Genetic Map Construction with Constraints

Dominic A. Clark, Christopher J. Rawlings and Sylvie Doursenot

Biomedical Informatics Unit
Imperial Cancer Research Fund
P.O. Box 123, Lincoln's Inn Fields
London, WC2A 3PX, UK.
{dac,cjr}@biu.icnet.uk

Abstract

A pilot program, CME, is described for generating a physical genetic map from hybridization fingerprinting data. CME is implemented in the parallel constraint logic programming language ElipSys. The features of constraint logic programming are used to enable the integration of pre-existing mapping information (partial probe orders from cytogenetic maps and local physical maps) into the global map generation process, while parallelism enables the search space to be traversed more efficiently. CME was tested using data from chromosome 2 of *Schizosaccharomyces pombe* and was found able to generate maps as well as (and sometimes better than) a more traditional method. This paper illustrates the practical benefits of using a symbolic logic programming language and shows that the features of constraint handling and parallel execution bring the development of practical systems based on AI programming technologies nearer to being a reality.

Introduction

Genetic and physical maps are constructed to provide a long range view of the genome by locating landmarks (genes and other genetic markers) on particular chromosomes. Some of these are positioned as a result of genetic linkage analysis (e.g. inheritance patterns of genes), others by physical analysis of the genome (e.g. hybridization fingerprinting, Lehrach et al.1990). The landmarks provide a high level index, by which the positions of DNA clones[1] can be located and then sequenced. Assuming the landmark maps are correct, it is theoretically possible to sequence the entire genome by sequencing the minimal subset of the clones which in combination span the whole genome and then ordering them according to the higher level maps. More practically, physical maps can be used as an index to libraries of cloned DNA used to focus in on areas of the genome of particular interest (e.g. to locate the positions of genes) which are known to contain those markers.

Hybridization Fingerprinting

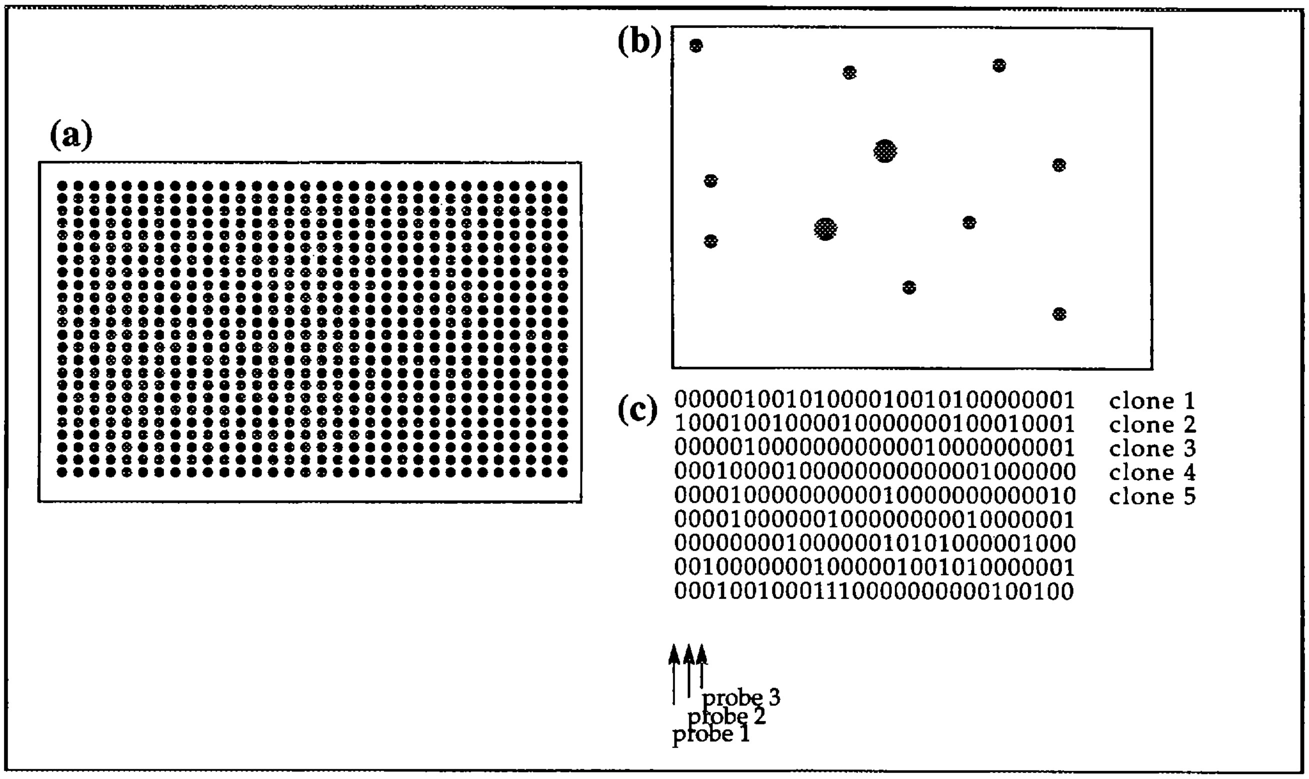
The establishment of ordered clone libraries relies on the identification of clones from overlapping segments of the genome. Groups of contiguous clones with established overlaps are called a contig. A well established technique for detecting overlaps between DNA clones is called hybridization fingerprinting (Lehrach et al. 1990). After fragmentation of the genome, the fragments are cloned into a host so that they may be easily stored, replicated and analysed. Samples of each clone are spotted in a grid (by robot) onto large nylon filter membranes and then they are hybridized with radioactive markers (probes). Assuming that the clones are randomly distributed over the genome (their locations unknown), similarities in the hybridization patterns of different clones with sets of probes imply that they derive from overlapping segments of the genome. The pattern of hybridization revealed by exposed spots observed on autoradiographs of the filter grids is then recorded (Figure 1).

