A Constraint-Based Approach for Signal Acquisition Control in Magnetic Resonance Imaging and Spectroscopy (MRI/MRS)

Xiaowei Song
Evangelos Milios
Malcolm Heywood
Benjamin Rusak

Technical Report CS-2003-02

March 5, 2003

Faculty of Computer Science
6050 University Ave., Halifax, Nova Scotia, B3H 1W5, Canada

A Constraint-Based Approach for Signal Acquisition Control in Magnetic Resonance Imaging and Spectroscopy (MRI/MRS)

Xiaowei Song[1,3], *Evangelos Milios*[1], *Malcolm Heywood*[1], *Benjamin Rusak*[2]

[1] Faculty of Computer Science,

Dalhousie University, Halifax, N.S., Canada B3H 1W5

`xsong@dal.ca, {eem,mheywood}@cs.dal.ca`

[2] Departments of Psychiatry and Psychology

Dalhousie University, Halifax, N.S., Canada, B3H 4J1

`Benjamin.Rusak@dal.ca`

[3] Geriatric Medicine Research Unit,

Queen Elizabeth II Health Sciences Center,

Halifax, NS B3H 2E1, Canada

Abstract

Magnetic resonance imaging technology is one of the most promising investigative and diagnostic methodologies in brain research and in clinical neurological practice. To acquire informative magnetic resonance signals, a variety of equipment parameters require identification. However, depending on which neuronal metabolites and brain tissues are of interest and on what types of clinical or research questions are being asked, the relation between equipment parameters is often complex. This complexity means that computerized tools that can design valid novel settings to control the activities of the magnet during data acquisition would be beneficial. The goal of the present research is to build such a tool using constraint satisfaction based on the constraints associated with different imaging protocols. A constraint-resolving program, which is generated automatically from the constraints provided by the user in text form, enumerates the value domains of the variables, enforces the constraints associated with each possible setting, and produces a complete set of all valid parameter settings for controlling the magnetic device during signal acquisition.

List of Abbreviations

AC	Arc Consistency
AVG	Average Number of Scans
BCSP	Binary Constraint Satisfaction Problem
CFG	Context-Free Grammar
CGN	Constraint Graph Network
CO	Constraint Optimisation
CP	Constraint Programming
CSP	Constraint Satisfaction Problem
CT	Computerized Tomography
CTQ	Consistency Technique
DP	Data Point
FOV	Field of View
FT	Fourier Transform
GT	Generate and Test
LA	Look Ahead
LB	Look Back
MR	Magnetic Resonance
MRA	Magnetic Resonance Angiography
MRI	Magnetic Resonance Imaging
fMRI	Functional Magnetic Resonance Imaging
MRS	Magnetic Resonance Spectroscopy
NEX	Number of Averages
NMR	Proton Nuclear Magnetic Resonance
PC	Path Consistency
PET	Positron Emission Tomography
RF	Radio Frequency
SE	Spin Echo
SNR	Signal to Noise Ratio
T1	Longitudinal Relaxation Time or Spin-Lattice Relaxation Time
T2	Transverse Relaxation Time or Spin-Spin Relaxation Time
TE	Echo Time
TI	Inversion Time
TR	Repetition Time
TT	Total Time

Contents

1 Introduction

This report addresses the use of constraint programming (CP) to solve a constraint satisfaction problem (CSP) in the field of magnetic resonance imaging. A generic solution is presented for the control of parameter settings and signal acquisition for magnetic resonance imaging and spectroscopy (MRI/MRS). The associated MRI/MRS constraints are investigated and algorithms developed to represent, recognize, parse, and resolve the associated constraints.

Imaging of the human brain using magnetic resonance (MR) technology includes magnetic resonance imaging (MRI), magnetic resonance spectroscopy (MRS), and functional magnetic resonance imaging (fMRI). MR technology is a promising modern method for the study of brain structures and functions [13, 40]. It is also very important in the diagnosis of brain damage and brain disease [13] and can be used to map physiological parameters of neural metabolism to ongoing brain activities and mental processes [4, 38].

Acquisition of the raw magnetic resonance signal is the first step for a MR examination. MRI reconstructs the signals to provide remarkably high-resolution images of brain anatomy on the computer screen. MRS extracts information from the signals to determine the nature and the amount of certain neurotransmitter and neural metabolite molecules existing in a well-defined volume of the brain [30, 35]. Depending on which neural processes or neural metabolites are being imaged, the size and location of regions of interest to be analysed, and the types of clinical or research questions being asked, the MR signal acquisition parameters need to be varied in complex ways [23, 33, 35].

Because no standard sequence patterns are available for many combinations of target imaging features, individual researchers often must develop their own scanning protocols for individual MR devices. Unfortunately, devising appropriate stimulation sequences and other necessary parameters for obtaining the MR signals, and programming these into a computer that controls the performance and operation of the MRI system is a very laborious and time-consuming task [19]. In addition, the sequence settings, developed with great effort, may turn out not to be useful because they failed to address the complex effects of the diverse variables associated with imaging properties [28].

The overall objective of the research reported here is to develop algorithms to support the control of MRI/MRS signal acquisition. This consists of the following objectives: a) To define the MRI/MRS parameter setting problem; b) To extract heuristics from MR physics and neuroscience knowledge bases and to apply them in the development of constraints on parameter settings; c) To implement and evaluate a constraint satisfaction program for this problem.

The research has successfully fulfilled the following objectives.

a) The MRI/MRS data acquisition control problem has been characterized as a constraint satisfaction problem (CSP) that can be tackled by constraint programming (CP; [8, 41]).

b) The MRI signal acquisition control parameters that determine imaging features have been interpreted as appropriate input and output entities (variables). The value domains of these variables have been defined based on MR physics and neuroscience knowledge.

c) The complex interrelations among the entities have been defined, interpreted, and formulated into several types of constraints. Users are allowed to specify and/or modify the constraints to meet their special measurement needs.

d) A constraint-resolution program has been designed and implemented to propagate the constraints and obtain valid parameter settings.

e) Search strategies based on CP have been applied to reduce the search space to a practical size and enumerate all feasible parameter settings in a given imaging context.

Section 2 of this report introduces MRI/MRS technology and its applications in neuroscience research, and the principles of MRI/MRS data acquisition control. It then defines the MRI parameter-setting problem and reviews previous efforts to provide a solution.

Section 3 reviews relevant techniques in Constraint Satisfaction and Constraint Programming.

Section 4 then formulates the MRI/MRS parameter-setting problem into a Constraint Satisfaction Problem.

Section 5 describes the algorithms and data structures constituting the generic constraint-resolving engine. It gives detailed information on how the constraints are formulated, recognized, parsed, enforced, and resolved and how implementation efficiency has been achieved with the use of heuristics.

Section 6 shows system performance and sample outputs of the generic constraint-resolving engine.
Section 7 summarizes the contributions of the research, and discusses its limitations and further research directions.

2 Related Work

2.1 Magnetic Resonance Technology and Brain Imaging

MR technology relies on the differential decay and recovery characteristics of the proton nuclear magnetic resonance (NMR) signal (i.e. relaxation behaviour) to detect differences in magnetic resonance in localized regions [26]. Discovered in 1946, the NMR phenomenon was originally developed and used for chemical and physical molecular analysis.
NMR began to be considered for medical purposes in the early 1970's when the nuclear magnetic relaxation times of tissues were found to be different from those of tumours [13]. It was demonstrated on small test tube samples [25] shortly after x-ray-based computerized tomography (CT) was introduced. The localisation technology was significantly improved in 1975 when phase and frequency encoding and the Fourier Transformation (FT) were applied [13]. In 1977, at the time when positron emission tomography (PET) was developed, MRI of the whole body was demonstrated. In 1987, MRI started to be used to perform real-time imaging of a single cardiac cycle. Since then, efforts have been made to reduce imaging time, and the time required to obtain a single image has finally reached only a few seconds [39].
In 1993, three years after magnetic resonance angiography (MRA) was perfected, allowing imaging of flowing blood without the use of radioactive contrast agents, the functional MRI (fMRI) method was first developed [4, 6]. This opened up a new application era for MR technology by providing views of the functional activity of living brains. Thus researchers can not only observe the structure, but also investigate the functions, diagnose diseases and monitor activity of the living brain and other tissues [21, 16].
In comparison with other modern brain imaging technologies such as CT and PET, MR technology has the following advantages. It provides detailed views of the brain in spatial

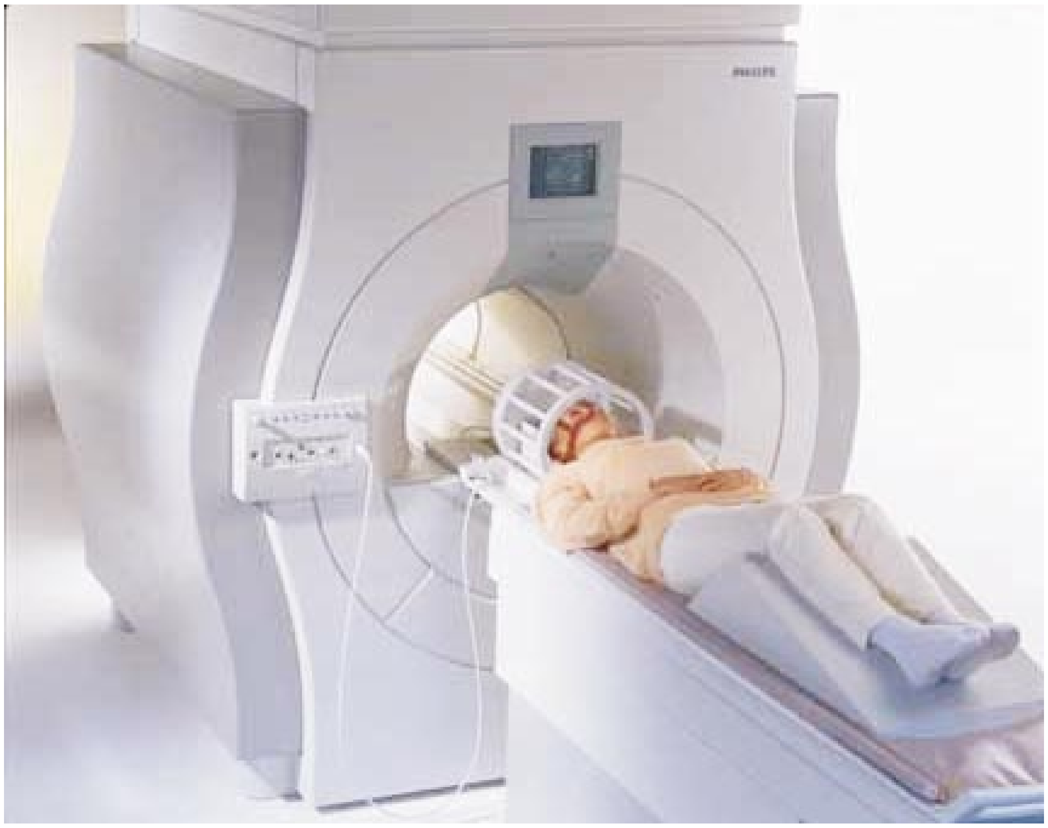

Figure 1: A typical scanner for MRI/MRS exams

dimensions and/or information on important neurochemicals with no x-ray or radioactive isotope being used. It is non-invasive, safe, and painless. It involves no blood sampling or special preparations. With the ongoing progress in MR technology aimed at improving the signal acquisition speed, the quantitative accuracy for localizing characteristic signals and data processing quality and spatial resolution, MRI will play an important role in many aspects of brain research. These include basic research, clinical investigations and treatments related to the nervous system and other organs [5, 26, 16].

