combination by $W_{i\alpha} \oplus W_{i\beta}$, we can place the following reasonable conditions on the combination operator $\oplus$:

(i) $\oplus$ should be a continuous, *monotonically increasing* function of each of its arguments, and

(ii) $\oplus$ should be *associative*, *i.e.* it should not matter in what order evidence is considered: $(W_{i\alpha} \oplus W_{i\beta}) \oplus W_{i\gamma} = W_{i\alpha} \oplus (W_{i\beta} \oplus W_{i\gamma})$

Under these conditions the weights $W_{i\alpha}$ together with the combination operator $\oplus$ can be shown to form an Abelian group isomorphic to the additive group of real numbers on $(-\infty,\infty)$, a result which has been independently discovered by a number of researchers (*e.g.*, Cox 1946; Good 1960, 1968; Hájek 1985, Cheng & Kashyap 1988). Thus with no loss of generality we can take the $W_{i\alpha}$ to be real-valued functions that combine simply by addition.

The additive form of the weights $W_{i\alpha}$ is often not the most convenient. It is common in scheduling problems to have numerous *incompatibility constraints* that specify times when an activity is *not* permitted to be scheduled. These are particularly important since they allow the scheduler to eliminate blocks of time from further consideration. In terms of the weights, these times should have highly unfavorable weight values, *e.g.*

$$W_{i\alpha}(t;t_{j\neq i}) = -w_0,$$

where w_0 is sufficiently large to indicate *overwhelming* evidence against scheduling activity A_i at time t. A convenient representation suitable for implementation on digital computers is obtained by adopting a limiting multiplicative form for the weights denoted by $B_{i\alpha}$:

$$B_{i\alpha}(t;t_{j\neq i}) = \lim_{w_0\to\infty} \exp\left(W_{i\alpha}(t;t_{j\neq i})\right). \tag{A-1}$$

Except where $B_{i\alpha}(t;t_{j\neq i})=0$, the *additive* combination of the weights $W_{i\alpha}(t;t_{j\neq i})$ is equivalent to the *multiplicative* combination of $B_{i\alpha}(t;t_{j\neq i})$; when $B_{i\alpha}(t;t_{j\neq i})=0$, multiplicative combination provides precisely the desired behavior, *i.e.* if there is overwhelming evidence against scheduling A_i at t from *any* source, then no amount of evidence from other sources can counteract this. We have found that the multiplicative formulation is particularly convenient for representing practical constraints by scheduling experts: in the HST domain we have further adopted the convention that a value $W_{i\alpha}=0 \Leftrightarrow B_{i\alpha}=1$ represents the absence of evidence either for or against a scheduling decision. In practice, the $B_{i\alpha}$ are defined by analysis of the constraints and preferences in consultation with telescope scheduling experts.

It is computationally infeasible to work with the full N-dimensional form of the $B_{i\alpha}(t;t_{j\neq i})$ in any practical scheduling problem. The approach adopted in SPIKE consists in projecting the $B_{i\alpha}$ onto functions of one time variable only:

$$S_{i\alpha}(t) = \max_{\{t_{j\neq i}\}} B_{i\alpha}(t;t_{j\neq i}), \tag{A-2}$$

where the maximum operator ranges only over times t_j where activity A_j is permitted to be scheduled (based on the current state of the schedule). $S_{i\alpha}(t)$ is zero only when, due solely to the constraint α, *no* possible choices for scheduling activity $A_{j\neq i}$ will permit A_i to be scheduled at t; otherwise its value is the best (most preferable) value of

$B_{i\alpha}$ that can be obtained by scheduling A_j at t_j, regardless of where any other activities are scheduled. The former property of $S_{i\alpha}(t)$ ensures that no times are excluded prematurely unless provably in violation of a strict constraint. The latter provides an important indicator of optimal scheduling choices to the scheduling agent by always indicating the *best* that can be achieved, regardless of future scheduling decisions. We call $S_{i\alpha}(t)$ the *suitability function for activity* A_i *due to constraint* C_α. The *total suitability function* for an activity $S_i(t)$ is the product of the suitability functions determined from each of its constraints, multiplied by a "restriction" operator $R_i(t)$ indicating any scheduling decisions made so far in constructing the schedule ($R_i(t) \equiv 0$ for excluded times, 1 otherwise):

$$S_i(t) = R_i(t) \prod_\alpha S_{i\alpha}(t) \tag{A-3}$$

$S_i(t) = 0$ if activity A_i is excluded from being scheduled at time t, either because a strict constraint would be violated or because of prior scheduling decisions. Thus we can rewrite Eqn. (A-2) in terms of the set of times over which the maximum is to be taken:

$$S_{i\alpha}(t) = \max_{\{t_{j\neq i} \,\&\, S_j(t)\neq 0\}} B_{i\alpha}(t;t_{j\neq i}) , \tag{A-4}$$

Eqns. (A-3) and (A-4) implicitly determine the suitability function for an activity and can be solved by an iteration procedure corresponding to the propagation of constraints.

The suitability function concept can be illustrated by a simple example: consider a preference constraint of the form: "Schedule A_j as soon as possible after the end of A_i, but starting no sooner than *x* minutes after and ending no later than *x+y* minutes after." We can represent this by choosing $B_{j\alpha}(t;t_i) = f(t-t_i)$ where f is a function indicating the judgement (objective or subjective) of how much "better" or "worse" it is to delay scheduling A_j after A_i. Given the suitability function $S_i(t)$ it is straightforward to construct $S_{j\alpha}(t)$ due to this constraint as illustrated in Fig. A-1 for a hypothetical example. Fig. A-1(a) shows what the $B_{j\alpha}(t;t_i)$ could look like for a plausible choice of $f(t-t_i)$. Fig. A-1(b) shows what the suitability function $S_i(t)$ might be at some stage in the scheduling: in this case there are two disjoint candidate intervals where A_i could be scheduled. Applying the definition of $S_{j\alpha}(t)$ yields the result shown in Fig. A–1(c).

Exploiting consistency methods

Consistency methods have long been known to improve search efficiency for discrete CSPs (see, *e.g.*, Dechter & Meiri 1989 and references therein). These techniques make explicit information that is implicit in the constraints. A full discussion of consistency methods is beyond the scope of this paper: here we only highlight the use of low-order techniques which we have found to significantly speed up scheduling search in the network representation.

1. *Node-consistency* refers to the removal from consideration of domain values which cannot be part of any solution, where this determination is made based on unary constraints. In our formulation this is explicit in the definition of suitability (Eqn. A-4) and in the definition of bias values based on the combination of all unary constraints.
2. *Arc-consistency* refers to removing from consideration domain values based on permitted domain values and binary constraints. This technique is best illustrated by example: suppose A_j is constrained to follow A_i with a minimum end-to-start separation of Δt, that both activities have unit durations and are restricted to be scheduled in the interval $[t_A,t_B]$. Then the interval $[t_B-\Delta t-1,t_B]$ is excluded for A_i, and the interval $[t_A,t_A+\Delta t+1]$ is excluded for A_j. To introduce arc-consistency into the network, we restrict activities to fall within the overall scheduling interval $[t_A,t_B]$, propagate constraints as specified by Eqn. A-4, then set the bias values based on the resulting activity suitabilities.
3. *Path-consistency* refers to the inference of additional binary constraints based on those explicitly stated. Again an example makes the principle obvious: suppose A_i must precede A_j which must in turn precede A_k: by explicitly stating the precedence A_i precedes A_k we can immediately represent the implication of a decision on scheduling A_i which would otherwise require a further decision about A_j. For simple precedence the additional constraints inferred by path consistency are simply those derived from the transitive closure of the precedence relationship. However, for binary constraints which depend on time differences only it is possible to generalize this as follows:

Suppose A_i constrains A_j as specified through a weight function $B_j(t_j-t_i)$ which is a function of the time difference t_j-t_i only, and that a similar constraint exists $B_k(t_k-t_j)$ exists between A_j and A_k. Then we can infer a constraint between A_i and A_k specified by $B(t_k-t_i)$ defined to be:

$$B(t) = \max_{\{\tau | B_j(\tau) \neq 0\}} \phi[B_k(t-\tau)] \qquad \text{(A-5)}$$

where $\phi(x)=1$ if $x>0$, 0 otherwise. We have found that the derivation of additional constraints by this technique dramatically improves scheduling search. There is, however, a significant pre-processing computational cost which must be traded against the speed-up in search.

An alternative to pre-computing inferred constraints is their derivation during search. This technique (a type of machine learning), in conjunction with delayed evaluation of network connection weights, can be used to significantly reduce the computational costs of applying consistency methods. A version of this technique has been used by Dechter (1986) for classical CSPs, and has also been successfully employed in the neural network context (Johnston & Adorf 1989).

Appendix B: A toy "scheduling" problem

The concept of a neural network based CSP solver, which is derived from the problem's constraint graph, is best illustrated with a small example. Consider the following simple "scheduling" problem: Place (if possible) N activities $A_1, ..., A_N$ onto a square timetable with N time-units $t_1, ..., t_N$ subject to the following (admittedly odd) highly regular incompatibility constraints:

1. No activity should be scheduled twice.
2. No two activities should be scheduled simultaneously.
3. No two activities should occur on the same "diagonal" of the timetable (*i.e.* if activity A_i is scheduled at time t_m, then activity A_j is not schedulable at times $t_{m\pm|i-j|}$).

For $N \geq 4$ our toy problem always admits at least 1 feasible solution with all N activities scheduled, which we call "globally" optimum. Fig. B-1a and b show for N=4 two fragments of the corresponding constraint graph G = (V, A) realized with "inhibitory" arcs.

When the constraint graph is equipped with a suitable vertex dynamics, such as the random sequential Hopfield (1982) dynamics, the graph is turned into an active binary neural network. When the neurons are slightly excited (not shown), the network always converges to a stable fixed-point configuration with $K \leq N$ neurons being "on" and $N^2 - K$ neurons being "off". (In graph-theoretic language, the K "on" nodes form a dominating, independent vertex set of the constraint graph, *i.e.* a clique in the constraint graph's complement, and the "off" nodes form a vertex cover. The "scheduling" problem stated above amounts to finding a (globally) maximum independent set of vertices in the constraint graph.)

Fig. B-2a shows the only feasible globally "optimal" (K = N) solution for N=4. Fig. B-2b shows a feasible configuration, corresponding to an attractive, stable fixed-point configuration of the network dynamics, which cannot be extended to a globally optimal solution by placing activity A_3, which is still missing on the timetable. (This situation corresponds to a "locally" maximum independent set of vertices.) The problem that stable attractors exist which correspond to locally but not globally optimum feasible solutions is quite a general one for "direct" neural network problem solvers which are implemented solely with visible neurons and has motivated the introduction of guard neurons (Fig. B-3).

To clarify the definition of the biases b_{im} and weights $W_{im,jn}$, we can specify a set of appropriate numeric values for the case N=4 which satisfy the constraints (1)-(3) above as well as the conditions discussed in §3 and §5. From §5.3 we have the condition $\omega_o \ll g_o < b_o$, where b_o and ω_o are the inhibitory values for biases and weights (see §3.2), and g_o is the guard-to-neuron weight. For specificity, let us choose:

$$\omega_o=2,\ g_o=10, \text{ and } b_o=11.$$

In the following, the indices i and j indicate activities, m and n indicate times, and both range over 1,...,4.

- **biases**: since there are no preferred or excluded times we can choose $b_{im} = 1$. Excluded times would be indicated by $b_{im} = -b_o$. Preferences for different time values would be indicated by values of b_{im} between 0 and 1.
- **weights**: we consider each of the constraints separately (see Fig. B-1a,b).

  (1) An activity should not be scheduled more than once: $W_{im,in} = -\omega_o$, $m \neq n$.

  (2) Two activities should not be scheduled simultaneously: $W_{im,jm} = -\omega_o$, $i \neq j$.

  (3) No two activities on same "diagonal": $W_{im,jn} = -\omega_o$, $i \neq j$, $m \neq n$, and $|m-n| = \pm|i-j|$.

  Note that, as stated, these constraints specify uniform inhibition only: if there were varying *degrees* of inhibition associated with the constraints, then the values of $W_{im,jn}$ could be adjusted to incorporate them.
- **guards**: Illustrated in Fig. B-3, the guards can have bias values $b^G_i = 0$ (i.e. they are stable and "on" with no additional stimulus.) Then the neuron-to-guard weights $W^G_{im,i}$ need only be <0 to ensure that the guard is "off" when any neuron in the row it guards is "on" (see §5.3). The guard-to-neuron weight $W^G_{i,im}$ has the value g_o. The condition $\omega_o \ll g_o$ ensures that the input to neuron im from a guard which is "on" will raise the input level x_{im} to be >0 (see Eqn. 5-1).