The computational problem is to take a digital representation of the hybridization pattern from all the filters covering the chromosome and determine both the most complete order of probes containing the greatest number of contiguous regions and the order of clones along chromosome as defined by this probe ordering. In the absence of experimental noise this would be a trivial problem and could be solved using a number of simple algorithms (e.g. the greedy algorithm whereby probes/clones were ordered by firstly selecting the most similar pair and repeatedly adding the most similar). However, noise from the following four sources makes the problem non-trivial:

- random noise (coming from the experimental methods or from threshold effects in image interpretation),
- co-ligated clones (clones which are a combination of two or more distinct parts of the genome),
- clones with internal deletions and
- non single-copy probes (DNA sequences that occur at more than one place in the genome).

The problem of probe ordering is essentially combinatorial in nature and computationally very intensive for large scale mapping experiments. The main approaches developed so far are either entirely statistical (Michiels 1987, Branscomb 1990, Torney et al. 1990, Mott et al. 1993) and treat noise as an artefact to be minimized, or are based on heuristic considerations (Mott et al. 1993) where the aim is to eliminate noisy data with a concomitant loss of

1. A clone is a contiguous short subsection of a genome inserted into the genome of host species in order to be replicated, stored and analysed.The type of host used determines the insert size of the clone (lambda, 15 kbp (kilo base pair); cosmid (*E. coli*) 40 kbp; YAC (yeast artificial chromosome) (*S. cerevisiae)* 200-600 kbp; Radiation Fusion Hybrid (mouse) 1000 kbp+).



```
00000100101000010010100000001   clone 1
10001001000010000000100010001   clone 2
00000100000000000001000000001   clone 3
00010000100000000000001000000   clone 4
00001000000000010000000000010   clone 5
00001000000100000000010000001
00000000100000010101000001000
00100000000100000100101000001
00010010001110000000000100100
```

Figure 1: Hybridization Fingerprinting. Clones spotted onto nylon membrane in a grid by robot (a). Probes only bind to clones containing the probe sequence and are identified because they "light-up" on the autoradiograph. (b) which through image processing is transformed to a binary signature for each clone (c) where '1' indicates hybridization.

completeness. In both approaches the rationale for minimizing or eliminating noise is to reduce or eliminate subsequent wasteful experiments predicated upon the assumption that the map is correct. In other words, the fewer the errors in the map, the less the wasted research effort.

The advantages of statistical approaches are that the problem can be treated as an instance of a well-known class (in Mott's case the symmetric travelling salesman problem) and that reasonable solutions can be generated in a relatively short space of time (e.g. matters of minutes) by standard methods such as simulated annealing or other optimization methods. The disadvantages are that approximation methods with random components such as simulated annealing can produce different solutions on different runs with the same dataset. More importantly, these methods make it difficult to extend or modify the map construction problem to integrate mapping data from other sources such as the genetic map. Some of the existing techniques can only do this by either hand-crafted cost functions or by manual intervention in the map construction process.

The heuristic method of Grigoriev (described in Mott et al. 1993) assumes that there is a large over-coverage of the genome provided by the clone dataset and therefore possibly noisy data can reasonably be eliminated without loss of coverage. The advantages of this approach is that it produces a very reliable probe ordering but the disadvantages are that the resulting solution is not necessarily from the maximal consistent set and too much data is eliminated so that fewer markers are ordered. Furthermore, human intervention is required to resolve indeterminacy in the elimination of non-informative probes.

The potential for improving the existing approaches to the contig map construction problem comes from the need during the map construction process to exploit *a priori* information such as local gene order at some chromosome locations or partial maps constructed by other mapping techniques and to integrate intersecting data sets. Furthermore, we wish to combine elements of the heuristic and algorithmic approaches and develop methods for automating the parts of the process that are presently subject to human judgement. In order to address these issues we have chosen to adopt software tools from AI and in particular logic programming.

In this paper we present a formulation of contig map construction as a constraint satisfaction problem using the parallel constraint logic programming language ElipSys (Veron et al. 1993). We show the effect of different constraining data on reducing the search space and how the parallel programming features of ElipSys can contribute to the realisation of software able to tackle large scale molecular biology problems using declarative programming languages.

```
generate_probe_order(Template, Orders, First, Threshold1, Threshold2):-
    create_probes_template_with_N_slots(NofSites,Template),             (a)
    extract_list(Template,Sites),
    all_different(Sites),                                               (b)
    apply_partial_order(Orders,Template),                               (c)
    apply_neighbourhood_constraints(Threshold1,Template),               (d)
    apply_adjacency_constraints(Threshold2,Template),                   (e)
    build_cost_function(Template,CostFunction),                         (f)
    initial_solution(First),                                            (g)
    minimize(instantiate(Sites),CostFunction)                           (h)
    generate_clone_order(Sites).                                        (i)
```

Figure 2: The top level goal and subgoals of CME.

Constraint Satisfaction and Constraint Logic Programming

Constraint satisfaction problems are ubiquitous in AI and operations research and have received extensive attention since the early 70s (Nadel 1990). In general, constraint satisfaction problems can be characterised as having four components: objects (in CME these are the probes); variables associated with those objects (position); values associated with those variables ([1..N] for each probe); constraints on the values that variables of specific objects can take (Adjacency, Neighbourhood, Partial orders). The goal of constraint satisfaction algorithms is to find values for the variables for each object such that all constraints are simultaneously satisfied.