The milestones of MRI technology are listed in appendix B.

MRI/MRS is based on absorption and emission of energy by protons (mainly hydrogen, ^1H) in the radio-wave frequency range of the electromagnetic spectrum. Measurement and interpretation of the emitted energy in brain tissues can provide an anatomical view of the brain as well as neurochemical and functional information [17]. Figure 1 shows a typical scanner used for MRI/MRS studies.

Tissue components such as lipid and water in the brain have a large number of hydrogen and other atoms. The atomic nuclei spin on their axes. Moreover, each spinning positive electronic charged particle acts as a magnet with the poles located in the axis of spin. When a patient is subject to a strong, externally applied superconductor magnetic field, the spin axes of all these tiny magnets align with the field. This creates an average vector of magnetization of the nuclei that is oriented parallel to the magnetic field [17].
When a radio frequency (RF) pulse is broadcast toward the patient in a line perpendicular to the magnetization vector, it causes the net magnetization vector to deviate from the main magnetic field and the magnetization of individual nuclei to diverge at different angles from their original axes (Figure 2). The intensity and duration of the RF pulse control the amount by which the magnetization vector tilts away from the z-axis.
After deactivation of the RF pulse, the net magnetization vector gradually (over a period of $200 - 2,000$ $msec$ for biological substances) returns to the state of being parallel with the main field. The time that this takes is referred to as the *T1 relaxation time* (longitudinal relaxation or spin-lattice relaxation; Figure 2). In addition, the average precession angle of the nuclei returns to its normal low value over a shorter period of $30 - 200$ $msec$, referred to as the *T2 relaxation time* (transverse relaxation or spin-spin relaxation; Figure 2). The rate at which this proceeds differs from tissue to tissue ([17]; Figure 2).

As the nuclei relax with respect to T1 and T2, each becomes a miniature radio transmitter, providing a characteristic resonance signal that changes over time. Based on the Larmor relationship, when nuclei relax, the frequency that they transmit is positively correlated with the strength of the surrounding magnetic field (Appendix C). Different nuclei resonate at different frequencies and similar atoms in different environments have different resonance frequencies [35]. For example, a hydrogen attached to an oxygen and a hydrogen attached to a carbon, or a hydrogen in lipids and a hydrogen in water have different microenvironments, and thus transmit different signals. Different tissues give characteristic resonance signals due to the different water-to-fat ratios [9, 35]. These signals are acquired as raw resonance signals and are the basis for determining the chemical composition of the target brain tissues (i.e. MRS; Figure 3) and/or for forming anatomical images of the brain (i.e. MRI; Figure 4).

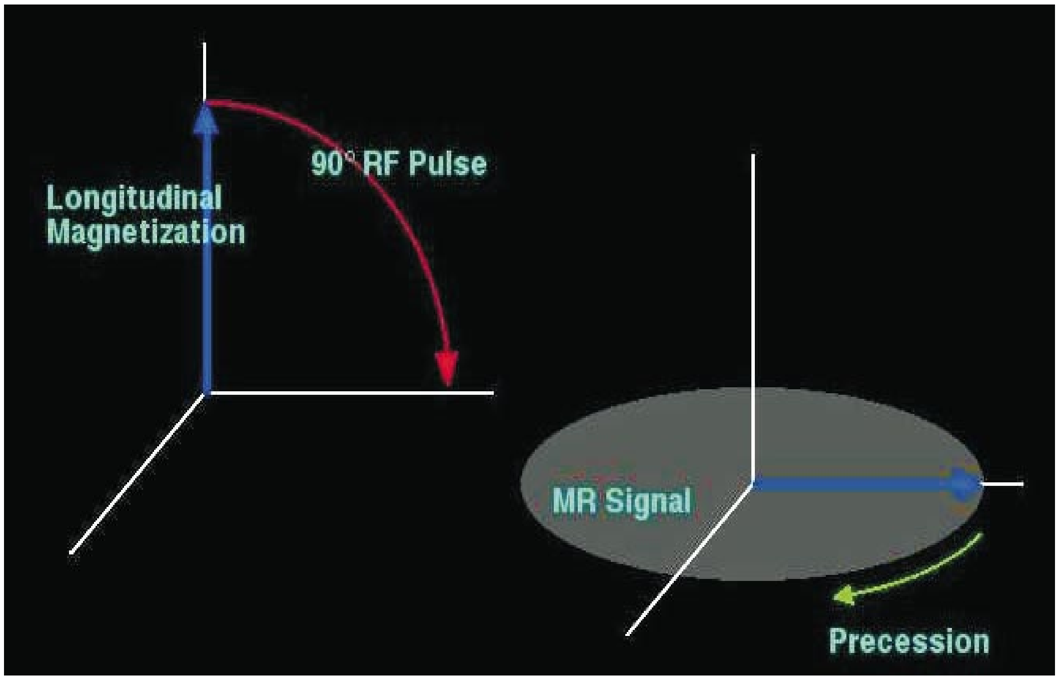

Figure 2: Tilt of the axes of the nuclei precessing when a radio frequency pulse is applied

In the spin-echo (SE) method that is usually applied in MRI/MRS measurements, a magnetic field gradient system consisting of gradients in x, y, and z dimensions is employed to perform three-dimensional signal localisation [35]. Information from various dimensions is extracted and transformed by means of Fourier transformation (Appendix D).
The SE technique applies a 90^o RF pulse followed by a series of 180^o RF pulses (i.e., a pulse sequence). The entire sequence must be repeated and averaged multiple times in order to get stable enough resonance signals with a good enough signal-to-noise ratio (SNR)[35]. The *repetition time (TR)* refers to the time between repetitions of the RF pulse sequence [17]. The *echo time (TE)* refers to the time between the 90^o pulse and the maximum magnetization in the echo in a spin-echo sequence [17].

2.2 MRI Signal Acquisition Problem

A typical MRI system consists of the following components (Figure 5).

1. A large and extremely strong magnet (1.5 - 4 T), in which a patient is placed.

2. Gradient systems to determine 3-D scanning localization.

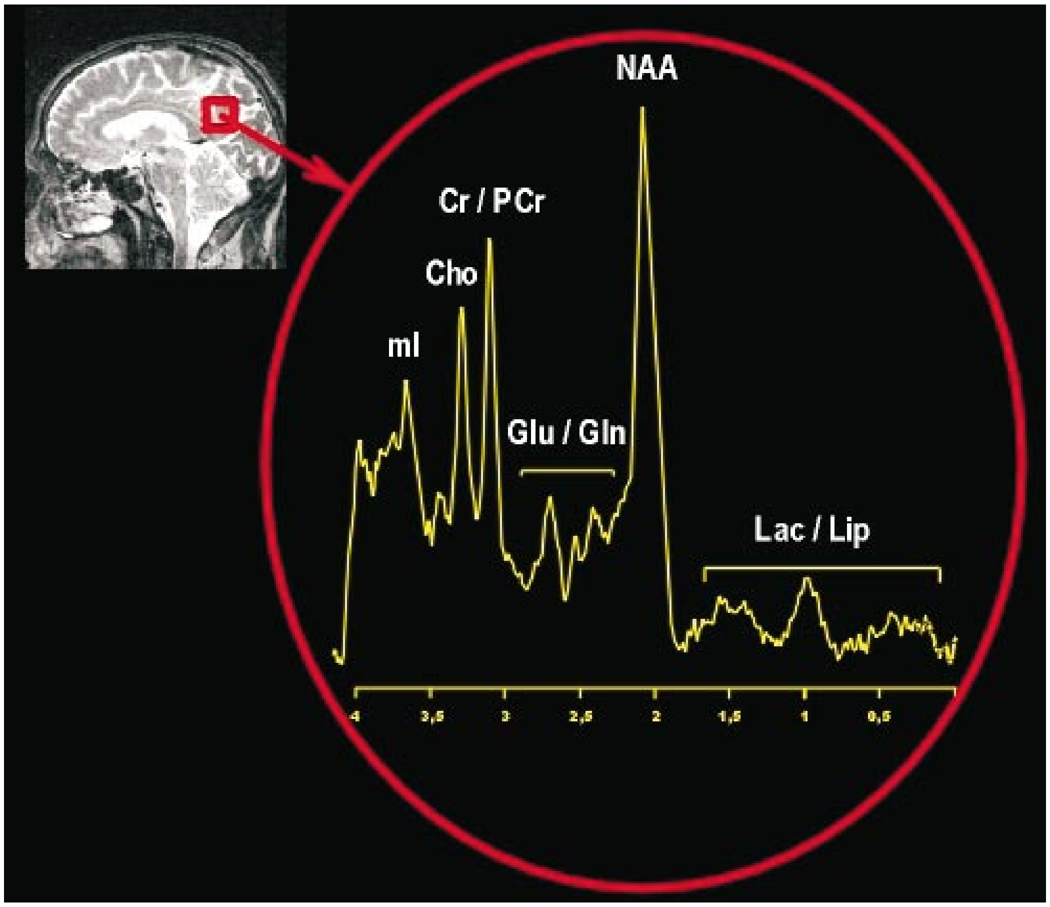

Figure 3: ^1H brain spectrum for some neural compounds. ml: myo-Inositol; Cho: Choline; Pcr: phosphocreatine; Cr: Creatine; Glu: Glutamate; NAA: N-Acetyl aspartate; Gln: Glutamine; Lac: Lactate; Lip: Lipid.

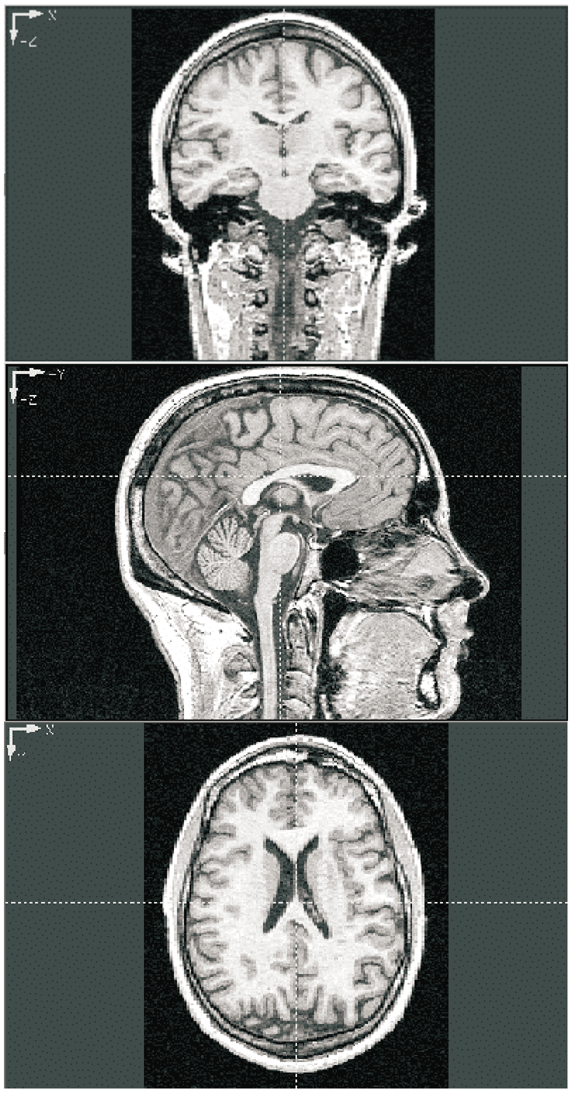

Figure 4: MRI brain images from 3-D views. Top: Coronal; Middle: Sagittal; Bottom: Transverse

3. A radio frequency transmitting and receiving system that sends RF signals ($10^4 - 10^8$ Hz) to the patient being measured and then gets resonance signals back.

