# Constraints and Global Optimization for Gene Prediction Overlap Resolution

Have, Christian Theil

*Publication date:*
2011

*Document Version*
Early version, also known as pre-print

*Citation for published version (APA):*
Have, C. T. (2011). *Constraints and Global Optimization for Gene Prediction Overlap Resolution*. Paper presented at Workshop on Constraint Based Methods for Bioinformatics, Perugia, Italy.

# Constraints and Global Optimization for Gene Prediction Overlap Resolution

Christian Theil Have

Research group PLIS: Programming, Logic and Intelligent Systems
Department of Communication, Business and Information Technologies
Roskilde University, P.O.Box 260, DK-4000 Roskilde, Denmark
E-mail: cth@ruc.dk

**Abstract.** We apply constraints and global optimization to the problem of restricting overlapping of gene predictions for prokaryotic genomes. We investigate existing heuristic methods and show how they may be expressed using Constraint Handling Rules. Furthermore, we integrate existing methods in a global optimization procedure expressed as probabilistic model in the PRISM language. This approach yields an optimal (highest scoring) subset of predictions that satisfy the constraints. Experimental results indicate accuracy comparable to the heuristic approaches.

## 1 Introduction

Traditionally, gene finding has been considered as a classification task which could be performed without much context [6]. This ignores the problem of the constraints between the set of predicted genes and their placement in the genome. A common problem occurs with overlapping genes. Overlapping genes are rare in prokaryotic genomes, but they do occur [12, 8].

The traditional intrinsic gene finding methods have a tendency to predict too many overlapping genes (particularly in GC rich genomes) because the feature patterns of a gene predicted in one reading frame give rise to similar feature patterns in other reading frames. This effect is known as shadow genes.

Several gene finders deal with the problem of overlapping genes by discarding some of the overlapping predictions in a post-processing step. In this paper we consider and compare such post-processing techniques and give unified presentation using Constraint Handling Rules [7]. We demonstrate how such rules can be formulated as constraints and integrated with a global optimization procedure implemented as a constrained Markov chain in the PRISM system [13].

We adopt a divide and conquer approach to gene finding, which can be seen as composed of two steps:

1. A gene finder supplies a set of candidate predictions $p_1 \dots p_n$, called the *initial set*.
2. The *initial set* is pruned according to certain rules or constraints. We call the pruned set the *final set*.

The present paper is concerned with methods for the second step. The purpose of this step is to repair effects of flawed assumptions in the first step, i.e. leading to over-prediction of overlapping genes, and more specifically to improve accuracy by pruning false predictions. We consider this step as a Constraint Satisfaction Problem (CSP).

**Definition 1.** *A Constraint Satisfaction Problem is a triplet* $\langle X, D, C \rangle$. *X is a set of n variables,* $X = x_1, \ldots, x_n$, *with domains* $D = D(x_1), \ldots, D(x_n)$. *The constraints C impose restrictions on possible assignments for sets of variables. A solution is an assignment of a value* $v \in D(x_i)$ *to each variable* $x_i \in X$, *consistent with C.*

We introduce variables $X = x_i \ldots x_n$ corresponding to each prediction $p_1 \ldots p_n$ in the initial set. All variables have boolean domains, $\forall x_i \in X, D(x_i) = \{true, false\}$ and $x_i = true \Rightarrow p_i \in$ **final set**.

If there are multiple solutions, then we are usually interested in the "best" one. We interpret "best" as meaning a solution that contains as many real genes as possible and as few incorrect predictions as possible. We do not know in advance which predictions are correct, but optimize the probability (or a similar measure) that the predictions are correct. This extends the problem as a constraint optimization problem.

**Definition 2.** *A Constraint Optimization Problem (COP) is a CSP where each solution is associated with a cost and the goal is to find a solution with minimal cost*[1].

## 2 Local heuristic methods

An approach taken by many gene finders is to employ local heuristic pruning rules to post-process a set of gene predictions. These rules make pruning decisions based on the context of only a subset of the predictions. Typically, the rules consider overlapping predictions on a case by case basis and deletes inconsistent predictions based on various criteria. The rules essentially work as propagators that reduce the domains of variables, e.g. a deletion corresponds to reducing the boolean domain of the corresponding variable to $false$. The drawback is that the rules are generally not guaranteed to yield a globally optimal solution and that they may produce different solutions depending on the order in which they are applied.