Most constraint satisfaction algorithms are based on two general approaches, tree search and arc consistency. In tree search, the potential solution space is traversed from the root to the branches by sequentially instantiating the variables in some order. Conversely, arc consistency methods are not tree based but use resolution techniques to generate consistent sets of values for each of the variables of each object and remove inconsistent values (Nadel, 1990). Many approaches have been adopted for programming constraint satisfaction problems and in general the software has been specific to a particular application and programmed in an imperative language such as 'C' for performance reasons. Such approaches, however, often lead to inflexible and hard to maintain programs where simple conceptual changes in the problem specification or the resolution strategy require extensive reprogramming.

A more general approach uses a constraint handling extension to logic programming (LP): constraint logic programming (CLP). CLP can be understood as a generalization of LP where the basic operation of unification is replaced by constraint checking (Jaffar and Lassez 1987, Van Hentenryck 1991). CLP integrates efficient constraint-solving methods from algebra, artificial intelligence, operations research and logic to support, in a declarative way, computational paradigms such as partial enumeration, constraint satisfaction and branch and bound search. These employ both search and consistency methods. CLP languages therefore combine the advantages of LP (declarative semantics, relational form and non-determinism) with the efficiency of special purpose problem-solving (search) algorithms. From a user's perspective CLP programs have great expressive power due to the natural way in which constraints are integrated into the logic programming framework. Program design can also be more intuitive because the programmer works directly in the domain of discourse. Moreover, the runtime efficiency of the resulting program is also enhanced because of the exploitation of computation techniques over specific domains.

ElipSys (ECRC Logical Inferencing Parallel System) is a parallel CLP language developed at ECRC which includes constraint handling on finite domains. The parallel execution model employed by ElipSys allows programs to be run concurrently on either shared or distributed memory computer architectures. Exploitation of parallelism is enabled by user annotations of non-deterministic predicates or through the use of built-in parallel predicates such as `par_member/2`. A useful way of viewing the interplay between the constraint propagation and parallel execution strategies is that the constraints provide the mechanism for *a priori* pruning of the hypothesis space, while parallelism allows it to be traversed more efficiently. ElipSys has previously been employed in RNA secondary structure prediction (Heuze 1989) and Protein Topology Prediction (Clark et al. 1993a, 1993b).

Contig Mapping with ElipSys (CME)

The CME program uses the CLP paradigm of constrain and generate rather than the traditional paradigm of generate and test.

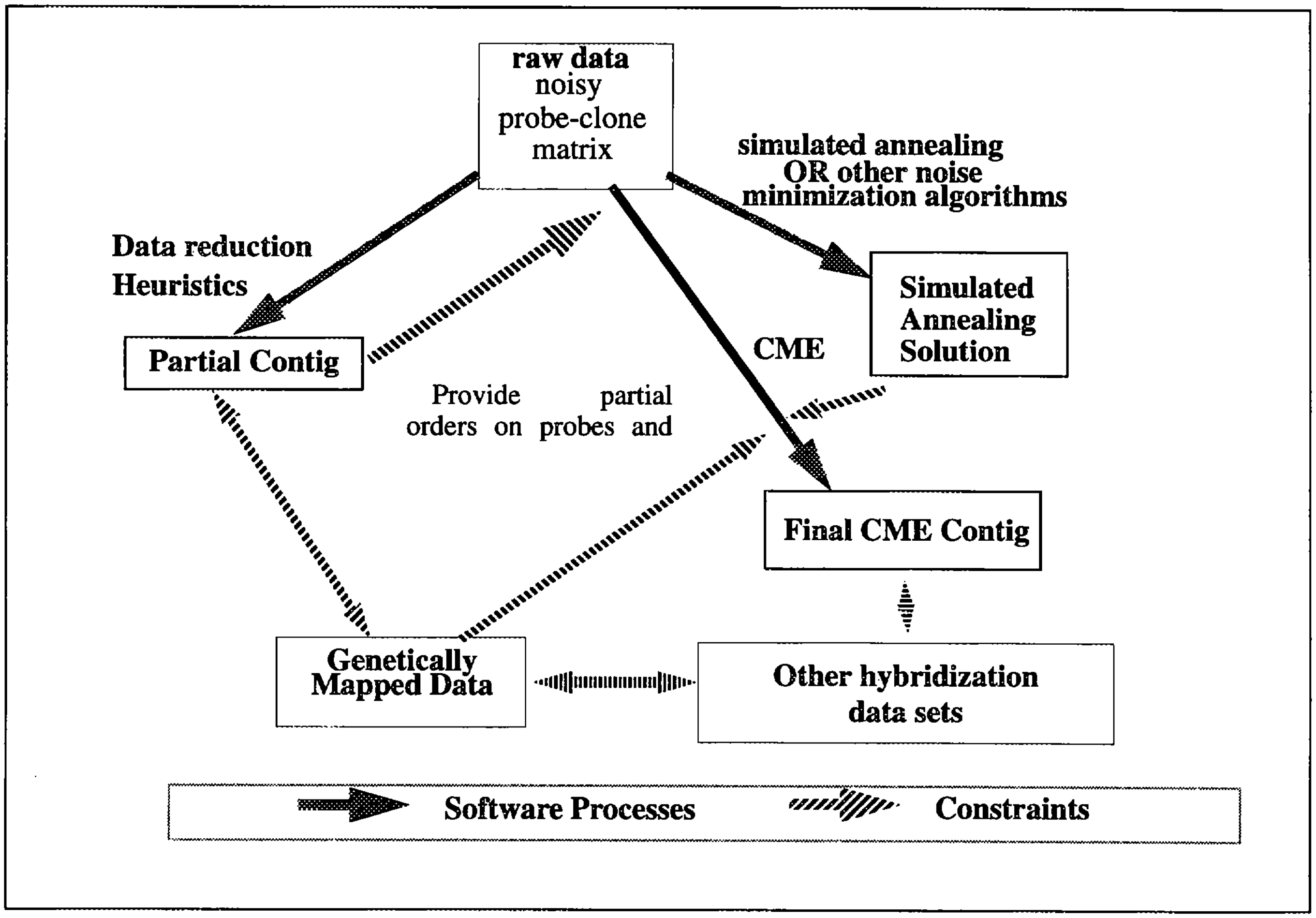


Figure 3: Relation of CME to simulated annealing and other data and analysis techniques. Ensuring consistency with other intersecting data sets is discussed in the final section

The constrain and generate paradigm entails:

- defining the hypothesis space
- specifying constraints
- labelling (or search)

If the intention is to minimize some cost function to direct the search, then the final stage is:

- defining a cost function
- labelling the variables in order to minimise the cost function

Figure 2 shows the top level goals of CME and Figure 3 illustrates the relationship of CME to other aspects of the physical map generation process.