Appendix C: Stochastic algorithms

The fact that most interesting combinatorial optimization problems are NP-hard — a property related to an algorithm executing on a serial, deterministic Turing machine — has motivated recent research in parallel and stochastic algorithms. While the use of random numbers is very natural in computer simulations of random processes, the idea of embedding elements of stochasticity into an algorithm for solving *deterministic* combinatorial problems is less obvious, but is slowly penetrating the computer science community and has recently become even quite popular (see *e.g.* Andreatta *et al.* 1986). Sometimes a simple randomization of the input is sufficient to achieve a considerable speed-up compared to the truly deterministic algorithm (Maffioli 1987a).

The perhaps most famous examples of stochastic algorithms are two related methods for testing the primality of integers proposed independently by Rabin (1976, 1980) and by Solovay & Strassen (1977): they output "prime" whenever the input number is prime, but output either "prime" or "composite" with some stateable (bounded) probability, when the input is composite. In other words: the output "composite" is always correct, the output "prime" is mostly. The statistical behavior of the primality testers is characterized by the four transition probabilities or *likelihoods* (only two of which are independent)

$$\Pr(\text{prime}_{out}|\text{prime}_{in}) = 1 \qquad \Pr(\text{composite}_{out}|\text{prime}_{in}) = 0$$

$$\Pr(\text{prime}_{out}|\text{composite}_{in}) = \varepsilon \qquad \Pr(\text{composite}_{out}|\text{composite}_{in}) = 1 - \varepsilon$$

By repeating the algorithm a number of times one can efficiently test even large numbers for primality to any desired certainty, even when ε is not small.

The primality testers are typical examples of stochastic algorithms of the *Monte Carlo*-type which always terminate in polynomial time, but sometimes lie, as opposed to *Las Vegas*-type algorithms which always tell the truth, but sometimes fail to stop (see Johnson 1984). Thus Las Vegas algorithms appeal to those who prefer failure to terminate to an unreliable answer.

A general Monte Carlo algorithm for a decision problem can be viewed as a probabilistic classifier (Fig. C-1) sorting problem instances from two classes A_{in} and B_{in} into one of two classes A_{out} and B_{out} with *likelihoods*

$$\Pr(A_{out}|A_{in}) = 1 \qquad \Pr(B_{out}|A_{in}) = 0$$

$$\Pr(A_{out}|B_{in}) = \varepsilon \qquad \Pr(B_{out}|B_{in}) = 1 - \varepsilon$$

which depend on the problem's and algorithm's characteristics.

In practical applications, one is not so much interested in the likelihoods, but in the *posterior probabilities*, which allow to infer the probable input, given the somewhat uncertain output. The posterior probabilities can be computed from the likelihoods and the probability distribution of the input instances via Bayes theorem of statistics

$$\Pr(A_{in}|A_{out}) = \Pr(A_{out}|A_{in})\Pr(A_{in}) / \Pr(A_{out}) = \Pr(A_{in}) / [\Pr(A_{in}) + \varepsilon \Pr(B_{in})]$$

$$\Pr(B_{in}|A_{out}) = \Pr(A_{out}|B_{in})\Pr(B_{in}) / \Pr(A_{out}) = \varepsilon \Pr(B_{in}) / [\Pr(A_{in}) + \varepsilon \Pr(B_{in})]$$

$$\Pr(A_{in}|B_{out}) = \Pr(B_{out}|A_{in})\Pr(A_{in}) / \Pr(B_{out}) = 0$$

$$\Pr(B_{in}|B_{out}) = \Pr(B_{out}|B_{in})\Pr(B_{in}) / \Pr(B_{out}) = 1$$

Here we have used the expressions

$$\Pr(A_{out}) = \Pr(A_{out}|A_{in})\Pr(A_{in}) + \Pr(A_{out}|B_{in})\Pr(B_{in}) = \Pr(A_{in}) + \varepsilon \Pr(B_{in})$$

$$\Pr(B_{out}) = \Pr(B_{out}|A_{in})\Pr(A_{in}) + \Pr(B_{out}|B_{in})\Pr(B_{in}) = (1 - \varepsilon)\Pr(B_{in})$$

for the output probabilities appearing in the denominators. It is important to clearly distinguish between likelihoods and posterior probabilities, since the latter depend on the input distribution.

Recently the potential of embedding elements of stochasticity into algorithms has started to be explored more widely (Maffioli 1979, p. 77; Gelfand & Mitter 1989). The idea behind stochastic (or probabilistic or randomized heuristics) algorithms in combinatorial optimization often amounts to using random numbers as "noise" to escape from disappointing local optima and thereby opening the possibility of exploring more globally the feasible regions (cf. Maffioli 1987a, b). It is usually required that the problem admits the definition of a neighborhood structure and that an efficient algorithm exists for searching neighborhoods.

Appendix D: A historical remark

Our interest in artificial neural networks and their use for the HST scheduling problem was motivated by Hopfield's representation of the travelling salesman problem (Hopfield & Tank 1985). At the end of 1987 one of us (HMA) started with an implementation of the graded Hopfield model (Hopfield 1984), at that stage using the object system and the user interface of the KEE™ expert system shell on a Texas Instruments Explorer™ Lisp machine. As a test problem we used the well known N-queens problem, since it was easily scalable and mathematically proven to possess solutions for any $N \geq 4$. Initial trials looked promising; the network functioned as a content-addressable, autoassociative memory (CAM) — provided the symmetry of the empty board was broken by a suitable "partial solution" stimulus.

However, because of serious performance problems, one of us (MDJ) soon instead tried the binary neuron model (Hopfield 1982) in conjunction with a heuristically controlled stochastic neuron selection rule and also invented the guard neurons. (We recently learnt from an article by Eliashberg (1988) that the idea of such auxiliary neurons can historically be traced back to the year 1904.) After a period of joint optimization, where *e.g.* lazy evaluation of the weights was introduced allowing the execution of much bigger problems, the fundamental GDS-network algorithm stabilized in 1988 — an early account is given in Adorf & Johnston (1988) — and the core code has essentially remained unchanged since then.

We were impressed by the performance of the GDS-network on the N-queens problem, for which the first N=1024 solution was found early in 1988, a considerable increase over the previously published maximum of N=96 (Stone & Stone 1987). Since then the basic algorithm has formed the most efficient search algorithm of HST's long-range scheduler SPIKE, only recently being superseded by an even more efficient algorithm of Minton *et al.* (1990), which uses the same representational concepts but a different variable/neuron selection rule.

In view of these successes, we were surprised to learn that the integer programming formulation of scheduling problems has been essentially dismissed (see, *e.g.*, French 1986, p. 135) as being computationally infeasible.

References

Aarts, E., Korst, J.: 1989, "Simulated Annealing and Boltzmann Machines — A Stochastic Approach to Combinatorial Optimization and Neural Computing", Wiley-Interscience Series in Discrete Mathematics and Optimization, J. Wiley & Sons, Chichester

Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: 1985, "A Learning Algorithm for Boltzmann Machines", *Cognitive Science* **9**, 147–169

Adorf, H.-M.: 1989, "Connectionism and Neural Networks", in: *"Knowledge-Based Systems in Astronomy", A. Heck & F. Murtagh (eds.),* Lecture Notes in Physics, Vol. **329**, Springer-Verlag Heidelberg, pp. 215–245

Adorf, H.-M.: 1990, "The processing of HST observing programmes", *ST-ECF Newsletter* **13**, 12–15

Adorf, H.-M., Johnston, M.D.: 1988, "Artificial Neural Nets in Astronomy", in: *Proc. Workshop "Konnektionismus", Bonn, Feb 1988, C. Lischka & J. Kindermann (eds.),* Arbeitspapier der Gesellschaft für Mathematik und Datenverarbeitung **329**, pp. 3–5

Adorf, H.-M., Johnston, M.D.: 1990, "A Discrete Stochastic Neural Network Algorithm for Constraint Satisfaction Problems", in: *Proc. Int. Joint Conf. on Neural Networks (IJCNN 90), 17–21 Jun 1990, San Diego, CA,* Vol. **III**, pp. 917–924

Ae, T., Aibara, R.: 1990 , "Programmable real-time scheduler using a neurocomputer", *Real-Time Systems* **1**, 351–363

Ali, M.: 1990, "Intelligent systems in aerospace", *Knowledge Engineering Review* **5**, p.147–66

Allen, J.F.: 1983, "Maintaining Knowledge About Temporal Intervals", *Comm. ACM* **26**, 832–843

Andreatta, G., Mason, F., Serafini, P. (eds.): 1986, "Advanced School on Stochastics in Combinatorial Optimization", CISM, Udine, Italy, 22–25 Sep 1986, World Scientific, Singapore

Balas, E., Padberg, M.W.: 1979, "Set Partitioning — A Survey", in: *"Combinatorial Optimization", N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds.),* John Wiley & Sons, Chichester, pp. 151–210

Barhen, J., Einstein, R., Jorgensen, C.C.: 1987a, "Advances in concurrent computation for machine intelligence and robotics", in: *Proc. "Second International Conference on Supercomputing (ICS 87)", 4–7 May 1987, San Francisco, CA,* Vol. **2**, pp. 84–97

Barhen, J., Toomarian, N., Protopopescu, V., Clinard, M.: 1987b, "Concurrent neuromorphic algorithms for optimization of the computational load of a hypercube supercomputer", in: *Proc. "IEEE First Int. Conf. Neural Networks", 21–24 Jun 1987, San Diego, CA, Caudill, M., Butler, C. (eds.),* SOS Printing, San Diego, CA, Vol. **4**, pp. 687–696

Barhen, J., Toomarian, N., Protopopescu, V.: 1987c, "Optimization of the computational load of a hypercube supercomputer on-board a mobile robot", *Applied Optics* **26**, 5007–5014

Bellman, R., Esogbue, A.O., Nabeshima, I.: 1982, "Mathematical Aspects of Scheduling & Applications", Int. Ser. Modern Appl. Math. Comp. Science, Vol. **4**, Pergamon Press, Oxford

Bourret, P., Goodall, S., Samuelides, M.: 1989, "Optimal scheduling by competitive activation: application to the satellite antennae scheduling problem", in: *Proc. Int. Joint Conference Neural Networks (IJCNN 89)", 18–22 Jun 1989, Washington, DC,* Vol. **1**, pp. 565–172

Cerny, V.: 1982, "A thermodynamic approach to the travelling salesman problem: an efficient simulation algorithm", *Report, Inst. Physic & Biophsyics, Comenius Univ., Bratislava*

Cerny, V.: 1985, "Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm", *J. Optimization Theory & Applications* **45**, 260–266

Cheng, Y, Kashyap, R.L.: 1988, "An Axiomatic Approach for Combining Evidence from a Variety of Sources", *J. of Intelligent and Robotic Systems* **1**, 17–33

Christofides, N., Mingozzi, A., Toth, P.: 1979a, "The vehicle routing problem", in: *"Combinatorial Optimization", N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds.),* John Wiley & Sons, Chichester, pp. 315–338

Christofides, N., Mingozzi, A., Toth, P.: 1979b, "Loading problems", in: *"Combinatorial Optimization", N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds.),* John Wiley & Sons, Chichester, pp. 339–369

Cox, R.T.: 1946, "Probability, Frequency and Reasonable Expectation", *J. American Physics* **14**, 1–13

Dagli, C.H., Lammers, S.: 1989, "Possible applications of neural networks in manufacturing", in: *Proc. Int. Joint Conf. Neural Networks (IJCNN 89), 18–22 Jun 1989, Washington, DC,* Vol. **2**, p. 605

Davis, D.L., Gillenwater, E.L., Johnson, J.D.: 1990, "An artificial neural systems framework for delivery truck scheduling", in: *Proc. "Twenty-Third Annual Hawaii International Conference on System Sciences", 2–5 Jan 1990, Kailua-Kona, HI, USA,* IEEE Comput. Soc. Press, Los Alamitos, CA, USA, Vol. **3**, pp. 327–333

de Kleer, J.: 1989, "A comparison of ATMS and CSP techniques", in: *Proc. 11th Int. Joint Conf. on Artificial Intelligence (IJCAI 89), Aug 1989, Detroit, MI,* Vol. **I**, pp. 290–296

Dechter, R.: 1986, "Learning while searching in constraint satisfaction problems", in: *Proc. Fifth National Conf. Artificial Intelligence (AAAI 86), Philadelphia,* Vol. I, pp. 178–183

Dechter, R., Meiri, I.: 1989, "Experimental evaluation of preprocessing techniques in constraint satisfaction problems", in: *Proc. 11th Int. Joint Conf. on Artificial Intelligence (IJCAI 89), Aug 1989, Detroit, MI,* Vol. I, pp. 271–277

Dechter, R., Meiri, I., and Pearl, J. 1989: "Temporal Constraint Networks", in: *Proc. First Int. Conf. on Principles of Knowledge Representation and Reasoning, May 1989, Toronto,* pp. 83–93.