These types of rules are conveniently expressed as *simplification* rules in the Constraint Handling Rules (CHR) language. Such rules work on a constraint store, which starts out as the initial set. The simplification rules remove predictions from the constraint store, until no more rules apply. Then, the constraint store represents the final set.

As example, consider the post-processing procedure of the Genemark frame-by-frame gene finder [11] expressed as a single rule in CHR:

[1] Or equivalently, a solution with maximal negative cost (utility).

```
prediction(Left1,Right1), prediction(Left2,Right2) <=>
        Left1 =< Left2,  Right1 >= Right2
        |  prediction(Left1,Right1).
```

The head of the rule — the part before `<=>` — matches two predictions in the constraint store. The rule replaces both predictions with the first prediction if the first prediction completely overlaps the second prediction. This condition is expressed in the guard of the rule – the part between the head and the `|` character. The rule is applied for all predictions matching the head and the guard, effectively removing all predictions which are completely overlapped by another prediction. With this rule it does not matter in which order the predictions are processed – the final set will be same. This is a consequence since the program consisting of the unique rule presented is *confluent* [1], i.e. it is not sensitive to the order of execution.

As an example of non-confluent rules, consider the scheme used in the ECOPARSE gene finder [9] which addresses partial overlaps and the score of the predictions:

```
prediction(Left1,Right1,Score1), prediction(Left2,Right2,Score2) <=>
    overlap_length((Left1,Right1),(Left2,Right2),OverlapLength),
    length_ratio((Left1,Right1),(Left2,Right2),Ratio),
    length(Left1,Right1,Length1), length(Left2,Right2,Length2),
    OverlapLength > 15, Score1 > Score2
    ((Length1 > 400, Length2 > 400) ; Ratio > 0.5),
    | prediction(Left1,Right1,Score1).

prediction(Left1,Right1,Score1), prediction(Left2,Right2,Score2) <=>
    overlap_length((Left1,Right1),(Left2,Right2),OverlapLength),
    length_ratio((Left1,Right1),(Left2,Right2),Ratio),
    length(Left1,Right1,Length1), length(Left2,Right2,Length2),
    OverlapLength > 15,  Ratio =< 0.5,  Length1 =< Length2
    | prediction(Left1,Right1,Score1).
```

If two predictions overlap by more than 15 bases, then one of them is removed. If the ratio between the longest and shortest of the predictions is more than 0.5, then the lowest scoring is removed (first rule) otherwise the shortest one is removed (second rule). Note how this may lead to different effects depending on the order in which predictions are considered, as illustrated in figure 1.

There are other approaches which employ more complex local heuristics. An example is heuristics of the RescueNet gene finder [10] which has rules considering scores, percent overlaps and local overlaps between up to three predictions. These heuristics can be implemented with nine CHR rules (not shown), but the resulting program is not confluent.

It is a general theme for the heuristics to be based on two central characteristics of overlapping predictions – the score of the predictions and the (relative) lengths of the predictions and the overlap.

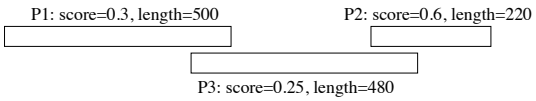


**Fig. 1.** ECOGENE post processing: We have two predictions $P1$ and $P2$ that overlap each end of a third prediction $P3$ by more than 15 bases. If $P1$ and $P3$ is are considered before $P2$ and $P3$ then $P3$ will be removed by the first rule. Consequently $P2$ does not overlap and is kept. If they are considered in opposite order, however, then $P2$ will be removed by the second rule and subsequently $P3$ is removed by the first rule.

## 3 Global optimization

We would like the final set to reflect the relative confidence scores in the predictions assigned by the gene finder and at the same time be consistent with the overlap constraints. To accomplish this we reformulate the problem as a constraint optimization problem.

Let the scores of $p_1 \dots p_n$ be $s_1 \dots s_n$ and $s_i \in \mathcal{R}^+$. The scores are the confidence scores given by the underlying gene finder, i.e. they reflect the supposed probability that a prediction constitutes a real gene. Such scores are commonly expressed as probabilities, but need not be.