Defining the Hypothesis Space

Like the methods of Mott and Grigoriev, described above, the strategy used by CME is to firstly order the probes and then order the clone library. In general in CLP, the definition of the hypothesis space is achieved through the use of domain variables. The domain definitions determine the possible binding values of each variable. In CME the hypothesis space is defined by a procedure to

- create a template with N slots Fig 2(a)

Here, N is the number of probes, the call to `create_probes_template_with_N_slots/2` uses the ElipSys finite domain variable constructor operators and returns a list of FD variables $S_{1\text{-}N}$ each of which has a domain 1..N to represent the site or probe position

Template = [S_1-_{1, 2, 3,..., N},S_2-_{1, 2, 3,..., N},
... S_N-_{1, 2, 3,..., N}].

Specifying Constraints

In ElipSys, all equalities and inequalities are specified as arithmetic constraints. These are not simply instantiated and then checked. Additionally, the ElipSys runtime system employs look ahead inference rule which attempts to remove inconsistent values from the domain of a variable. The particular constraints applied are spatial constraints (no two probes can co-occur in the same location), threshold constraints (thresholds on the allowable dissimilarity values for neighbouring probes) and partial order constraints (e.g. from a genetic map, or a reliable existing contig).

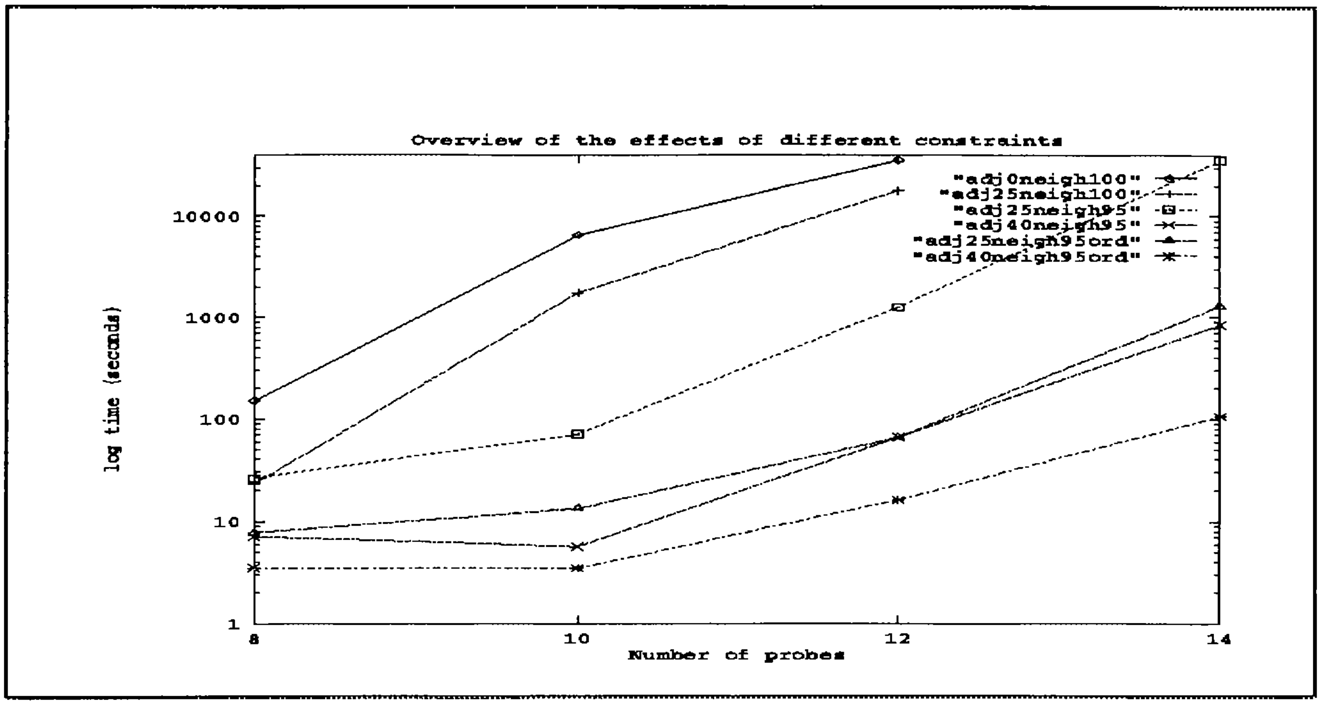


Figure 4: Overview of scaling data - effects of different constraints

- spatial constraints Fig 2(b)

In `all_different(Sites)` the values of the variables all have to be different, i.e. once a probe is positioned it is not considered further. This constraint is supported by the built-in predicate `all_different/1`.

- partial order(s) constraints (Orders) Fig 2(c)

If it is known that probe x is localised before probe y then the domain variable representing the sites associated with probe x will be smaller than the one associated with probe y:

Sx =_{1, 2,..., N-1} and Sy =_{2, 3,..., N}

- Threshold constraints Fig 2(d,e)

The neighbourhood constraint ensures that two or more probes with a pairwise dissimilarity (Eq 2) less than the threshold must be neighbours and its logical opposite, the adjacency constraint, ensures that probes with a pairwise dissimilarity value greater than threshold2 cannot be adjacent.