Dechter, R., Pearl, J.: 1988, "Network-based Heuristics for Constraint-Satisfaction Problems", *Artif. Intell.* **34**, 1–38

Demongeot, J.: 1987, "Random automata networks", in: *"Automata networks in computer science", F. Fogelman-Soulié, Y. Robert, M. Tchuente (eds.),* Princeton University Press, Princeton, N.J.

Eliashberg, V.: 1988, "Neuron layer with reciprocal inhibition as a mechanism of random choice", in: *Proc. IEEE Int. Conf. Neural Networks (ICNN 88),* Vol. I, pp. 17–25

Fang, L., Wilson, W.H., Li, T.: 1990, "A neural network for job sequencing", in: *Proc. Conf. "Parallel Processing in Neural Systems and Computers", R. Eckmiller, G. Hartmann, G. Hauske (eds.),* Elsevier Science Publ. B.V., (North Holland), pp. 253–256

Feldman, R., Golumbic, M.C.: 1989, "Constraint satisfiability algorithms for interactive student scheduling", in: *Proc.11th Int. Jont Conf. Artif. Intell. (IJCAI 89), 20—25 Aug 1989, Detroit, MI,* Vol. II, pp. 1010–1016

Feldman, R., Golumbic, M.C.: 1990, "Optimization algorithms for interactive student scheduling via constraint satisfiability", *Computer J.* **33**, 356–365

Foo Y.-P.S., Takefuji, Y.: 1988a, "Stochastic neural networks for solving job-shop scheduling. I. Problem representation", in: *Proc. "IEEE International Conference on Neural Networks (ICNN 88), 24–27 Jul 1988, San Diego, CA, USA,* IEEE, New York, NY, USA, Vol. 2, pp. 275–282

Foo Y.-P.S., Takefuji, Y.: 1988b, "Stochastic neural networks for solving job-shop scheduling. II. Architecture and simulations", in: *Proc. "IEEE Int. Conf. Neural Networks (ICNN 88), 24–27 Jul 1988, San Diego, CA, USA,* IEEE, New York, NY, USA, Vol. 2, pp. 283–290

Foo Y.-P.S., Takefuji, Y.: 1988c, "Integer linear programming neural networks for job-shop scheduling", in: *Proc. "IEEE Int. Conf. Neural Networks (ICNN 88), 24–27 Jul 1988, San Diego, CA,* USA, IEEE, New York, NY, USA, Vol. 2, pp. 341–348

Fox, M.S.: 1987, "Constrained-Directed Search: A Case Study of Job-Shop Scheduling, Morgan Kaufmann Publishers, Inc.

French, S.: 1986, "Sequencing and Scheduling – An Introduction to the Mathematics of the Job-Shop", Ellis Horwood Series in Mathematics and its Applications, John Wiley & Sons, New York

Galland,C.C., Hinton, G.E.: 1991, "Deterministic Boltzmann Learning in Networks with Asymmetric Connectivity", in: *Proc. 1990 Summer School on "Connectionist Models", Univ. California at San Diego (UCSD), D.S. Touretzky, J.L. Elman, T.J. Sejnowski, G.E. Hinton (eds.),* Morgan Kaufmann Publ., Inc., San Mateo, CA, USA, pp. 3–9

Garey, M.R., Johnson, D.S.: 1979, "Computers and intractability – A Guide to the Theory of NP-Completeness", W.H. Freeman and Co., New York

Gaspin, C.: 1989, "Mission scheduling", in: *Proc. "1989 Goddard Conf. on Space Applications of Artificial Intelligence", 16–17 May 1989, Greenbelt, MD,* reprinted in: *Telematics and Informatics* **6**, 159–169

Gelfand, S.B., Mitter, S.K.: 1987, "Simulated Annealing", in: *"Proc. Advances School on Stochastics in Combinatorial Optimization", CISM, Udine, Italy, 22–25 Sep 1986, Andreatta, G., Mason, F., Serafini, P. (eds.),* World Scientific, Singapore, pp. 1–51

Gerb A.: 1991, "Transformation Reborn: A New Generation Expert System for Planning HST Operations", in: *Proc. 1991 Goddard Conference on Space Applications of Artificial Intelligence,* (to appear)

Gianessi, F., Nicoletti, B.: 1979, "The Crew Scheduling Problem: A Travelling Salesman Approach", in: "Combinatorial Optimization", Cristofides, N., Mingozzi, A., Toth, P., Sandhi, C. (eds.), J. Wiley & Sons, Chichester, pp. 389–408

Gianoglio, P.: 1990, "Application of neural networks to timetable construction", in: *Proc. Third Intern. Workshop "Neural Networks and Their Applications (Neuro-Nimes 90)", Nimes, 12–16 Nov. 1990,* EC2, Nanterre, France, pp. 53–67

Gislen, L., Peterson, C., Söderberg, B.:, 1989, "Teachers and classes with neural networks", *Int. J. Neural Systems* **1**, 167–176

Goles, E.: 1987, "Lyapunov functions associated to automata networks", in: *"Automata networks in computer science", F. Fogelman-Soulié, Y. Robert, M. Tchuente (eds.),* Princeton University Press, Princeton, N.J., pp. 58—81

Good, I.J.: 1960, "Weight of Evidence, Corroboration, Explanatory Power, Information and the Utility of Experiments", *J. Royal Statistical Soc.* **B 22**, 319–331

Good, I.J.: 1968, "Corrigendum: Weight of Evidence, Corroboration, Explanatory Power, Information and the Utility of Experiments", *J. Royal Statistical Soc.* **B 30**, 203 ff.

Gulati, S., Iyengar, S.S., Toomarian, N., Protopopescu, V., Barhen, J.: 1987, "Nonlinear neural networks for deterministic scheduling", in: *Proc. "IEEE First Int. Conf. Neural Networks (ICNN 87)", 21–24 Jun 1987, San Diego, CA, Caudill, M., Butler, C. (eds.),* SOS Printing, San Diego, CA, Vol. 4, p.745–752

Hájek, P.: 1985, "Combining functions for certainty degrees in consulting systems", *Int. J. Man-Machine Studies* **22**, 59–76

Hinton, G.E., Sejnowski, T.J.: 1983, "Optimal perceptual inference", in: *Proc. IEEE Conf. Computer Vision & Pattern Recognition, Washington, DC*, pp. 448–453

Hinton, G.E., Sejnowski, T.J., Ackley, D.H.: 1984, "Boltzmann machines: constraint satisfaction machines that learn", *Carnegie-Mellon University, Technical Report* **CMU-CS-84-119**

Hopfield, J.J.: 1982, "Neural networks and physical systems with emergent collective computational abilities", *Proc. Nat. Acad. Sci.* **79**, 2554–2558

Hopfield, J.J.: 1984, "Neurons with graded response have collective computational properties like those of two-state neurons", *Proc. Nat. Acad. Sci.* **81**, 3088–3092

Hopfield, J.J., Tank, D.W.: 1985, ""Neural" Computation of Decisions in Optimization Problems", *Biol. Cybern.* **52**, 141–152

Hopfield, J.J., Tank, D.W.: 1986, "Computing with Neural Circuits: A Model", *Science* **233**, 625–633

Hutchison, W.R., Stephens, K.R., Bell, T.M.: 1990, "Adaptive networks for scheduling and inventory control: resource allocation in changing environments", in: *Proc. "Third European Seminar on Neural Computing: The Marketplace", 15–16 Feb. 1990, London,* IBC Tech. Services, London, p. 7

Iida, I., Chugo, A., Yatsuboshi, R.: 1989, "Autonomous routing scheme for large scale network based on neural processing", in: *Proc. "1989 IEEE International Conference on Systems, Man and Cybernetics", 14–17 Nov. 1989, Cambridge, MA,* Vol. **1**, pp. 194–199

Jackson, R., Johnston, M., Miller, G., Lindenmayer, K., Monger, P., Vick, S., Lerner, R., Richon, J.: 1988, "The Proposal Entry Processor: Telescience Applications For Hubble Space Telescope Science Operations", in: *Proc. " Goddard Conference on Space Applications of Artificial Intelligence", U. Rash, P. Hughes (eds.)*, NASA Conf. Publ. **3009**, pp. 107-123

Jeffrey, W., Rosner, R.: 1986, "Optimization Algorithms: Simulated Annealing and Neural Network Processing", *Astrophys. J.* **310**, 473–481

Johnson, D.S.: 1984, "The NP-Completeness Column: An Ongoing Guide", *J. Algorithms* **5**, 433–447

Johnston, M.D.: 1987, "HST Planning Constraints", *STScI SPIKE Tech. Report* **87-1**

Johnston, M.D.: 1988a, "Artificial Intelligence Approaches to Spacecraft Scheduling", in: *Proc. ESA Workshop on Artificial Intelligence Applications for Space Projects, Nov 1988, ESTEC, Noordwijk, A. Scheffer (ed.)*, pp. 5–9

Johnston, M.D.: 1988b, "Automated Observation Scheduling for the VLT", in: *Proc. ESO Conf. on "Very Large Telescopes and their Instrumentation", European Southern Observatory, Garching bei München, Mar 1988, M.-H. Ulrich (ed.)*, Vol. **2**, pp. 1273–1282

Johnston, M.D.: 1988c, "Automated Telescope Scheduling", in: *Proc. Conf. on "Coordination of Observational Projects in Astronomy", Strasbourg, Nov 1987, C. Jaschek & C. Sterken (eds.)*, Cambridge University Press, Cambridge, pp. 219–226

Johnston, M.D.: 1989a, "Knowledge Based Telescope Scheduling", in: *"Knowledge-Based Systems in Astronomy", A. Heck & F. Murtagh (eds.)*, Lecture Notes in Physics, Vol. **329**, Springer-Verlag Heidelberg, pp. 33–49

Johnston, M.D.: 1989b, "Reasoning with Scheduling Constraints and Preferences", *SPIKE Tech. Rep.* **1989-2, Rev. A**, pp. 1–21

Johnston, M.D.: 1990: "Spike: AI Scheduling for NASA's Hubble Space Telescope", in: *Proc. 6th Conf. on Artif. Intell. Applications (CAIA 90), Mar 1990, Santa Barbara, CA*, pp. 184-190

Johnston, M.D., Adorf, H.-M.: 1989, "Learning in Stochastic Neural Networks for Constraint Satisfaction Problems", in: *Proc. NASA Conf. on Space Telerobotics, 31 Jan – 2 Feb 1989, Pasadena CA, G. Rodriguez and H. Seraji (eds.)*, Vol. **II**, pp. 367–376

Johnston, M.D., Miller, G.: 1988, "Artificial Intelligence Approaches to Astronomical Observation Scheduling", in: *Proc. Workshop "Data Analysis in Astronomy III", Erice, June 1988, V. Di Gesù, L. Scarsi, P. Crane, J.H. Friedman & S. Levialdi (eds.)*, Ettore Majorana International Science Series, Plenum Press New York, pp. 205–214

Johnston, M.D., Miller, G.: 1990, "Artificial intelligence scheduling for NASA's Hubble Space Telescope", in: *Proc. "The Fifth Annual AI Systems in Government Conference", 6–11 May 1990, Washington, DC, USA, Silverman, B., Hwang, V., Post, S. (eds.),* IEEE Comput. Soc. Press, Los Alamitos, CA, USA, pp. 33–39

Karp, R.M.: 1975, "On the Computational Complexity of Combinatorial Problems", *Networks* **5**, 45–68

Karp, R.M.: 1986, "Combinatorics, Complexity, and Randomness", (Turing Award Lecture), *Comm. ACM* **29**, 98–109

Kasahara, H.: 1990, "Current status of optimal parallelizing compilers", *J. Institute of Electronics, Information and Communication Engineers* **73**, pp. 258–266

King, J., Spachis, A.: 1980, "Scheduling: Bibliography and Review", *J. Phys. Distr. Materials Managem.* **10**, 105–131