We would like to maximize the sum of the scores $\sum_{i=1}^{n} s_i$ since it is directly related to the criteria of the model that produced the initial set. With this criteria, the inclination to prune a prediction in the final set is inversely proportional to the score which is expected to reflect the underlying models belief that the prediction is a real gene.

To perform global optimization with a set of constraints, we propose to use a constrained first-order Markov chain. We assume that a gene finder has produced initial set of predictions, $p_1 \dots p_n$, and further require these to be sorted by the position of their left-most base, such that $\forall p_i, p_j, i < j \Rightarrow \text{left-most}(p_i) \leq \text{left-most}(p_j)$. The variables $x_1 \dots x_n$ of the CSP are given the same ordering.

The Markov chain has a *begin* state, an *end* state and two states for each variable $x_i$ corresponding to its boolean domain $D(x_i)$. The state corresponding to $D(x_i) = true$ is denoted $\alpha_i$ and the state corresponding to $D(x_i) = false$ is denoted $\beta_i$. In this model, a path from the begin state to the end state corresponds to a potential solution of the CSP. The Markov model is illustrated in figure 2. The *begin* state has transitions to $\alpha_1$ with probability $P(\alpha_1|begin) = \sigma_1$ and $\beta_1$ with probability $P(\beta_1|begin) = 1 - \sigma_1$. The last two prediction states, $\alpha_n, \beta_n$ can only transit to the end state, i.e. $P(end|\alpha_n) = P(end|\beta_n) = 1$. For all other states, we have the transition probabilities,

$$P(\alpha_i|\alpha_{i-1}) = P(\alpha_i|\beta_{i-1}) = \sigma_i \text{ and } P(\beta_i|\alpha_{i-1}) = P(\beta_i|\beta_{i-1}) = 1 - \sigma_i$$

We normalize the scores to the interval $(0.5, 1]$, yielding the normalized probability scores $\sigma_1 \dots \sigma_n$, in the following way,

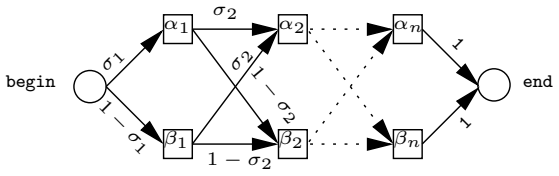


**Fig. 2.** Illustration of the Markov chain used. The transitions are marked with their corresponding probabilities. Only the first few and the last states are included - the dotted transition arrows symbolize the omitted $\alpha_3 \ldots \alpha_{n-1}$ and $\beta_3 \ldots \beta_{n-1}$ states and their transitions, which follows the same principle as the previous.

$$\sigma_i = 0.5 + \lambda + \frac{(0.5 - \lambda) \times (s_i - min(s_1 \ldots s_n))}{max(s_1 \ldots s_n) - min(s_1 \ldots s_n)}$$

$\lambda$ is a small pseudo-count to ensure that all $\sigma$ scores are above 0.5. Since $\alpha$ probabilities are always larger than 0.5, the model prefers $\alpha$ states over their corresponding $\beta$ states. Hence, a most probable path from the *begin* state to the *end* state will not include any $\beta$ states. The predictions that maximize the product of the $\sigma$ scores will also maximize the sum of the original scores, since the normalized $\sigma$ scores are monotonic to the original scores, $\sigma_i \geq \sigma_j \iff s_i \geq s_j$.

For inference with the model we use the Viterbi algorithm [16], which returns a most probable state sequence $\{begin, S_1, S_2 \ldots S_n, end\} | S_i \in \{\alpha_i, \beta_i\}$.

Constraints are defined on states that are not allowed to occur together in a path. These constraints force the Viterbi algorithm to choose a most probable path, consistent with the imposed constraints, i.e. this path may include $\beta$ states. The constraints are formulated as CHR rules similar to those of the local heuristics, but instead of removing predictions they define conditions for inconsistency. We call these *inconsistency rules*. Inconsistency rules match predictions corresponding to $\alpha$ and $\beta$ states in the head of the rule. The guard of the rule ensures that the additional criteria for rule application are met and the implication of the rules is always failure. Note that such rules are necessarily confluent.