Cost Function

The two costs described here are each defined as an additive linear function. Only one cost is used at a time for minimization purposes, though the program computes both for each run. They are built by the clause `build_cost_function(Template,CostFun)` Figure 2(f).

Cost1 is the sum of all the dissimilarities values between the probes for a given order and is the classic pairwise dissimilarity measure employed in a variety of techniques. Supposing T_{xy} is the number of clones that hybridize to both probes and R_{xy} the number of clones hybridizing to either probe; then the maximum likelihood of the distance between the probes is:

$$\mathbf{d_{xy} = (T_{xy} - R_{xy}) / T_{xy}} \qquad \textbf{(Eq. 1)}$$

These values are pre-computed. The total cost, P(π), for a permutation π of the probes is therefore:

$$P(\pi) = \sum d_{\pi_i \, \pi_{i+1}} \qquad \textbf{(Eq. 2)}$$

Cost2 is defined as the sum of the costs for each clone, defined for each clone as the number of positive hybridizations not in the largest contig. It is achieved by using a built-in symbolic constraint defined in `sequences(+ListItems, +Item, +MaxLength, -Occurrences)`.

Note that although this cost function is computed across clones, the particular search tree traversed consists of probe orders with dependencies propagated using an active constraint mechanism that links probe and clone sites using a goal suspension mechanism.

Search Strategy and Minimization

The user can assign an initial starting probe order, which may be the result of a previous analysis in the predicate `initial_solution(?First)`, Figure 2(g). This initial solution is used both to compute the initial value of the cost function for subsequent minimization and as the configuration from which search may begin, unless it violates a hard constraint. A user-supplied configuration might be the output from an optimization procedure such as

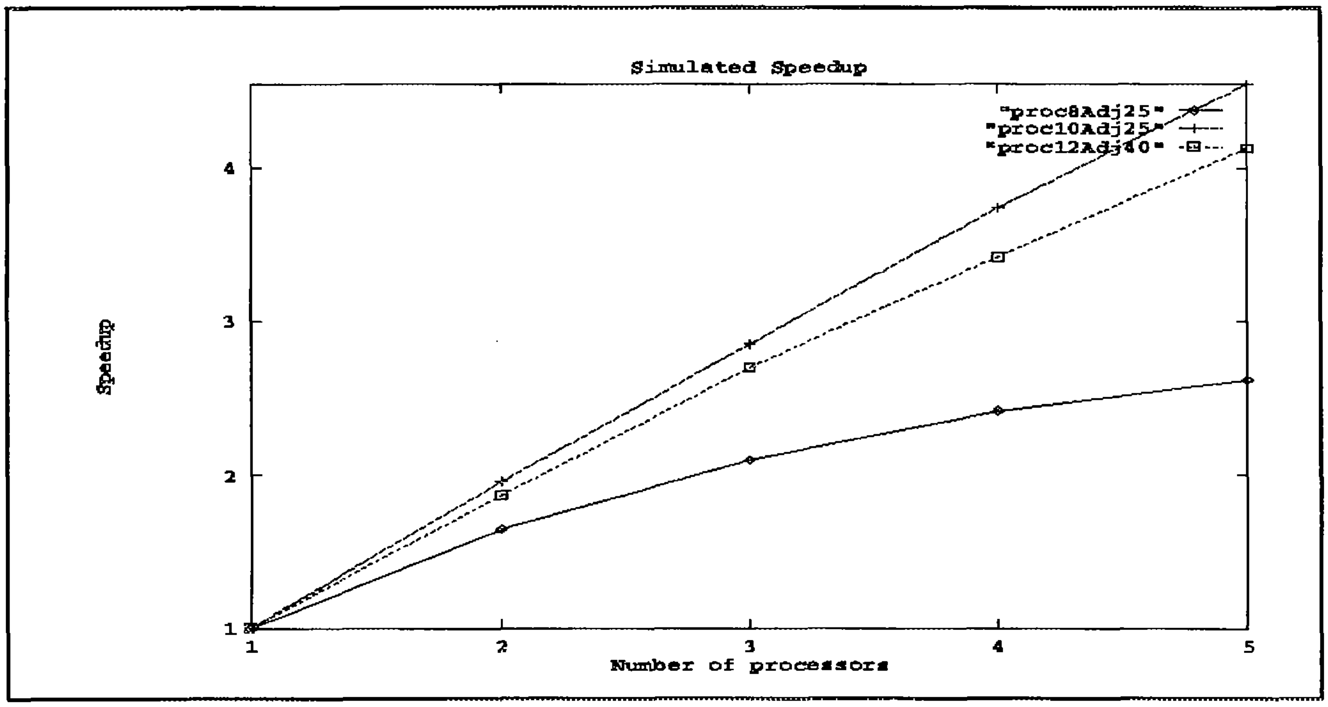


Figure 5: Simulated speed-up due to parallel processing

simulated annealing. In the absence of a user-supplied starting configuration an initial solution is generated using the "greedy algorithm" where the first probe is taken as that with the smallest set of possible neighbours.

- minimization of cost Fig 2(h)

The cost minimization is performed using the built-in predicate `minimize(Term,Cost)` where *Term* is the predicate attributing the values to be used in the cost function, here `instantiate(Sites)`. This predicate calls a built in branch-and-bound algorithm which prunes branches of the search tree for which the partial cost is already greater than the existing minimum and is guaranteed to find a global minimum cost solution within the constraints, given unlimited computational resources.