Kirkpatrick, S., Gelatt, C., Vecchi, M.: 1982, "Optimization by simulated annealing", *IBM Research Report* **RC 9355**

Kirkpatrick, S., Gelatt, C., Vecchi, M.: 1983, "Optimization by simulated annealing", *Science* **22**, 671–680

Kobayashi, Y., Nonaka, H.: 1990, "Application of neural networks to schedule integration in plant engineering", in: *Proc. "International Neural Network Conference (INNC 90)", Paris, France,* Kluwer, Dordrecht, Netherlands, Vol. **1**, pp. 287–290

Li-wei Bao, Yong-zai Lu: 1990, "Neural network based real-time production scheduling for industrial processes", in: *Proc. IFAC Workshop "Artificial Intelligence in Real-Time Control 1989", 19–21 Sep 1989, Shenyang, China, Rodd, M.G., Hua-Tian Li, Shi-Quan Su (eds.)*, Pergamon, Oxford, UK, pp. 85–90

Mackworth, A.K., Freuder E.C.: 1985, "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction", *Artif. Intell.* **25**, 65–74

Mackworth, A.K.: 1977, "Consistency in Networks of Relations", *Artif. Intell.* **8**, 99–118

Maffioli, F.: 1979, "The complexity of combinatorial optimization algorithms and the challenge of heuristics", in: *"Combinatorial Optimization", N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds.)*, John Wiley & Sons, pp. 107–129

Maffioli, F.: 1987a, "Randomization for more efficient algorithms", in: *"Proc. Advances School on Stochastics in Combinatorial Optimization", CISM, Udine, Italy, 22–25 Sep 1986, Andreatta, G., Mason, F., Serafini, P. (eds.)*, World Scientific, Singapore, pp. 52–75

Maffioli, F.: 1987b, "Randomized heuristics for NP-hard problems", in: *"Proc. Advances School on Stochastics in Combinatorial Optimization", CISM, Udine, Italy, 22–25 Sep 1986, Andreatta, G., Mason, F., Serafini, P. (eds.)*, World Scientific, Singapore, pp. 76–93

Mars, P.: 1989, "Optimising PSS using learning automata", *IEE Colloq. "Advances in Optimisation", 20 Mar 1989, London*, IEE (Digest No.45), London, p. 3/1–3

Meseguer, P.: 1989, "Constraint Satisfaction Problems: An Overview", *AI Communications* **2**, 3–17

Miller, G., Johnston, M., Vick, S., Sponsler, J., Lindenmayer, K.: 1988, "Knowledge based tools for Hubble Space Telescope planning and scheduling: constraints and strategies", in: *Proc. "1988 Goddard Conference on Space Applications of Artificial Intelligence", 24 May 1988, Greenbelt, MD, USA*, NASA Conf. Proc. **3009**, pp. 91–106, reprinted in: *Telematics and Informatics* **5**, 197–212

Miller, G.E, Johnston, M.D.: 1991, "Long Range Science Scheduling for the Hubble Space Telescope", in: *Proc. 1991 Goddard Conference on Space Applications of Artificial Intelligence* (to appear)

Miller, G., Rosenthal, D., Cohen, W., Johnston, M.: 1987, "Expert System Tools for Hubble Space Telescope Observation Scheduling", in: *Proc. 1987 Goddard Conf. Space Appl. of Artificial Intelligence* and Robotics, reprinted in *Telematics and Informatics* **4**, 301–311

Minton, S., Johnston, M.D., Philips, A.B., Laird, Ph.: 1990, "Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method", in: *Proc. Eighth National Conf. Artificial Intelligence (AAAI 90), 29 Jul–3 Aug 1990*, Vol I, pp. 17–24

Moussouris, J.: 1974, "Gibbs and Markov random systems with constraints", *J. Statist. Physics* **10**, 11–33

Oglesby, J., Mason, J.S.: 1989, "Dynamic scheduling for feed-forward neural nets using transputers", in: *Proc. "First IEE Int. Conf. on Artificial Neural Networks (ICANN 89)", 16–18 Oct 1989, London, UK*, pp. 257–260

Pearl, J. 1988: "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufmann, San Mateo CA

Peterson, C., Söderberg, B.: 1989, "A new method for mapping optimization problems onto neural networks", *Int. J. Neural Systems* **1**, 3–22

Poliac, M.O., Lee, E.B., Slagle, J.R., Wick, M.R.: 1987, "A crew scheduling problem", in: *Proc. IEEE Int. Conf. Neural Networks (ICNN 87), San Diego, CA*, Vol. IV, pp. 779–786

Price, C.C., Salama, M.A.: 1990, "Scheduling of Precedence-Constrained Tasks on Multiprocessors", *The Computer Journal* **33**, 219–229

Rabelo, L.C., Alptekin, S.: 1989a, "Using hybrid neural networks/expert systems for intelligent scheduling in flexible manufacturing systems", in: *Proc. Int. Joint Conf. Neural Networks (IJCNN 89), 18–22 Jun 1989, Washington, DC*, Vol. **2**, p. 608

Rabelo, L.C., Alptekin, S.: 1989b, "Integrating scheduling and control functions in computer integrated manufacturing using artificial intelligence", in: *Proc. "11th Annual Conference on Computers and Industrial Engineering", 15–17 Mar 1989, Orlando, FL, Computers & Industrial Engineering* **17**, 101–106

Rabin, M.O.: 1976, "Probabilistic Algorithms", in: "Algorithms and Complexity. New Directions and Recent Results", J.F. Traub (ed.), Academic Press, New York, pp. 21–39

Rabin, M.O.: 1980, "Probabilistic algorithm for testing primality", *J. Number Theory* **12**, 128–138

Rit, J.-F.: 1986, "Propagating temporal constraints for scheduling", in: *Proc. Fourth National Conf. Artificial Intelligence (AAAI 86)*, pp. 383–388

Rosenthal, D.: 1986, "Transformation of Scientific Objectives into Spacecraft Activities", in: *Proc. Symp. "Expert Systems in Government", 22–24 Oct 1986, McLean, VA, K.N. Karna, K. Parsaye, B.G. Sileverman (eds)*, IEEE Computer Soc. Press **738**, 430–435

Rosenthal, D., Monger, P., Miller, G., Johnston, M.: 1986, "An Expert System for Ground Support of the Hubble Space Telescope", in: *Proc. 1986 Goddard Conf. Artificial Intelligence Applications, 15 May 1986, Greenbelt, MD*, pp. 43–54

Sadeh, N., Fox, M.S.: 1988, "Preference Propagation in Temporal Constraints Graphs", *Tech. Rep., Intelligent Systems Laboratory, The Robotics Institute, Carnegie Mellon University*

Shapiro, R.D.: 1980, "Scheduling coupled tasks", *Nav. Res. Logist. Q.* **27**, 489–479

Smith, A.W., Baghernejad, M., Stevenson, I.A.: 1988, "Architectures for intelligent real-time control systems for manufacturing applications", in: Proc. *"FACTORY 2000: Integrating Information and Material Flow", 31 Aug–2 Sep 1988, Cambridge, UK,* IERE (Publ. No.80), London, p. 67–74

Solovay, R., Strassen, V.: 1977, "A fast Monte-Carlo test for primality", *SIAM J. Comput.* **6**, 84–85

Sponsler, J.L., Johnston, M.D.: 1990, "An Approach to Rescheduling Activities Based On Determination of Priority and Disruptivity", in: *Proc. 1990 Goddard Conference on "Space Applications of Artificial Intelligence", Greenbelt, MD, 1–2 May 1990, J. L. Rash (ed.),* NASA Conf. Publ. **3068**, pp. 63–74

Stone, H.A., Stone, J.M.: 1987, "Efficient search techniques – An empirical study of the N-queens problem", *IBM J. Res. Devel.* **31**, 464–474

Szu, H.: 1986, "Fast Simulated Annealing", in: *"Neural Networks for Computing", Proc. Conf. American Institute of Physics 151, Snowbird, UC, J.S. Denker (ed.),* pp. 420—425

Szu, H., Hartley, R.: 1987, "Fast simulated annealing", *Phys. Lett.* **A 122**, 157–162

Tank, D.W., Hopfield, J.J.: 1986, "Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", *IEEE Trans. Circuits & Systems* **CAS-33**, 533–541

Ullman, J.D.: 1975, "NP-Complete Scheduling Problems", *J. Computer & System Sciences* **10**, 384–393

Wilson, G.V., Pawley, G.S.: 1988, "On the stability of the travelling salesman problem in the algorithm of Hopfield and Tank", *Biol. Cybernetics* **58**, 63

Yong-zai Lu, Li-wei Bao, Yong-yao Yang, Bo-yin Li: 1990, "An integrated computer control system for a steel plate mill", in: *Selected Papers 6th IFAC Symposium "Automation in Mining, Mineral and Metal Processing", 4–8 Sep 1989, Buenos Aires, Argentina, Koppel, V. (ed.),* Pergamon, Oxford, pp. 293–297

Yu, T.L.: 1990, "Time-table scheduling using Neural Networks Algorithms", in: *Proc. Int. Joint Conf. Neural Networks (IJCNN 90), San Diego, Jun 1990,* Vol. **1**, pp. 279–284

Zhou, D.H., Cherkassky, V., Baldwin, T.R., Hong, D.W.: 1990, "Scaling Neural Network for Job-Shop Scheduling", in: *Proc. Int. Joint Conf. Neural Networks (IJCNN 90), San Diego, Jun 1990,* Vol. **3**, pp. 889– 894

Zhou, D.H., Cherkassky, V., Baldwin, T.R., Olson, D.E.: 1991, "A Neural Network Approach for Job-Shop Scheduling", *IEEE Trans. Neural Networks* **2**, 175–179

Symbols

Quantity	Symbol
set of activities	$\{A_i\}$
number of activities	N
activity	$i, j, A_i, A_j,$
activity duration	d_i
constraint	C_α
set of resources	$\{L_\rho\}$
resource	L_ρ
temporal resource bounds/capacity limits	Q^ρ_m
temporal resource requirement	q^ρ_{im}
time	t
minimum time separation	Δt
scheduling period	$[t_A, t_B]$
time slot/interval	m, n, I_m, I_n
sampled value of function f(t) at time t_m	$E_m[f(t)]$
two-point sampling operator	$E_{mn}[f(t_1,t_2)]$
additive degree of preference/utility	$W_{i\alpha}, W_{i\beta}, W_{i\gamma}$
multiplicative degree of preference/utility	$B_{i\alpha}$
suitability function due to constraint C_α	$S_{i\alpha}(t)$
total suitability function for activity A_i	$S_i(t)$
minimum schedulable suitability value	S_O
restriction operator	$R_i(t)$
overwhelming evidence	w_O
neuron input	x_{im}, x_{jn}
neuron output	y_{im}, y_{jn}
neuron transfer function	$\eta(x_{im})$
guard superscript	G
bias	b_{im}, b_{jn}
compatible/incompatible components of bias	b^+, b^-
connection weight/strength	$W_{im,jn}$
compatible/incompatible components of connection weight	W^+, W^-
set of vertices	V
set of directed edges/arcs	A
weighted directed graph/digraph	$G = (V, A)$
schedule	$\sigma: \{A_i\} \rightarrow \{t_m\}$

Acronyms

Acronym	Expansion
ANN	artificial neural network
BM	Boltzmann machine
CAM	content-addressable associative memory
COP	combinatorial optimization problem
CSA	classical simulated annealing
CSP	constraint satisfaction problem
EARN	European Academic Research Network
ESO	European Southern Observatory
FMS	flexible manufacturing system
FSA	fast simulated annealing
GDS	guarded discrete stochastic
HEAO	High-Energy Astronomy Observatory
HST	Hubble Space Telescope
KEE™	Knowledge Engineering Environment™
NN	neural network
MFA	mean field annealing
SA	simulated annealing
SPAN	Space Physics Analysis Network
SPSS	Science Planning and Scheduling System
ST-ECF	Space Telescope — European Coordinating Facility
STScI	Space Telescope Science Institute
TDRS	Tracking and Data Relay Satellite