4. A viewing console. The raw resonance signals are transformed and converted into information about the MR images and spectra (and often further quantified as neurochemical concentration) using analytic software on a computer system.

5. A scan controller, that is usually located in the same computer as the viewing console. It is the component that directs all the actions in the MRI/MRS measurements to acquire and process the data. It controls the variables that determine when to turn on and off the gradient system and the RF transmitter to obtain the proper pulse sequence. It also controls the RF receiver amplifier and the A-D converter to digitize the signal. Through these, it determines which characteristic resonance signals will be generated.

Since the MRI/MRS characteristics depend on many factors, including proton density, T1 and T2 relaxation effects, and effects of flow, diffusion and acquisition time etc, there are several types of control information.

1. Localization and orientation factors, including regions of interest and volume of elements.

2. Tissue contrast and SNR factors such as T1-weighted or T2-weighted, timing and RF pulse sequence.

3. Proton resonance characteristics, such as composition and properties of neural metabolites of interest (Appendix E).

4. Spatial resolution and spectral dispersion, such as number of slices and averages, etc.

This information is conveyed to the controller by entering the values of a number of parameters that ultimately determine the characteristics of the signals recorded. It is necessary that the set of parameter values used be properly determined for a required MRI/MRS protocol before the scanning process can start [23].

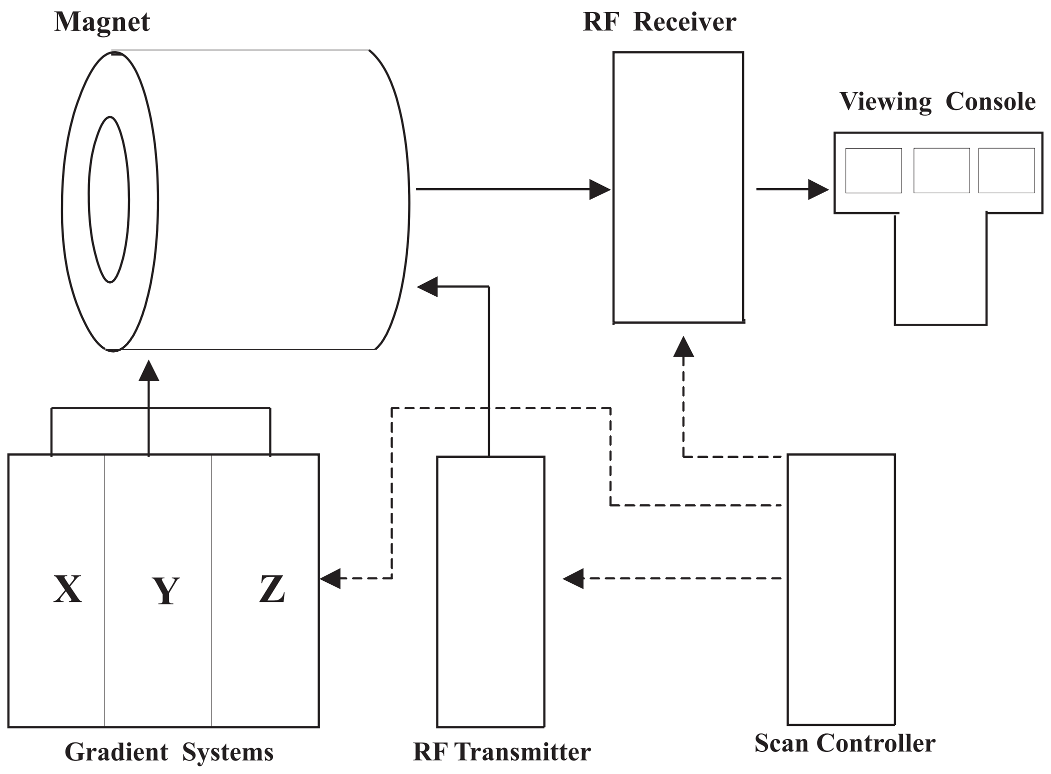

Figure 5: Major MRI system components

The parameters that need to be determined for a typical single voxel MRI/MRS study with a system such as the one being used in the QEII Health Sciences Centre in Halifax [19] are listed in Appendix D. The variable domains for these parameters from which the values of each parameter may be chosen are also listed (Appendix F). The MRI/MRS signal acquisition parameters are of different types. Some of them can be determined easily from the requirements of given scanning tasks. Examples of this kind of parameter include the neurochemicals and region of interest. Other parameters, however, have to be derived by dealing with the complex relationships among the parameters that are related to the physics involved or to basic features of the physiology of the targeted brain tissue. Examples of these kinds of parameter include the total number of acquisitions that are feasible, the gradient orders and RF pulse sequence.

Various combinations of the parameter values allow researchers to tailor their signal acquisition to particular research questions, varying localization strategies and evaluating different features of neuronal function. Because the effects of these parameters on MRI/MRS signal characteristics are interrelated, changing the value of one parameter often influences the others. Adjustment of many of these parameters involves tradeoffs, depending on which neuronal processes, neurotransmitters or other neural metabolic processes are more important to examine [18]. The tradeoff table for selecting parameter values is given in appendix G.

Furthermore, there are MRI system constraints as well as scanning task-related constraints that restrict the choice of values for each parameter. For a MRI/MRS system to accept a certain parameter setting for a scanning protocol and generate meaningful results, all the constraints have to be satisfied. This requires not only identifying the variety of signal acquisition parameters that need to be set appropriately, but also resolving conflicts arising from the complex interrelations among these parameter values.

For instance, to be able to obtain spectra of certain neurochemicals and metabolites with clearly displayed chemical shift dispersion (Fig. 3) as well as a good spatial resolution (Fig. 4), a TR value has to be properly chosen which should be three to five times the T1 value for those neurochemicals. The proper choice of a value for TR is influenced, however, by the assigned TE value. The choice of TE is also dependent on what the neurochemical targets of interest are, which part(s) of the brain is/are the target to be examined, and what kinds of

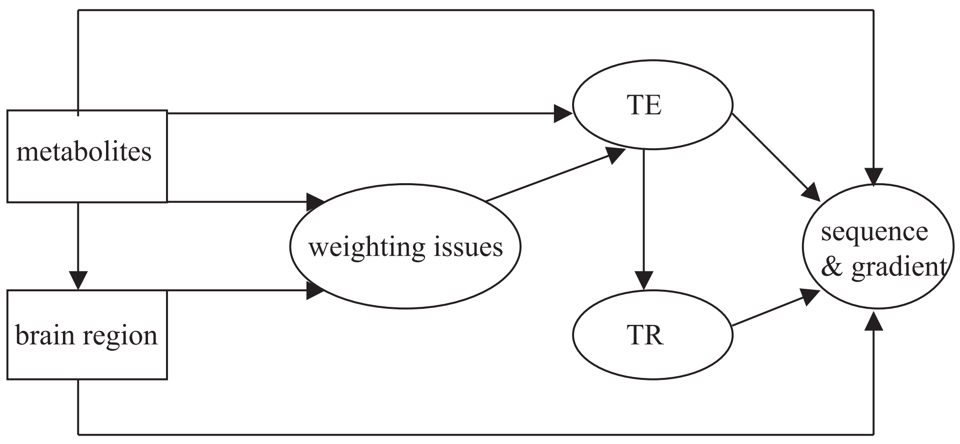

Figure 6: An example of MRI parameter value determination

imaging features (i.e., T1-weighted or T2-weighted) are of most interest. Generally speaking, with increasing TE, the spectra of those compounds with the shorter T2 become less clear, while those with longer T2 become more prominent. Similarly, with decreasing TR, the spectra of compounds with a shorter T1 will appear more prominent [9, 37]. In addition to resolving all of these issues, the gradient order and RF sequence may also need to change, depending on the combinations of values selected for TE, TR etc. The relation among these parameters is illustrated in Fig. 6.

It is easy to appreciate that manually resolving the relations among these parameters to determine the appropriate value for each of the many parameters involved can be quite difficult. As a result, MRI researchers may devote a great deal of time to working out a single setting when developing a new MRI research protocol. Nevertheless, the array of settings selected may be found to be unacceptable by the system, that is, unresolved conflicts among experimental or system constraints can arise. Alternatively, the system may accept the derived settings, but the output signals may not produce meaningful MRI/MRS results because of emerging constraints that have not been addressed.

The above difficulties associated with the identification of customized, manual setting of resonance signal acquisition parameters, provide a requirement for the development of computerized tools to determine valid parameter settings for various scanning tasks. This would help MR researchers and benefit the development of new MRI/MRS application protocols. Such a parameter setting tool should therefore provide the following functions:

1. Determine valid MRI system parameter settings for any given scanning task. After getting a list of input parameters that specify the measurement task, the program should output all the valid MRI system parameter settings that may be used.

2. Optimise the valid MRI system parameter settings. If there is more than one set of valid parameter settings for a certain task, the program should indicate those that best meet the task requirements.

3. Provide setting suggestions. If there is no valid set of MRI system parameter settings for the input values provided by the user, the program should detect the critical conflicts. In addition, if possible, it should provide suggestions for how to modify input parameters to make the task feasible.

3 Constraint Satisfaction Problems and Constraint Programming

In this section we review the main ideas in the literature of constraint satisfaction and constraint programming. In the following section we apply these ideas to the MRI parameter setting problem.

3.1 Constraint Satisfaction Problem

A constraint satisfaction problem (CSP); $P = (X, D, C)$ consists of:

1. A finite set of variables; $X = \{x_1, x_2, ...x_n\}$.

2. A finite set of domains; $D = \{D(x_1), D(x_2), ...D(x_n)\}$. Set $D(x_i)$ represents a finite set of possible values for variable x_i. All of the variables have a corresponding finite domain of values.

3. A set of constraints associated with the variables; $C = \{c_1, c_2, ...c_m\}$. Each of the constraints C_i expresses a logical relation among some of the variables in X. An entire set of constraints of a CSP thus represents complete information about the relations among the variables, and restricts the number of acceptable combinations of values that can be assigned to the variables [41].

Typically, constraints in a CSP share the following properties [2].

1. Declarative: Constraints specify what relation must hold without clarifying a procedure to enforce that relation.
2. Non-directional: A constraint implies a particular value for the variable X, given a particular condition being applied to variable Y and vice versa.
3. Representative: Constraints may specify only a partial relation and need not uniquely specify all the values in the variable domains.
4. Additive: While the conjunction of constraints at the end of enforcement does determine the correctness of the solution, the order of constraint imposition does not matter.
5. Dependent: Limits to a given constraint set often share variables in the relations that they define.

3.2 Constraint Programming

Specification of constraints is a natural mechanism to describe problems in human activities, and satisfaction of multiple, sometimes conflicting, constraints represents one of the most common problems in the real world that require efficient software strategies to address [41, 42]. Constraint programming (CP) emerged to meet this need. The objective of CP is to study computational systems based on constraints [20].
Research on CP started in the 1970s, and the systematic use of CP began in the ¡ 1980s [20]. CP has already become a popular and powerful technique that declaratively describes and

effectively solves many practical constraint satisfaction problems, especially large combinatorial problems in areas such as planning and scheduling [2].

A typical CP process may be described as follows. The user provides requirements (i.e., constraints) and the program generates solutions to the CSP by providing a set of values for all the variables, which do not violate any of the constraints. Alternatively, CP may find that there is no solution to the CSP, if no such values can be found. Thus, in fact, the task of CP is a combinatorial search problem in finding one or all value assignments to the variables satisfying all the constraints [2].