As example, version 3 of the Glimmer gene finder [5] use a similar approach with a constraint that enforce a maximal length of overlaps (110 for *E.coli*). In our system, this constraint is formulated as,

```
alpha(Left1,Right1), alpha(Left2,Right2) <=>
    overlap_length((Left1,Right1),(Left2,Right2),OverlapLength),
    OverlapLength > 110
    | fail.
```

The Genemark heuristic rule is represented as two inconsistency rules,

```
alpha(Left1,Right1), alpha(Left2,Right2) <=>
    Left1 =< Left2, Right1 >= Right2 | fail.
beta(Left1,Right1), alpha(Left2,Right2) <=>
    Left1 =< Left2, Right1 >= Right2 | fail.
```

The first rule states that one prediction may not completely overlap another and the second says that we cannot include a prediction if a pruned prediction completely overlaps it. Since the heuristic is confluent it may also be applied to the initial set as a filtering algorithm before the process of global optimization. We can reformulate the two ECOGENE rules in the same fashion (guard is omitted, but it is the same as in the heuristic rules),

```
alpha(Left1,Right1), alpha(Left2,Right2) <=>  ... | fail.
beta(Left1,Right1), alpha(Left2,Right2) <=>  ... | fail.
```

Note that the `Score` arguments have been removed. They are now implicitly integrated in the optimization algorithm. The confluence issue is resolved due to the optimization procedure. In effect, the execution strategy that maximizes the score is applied.

### 3.1 Implementation in PRISM

A PRISM program that implements the constrained Markov chain is created from the initial set of predictions and constraints expressed as CHR rules. PRISM is an extension of Prolog with special goals representing random variables. A derivation of the PRISM program corresponds to a path through the Markov chain. The Markov chain is implemented as a recursive predicate, such that in the $i'th$ recursive call, the (random) variable $x_i$ is assigned a value corresponding to a Markov chain state; $\alpha_i$ or $\beta_i$. After each recursion — an attempted transition in the Markov model — the constraints are checked.

*Relevant recent states* As part of a derivation we maintain a list of recent states ($m_i$) sorted by the right-most position of the corresponding predictions. Constraints are only checked for predictions corresponding to elements of $m_i$. In step $i$, we construct $m_i$ as the maximal prefix of $x_i + m_{i-1}$, such that $x_j \in m_i \iff \text{right-most}(p_j) \geq \text{left-most}(p_i)$. If the constraints propagate `fail`, then the PRISM derivation fails and the (partial) path it represents is pruned from the solution space.

The most probable consistent path is found using PRISMs generic adaptation of the Viterbi algorithm for PRISM programs [14].

## 4 Evaluation

In lack of a true golden standard, we use an accepted reference set to define the set of "correct" genes. A slight complication of this approach is that the reference set itself may have incorrect and missing annotations. True positives

are gene predictions in the final set which are included (exactly) in the reference set and false positives are those predictions that are not.

Traditionally, in gene finding, accuracy is measured in terms sensitivity and specificity. Sensitivity measures the fraction of reference genes exactly predicted by the approach and specificity measures the fraction of predicted genes that are correct. Since the starting point is the initial set of predictions (which may omit some potential genes) we cannot improve on sensitivity. The goal of a pruning approach is then to improve on specificity with minimal impact to sensitivity.

We consider a pruning approach *successful* wrt. to an initial set when it prunes false positives at a higher rate than it prune true positives. This is reflected by the difference in sensitivity and specificity of the final set compared to the initial set.

We consider constraints *safe* when the constraints prune only false positives. Neither of the examined constraints are safe with respect the RefSeq annotation of *E.coli*, *NC*_000913. Three of the reference genes are completely overlapped by another reference gene. These would be removed by the genemark heuristic and hence it is not safe, although the negative impact of sensitivity would negligible. Similarly with the Glimmer constraint – the reference annotation have four overlaps longer than 110 bases which would be removed by this constraint. There are 93 overlaps longer than 15 bases. All of these would be removed by the ECOGENE constraints, which is therefore expected to have a noticeable negative sensitivity impact.