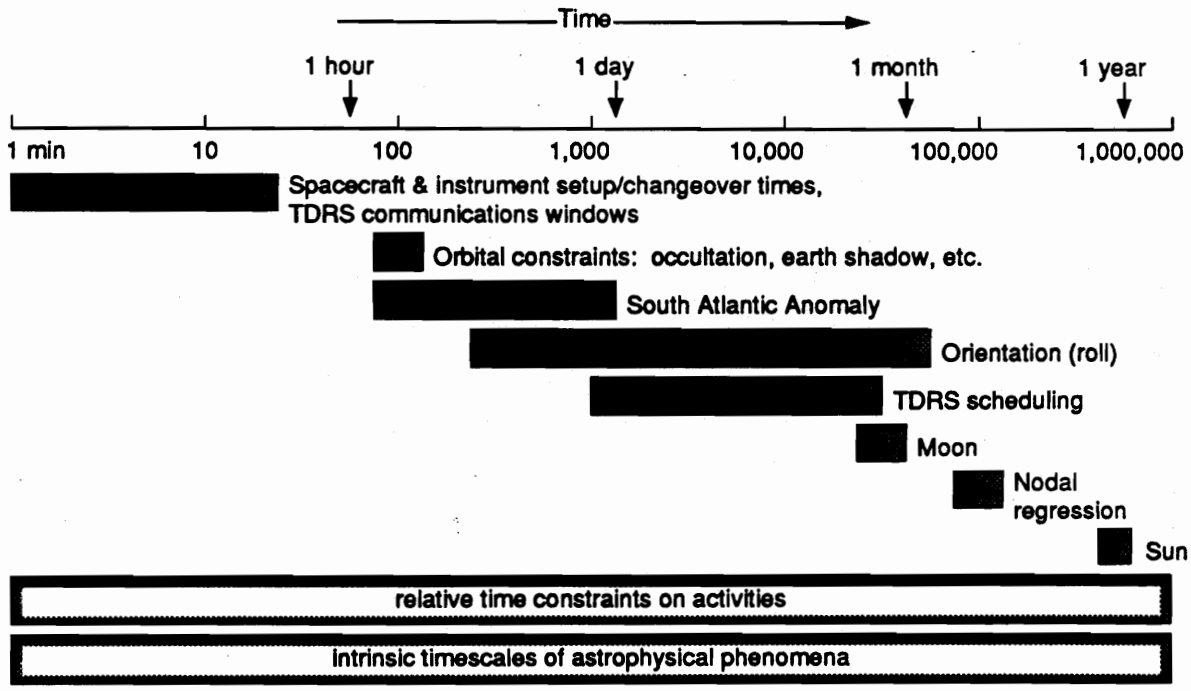

Fig. 2-1: The range of timescales of important constraints in the Hubble Space Telescope scheduling domain: the gray bars show the approximate timescales over which important classes of constraints can affect scheduling choices. The variation is over more than six orders of magnitude.

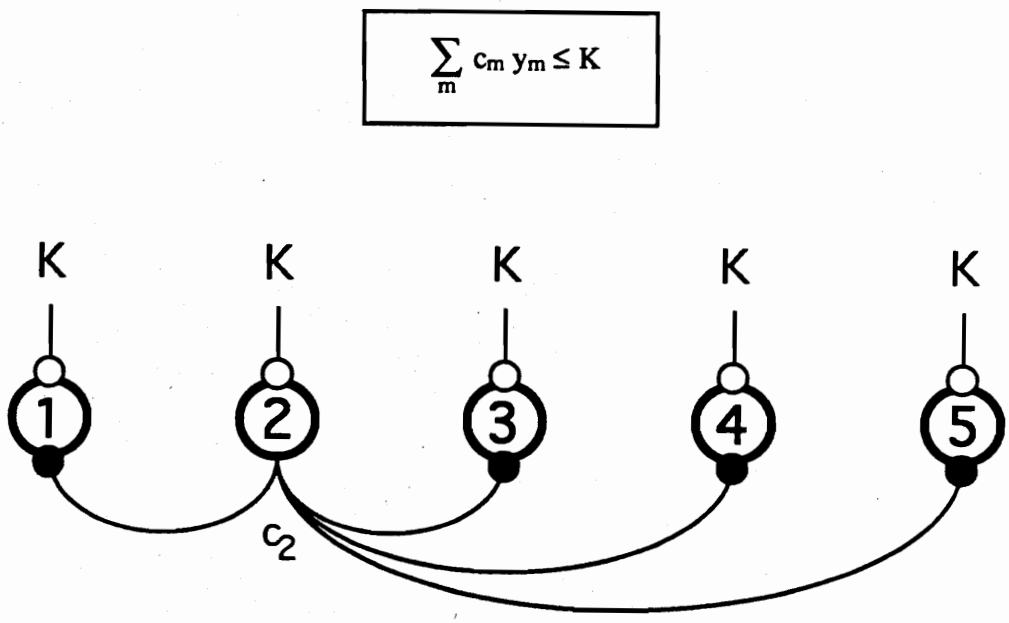

Fig. 5-1a: A complete "lateral" inhibition of a set of neurons implementing a (generalized "at *most* K neurons") upper bound inequality constraint. Large circles with their incoming and outgoing lines represent artificial neurons with their "dendrites" and "axons". Small unfilled and filled circles represent excitatory and inhibitory "synapses". (Note that the latter are introduced only as a visual aid; they are not meant to invert the incoming weight, contrary to natural synapses). Only the inhibitory connections outgoing from neuron 2 are shown; analogous connections would have to be made for each other neuron.

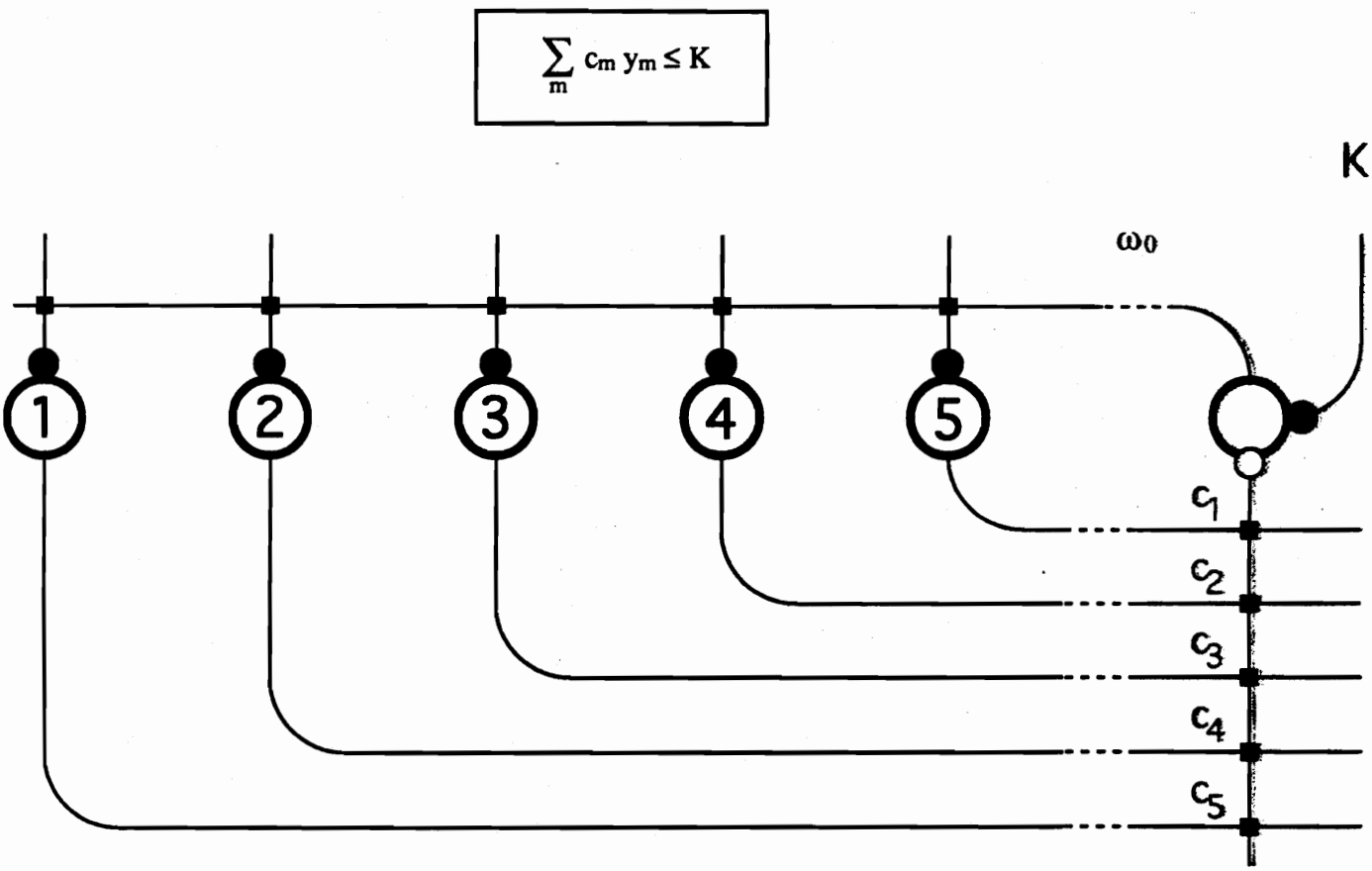

Fig. 5-1b: An asymmetrically coupled inhibitory guard as an alternative means for implementing a (generalized "at *most* K neurons") upper bound inequality constraint.

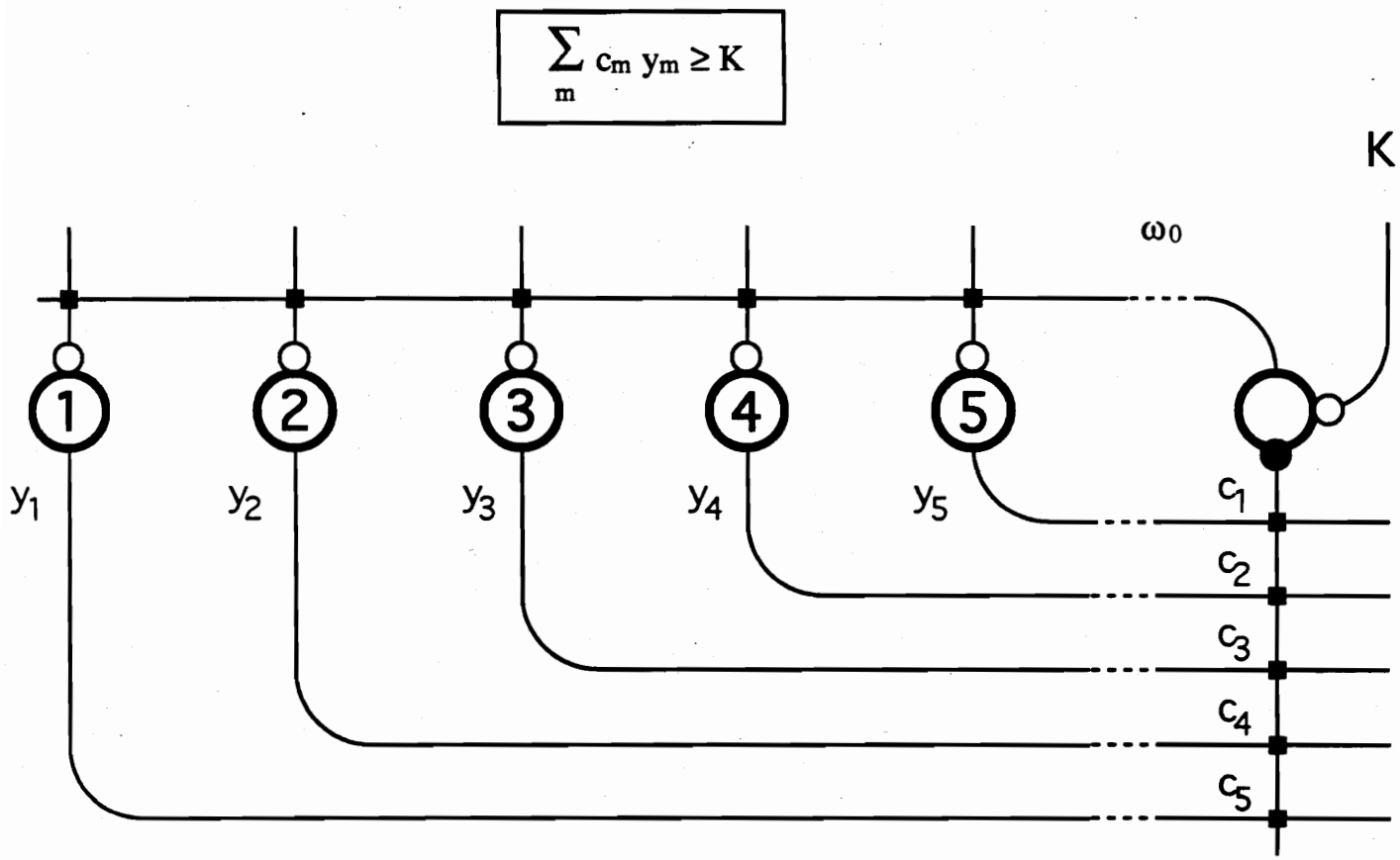

Fig. 5-2: An asymmetrically coupled excitatory guard as a means for implementing a (generalized "at *least* K neurons") lower bound inequality constraint.