- explicit instantiation Fig 2(h)

Search is initiated using the `instantiate/1` predicate. Only combinations of variable assignments not ruled out by the constraints are considered. The precise search strategy employs a best first search based on the pairwise measures in Eq. 1. It is this aspect of the program that is potentially executed in parallel.

Finally, Figure 2(i), clones are ordered using the dynamic programming algorithm of Mott et al. (1993) implemented in C.

Results and Evaluation

We describe the ordering of a YAC library from *S. pombe*, a unicellular yeast with a genome of 15 million base pairs (about one-tenth of the human X chromosome) divided into 3 chromosomes. It has all the basic characteristics of a eukaryotic genome. Over 450 genetic markers have been identified, with over 200 markers genetically localized on the three chromosomes. Because of its small size and a low abundance of repeated sequences, the *S. pombe* genome is useful for prototyping new mapping techniques. Cosmid and YAC genomic libraries have been constructed by the ICRF Genome Analysis Laboratory to assist in the development of large scale mapping methods. Data from hybridizations between a panel of probes and the ICRF YAC *S. pombe* library was separated into three subsets corresponding to the three chromosomes (according to the results obtained by the statistical approach of Richard Mott). In this study we focused on the data for Chromosome 2. Since most of the probes correspond to genetically identified markers, such experiments should also allow the alignment of the genetic map with the physical map.

Scaling-up

In order to evaluate the scaling properties of CME a structured set of samples of increasing size were selected from the chromosome 2 data set. Results for scaling data are summarized in Figure 4 for runs on a SparcStation2. Each dataset contained N probes (N $\in$ {8, 10, 12, 14,16} with 3N clones and N hybridizations due to noise. The data presented are the medians of 9 runs (3 runs on each of 3 random data orders) measuring CPU time spent in user mode. The six curves represent the effects of introducing various constraints into the search. Assuming that complete dissimilarity between probes corresponds to a value of 100, we constrain all pairs of probes which have a dissimilarity

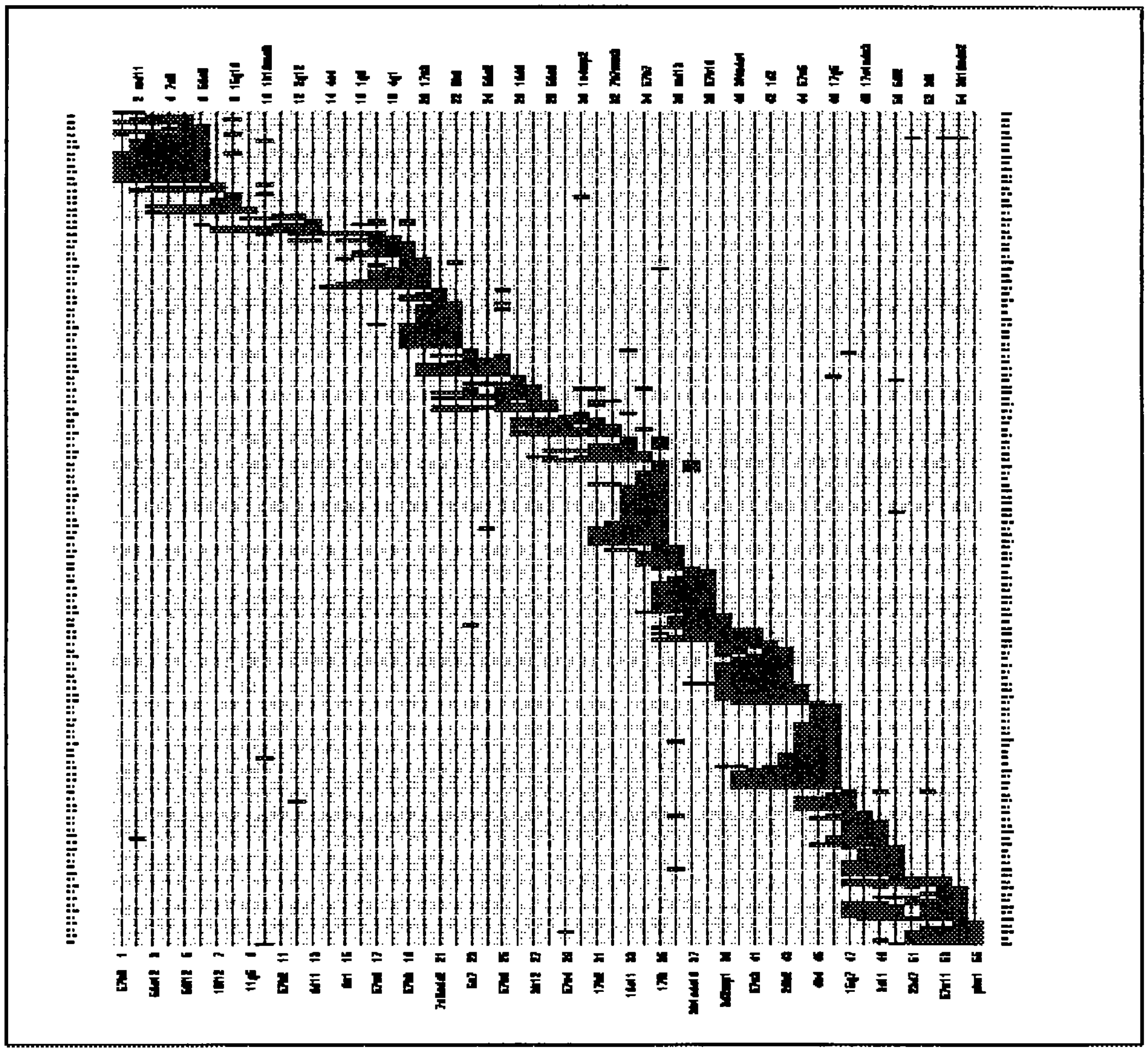


Figure 6: CME generated physical map for *S. pombe* chromosome 3 for 55 probes, 269 clones, (3rd solution)