A straightforward but expensive algorithm to accomplish the search task is that of systematically generating and testing (GT) all value combinations. The approach is prohibitively expensive for all but the smallest problems. Therefore, developing strategies that accomplish the task more efficiently, at least for a subset of problems, has been one of the most important aspects of modern CP research [29].

While recent research on CP concentrates on generating novel models and faster algorithms, classic CP approaches have provided, and continue to provide, the most reliable and complete strategies [1, 15]. Hence, classic CP technologies are widely and successfully applied in solving practical CSPs in the real world [42, 29]. Some of the most important CP techniques are introduced in the next subsection.

3.3 Classic Constraint Programming Techniques

Classic CP algorithms that are often employed to tackle typical CSPs in practice include the following [1, 2, 22, 29].

1. *Look back (LB).* The validity of the constraints is tested immediately after the variables relevant to the constraints are added into the assignment list. This algorithm solves a CSP by incrementally extending a partial solution that specifies consistent values for some of the variables towards a complete, consistent solution. If a partial solution violates any of the constraints, backtracking is performed to the most recently instantiated variables. As extensions of backtracking-type algorithms of LB, the backjumping method has been introduced to avoid repeated failures caused by a single class

of violation [10], and the backmarking method has been employed to avoid redundant discoveries of unmarked conflict values [12].

2. *Look ahead (LA).* Algorithms of LA also overcome the shortcomings of late discovery of inconsistency by preventing future conflicts before assigning values for variables. They perform the consistency checks for the not-yet-assigned variables against the consistent assignments. When a value is assigned to the current variable, any value in the domain of a future variable and/or its associations, which conflicts with this assignment is temporarily removed from the domain. LA can often further suppress the space on a subsolution basis.

3. *Consistency Technique (CTQ).* This technique removes inconsistent values from the domains of the variables based on the clues derived from the constraint specifications. The algorithms of this technique generate the complete value assignments by avoiding conflicts before they occur [34].

   Because we employ CTQ extensively in our research in solving this MRI problem to prune the search space by using constraints actively, it is discussed in a little more detail here. A CSP can be represented as a constraint graph network (CGN) where nodes correspond to variables and edges are labelled by constraints [32, 24]. If (and only if) for every value x in the current domain of V_i, there is some value y in the domain of V_j such that $V_i = x$ and $V_j = y$ is permitted by the constraint between V_i and V_j (Figure 7), the arc (V_i, V_j) has *arc consistency* (AC). The *Path consistency* (PC) technique of CGN further ensures that for every pair of values of two variables X and Y, there exists a value for each variable along some path between X and Y such that all constraints in the path are satisfied. The critical point of CTQ is that the values that would result in inconsistency have been removed from the variable value domains before the values are assigned. The most popular CTQ algorithms, such as AC-3, AC-4, PC-1, PC-2, and their extensions re-revise affected nodes/arcs/paths from previous node/arc/path revision, individual pairs of values, or values of multiple variables [32, 24, 2].

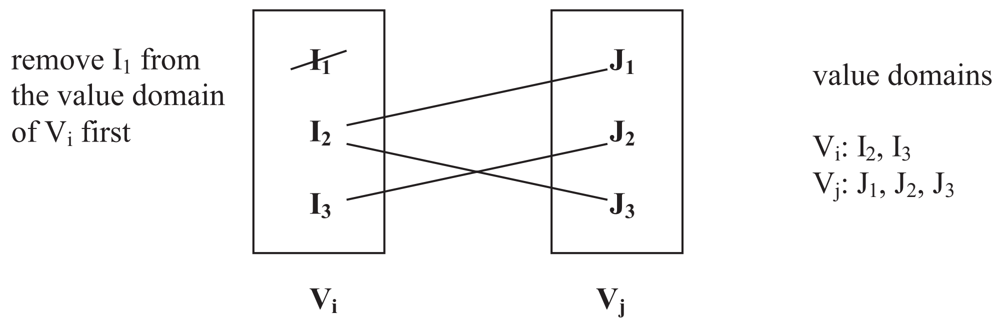

Figure 7: Consistency technique. Local inconsistencies have been removed from value domains before assigning consistent pairs of values.

4. *Complete and heuristic techniques.* These techniques include local search strategies of greedy algorithms equipped with various randomizing heuristics to search the space of complete labellings. Among these techniques, the hill climbing algorithm starts with a randomly generated labelling of variables and, at each step, it changes a value of some variables in such a way that the resulting labelling satisfies more constraints, until a global satisfaction is reached. To avoid cycling or getting trapped in local minimization, many heuristic methods have been developed [22, 11, 31].

In addition to the classic CP techniques, modern algorithms have been recently proposed, including the following:

1. *Binary constraint techniques.* One recent development of CP research has been centred on solving binary constraint satisfaction problems (BCSP) using evolutionary computation techniques. Nevertheless, this strategy has been found especially interesting in the field of constraint optimisation (CO) rather than CSP, to reduce the large number of constraint tests associated with a systematic search.

   BCSP is a more abstract research model that only allows constraints over a maximum of two variables. General CSPs may be dealt with using this model by first converting an arbitrary CSP into a binary form and then applying BCSP techniques [7, 14, 15, 41].

2. *Evolutionary algorithms.* These algorithms generally start with a random population of value assignments, and genetically evolve the population, based on fitness functions

that count the number of constraints violated. Breakout mechanisms are used to reduce the chance of getting stuck in local optima. Genetic algorithms generally do not know when to quit, and thus lack the ability to detect if a CSP in fact has no solution [15]. Some of the most popular evolutionary algorithms include microgenetic iterative descent and stepwise adaptation of weights, although some hybrid evolutionary and systematic search methods have started to become available [3, 36, 14].

Combinations of CTQ and systematic search techniques have been applied extensively in this research, together with the application of LB, LA, and heuristics in order to achieve completeness in constraint propagation and a good tradeoff between effectiveness and complexity.

4 Solving the MRI Problem with Constraint Programming

The MRI problem is a typical CSP. It can be properly formulated in terms of a set of variables, each having a value domain, and a set of constraints on the variables. The number of variables and their domains and the constraints are based on MRI physics and brain imaging and/or spectroscopy characteristics, and are specified by the user. The search space consists of all possible combinations of the values for each and every variable in the variable set.

4.1 The Enumeration Nature of the Original Problem

The computational complexity of complete enumeration of the search space is an exponential function of the number of variables. If it is possible for a variable to be assigned a set of values, instead of a single value, then the computation required for enumeration would be even more expensive.

In Figure 8 we illustrate the complexity of enumeration through an example. Assume that the MRI system had two variables a and b, with the domain of each being $\{a_1, a_2, a_3\}$ and

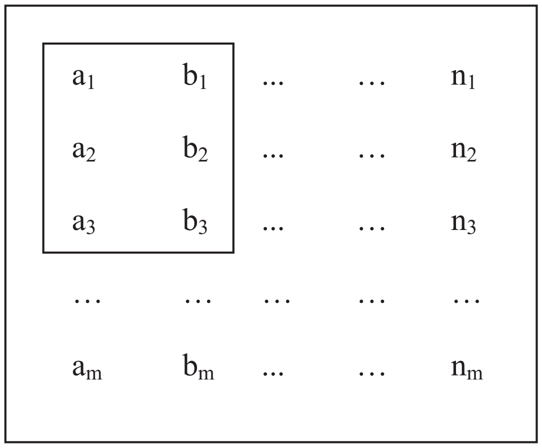

Figure 8: Enumeration nature of the problem

$\{b_1, b_2, b_3\}$ respectively. The complete set of value combinations includes a_1b_1, a_1b_2, a_1b_3, a_2b_1, a_2b_2, a_2b_3, a_3b_1, a_3b_2, a_3b_3, i.e. a total of $3^2 = 9$ value computations.

In general, if there are n variables, with m values each, the total number of value combinations is m^n, if each variable can be assigned a single value only. The problem becomes more complicated if variables can be assigned sets of values, instead of single values.

In the above 2-variable system, if multiple values are allowed for a, the possible values for a are: $\{\}$, a_1, a_2, a_3, a_1a_2, a_1a_3, a_2a_3, $a_1a_2a_3$. The number of value assignments for variable a is 2^3, including the empty set of values. When these are combined with the 3 value choices for b, the total number of value combinations would be $2^3 \times 3$. When multiple values are allowed for both variables a and b, the number of value combinations would be $2^{3\times 2}$. In general, the total number of value combinations for a system consisting of n variables with m values each thus would be $2^m \times m^{n-1}$ if one variable were allowed sets of values, and it would be as big as: $2^{m\times n}$, if all variables were allowed to have sets of values.

For a typical MRI parameter-setting CSP in the real world, it is realistic to assume there could be as many as 12 variables, each with 10 values. Thus the search space for this MRI/MRS CSP would be 10^{12}, if variables take on a single value, and $(2^{10})^{12} \simeq 10^{36}$ value combinations, if all variables are allowed to have sets of values.

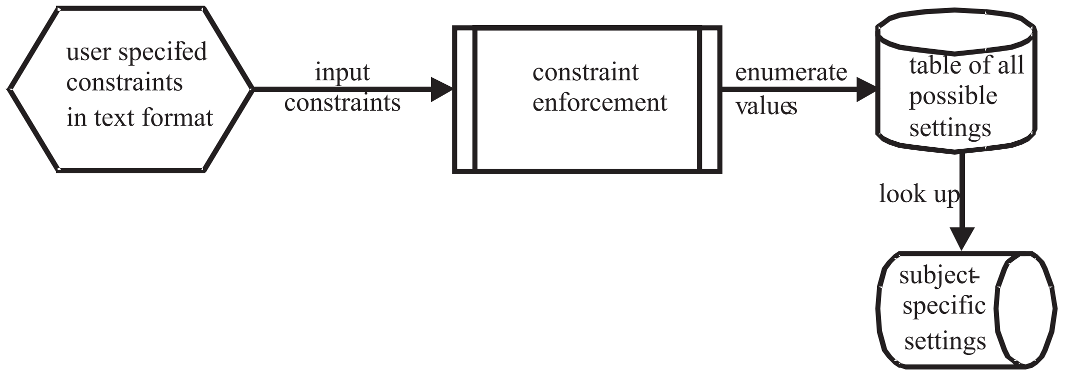
user specifed constraints in text format
input constraints
constraint enforcement
enumerate values
table of all possible settings
look up
subject-specific settings

Figure 9: A solution to the MRI parameter setting problem.

4.2 Design of a Constraint Satisfaction Program for MRI Parameter-Setting

In addition to solving the CSP, a program should be easy for the practitioner to use. Therefore, a particular MRI parameter-setting CSP should be specified in declarative form in a language that is easy for someone who is an expert in the operation of MRI equipment but not necessarily a computer programmer. The proposed design consists of a generic constraint-resolving engine, that reads an input file specifying the variables, their domains and the associated constraints, and enumerates all feasible value combinations (Figure 9). The input file contains textual constraint specifications prepared by the user according to the rules of a grammar. The grammar ensures the constraint specifications can be recognized and parsed by the program.

The constraint resolving engine consists of a case-independent part and a case-specific part. The former consists of hard coded parameters and formulas which are independent of the constraint set provided by users and do not differ among executions of the program. The latter is dependent upon the content of the constraint set and may vary from one execution of the program to another. We discuss them in the next subsection.

4.3 Case-Independent Parts of the Constraint Resolving Engine

Some of the software components represent characteristics of the MRI/MRS brain research applications. These aspects thus should be pre-defined and coded in the software. Usually, their values are determined by either physical principles related to MRI or task-specific features.