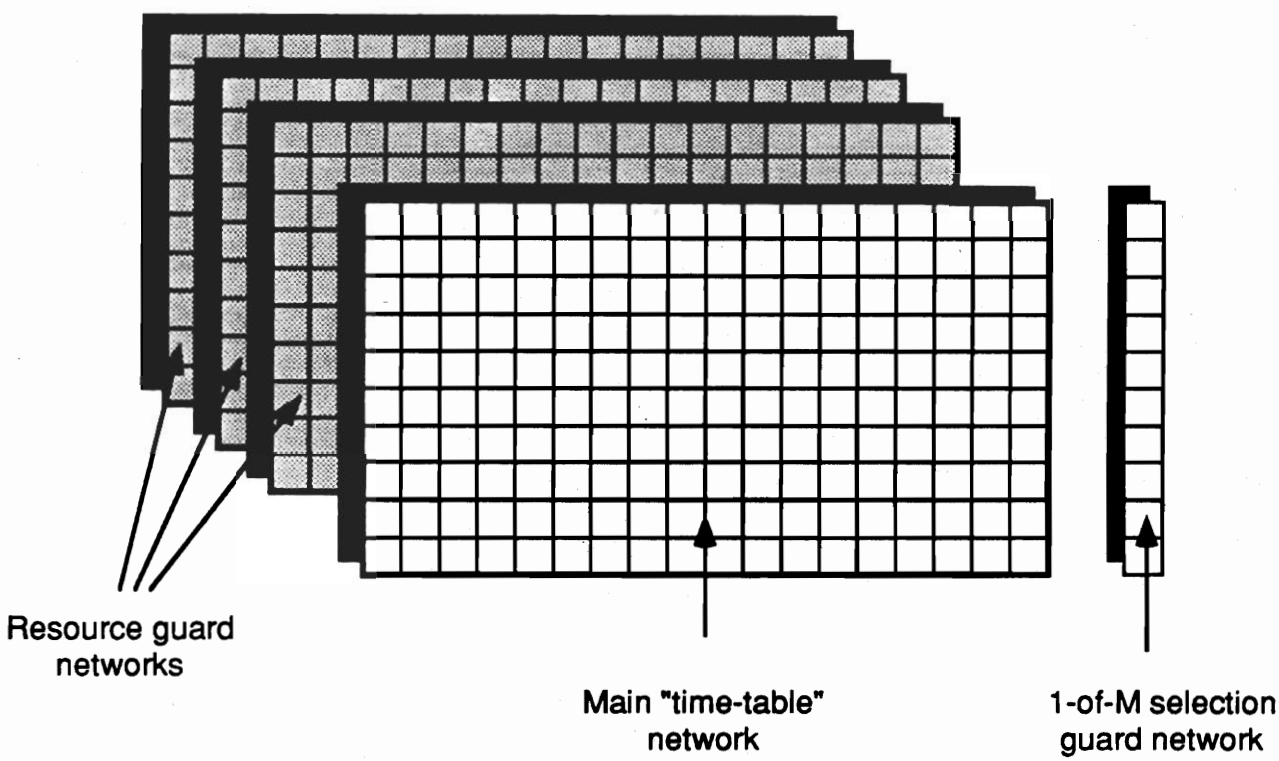

Fig. 5-3: Architecture of connected neural networks for scheduling: the main network consists of an array of N x M neurons (= 0-1 variables = active graph vertices) representing the major part of the combinatorial optimization problem. A network of fast guard neurons, asymmetrically-coupled to the main network, supervises the fulfillment of strict lower-bound constraints. Sheets of resource guard neurons, also asymmetrically-coupled to the main network, guarantee that the resource capacity constraints are fulfilled.

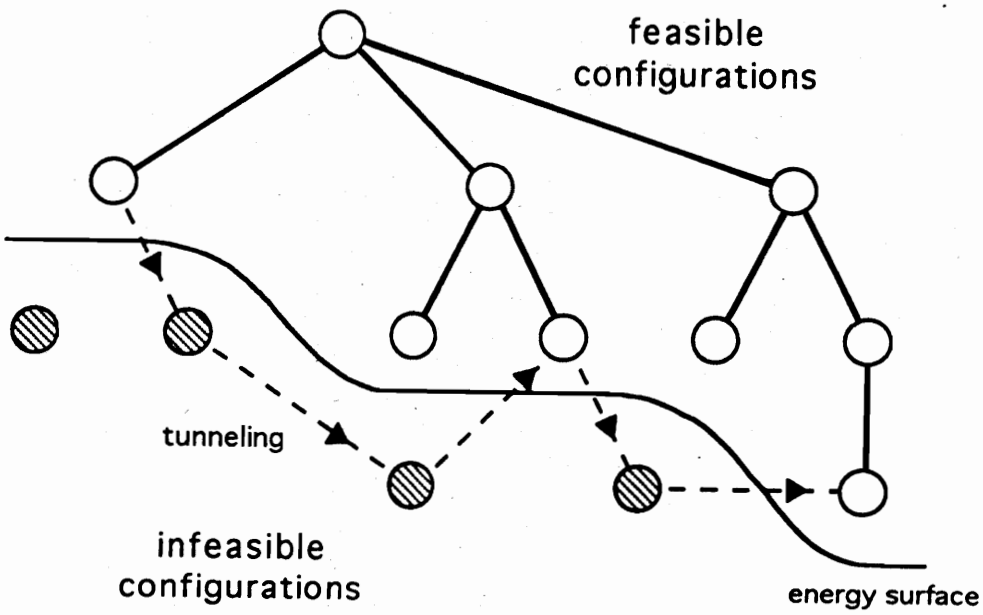

Fig. 6-1: Tunneling through the region of infeasible configurations as a means for efficiently finding good solutions: here each circle indicates a network configuration, arranged in form of a search tree. The GDS-network is allowed to explore promising configurations even when they are infeasible.

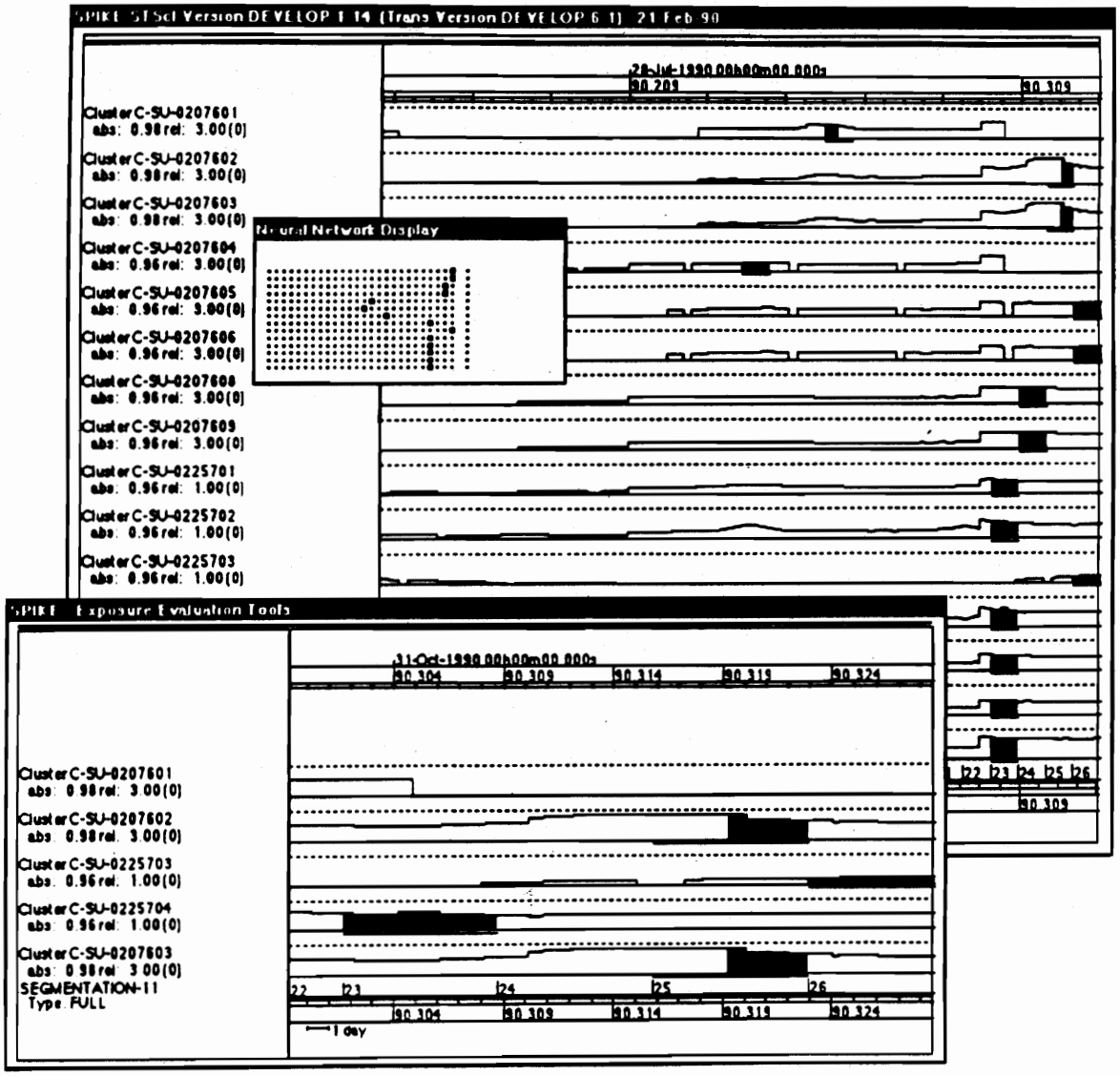

Fig. 7-1: A sample screen showing SPIKE scheduling windows for a small number of activities. The two larger windows display suitability functions for several activities (the one at the bottom is simply a view at higher time resolution). The smaller superimposed window shows a graphical display of the neural network for the same activities: darker squares represent neurons in their "on" state. The displaced row on the right displays "guard" neurons which attempt to ensure that each activity is scheduled at some time.