value less than a threshold to be adjacent in the final probe order. The far left curve (adj0neigh100) is obtained without any constraints, while the second curve (adj25neigh100) shows the effect of increasing the adjacency constraint. If we now introduce a further constraint which states that probes with greater than 95% dissimilarity cannot be adjacent the search time is further decreased (adj25neigh95). Finally the last two curves demonstrate the effect of partial order constraints, for a partial order among N/2 of the probes (adj40neigh95ord). The overall result is clear and shows that the introduction of each additional constraint vastly reduces computation time. However it is also clear that the resulting curves are still exponential in the number of probes. Figure 5 shows the possible speed-ups by use of parallelism. These data were derived using simulated parallelism on a SparcStation2. The point to note is that the speed-up was greatest for the largest search space (10 probes with an adjacency constraint of 25%) - a factor of approximately 4.5 with 5 processors. We are currently replicating these results on an ICL DRS6000 with 4 Sparc2 processors.

Full data sets

Results for CME with the data for chromosome 2 (55 probes 502 clones) are shown in Table 1. These were obtained using the best solution of Mott et al. (1993) as the starting configuration and the final heuristically produced partial map using the method of Grigoriev as a partial order constraint. CME demonstrated that using Cost1 (above), moderate adjacency and neighbourhood constraints and the partial order constraints from Grigoriev's map (Mott et. al. 1993), the simulated annealing solution of Mott (1993) was a minimum solution for this cost function. Using the alternative cost function (Cost2), however, CME found a slightly superior solution after 6 minutes of CPU usage.

Table 1:

Method	Cost 2 (CPU time)	Cost 1 (CPU time)
Simulated Annealing (Mott et al., 1993) CME Starting Solution	200	2256 (few mins)
CME 2nd solution	196 (6 minutes)	2256 (11 hours)

In a second analysis, clones having 0, 1 or more than 7 positive hybridization were eliminated from the data set. As with the heuristic method of Grigoriev, the justification for this is that clones with 0 or 1 hybridizations do not contribute to contigs, while clones with more than 7 hybridizations are very likely to contain noise. Thus the reduced data consisted of 55 probes and 239 clones (from 502 in the initial dataset) (Table 2). We have not yet had the opportunity to run simulated annealing with this reduced data set and hence the initial starting configuration was based on the simulated annealing solution for the whole data set (502 clones). However, CME was able to improve significantly upon the best result for simulated annealing with the larger set using the alternate cost function. Figure 6 shows the resulting contig for the third CME solution.

Table 2:

Method	Cost2 (time)	Cost1 (time)
Simulated Annealing (Mott et al., 1993) CME Starting Solution	138	2546 (few mins)[a]
CME 2nd solution	116 (5 minutes)	2523
CME 3rd solution	110 (7 minutes)	2459

a. Based on simulated annealing solution of full set of clones.

Discussion and Conclusions

In this paper we describe the ordering of a subset of genetic markers in a YAC library from the yeast *S. pombe* using a program written in the ElipSys parallel constraint logic programming language. The generation of long range physical maps is a problem with combinatorial complexity that also requires complex human judgement in order to integrate the range of data necessary to yield satisfactory solutions. Problems with similar properties have been addressed previously using AI languages and techniques. In particular the contig mapping problem shares many of the difficulties of restriction endonuclease mapping (Stefik 1978). Other applications of AI techniques to molecular biology can also be considered as examples of constraint solving. For example, DENDRAL (Buchanan and Feigenbaum 1978) the MOLGEN project (Stefik 1981a,b) and the PROTEAN system (Hayes-Roth 1987).

These earlier systems were either written in LISP or developed in general problem solving architectures built in LISP. Although there are many advantages in the use of logic programming languages (e.g. Prolog) for building knowledge-based systems, LP languages have not often been considered as appropriate vehicles for large scale AI applications.The main problem attributed to LP arises from its uniform but simple depth-first, left-to-right computational model. This makes LP programs equivalent to simple generate and test with standard backtracking. However, in generate and test, constraints are only used to test if the overall assignment is a solution. No pruning occurs in the search except of the search tree subordinate to the current node as defined by the current set of variable bindings. Thus generate and test potentially explores too much of the search space and is very inefficient for larger search spaces.

The inefficiency of generate and test in LP may be contrasted with the result of search procedures based on consistency techniques (local value propagation, data driven computation, forward checking and look ahead). They are based on the idea of *a priori* pruning, that is the using of constraints to reduce the search space before the discovery of a failure. The pruning in consistency techniques is achieved by spending more time at each node of the search tree removing combinations of values that can not appear in a solution. Thus these procedures are oriented towards the prevention of failures and enable both an early detection and a reduction of the backtracking and the constraints check.

Constraint logic programming therefore offers the best of two worlds: the declarative semantics, relational form and problem abstraction features of LP languages which when combined with consistency techniques make it possible to preserve most of the efficiency of specialized programs. This leads to the idea of embedded consistency techniques into logic programming so that logic programs keep their descriptive features while being more efficient. This new paradigm can be characterised as "constrain and generate".

The research described in this report has shown that using techniques from CLP as currently implemented in ElipSys it is possible to tackle the problem of building probe hybridization map for data sets of limited size, yielding comparable or improved solutions over other methods. Furthermore it was demonstrated that the addition of each type of constraint can potentially decrease the total search time by an order of magnitude. Finally it should be emphasised that CME is complimentary to other contig mapping approaches in so far as the results of previous analyses can either be used as partial order constraints or as initial starting positions.However, the huge search space which has to be explored would still make the application very slow with large data sets (e.g. human X chromosome). There are a number of developments that might be considered for improving this limitation.