Parameters determined by physics of a MRI device may include the following:

1. The parameters used to calculate the strength of a magnetic field and derived resonance are based on the Larmor equation (as listed in Appendix C). They were coded in the solution as constants.

2. Time of a single scanning cycle (duration to complete a single sequence) is a constant tied to the MRI equipment, ranging from 2 to 10 seconds. The exact value of time per scan is a constant factor at any given value combination of TE and TR. The relationships generating correct time of any single scan were pre-specified in a machine dependent way.

3. Equations derived from mathematical expressions and operations were coded in certain program functions.

Parameters determined by biological features of the imaging task may include the following:

1. Value type for the various entities (i.e., whether a certain entity takes values taht are discrete numbers, strings or continuous numbers).

2. Constraint type. Based on the interpretation of the possible relations among the parameters in the real world application, several different constraint types were defined.

4.4 Case-Specific Parts of the Constraint-Resolving Engine

As a generic solution that has the potential to be applied to a wide range of tasks, many components of the software were not fixed but viewed as case-specific. These portions may change from one execution of the program to another, depending on what set of constraint specifications the program is executing. These portions are determined after the constraints are parsed. They include:

1. Values of all entities. While the identifiers and types of the entities are hard-coded in the program, the value domains of the entities may vary. For example, entity “measurement” may include any neuronal compound(s) of interest.
2. Values of all constraints. While the form of the constraints must be predefined in the software, the value domains of the constraints can be specified by the user; i.e., a constraint may involve any values of any entity.
3. Relations among entities. New relations among entities may be derived with any type of constraint. For example, some values of entity “measurement” may be involved in various types of constraints with entity “location”.
4. Constraint-resolving program. The program is generated automatically based on the input specified.

4.5 System Structure and Data Flow

The system consists of three major parts: the Parser, the Compiler and the Resolver. The Parser reads the constraint specifications and checks their syntactical correctness. The Compiler program interprets the specifications, and constructs the Resolver program dynamically. Interpretation involves the extraction of the variable identifiers and value domains as well as constraint identifiers and associated values. The Resolver iterates on the variable values and enforces the constraints to produce the settings that form the output.

4.6 Constraint Resolving Procedure

With all the required data structures and other information on entities and constraints prepared by the Parser, the Resolver resolves the constraints, i.e. it assigns values for all the entities consistent with the constraints. As shown in Figure 10, this is essentially a search procedure applying CT, heuristics, and systematic search CP techniques.

1. Constraint resolving begins by selecting an initial entity that has the lowest number of value choices that are independent of any other entity values.

2. The Resolver selects a value for the initial entity randomly from its value domain and stores this value as a member of the accumulated local solution.

3. From the constraints involving this value, it finds one other entity that has constraint relations with the value.

4. It determines all values of the entity satisfying the constraint and stores each of these values in the accumulated value group, together with the value of the previous local solution.

5. For each partial solution, it repeats steps 3 and 4 until a global solution is obtained, which assigns a value to each entity. The global solution is stored.

6. It repeats steps 2 to 6 to get more global solutions.

7. When all the values of the initial entity have been processed, the search space is complete, and all the global solutions have been found.

Constraints of a relatively simple form and those that bind fewer variables/values are enforced prior to those of more complicated types. Continuous numerical data are discretized by means of thresholds that are associated with distinguishable results.

5 Algorithms and Implementation

This section presents the details of the constraint resolution algorithms and data structures.

5.1 Definitions of Entities and Their Value Domains

Entities are of two different types, depending on whether they can assume a single value or multiple values from their domain.

Definition 5.1 (Single-Value Entity) *An entity that can only take single values from its domain is called a single-value entity.*

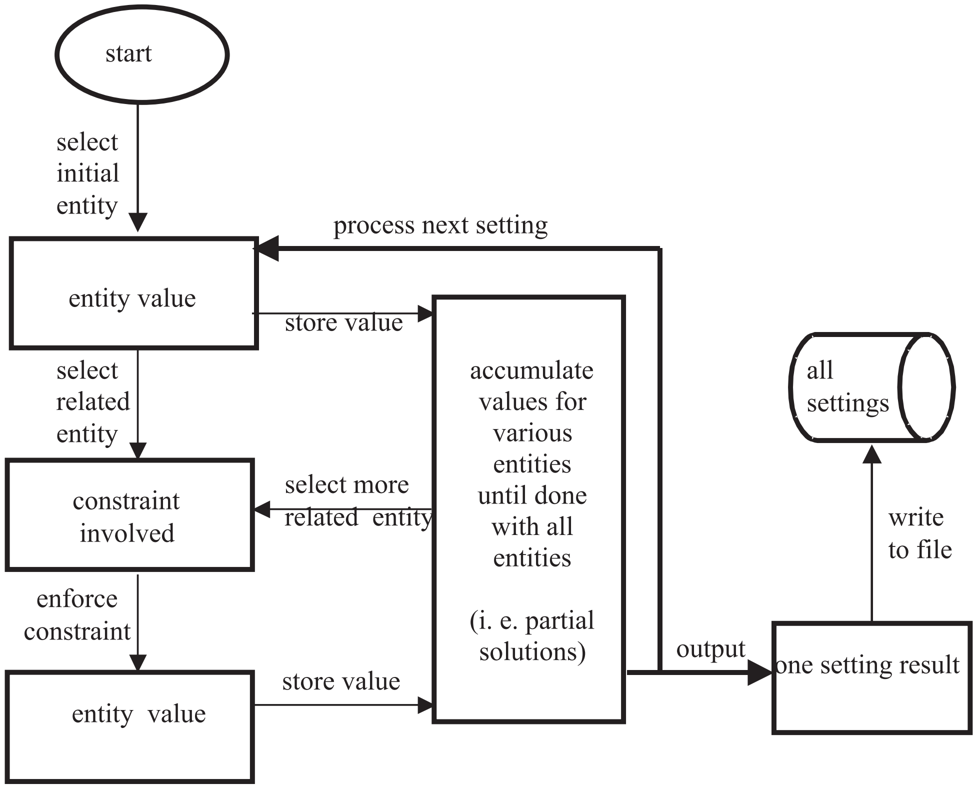

Figure 10: The value enumeration process. The entity value domains obtained after constraint enforcement are jointly enumerated to generate complete parameter settings. All complete parameter settings form the final output of the system and are written to disk.

An example of a single-value entity is "*avg*", the number of acquisitions to average (Table 1 and Appendix F).

Definition 5.2 (Multiple-Value Entity) *An entity that can take more than one value from its domain simultaneously is called a multiple-value entity.*

An example of a multiple-value entity is *mea*, the neural compounds of interest (Table 1 and Appendix F). In subsection 4.1 we demonstrated that the size of the search space is an exponential function of the sizes of the entity value domains. However, for the practical cases of the MRI parameter setting problem, it turns out that the size of the search space is not prohibitively large, after constraints have been applied to prune it.

Definition 5.3 (Value Combinations) *A value combination is an assignment of values to a set of entities, that is consistent with the constraints of the problem. Entities can be either single-value or multiple-value.*

Definition 5.4 (Partial Value Combinations) *A partial value combination is a value combination that involves only a subset of the entities. It cannot be used in MRI parameter setting yet, because it is a partial solution.*

Definition 5.5 (Complete Value Combinations) *A complete value combination is a value combination that involves all of the entities. It can be used in MRI parameter setting, because it is a complete solution. The output of our system is the full set of all possible complete value combinations.*

An example of a value combination for the set of entities (mea, loc, te, tr) is $((naa\ cre), (tem)), (30), (3000))$. In this example, *mea* is a multiple-value entity, while loc, te, tr are single-value entities. The assigned values of a single entity are represented as a list. If the list is a singleton (i.e. it has one element), then a single value has been assigned. The next subsection describes the constraints that must be satisfied by acceptable value assignments to individual multiple-value entities, and to acceptable value combinations.

Entities	*Variable Identifiers*	*Examples of values*
Measurement	mea	choline, naa, gaba
Region	loc	temporal cortex, orbital
Echo time (TE)	tev	10, 30, 68
Repetition time (TR)	trv	1500, 2000, 4000
Total time (TT)	time	1 .. 100, increment by 1
Voxel size	vox	1 .. 50, increment by 0.1
Sequence	sqz	PRESS, STEAM
Data point (DP)	dat	512, 1024, 2048
Gradient order	ord	y-x-z, z-y-x
Filter type	fil	double,no filter
Average (AVG)	avg	32, 128, 512
Single scan	sgl	2, 3, 10

Table 1: Entities handled by constraint parser

Type	*Class Name*
One	Incompatible value groups of a single entity
Two	Value incompatibility between two entities
Three	Single-single value dependency between two entities
Four	Multiple-single dependency between multiple entities
Five	Multiple-multiple dependency between multiple entities

Table 2: MRI/MRS constraint classification

5.2 Constraint Formulation

The constraints are imposed on 12 entities, that represent the 12 MRI/MRS parameters determining signal features. Table 1 lists the 12 entities, together with their identifiers and examples of typical values.

The types of constraints correspond to the types of relations among the entities. Constraints belong to the types shown in Table 2. Definitions and examples of these types of constraints are given next.

5.2.1 Constraint Type One

Type 1 constraint applies to multiple-value entities only. Its effect is to constrain the value sets that the entity can take. In order to formulate a type one constraint, the notion of value group is introduced.

Definition 5.6 (Value group) *A subset of values of a multiple-value entity is called a value group.*

Definition 5.7 (Partition of value domain) *A partition of a value domain of a multiple-value entity is a set of value groups that are mutually exclusive and their union is the value domain.*

A partition of a value domain is specified by the user in the constraint file.

Definition 5.8 (Constraint type one) *A constraint of type one applies to a value group G together with a set of other value groups $\{G_i\}$, all of the same multiple-value entity. Value sets from G can appear together with value sets from zero, one or more of the value groups. Value sets from one or more of the value groups cannot appear by themselves without the presence of one or more values from G.*

Example 5.1 *The multiple-value entity "measurement" (mea) has six value groups $G_1 - G_6$. When assigning values, values in groups G_3, G_4, or G_5 may be used together with other values of the same group only. Values in G_2 and G_6 may be assigned also with values of G_1. Values in G_1 may be assigned with values in G_1, G_2 and/or G_6. In this case, we have six constraints of type 1. The first five of them apply to value groups G_2, G_3, G_4, G_5, G_6 where the corresponding sets $\{G_i\}$ are empty sets. These five constraints give us the powersets of these value groups as acceptable value sets of the entity. The last constraint involves value group G_1 and the set of value groups $\{G_2, G_6\}$. Therefore, the acceptable value sets from this constraint is the union of the following powersets:*
$P(G_1)$,
$P(G_{12}) = P(G_1 \cup G_2) - (P(G_1) \cup P(G_2))$,
$P(G_{16}) = P(G_1 \cup G_6) - (P(G_1) \cup P(G_6))$,
$P(G_{126}) = P(G_1 \cup G_2 \cup G_6) - (P(G_1) \cup P(G_2) \cup P(G_6) \cup P(G_1 \cup G_2) \cup P(G_1 \cup G_6) \cup P(G_2 \cup G_6)$.
Hence, it would generate a conflict if values in incompatible groups (e. g. "choline" from G_1 and "gaba" from G_5) were to be assigned simultaneously.

5.2.2 Constraint Type Two

Type two constraints refer to value incompatibility between entities. This type of constraint indicates the situation when particular values of two different entities cannot be assigned simultaneously.

As an example, consider two multiple value entities, “measurement” and “region”. Certain values of the first entity (e.g. “myo-inositol”) are not compatible with certain values of the second entity (e.g. “orbital”).