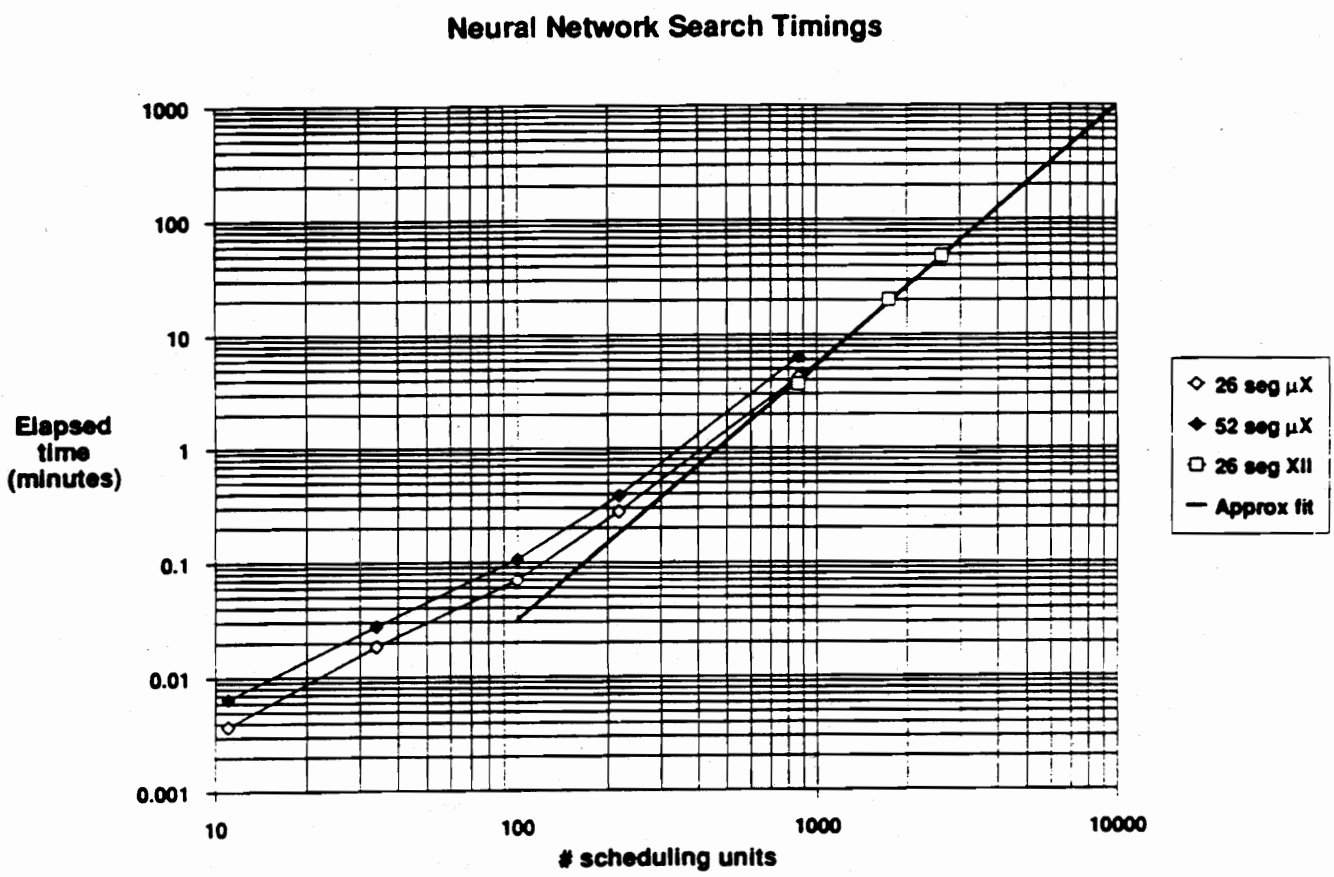

Fig. 7-2: Performance of the neural network as a function of problem size. The open squares show the wall-clock time in minutes as a function of problem size (number of scheduling units) when run on a TI Explorer II workstation (XII): 3000 scheduling units represent about six months of HST scheduling, which can be accomplished in about an hour. The dark line shows the approximately quadratic scaling behavior with problem size. The diamonds represent runs made on a TI MicroExplorer (μX) showing scheduling over six months (26 segments) and one year (52 segments) at one-week resolution.

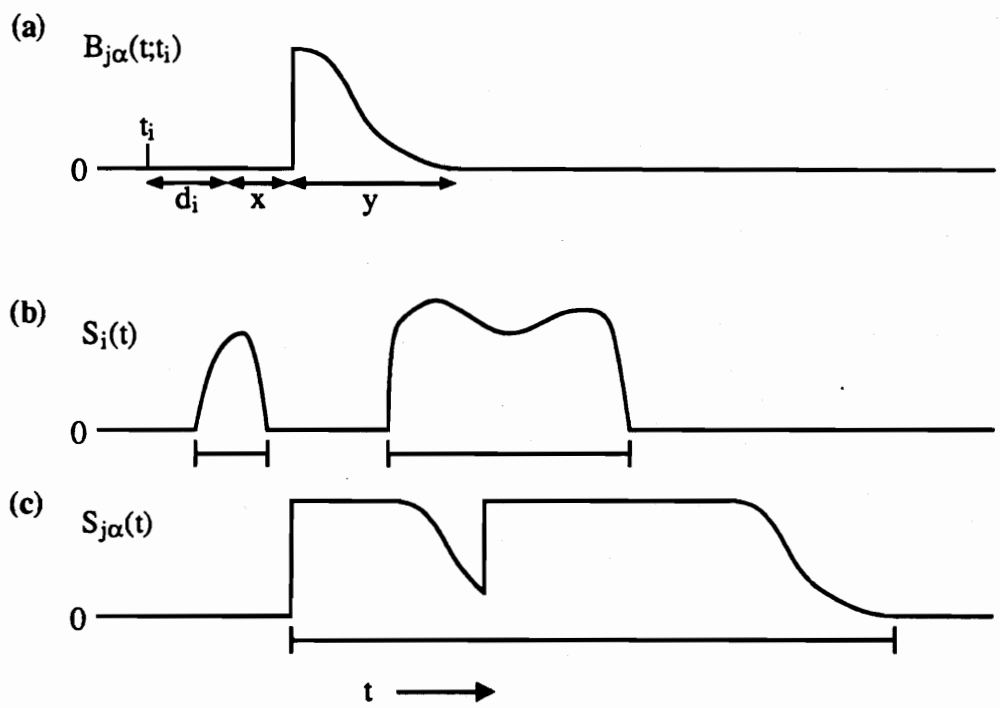

Fig. A–1: Illustration of suitability functions for the case of a binary interval preference constraint: (a) $B_{j\alpha}(t;t_i)$ represents the suitability of scheduling A_j at t given that A_i is scheduled at t_i. The duration of A_i is d_i and A_j is constrained to start no sooner than x and no later than $x+y$ after the end of A_i; (b) hypothetical suitability of A_i at some point in the scheduling process; (c) the resulting suitability of A_j. The intervals where each displayed function is non-zero are indicated by bars under the time axis in (b) and (c).

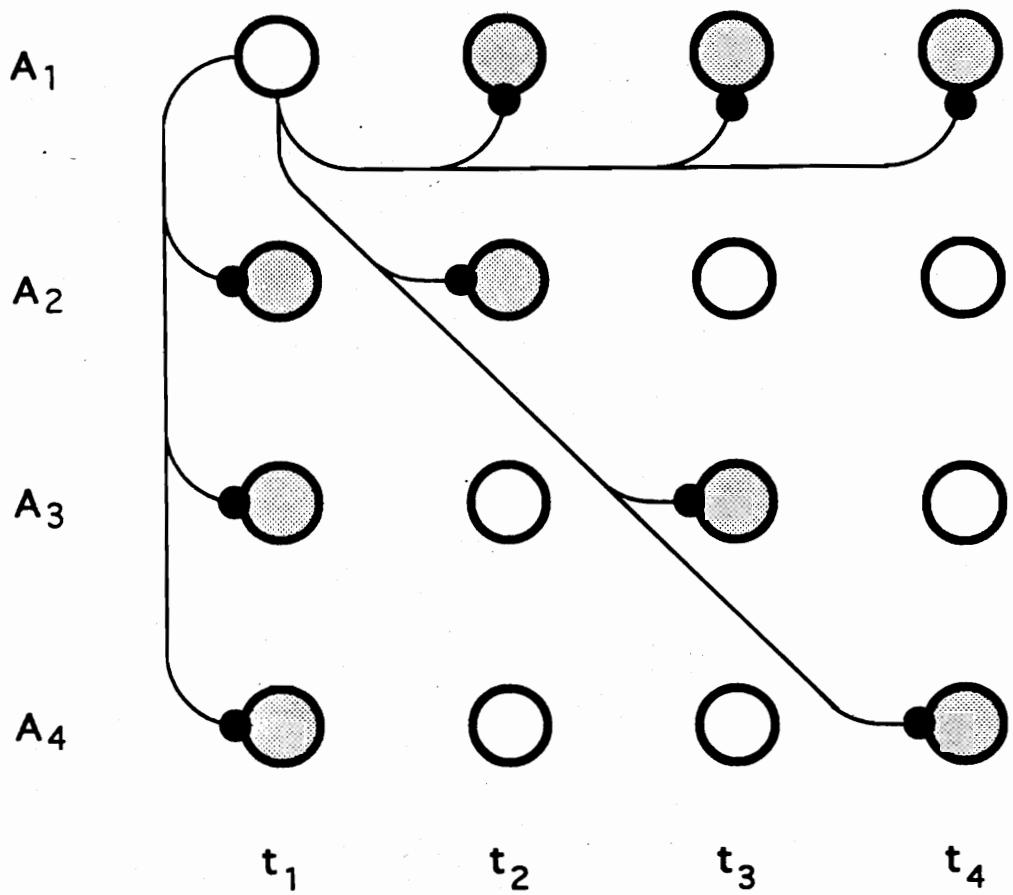

Fig. B-1a: A fragment of the constraint graph G = (V, A) of the toy scheduling problem for N=4. Suppose that activity A_1 is already scheduled at time t_1. The inhibitory edges shown implement the constraints that 1. *activity A_1 should not be scheduled twice* by complete lateral row inhibition, that 2. *no other activity should be scheduled at time t_1* by complete lateral column inhibition and that 3. *no other activity should occur on one of the two timetable diagonals coincident with place (1,1)* by complete lateral diagonal inhibition.

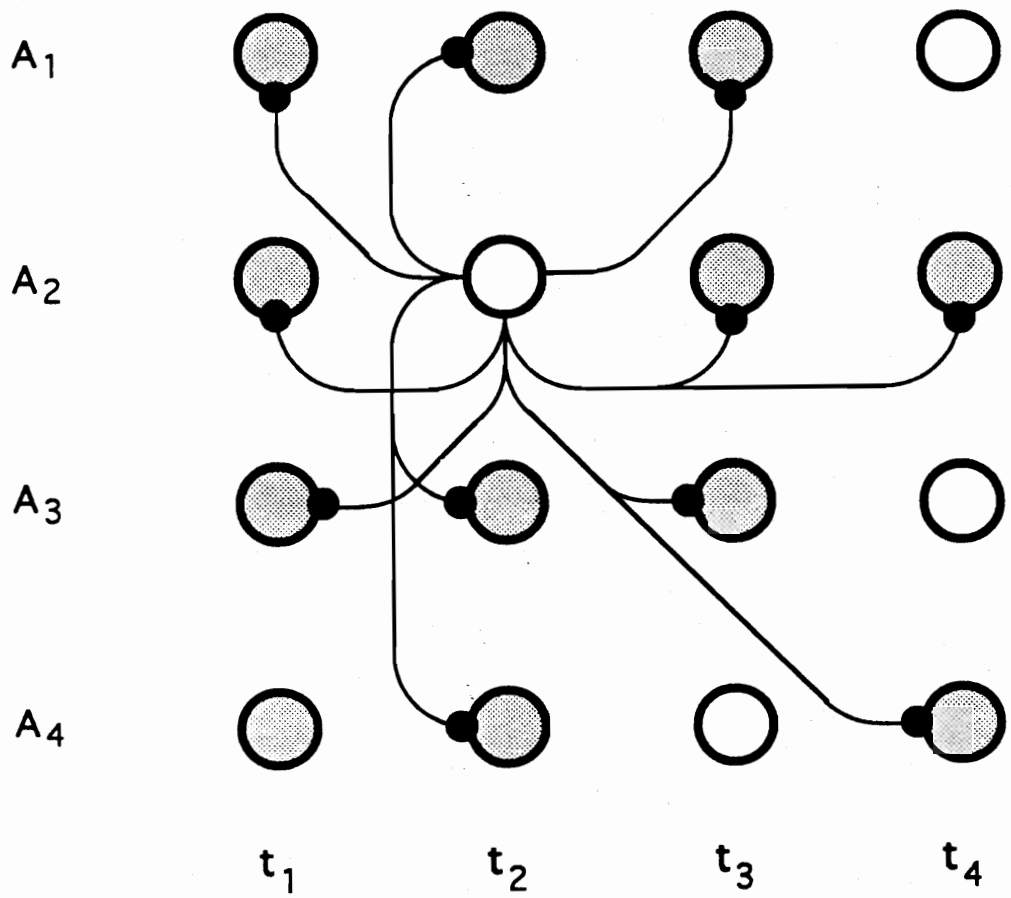

Fig. B-1b: Another fragment of the constraint graph G = (V, A) of the toy scheduling problem for N=4, implementing the outbound constraints for activity A_2 assumed to be scheduled at time t_2.