Recursive Algorithms and Parallelism

CME simultaneously orders all probes. An alternative approach would be to incrementally and recursively order increasingly large sets of probes taking the best previous order as a fixed ordinal level constraint and introducing the probes in a best first (i.e. lowest pairwise dissimilarity) basis.This has some similarities with the method described by Letovsky and Berlyn (1992). As we have demonstrated, parallelism resulted in near linear speed-ups. Use of massive shared memory parallelism can therefore be expected to play a significant role for larger data sets.

More constraints

In CME, constraints come from three sources: general properties of good solutions (adjacency and neighbourhood constraints), partial orders from other sources and the data themselves. Clearly if more uncertain data is introduced the computation can become more complex, because the cost function to be minimized has more free variables. However, if new constraints in the form of characteristics of good solution or partial orders or other "certain" constraining data can be identified then the problem potentially becomes simpler because more values from the domain of variables can be eliminated without the need for search.

Alternative cost functions

Use of pairwise difference as a cost measure has the effect of obscuring experimental artifacts such as clone co-ligation. The alternate cost function described in this paper makes a step in the direction of explicitly modelling this process. Subsequently, a new ElipSys built-in constraint has been devised explicitly to model co-ligation for use in minimization. The new constraint `contigs/5` should make it possible to explicitly model the process of co-ligation, rather than treat it as random noise. Indeed using such a constraint it is possible to specify constraints on solutions such that all clones with a given number of hybridizations should contain a given number of contigs.

We are currently extending the use of CME with the *S. pombe* dataset using elements of all these four. The long term aim for extending this approach for human chromosome physical mapping is probably most dependent upon the availability of constraints arising from other data sets.- in particular the use of mapping data at many different resolutions (e.g. RFHs, mega-YACs).

Acknowledgements

This research was funded by the Imperial Cancer Research Fund, the European Computer-Industry Research Centre and the European Union under ESPRIT III project 6708 "APPLAUSE". Thanks to Richard Mott (ICRF Genome Analysis Laboratory) for making data and display programs available and ECRC colleagues for ElipSys support.

References

Branscomb, E. et al. 1990. Optimizing Restriction Fragment Fingerprinting Methods for Ordering Large Genomic Libraries., *Genomics*. 8:351-366.

Buchanan, B. G. and Feigenbaum, E. A. 1978. DENDRAL and Meta-DENDRAL: Their Applications Dimension. *Artificial Intelligence*. 11:5-24.

Clark, D. A.; Rawlings, C. J.; Shirazi, J.; Veron, A. and Reeve, M. 1993(a). Protein Topology Prediction through Parallel Constraint Logic Programming, In Proceedings of the First International Conference on Intelligent Systems for Molecular Biology (eds. L. Hunter, D. Searls, and J. Shavlik) AAAI/MIT Press, Menlo Park CA.

Clark, D. A.; Rawlings, C. J.; Shirazi, J.; Li, L-L.; Reeve, M.; Schuerman, K. and Veron, A. 1993(b). Solving Large Combinatorial Problems in Molecular Biology Using the ElipSys Parallel Constraint Logic Programming System. *The Computer Journal* 36(8): 690-701.

Hayes-Roth, B.; Buchanan, B.G.; Lichtarge, O. et al. 1986. PROTEAN: Deriving Protein Structure from Constraints. Proceedings of National Conference of American Association of Artificial Intelligence 5: 904-909

Heuze, P. 1989. RNA secondary structure prediction in ElipSys. Technical Report ElipSys/10. European Computer-Industry Research, Centre, Arabellastrasse 17, D-8000, Munich 81, Germany.

Lassez, J-L. and Jaffar, J. 1987. Constraint Logic Programming. In Proceedings of the 14th ACM Symposium on Principles of Programming Languages, Munich, Germany.

Lehrach, H.; Drmanac, R.; Hoheisel, J. et al. 1990. Hybridization Fingerprinting in Genome Mapping and Sequencing, *Genome Analysis, 1, Genetic and Physical Mapping*. Cold Spring Harbour Laboratory Press, 39-81

Letovsky, S. & Berlyn, M.B. 1992. CPROP: A rule-based program for constructing genetic maps. *Genomics*. 12:435-446.

Michiels, F. 1987. Molecular approaches to genome analysis: a strategy for the construction of ordered overlapping clone libraries, *CABIOS* 3(3): 203-210.

Mott, R, Grigoriev, A, Maier, E, Hoheisel, J. and Lehrach, H. 1993. Algorithms and software tools for ordering clone libraries: Application to the mapping of the Genome of *Schizosaccharomyces pombe. Nucleic Acids Research* 21(8): 1965-1974.

Nadel, B.A. 1990. Constraint Satisfaction Algorithms. *Computational Intelligence* 5(4): 188-221.

Stefik, M. 1978) Inferring DNA Structures from Segmentation Data. *Artificial Intelligence* 11: 85-114

Stefik, M. 1981(a). Planning with Constraints [MOLGEN: Pt 1]. *Artificial Intelligence* 16: 111-140

Stefik, M. 1981(b). Planning and meta-planning [MOLGEN: Pt 2]. *Artificial Intelligence*16: 141-169

Torney, C. Schenk, K. Whittaker, C. White, S. 1990. Computational methods for physical mapping of chromosomes. The First International Conference on Electrophoresis, Supercomputing, and the Human Genome; (Cantor and Lim, eds): World Scientific, 268-278.

Van Hentenryck, P. 1989. *Constraint Satisfaction in Logic Programming*, MIT Press

Veron, A, Schuerman, K, Reeve, M. and Li, L-L. 1993. How and Why in the ElipSys OR-parallel CLP System. In Bode, A, Reeve, M. and Wolf, G (eds) PARLE 93, *Lecture Notes in Computer Science*, 694: 291-304. Springer-Verlag, Heidelberg.