Definition 5.9 (Constraint type two) *A constraint of type two applies to two different entities E_1 and E_2, and values v_{1i} and v_{2j} from their respective value domains. It states that v_{1i} and v_{2j}, or value groups containing them, cannot be assigned to E_1 and E_2 respectively in the same setting.*

5.2.3 Constraint Type Three

A constraint of type three refers to a dependency of single values between different entities. This type of constraint relates to the situation in which, if a certain value has been assigned to an entity, then particular value(s) must then be assigned to another entity.

An example of this constraint type is the following: When value “gaba” is used for entity “measurement”, the value for entity “TE” must be 68.

Definition 5.10 (Constraint type three) *A constraint of type three applies to two different single or multiple value entities E_1 and E_2, and values v_{1i} and v_{2j} from their respective value domains. It states that if v_{1i} has been assigned to E_1 alone or as a member of a value group if applicable, then v_{2j} must be assigned to E_2 alone or as a member of a value group if applicable.*

5.2.4 Constraint Type Four

Type four constraints are similar to type three constraints, but they involve more than two entities. For example, consider multiple value entity “measurement”, and single value entities “region”, and “TE”. The value choice of “TE” depends on the value combinations of the other two; e.g., if “measurement” takes value “choline” and “region” takes value “parietal”, then “TE” must take value either “10” or “275”.

Definition 5.11 (Constraint type four) *A constraint of type four applies to three or more single or multiple value entities E_1, E_2, E_3, ..., E_{n-1}, E_n and values v_1, v_2, ...,v_{n-1} and v_n from their respective value domains. It states that if v_1, v_2, ..., v_{n-1} have been been assigned to E_1, E_2,...,E_{n-1} respectively alone or as a member of a value group, then v_n must be assigned to E_n alone or as a member of a value group. Entities E_1, E_2,...,E_{n-1} are the independent entities of the constraint, while entity E_n is the dependent entity.*

5.2.5 Constraint Type Five

A type five constraint refers to multiple-multiple dependency. This constraint type occurs when value combinations of certain entities depend on value combinations of other entities. As an example, consider the following 6 different entities: “measurement”, “region”, “TT’, “TE”, “TR”, , and “AVG”. Choices of “TE”, “TR”, and “AVG” depend on what values have been assigned to the first three entities. If, for instance, “measurement” is “choline” and “region” is “parietal”, and “TT” is “70”, then “TE” must be “135” and “TR” must “1500”, and “AVG” must be “512”.

Definition 5.12 (Constraint type five) *A constraint of type five applies to more than three single or multiple value entities E_1, E_2, ...,E_k, E_{k+1},...,E_n and values v_1, v_2,...,v_k,v_{k+1},...,v_n from their respective value domains. It states that if v_1, v_2,...,v_k have been assigned to E_1, E_2, ...,E_k respectively, then v_{k+1},...,v_n must be assigned to E_{k+1},...,E_n respectively. Entities E_1, E_2, ...,E_k are the independent entities of the constraint, while entities E_{k+1},...,E_n are the dependent entities of the constraint.*

5.3 Constraint Representation

A simple context-free language [27] is defined to represent the constraints in a form that can be easily parsed. A grammar $G = (V, \Sigma, R, S)$ was defined for this purpose, where V represents the vocabulary, Σ represents the terminal symbols, $V - \Sigma$ represents the non-terminal symbols, R represents the production rules, and S is the starting state. The alphabet is as follows:

$\Sigma = \{$ `ent, grp, con,`
`type1, type2, type3, type4, type5, (, ), SPACE` $\} \cup$
$\{$set of integers $\} \cup$
$\{$user-defined symbols representing names and values of entities$\}$

$V - \Sigma = \{$ `NAME, NAMES, CONSTRAINTTYPE,`
`ENTITYLIST, ENTITYLISTPAIR, ENTITYVALUEPAIRS, ENTITYVALUEPAIRS1,`
`VALUES, VALUEGROUP, VALUEGROUPPAIR, VALUEGROUPPAIRS` $\}$

where `e` is the empty string, and "." is the string concatenation operation. The production rules are the following:

```
S -> ( (ent NAME) VALUES ) |
     ( (grp NAME) VALUEGROUPPAIRS) |
     ( (con CONSTRAINTTYPE) ENTITYLISTPAIR ENTITYVALUEPAIRS )

NAME -> an entity name
VALUES -> a list of entity values

VALUEGROUPPAIRS -> VALUEGROUPPAIR | VALUEGROUPPAIRS . VALUEGROUPPAIR
VALUEGROUPPAIR -> (VALUEGROUP VALUEGROUP)
VALUEGROUP -> a list of entity values (indexed)

CONSTRAINTTYPE -> type1 | type2 | type3 | type4 | type5
ENTITYLISTPAIR -> (ENTITYLIST ENTITYLIST)
ENTITYLIST -> (NAME NAMES)
NAMES -> NAME . NAMES | e
```

```
ENTITYVALUEPAIRS -> VALUEGROUPPAIRS | ENTITYVALUEPAIRS1
ENTITYVALUEPAIRS1 ->
        VALUEGROUPPAIRS | VALUEGROUPPAIRLIST . ENTITYVALUEPAIRS1 | e
```

In addition to the above context free grammar, there are semantic constraints that must be satisfied, that cannot be represented in the context-free grammar formalism. In the constraint specification, the number of entity name pairs should be the same as the number of value list pairs. Furthermore, the type of constraint should be consistent with the number of entities.

5.3.1 Generating Constraints

We now present some sample strings generated by the grammar defined above.

Example 5.2 (Strings generated by the grammar) *The strings generated by the grammar are of two kinds.*

1. Entity identifiers*, for example:*
   `(ent mea)`*, representing entity "measurement"*
   `(ent vox)`*, representing entity "voxel size".*

2. Constraint identifiers*, for example:*
   `((con type2) (((loc) (mea))` *((list of* `loc` *values) (list of* `mea` *values)))), representing a type 2 constraint relating variables loc and mea, i.e., incompatible values between entities "measurement" and "region".*
   `((con type4) (((tev)(mea loc))` *((list of* `tev` *values) ((list of* `mea` *values) (list of* `loc` *values))))) representing a type 4 constraint stating that combined values of entities* `mea` *(measurement) and* `loc` *(location) determine the values of entity* `tev` *(TE).*

5.3.2 Recognizing Constraints

For the program to extract the constraints from the constraint files, the format of the file should follow some rules. Table 3 shows a portion of a typical constraint file. A nested list notation for the constraints makes parsing of the constraints straightforward.

5.4 Generation of Partial Value Combinations by Enforcing Constraints

The goal of this system is to generate all feasible complete value combinations. This is done in two phases. Phase 1 enforces all the constraints to generate partial value combinations. Phase 2 uses the partial value combinations generated by phase 1 to construct complete value combinations. Constraints of type one are enforced first, then constraints of type two and type three, and finally constraints of type four and five.

5.5 Enforcement of Type One Constraints: Generating Multiple Value Sets of an Entity

For multiple-value entities all subsets of values that satisfy constraints of type one must be enumerated. This involves a powerset computation, which has exponential complexity. However, because the cardinality of the sets of values is relatively low in our problem, computing a solution is feasible.

Example 5.3 (Pruning of a Power set of a value set based on type 1 constraints) *Consider an entity E with a value domain equal to* $\{1,2,3,4,5,6\}$ *and potentially incompatible groups* $g_1 = \{1,2,3\}$, $g_2 = \{4,5\}$, $g_3 = \{6\}$*, then their power sets contain the following subsets:* $P(g_1) = \{\{\},\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$, $P(g_2) = \{\{\},\{4\},\{5\},\{4,5\}\}$, $P(g_3) = \{\{\},\{6\}\}$
If there exists a type one constraint prohibiting the mixing of values from g_1 *and* g_2 *as well as from* g_2 *and* g_3*, then the actual value domain of entity E would consist of the union of the powersets* $P(g_1), P(g_2), P(g_3)$ *with the following mixed value sets permitted by the type one constraint:* $\{\{1,6\},\{2,6\},\{3,6\},\{1,2,6\},\{1,3,6\},\{2,3,6\},\{1,2,3,6\}\}$.

The algorithm generates the power set of a set of n values. It parses the entity values together with the associated grouping information. It prunes based on type one constraints by producing and storing all the value combinations of an entity that may be used in a single setting.

```
;; general form
;; (title values)
;; title: (type name)
;; values: a list of values if type is ent (entity)
;;         a list of pairs of value groups if type is grp (grouping information)
;;         if type is con (constraint),
;;            first value is a pair of entity lists
;;            rest values are pairs of entity value groups
;;

;; -----  entity value specification
((ent mea)   ;; entity name
 (naa cho cre glx myo lip lac wat glu gab)) ;; list of entity values
((ent loc)
 (pre orb tem par occ bra sub cer))
((ent tev)
 (10 20  30  68  135 275))

....

;; ------ value group specification for multiple value entities
((grp mea)          ;; group values for entity
 (((naa) (cho cre))       ;; values on left can be grouped with values on the right
  ((glx) (wat glu))))
....
;; ------ constraint specification
((con type2)        ;; constraint and its type
 ((loc) (mea))      ;; entities: left list: constrained, right list: constraining
                    ;; corresponding value group pairs for entities above
 ((rob) (glx myo lip wat glu gab)))


((con type4)         ;; constraint and its type
 ((tev) (mea loc))  ;; entities: left list: constrained, right list: constraining
                    ;; corresponding value group pairs for entities above
 (((10)) ((glx myo lip wat glu) (pre orb par)))
 (((10)) ((naa cho cre) (pre par occ)))
 (((20)) ((glx myo lip wat glu) (pre orb tem par occ bra sub cer)))
 (((20)) ((naa cho cre) (pre par occ sub))))
...
```

Table 3: A sample textual constraint specification. Entities, value groups and constraints are represented as nested lists. Comments are preceded by ";;".

5.6 Enforcement of Type Two and Type Three Constraints: Defining Value Intervals that Can Be Treated as Single Values

Type two and type three constraints help generate consistent value combinations involving two entities. These value combinations are enumerated and stored.

The input of the algorithm includes the following:

a) An entity E_1 with a finely discretized numerical value domain.

b) An entity E_2 with a discrete value domain.

c) A set of type two and/or type three constraints, relating E_1 with E_2.

The output of the algorithm is a set of partial value combinations that involve intervals defined over the value domain of entity E_1 and values of E_2. Because partial value combinations are defined over intervals as opposed to over single values, they are represented much more compactly. The following is an example:

Example 5.4 (Intervals in partial value combinations) *Entities "time" and "avg" are associated via a set of constraints involving time for a single scan, so that inconsistently large values would not be assigned to "avg" if values of "time" are not sufficiently large. Suppose each single scan takes 10 seconds and value of "time" is 25 minutes, then the highest consistent value for "avg" is 128 because 25*60/10=150, which is smaller than the next acceptable value 256. Applying this algorithm, we are able to calculate and store the partial value combinations of "time" interval [22,42], together with "avg"=128; "time" interval [43,86] together with "avg"=516, etc.*

This algorithm computes discretizations of continuous entities. The algorithm then stores the discretized values in data structures, together with the associated values of the related entities. The algorithm is shown in Fig. 11.