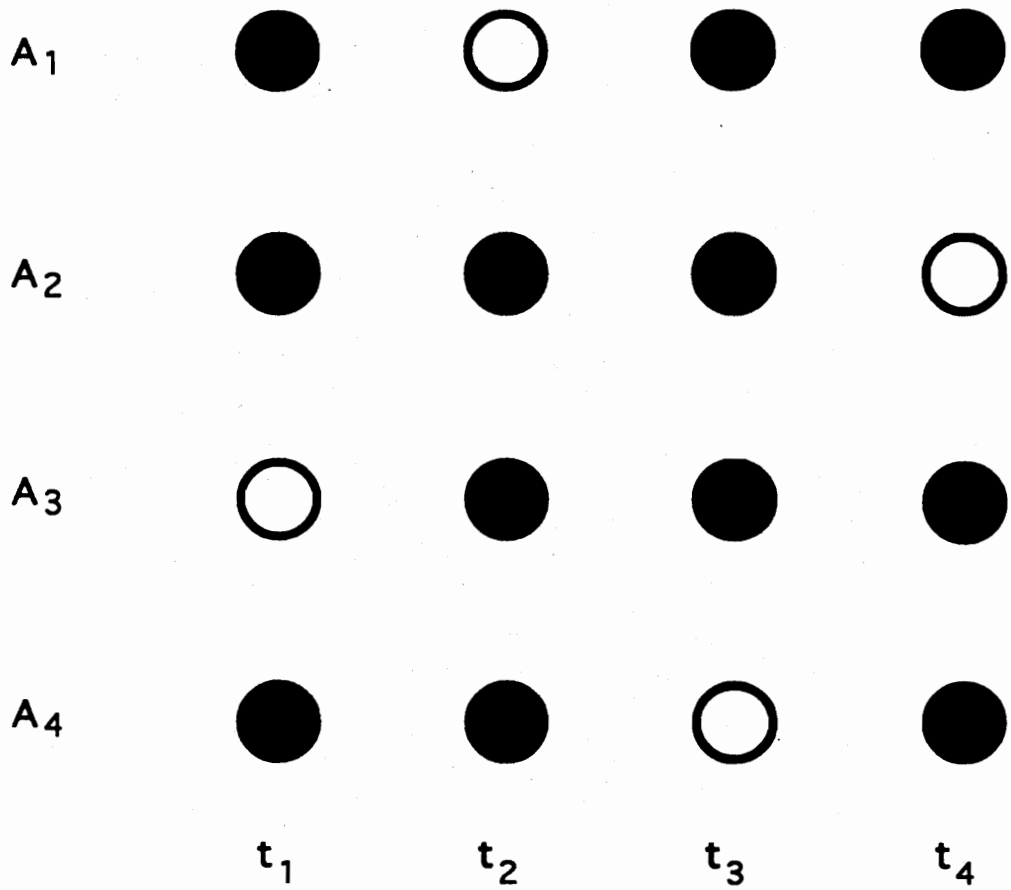

Fig. B-2a: A globally optimal (K = N) solution for the N=4 toy scheduling problem.

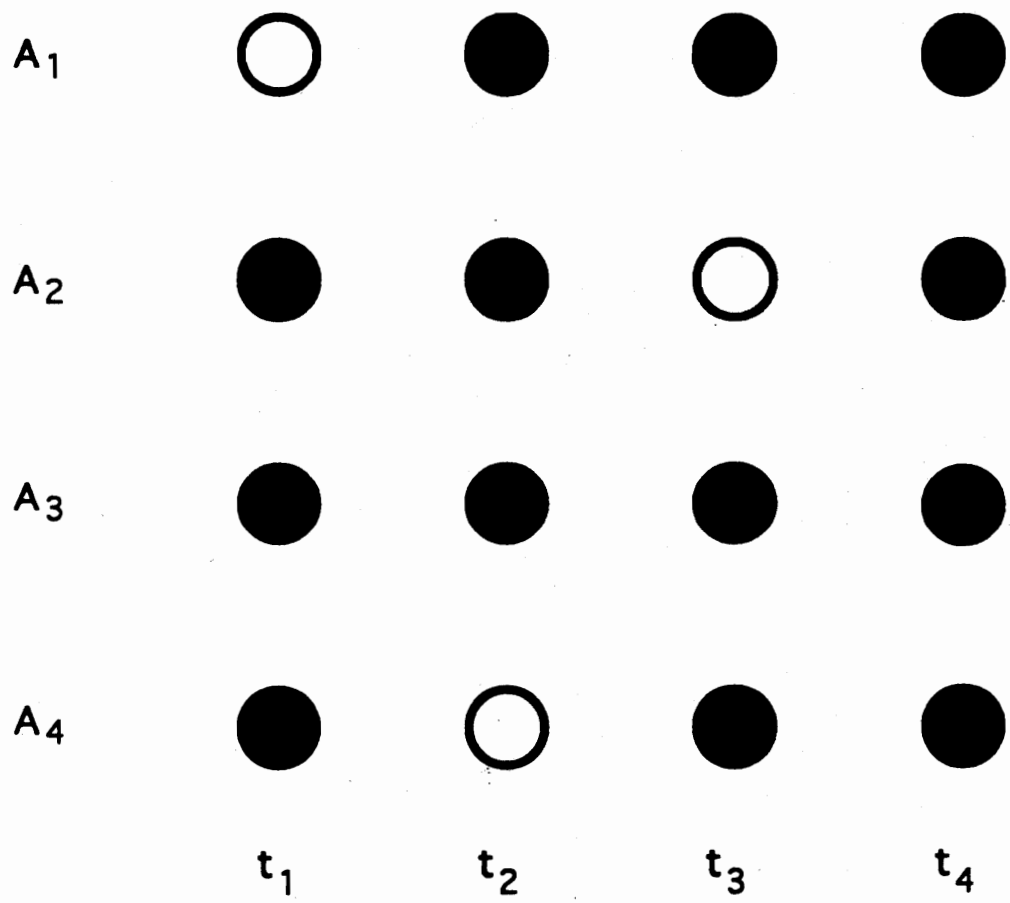

Fig. B-2b: A feasible configuration of our toy scheduling problem for N=4, corresponding to an attractive stable fixed-point of the network dynamics, which cannot be extended to a globally optimal (K = N) solution by placing activity A_3 somewhere.

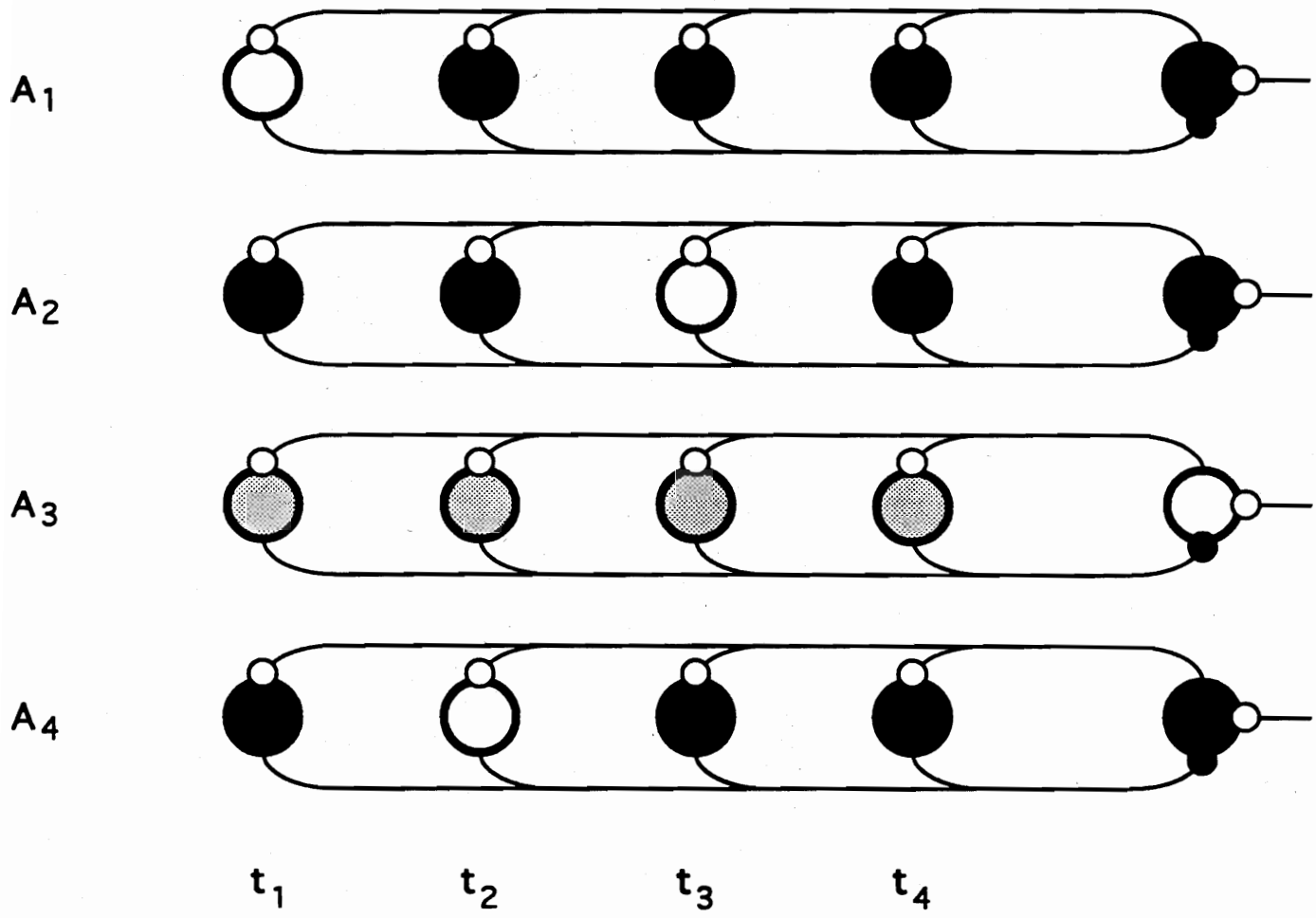

Fig. B-3: An additional network of fast "hidden" guard neurons asymmetrically coupled to the main network allows the implementation of the global constraint that only globally optimum (K = N) configurations of the original problem are feasible.

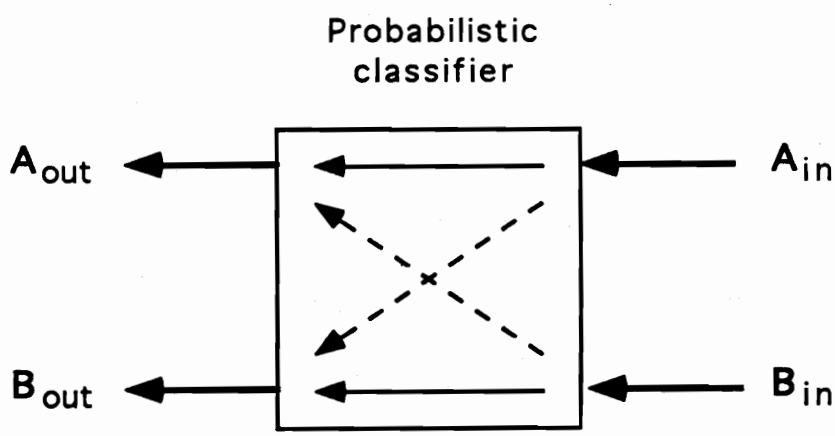

Fig. C-1: Solving instances of a binary decision problem with the help of an unreliable probabilistic classifier: being fed with a problem instance the classifier decides upon either of two output categories, where sometimes this decision is erroneous. Nevertheless, by repeating the classification process sufficiently often the uncertainty about the true category of the problem instance can (in a probabilistic sense) be made arbitrarily small. In a sense the GDS-network when acting on a CSP can be likened to a probabilistic classifier.

Biographies

Dr. Mark D. Johnston is an Associate Scientist at the Space Telescope Science Institute (STScI) where he is Deputy Head of the Science and Engineering Systems Division. He was responsible for the design and development of both the Proposal Entry Processor and SPIKE scheduling system. He received his Ph.D. from M.I.T. in 1978 with a thesis on High-Energy Astronomy Observatory (HEAO-1) X-ray observations of the Large Magellanic Cloud. He has extensive experience with satellite data and has authored or co-authored 30 papers based on the analysis of HEAO-1 X-ray data. His other research interests include numerical hydrodynamics, nucleosynthesis in supernovae, statistics and classification techniques, and the application of advanced software and computing techniques to astronomical research. In recent years he has authored or co-authored about 20 papers in these areas. He was an Assistant Professor of Astronomy at the University of Virginia before moving to STScI in 1982.

Hans-Martin Adorf, a theoretical physicist by education, has been working in astronomy since 1980. In 1985 he joined the Science Data and Software Group of the Space Telescope — European Coordinating Facility, where he is employed as data analysis scientist within the Hubble Space Telescope project. He is one of the founders of the "ST-ECF Artificial Intelligence Pilot Project" and has sporadically participated in the SPIKE project. While the center of gravity of his professional work has recently shifted to HST image restoration, he maintains an active interest in the application of AI techniques to astronomical problems. He has authored or co-authored more than 35 papers in the area of expert systems, neural networks, databases, observatory operations, data analysis, statistical pattern recognition and classification of astronomical objects.