5.7 Enforcement of Type Four and Type Five Constraints: Acceptable Value Combinations of Multiple Entities

Hash tables are used to enforce type four and type five constraints. Such constraints involve the relation of values or value sets of certain entities $E_1, E_2, \dots E_n$ with the acceptable value

```
let $d$ be the domain of $E_1$        // Input, the continuous entity value domain
let $\{p_1, p_2, \ldots p_n\}$ be the domain of $E_2$
// Output, is a set $C$ of partial value combinations involving
// intervals over $d$ associated with values $p_i$
// $C$ has $n$ elements, one for each $p_i$, $i = 1 \ldots n$
For each $p_i$        // Process all values in domain $p$ sequentially
        enforce all constraints on $p_i$ based on values from $d$;
        identify the lower and upper limits of $d$, $d_{lo}$ and $d_{hi}$
        construct the partial value combination $((interval\ d_{lo}\ d_{hi})(p_i)$
        add it to the set $C$
return $C$
```

Figure 11: Algorithm for Obtaining Threshold Values

Key (mea loc te tr)	*value set (avg)*
((*naa cre*) (*tem*) (30) (3000))	(1024)
((*gab*) (*bra*) (68) (4000))	(512)
...	

Table 4: Data structure for hashed value combinations. Value combinations for entities mea, loc, te, tr determine value for entity *avg*. Note that some entities may have multiple values, for example *mea*, which has values (*naa cre*). This is a set of values, but, as a hash key, the order of the values is important.

of another single value entity E_{n+1}. For each combination of entities, a separate hash table is used. A hash key is, in general, a nested list of values (i.e. a list of lists of values) for each of the entities $E_1, E_2, \ldots E_n$. The hash value is the list of acceptable values of entity E_{n+1}. Sample hash keys and values from the hash table corresponding to entities (*mea loc te tr avg*) are shown in table 4. If a key has ((*naa cre*) (*tem*) (30) (3000)) for (*mea loc te tr*), the value set of (1024) for entity *avg* is returned.

Implementation with hash tables allows us to compute and store acceptable value sets based on these constraints once and then look them up multiple times as needed in $O(1)$ time.

The algorithm that computes the entries of the hash table is shown in Fig. 12.

```
Input:  The value combinations of multiple entities $E_1, E_2, \ldots E_n$
        Single-value entity $E_{n+1}$.
        All constraints of type 4 or type 5 involving $E_1, E_2, \ldots E_n, E_{n+1}$
Output:Hash table
define hash table $H$
for each value combination $V$ of $E_1, E_2, \ldots E_n$
        resolve the constraints involving $V$ to get acceptable value set $v$ for $E_{n+1}$
        enter $V$ and $v$ into the hash table as a key/value pair
return $H$;
```

Figure 12: Algorithm for building the hash table implementing constraint types four and five.

6 Evaluation

The system was implemented and tested on a variety of constraint files.

Experiment 1: Simple constraint files that consist of subsets of specific types of constraint; e.g., the constraint file may contain extensive numbers of constraints of one or two types only.

Experiment 2: Complete constraint files that consist of all of the system and task-specific constraints for a specific real world MRI system and neuroscience imaging protocol.

Experiment 3: Constraints included in Experiment 2 plus additional unseen cases that were added for the purpose of testing the limits of the software system. The constraint files have been constructed following the rules of the grammar described previously.

6.1 Correctness

System correctness was evaluated by comparing the parameter setting results obtained from executing the program against the known correct sample settings provided by the manufacturer [19] and those obtained from MRI experts [28]. The comparison results in 10 identical measurement instances demonstrated that the solution always produced correct parameter settings.

6.2 Computational Complexity

The hash algorithm, and the algorithm to obtain the threshold values of numerical variables, were quite fast with a running cost of $O(1)$, or $O(n)$, where n represents the size of the value domain of the entities. The expensive algorithms, that determine the overall computational complexity of the solution, are the following.

a) Computation of the powerset to obtain the actual value domains for the multiple value entities in type one constraint enforcement. The complexity of the powerset is 2^n, where n is the size of the value domain of an entity. However, this complexity remains limited in practice for two reasons. First, in practical MRI applications, the total number of values for an entity is typically not more than 10. Second, the value domain of a multiple value entity is partitioned into subgroups, where values across subgroups are mutually exlusive. Thus only the powerset of values belonging to the same subgroup needs to be obtained, rarely exceed 5 in actual MRI/MRS brain research applications. Thus, the complexity in practice is a constant bounded by 2^5.

b) The algorithm for enforcing type four and type five constraints has a complexity of $O(n^k)$ where k is the number of independent entities of the constraint and n is the size of the value domains of the entities. Since many problems solved by Constraint Programming belong to the area of NP hard problems, the identification of restrictions that make the problem feasible to solve has always been very important in practice [2]. In our particular problem, brain MRI imaging/spectroscopy, k rarely exceeds 5, and n rarely exceeds 12.

c) The algorithm for enumerating complete value combinations (solutions to the parameter setting problem) based on partial value combinations generated by enforcing all the constraints has complexity $O(n^c)$, where n represents the size of the set of partial value combinations computed after enforcing all the constraints and c is the number of entity sets for which partial value combinations have been computed. In practice, n is in the range 20 to 30, and c is less than or equal to 4.

Execution time data were acquired as follows. On a SUN Microsystems 8-processor machine with 3GB of memory running a UNIX operating system (Solaris 7.0) (the time-shared server of the Faculty of Computer Science at Dalhousie University), the running time for one

Constraint set	Size of constraint set	No of multiple value entities	Maximum No of independent entities in type 4/5 constraints
1	< 30	0	1
2	< 30	1	2
3	< 30	1	3
4	30 - 60	0	1
5	30 - 60	1	2
6	30 - 60	1	3
7	> 60	0	1
8	> 60	1	2
9	> 60	1	3

Table 5: Constraint sets used in the experiments

complete program execution has never exceeded 20 *min* in all cases. Moreover, a significant percentage of this time was spent writing the results into secondary storage.

In order to experimentally estimate the computational complexity of the generic constraint-resolving engine, experiments were conducted to assess how the running time of the software changes as a function of input and how the search space is reduced by each filtering step, as constraints are applied.

Because the degree of complexity of the input of the program is related to both the number and the nature of the constraints that it contains, the complexity of input data was first classified using the criteria defined in Table 5. Nine constraint sets of various degrees of complexity were constructed and then the software was tested using these constraint sets. Information on the experimental results, including the reduction of search space by each filtering step and the time taken to accomplish the tasks, is provided in Table 6.

The results show that during the constraint-enforcement phase, the size of search space is the predominant factor in determining performance, although heuristics from constraints have served to prune the search space to a certain extent. During the value enumeration phase which enumerates complete solutions from partial value combinations obtained by enforcing the constraints, the performance is more closely related to the pruning power of the constraints involved.

As suggested by the literature, a general shortcoming of CP in solving real-world problem is

Set of Constraints	Original size of search space	Constraint enforcement		value enumeration		Disk write time (s)
		Time(s)	Size *	Time(s)	Size *	
1	6.2×10^8	26	5.0×10^8	3	25050	45
2	6.6×10^{10}	30	2.1×10^9	5	14204	29
3	6.7×10^{10}	32	1.9×10^9	5	7258	15
4	2.8×10^9	31	2.2×10^9	8	101640	180
5	2.8×10^{11}	36	1.77×10^{10}	36	71023	130
6	2.9×10^{11}	38	1.68×10^{10}	35	58564	121
7	4.2×10^{10}	40	2.8×10^{10}	56	142676	272
8	4.1×10^{12}	46	3.8×10^{10}	71	158197	290
9	4.4×10^{12}	48	2.3×10^{10}	50	120691	230

Table 6: Complexity results. The original size of the search space, as well as the size of the reduced search spaces after enforcement of the constraints, and enumeration of the complete value combinations, and the associated computation times in seconds, are shown.

that the efficiency of software is unpredictable. It is possible for small changes to the input data to lead to a significant change in performance [2]. The evaluation method applied here is by necessity ad hoc, since standard methods for assessing computational complexity of CP solutions have yet to be developed [2].

7 Discussion

The research reported here was a pioneering attempt to solve a complex problem in the selection of appropriate parameters for signal acquisition in MRI/MRS research and clinical applications by using algorithms and strategies derived from the field of Constraint Programming. We produced a generic program with an automatically constructed constraint resolving engine. This engine prunes the search space and generates all valid parameter settings as directed by the constraint specifications. It allows users to specify/modify constraints to deal with measurement protocols of interest and at the same time frees them from burdensome computations. Implementation of the program should make it possible to develop MRI/MRS applications more quickly and easily.

With its properties of generality and flexibility, the software is potentially applicable to a variety of MR tasks. It could be especially useful for designing applications for functional MRI, since there are no commercial software packages available for this purpose and no agreed-on, standardized settings for clinical use. Furthermore, the CP approach implemented in this program may be applicable to the solution of other classes of constraint satisfaction problems.

The constraint-resolving program is automatically generated by the generic constraint-resolving engine through its interpretation of the constraint specifications. Thus, the performance of the program and the quality of its output are heavily dependent on the quality of the constraint file. Preparing appropriate constraint specifications requires extensive technical expertise in the field of MRI/MRS brain imaging. It also requires a clear understanding of and strict adherence to the rules of the grammar and of file format specifications. Therefore, the use of the program may be limited to those with appropriate expertise in MRI/MRS technology and proper training in generating constraint files for the program.

It is likely that some training would be necessary initially, even for MRI experts, to be able to develop constraint files in the grammar formalism. For an MRI expert, writing the constraint file should, however, be relatively easy to manage due to the simplicity of the grammar and the limited number of terminal symbols defined in the language. Expert MRI users should quickly become familiar with the small set of grammar rules. Once this is accomplished, a considerable amount of time and effort would be saved because users would then not need to expend the time required to resolve the complex interrelations among parameters on their own, nor to implement a computer program to do so.

A better user interface may prove helpful in assisting the user to efficiently perform the task of preparing constraint files. A user interface could provide guidance in constraint representation and specification and ensure that the constraints were formalized and expressed correctly in terms of both syntax and semantics.

One limitation of the program is that it does not handle the constraint optimization problem associated with parameter setting; i.e., it generates an output listing all feasible settings, but does not prioritize them in terms their optimality. Optimization of MR settings is a difficult task and resolving this issue will require further input to and modification of the program. Additional inputs may include heuristics and declarations of priorities for the entities and values, derived from expert knowledge. Another promising approach to tackle the prioritization issue imay be by using genetic CP with extensive sets of training data. To implement this approach, the large number of parameter setting results generated by the present CP programming solution may be useful as training data.

A practical problem with assessing this program is that extensive performance testing in real-world settings will be difficult, in part because of minor safety concerns (although MR scanning is an inherently safe procedure), and in part because individual scans are very expensive in terms of personnel and time. It would, therefore, be both time-consuming and expensive to assess the appropriateness of a full range of output settings derived by the program. Thus, most of the evaluation criteria we have used to assess output quality have been indirect. We have compared output settings to standard settings made available by MR device manufacturers and to settings developed by local MR experts, based on their individual experience. Real-world evaluations of the full range of settings generated by the software may not be possible, and future evaluations will probably involve the development of simulation approaches.

Acknowledgements We sincerely thank Mr. F. MacMaster from Anatomy/Neurobiology, Psychiatry and the National Research Council of Canada, and Mr. L. Gates from Radiology for their technical cooperation in preparing data and in providing experimental samples and testing results. In particular, we thank Dr. R. Spiteri, Dr. A. Farrag, Dr. A. Sedgwick, Dr. P. Keast, and Dr. M. McAllister for discussions on algorithms applied in this research.

References

[1] R. Bartak. Constraint programming: A survey of solving technology. In *AIRONews 4*, pages 7–11, 1999.

[2] R. Bartak. Theory and practice of constraint propagation. In *Proceedings of CPDC2001 Workshop*, pages 7–14, Gliwice, 2001.

[3] J. Bowen and G. Dozier. Solving constraint satisfaction problems using a genetic/systematic search hybrid that realises when to quit. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 614–619, 1995.

[4] M. S. Cohen and S. Y. Bookheimer. Localization of brain function using magnetic resonance imaging. *Techniques in Neuroscience*, 17:268–277, 1994.

[5] J. P. Cousins. Clinical mr spectroscopy: Fundamentals, current applications, and future potentials. *Am. J. Radiology*, 164, 1995.

[6] I. J. Cox. Development and applications of in vivo clinical magnetic resonance spectroscopy. *Prog. Biophys. Mol. Biol.*, 65, 1996.

[7] G. Dozier, J. Bowen, and A. Homaifar. Solving constraint satisfaction problems using hybrid evolutionary search. *IEEE Transactions on Evolutionary Computation*, 2:23–33, 1998.

[8] E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 1, 1992.

[9] D. Gadian, T. E. Bates, S. R. Williams, J. D. Bell, S. J. Austin, and A. Connelly. Approaches to editing, assignment and interpretation of proton spectra. *NMR Biomed.*, 4:85–89, 1991.

[10] H. Gallaire. Logic programming: Further developments. In *Proceedings of IEEE Symposium on Logic Programming*, Boston, 1985.

[11] F. Glover and M. Laguna. *Modern Heuristics for Combinatorial Problems.* Blackwell Scientific Publishing, Oxford, 1993.

[12] R. M. Haralick and G. L. Elliot. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–314, 1980.

[13] K.H. Hausser and H.R. Kalbitzer. *NMR in Medicine and Biology.* Springer-Verlag, 1991.

[14] J. I. Hemert. Constraint satisfaction problems and evolutionary computation: A reality check. In *Proceedings on the Twelfth Belgium-Nederlands Conference on Artificial Intelligence*, 2000.

[15] J.I. Hemert. *Comparing classical methods for solving binary constraint satisfaction problems with state of the art evolutionary computation*, volume 2279 of *LNCS*, pages 81– 90. Springer-Verlag, New York, 2002.

[16] J. Hirsch, D. R-Moreno, and K.H.S. Kim. Interconnected large-scale system for the fundamental cognitive tasks revealed by functional mri. *Journal of Cognitive Neuroscience*, 13:389–405, 2000.

[17] A. L. Horowitz. *MRI Physics for Radiologists: A Visual Approach.* Springer- Verlag, New York, 3 edition, 1995.

[18] J. Hugg, A. A. Maudsley, M. W. Weiner, and G. B. Matson. Comparison of k-space sampling schemes for multidimensional mr spectroscopic imaging. *Magn. Reson. Imag.*, 6:453–459, 1996.

[19] Siemens Inc. Magnetom operating instructions: Sequence generator. Technical report, Siemens Medical Systems, 1992.

[20] J. Jaffar and M. J. Naher. Constraint logic programming: A survey. *J. Logic Programming*, 19:503–581, 1996.

[21] R. J. Clifford Jr., C. C. Lee, and S. J. Riederer. Functional brain imaging with a standard 1.5-t magnetic resonance imaging system. *Acad. Radiol.*, 2:916–923, 1995.

[22] R. Korf. *Artificial intelligence search algorithms.* CRC Press, Boca Raton, 1998.

[23] R. M. Kroeker. Numaris/2 magnetom sp user's library: User's manual. Technical report, Siemens AG UB Med, Erlangen, 1992.

[24] V. Kumar. Algorithms for constraint satisfaction problems: A survey. *AI Magazine*, 13:32–44, 1992.

[25] P.G. Lauterbur. Image formation by induced local interactions: Examples employing nuclear magnetic resonance. *Nature*, 242:190–191, 1973.

[26] B. D. Le. Functional magnetic resonance imaging of the brain. *Annals of Internal Medicine*, 122:296–303, 1995.

[27] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall, New Jersey, 2 edition, 1998.

[28] Frank MacMaster and Dr. Larry Gates. Personal communication.

[29] K. Marriot and P. Stuckey. *Programming with Constraints: An Introduction.* The MIT Press, Cambridge, 1998.

[30] A. A. Maudsley and M. W. Weiner. The future of magnetic resonance spectroscopy and spectroscopy imaging. In *Proceedings of JMRM Internet Conference*, University of California, Los Angeles, 1997.

[31] S. Minton, M. D. Johnston, and P. Laird P. Minimising conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:11–206, 1992.

[32] U. Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[33] H. Murck, T. Struttmann, M. Czisch, T. Wetter, A. Steiger, and D. Auer. Increase in amino acids in the pons after sleep deprivation: A pilot study using proton magnetic resonance spectroscopy. *Neuropsychobiology*, 45:120–123, 2002.

[34] B. Nadel. *Tree Search and Arc Consistency in Constraint Satisfaction Algorithms.* Springer-Verlag, 1988.

[35] J. H. Newhouse and J. I. Wiener. *Understanding MRI.* Little, Brown and Company, 1991.

[36] B. O'Sullivan and J. Bowen. A constraint-based approach to supporting conceptual design. In *Proceedings of the International Conference on Artificial Intelligence in Design*, pages 291–308, 1998.

[37] O. A. Rothman, C. Petroff, K. L. Behar, and R. H. Mattson. Localized 1h nmr measurements of gamma-aminobutyric acid in human brain in vivo. *Proc. Natl. Acad. Sci.*, 90:5662–5666, 1993.

[38] G.V. Simpson, M. E. Pflieger, J. J. Foxe, S. P. Ahlfors, H. G. Vaughan Jr., J. Hrabe, R. J. Ilmoniemi, and G. Lantos. Dynamic neuroimaging of brain function. *J. Clin. Neurophysiol.*, 12:406–429, 1996.

[39] D. D. Stark and W.G. Bradley Jr. *Magnetic Resonance Imaging.* Mosby Year Book, St. Louis, 1992.

[40] S. Thomas and R. Dixon. *NMR in Medicine: The Instrumentation and Clinical Applications.* AAPS, 1986.

[41] E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, London, 1993.

[42] M. Zweben and M.S. Fox. *Intelligent Scheduling.* Morgan Kaufmann Publishers, San Francisco, 1994.

A Milestones in the History of MRI Technology

1946	NMR phenomenon - Bloch & Purcell
1952	Nobel Prize - Bloch & Purcell
1950	NMR developed as analytical tool in physics and chemistry
1960	Start Computerized Tomography
1970	Computerized Tomography development
1972	Matured Computerized Tomography
1973	Backprojection MRI - Lauterbur
1975	Fourier Imaging - Ernst
1980	MRI demonstrated - Edelstein
1986	Gradient Echo Imaging
1987	NMR Microscope
1988	Angiography - Dumoulin
1989	Echo-Planar Imaging
1991	Nobel Prize - Ernst
1994	Initial techniques of fMRI

B The Electromagnetic Spectrum

Name	Wave Range
Gamma-ray	<1 $\AA$
X-ray	1-10 $\AA$
Soft X-ray	10-100 $\AA$
Vacuum UV	100 $\AA$ - 200 nm
UV	200-400 nm
Visible light	400-800 nm
Near IR	800 nm - 2.5 μm
Mid IR	2.5-25 μm
Far IR	25-400 μm
Microwaves	400 μm - 25 mm
Radio Waves	>25 mm

C The Larmor Relationship

The relationship between the strength of the main magnetic field and the precessional frequency of the magnetic vectors is known as Larmor relationship. Mathematically, it is stated as follows:

$f = \frac{\gamma B}{2\pi}$

where f is the frequency of precession, B is the strength of the net magnetic field, and γ is the gyro-magnetic ratio which is a constant for a certain nucleus.

D Fourier Transformation

Fourier transformation is a mathematical technique for converting time domain data to frequency domain data, and vice versa. Mathematically, it is stated as follows:

$F(v) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi vt}dt$

or

$f(t) = \int_{-\infty}^{\infty} 2\pi F(v)e^{i2\pi vt}dv$

where t is time and v is frequency.

E Important Neural Compounds Observed by ^1H MRS

Compound	Abbreviation	Biological Role
N-Acetyl aspartate	NAA	Neuronal marker.
Creatine/phosphocreatine	Cr	Source of phosphate to convert ADP to ATP.
Choline	Cho	Element of all derivatives of choline (acetylcholine neurotransmitter, membrane phosphatidylcholine, betaine etc). Reflection of neurodegenerative state.
myo-Inositol	mI	Glial marker.
Glutamate	Glu	Neurotransmitter (excitatory).
Glutamine	Gln	Product of Glu and ammonia reaction. Regulator of Glu and detoxification. Spectrum signals from Gln and Glu are indistinguishable and noted Glx.
Glucose	Glc	Energy source.
Lactate	Lac	Final product of anaerobic glycolysis.
g-Amino butyric acid	GABA	Neurotransmitter (inhibitory).

F Typical MRI/MRS Parameters

Parameter	Low Range	High Range	Explanation
TR	$\sim$100 ms	<20,000 ms	repetition time
TE	$\sim$10 ms	<300 ms	echo time
SE	1	32	spin echo, sequence timing-dependent
TI	$\sim$30 ms	<2,000ms	inversion time
FAA	$>0^o$	$<360^o$	flip angle alpha, increment by $\tilde{3}0^o$
Slicing	>0	<<	number of slices
VS	1mm	50mm	voxel size, incremented by 1mm
AVG	>1	<<	Selectable, number of acquisitions to average
MM			measurement size
TT			total measurement time
PS	>>	<2 s	preparation time
DP	512	<<	sequence timing-dependent, powers of 2
Filter			filter for noise level, select or default
Region			brain tissue of interest
mea			Measurements, i.e.neural compounds of interest
Priority			chemical-dependent, selectable
ETT	feedback	feedback	expected total time, setting-dependent
RF			radio pulse order
Order			gradient order, select or default
WS			water suppression, select or default
WR			water reference, boolean

G How Changes in Parameters affect MRI/MRS Characteristics

Changes in →	SNR	Resolution	TT	Range	Slice No
By increasing ↓					
Magnet	+	NC	NC	NC	NC
RF Width	-	NC	NC	NC	+
TE	-	NC	NC	NC	-
TR	+	NC	+	NC	+
Matrix Size	-	+	+	NC	NC
Thickness	+	-	NC	+	NC
Gap	+	-	NC	+	NC
FOV	+	-	NC	NC	NC
NEX	+	NC	+	NC	NC

H Sample Results from the Constraint Resolver

For a text file containing a set of MRI/MRS constraint specifications, the constraint-resolver generated all valid settings for all possible scanning tasks. Here, only those that matched the requirements as input to the simple resolver were listed. The program also provides the minimum total time for each measurement choice.

```
Measurement Region Echo Repeat Voxel  Size Seq Ord DataPt Filt Low High Inc Avg
......
cho  lac    bra    275  1500   1~50 at 0.5 pre acs 1024   nof  60  60   1   512
cho  lac    bra    275  1500   1~50 at 0.5 pre acs 1024   nof  34  60   1   256
cho  lac    bra    275  2000   1~50 at 0.5 pre acs 1024   nof  60  60   1   512
cho  lac    bra    275  2000   1~50 at 0.5 pre acs 1024   nof  39  60   1   256
cho  lac    bra    275  2500   1~50 at 0.5 pre acs 1024   nof  60  60   1   512
cho  lac    bra    275  2500   1~50 at 0.5 pre acs 1024   nof  34  60   1   256
cho  lac    bra    275  3000   1~50 at 0.5 pre acs 1024   nof  60  60   1   512
cho  lac    bra    275  3000   1~50 at 0.5 pre acs 1024   nof  43  60   1   256
......
```