### 4.1 Experimental validation

We compare the different approaches using the predictions from a very simple codon preference based gene finder – the simplest model described in [4]. The gene finder has been trained on *E.coli NC*_000913 and applied to predict genes in the same genome. It overpredicts quite a lot – a total of 10799 predictions for the genome, which has 4145 known genes.

We ran the constrained Markov chain using the gene finder predictions as initial set, applying our adaptations of the both the Genemark constraint, the ECOGENE constraint and the Glimmer3 constraint. We also tested the local heuristic versions of the Genemark and ECOGENE constraints. The results are summarized in table 1.

Both the Genemark and ECOGENE heuristics achieve quite impressive improvement compared to the initial set. Our global optimization achieves better sensitivity than ECOGENE and better specificity than Genemark, but seen as a combination of the measures, the result is not significantly better.

Note that the optimal or highest scoring set of predictions subject to the constraints is not necessarily the most successful, but it is the one that most faithfully reflects the confidence scores assigned by the gene finder.

The purely declarative CHR implementations of genemark or ECOGENE rules are quite slow (hours), e.g. it essentially considers each pair of constraints resulting in $\mathcal{O}(n^2)$ complexity, $n$ being the number of predictions in the initial

| Method | #predictions | Sensitivity | Specificity | Time (seconds) |
|---|---|---|---|---|
| **initial set** | 10799 | 0.7625 | 0.2926 | na |
| **Genemark rules** | 5823 | 0.7558 | 0.5379 | 1.4 |
| **ECOGENE rules** | 4981 | 0.7148 | 0.5947 | 1.7 |
| **global optimization** | 5222 | 0.7201 | 0.5714 | 75 |

**Table 1.** Accuracy of predictions using different overlap resolution approaches. Note that the results for the ECOGENE heuristic may vary depending on execution strategy - in case of above results, predictions with lower left position are considered first.

set. However, with proper control in place (using the relevant recent states optimization described in section 3.1), they can be made to run very fast (less than two seconds). The running time for the global optimization is slower – it takes a little more than one minute. This is still acceptable.

## 5 Conclusions

We presented a novel way to post-process gene prediction results based on constrained global optimization. Contrary to the heuristic approaches our approach provides an optimality guarantee – the final set of prediction will be the maximally scoring set that satisfies the imposed constraints. We have incorporated existing heuristic methods with the optimization procedure using inconsistency rules implemented in CHR. Currently, the approach has similar accuracy to the heuristic methods. The results indicate that maximizing the sum of scores have the effect of including more short predictions. This could be addressed weighting the scores by prediction length. We also plan to experiment with different constraints to achieve better accuracy. Our approach is limited to local overlap constraints and is not well-suited for global or long-distance constraints.

We are not the first to use dynamic programming based approaches to post-processing of gene predictions. Version 3 of Glimmer [5] use a custom dynamic programming algorithm which is similar to the present approach, but incorporates only the maximal overlap constraint. Another difference is that our approach is expressed as a declarative PRISM program and can therefore utilize the generalized Viterbi algorithm. Our approach is similar to constrained HMMs in PRISM, which has previously be applied to other biological sequence analysis tasks [2, 3]. A main difference is that we express constraints with CHR rules.

CHRiSM[15] already combines CHR and PRISM and is to our knowledge the first system to do so. CHRiSM assigns probabilistic semantics to CHR rules, which are interpreted as chance rules – e.g. even if a rule head is matched the rule is only applied with a certain probability. The main difference with our approach is that we use ordinary CHR rules in conjunction with a PRISM program, although ordinary CHR rules may be seen as a special case of CHRiSM rules, where the probability of invocation is one. Additionally, the form of the CHR rules we use is restricted (inconsistency rules) and they are only used in the constraint checking part of the PRISM program. It would be interesting to

use CHRiSM as a method of incorporating soft constraints with our approach, e.g. redefining the inconsistency rules as CHRiSM chance rules.

*Acknowledgement* This work is part of the project "Logic-statistic modeling and analysis of biological sequence data" funded by the NABIIT program under the Danish Strategic Research Council.

## References

[1] S. Abdennadher, T. Fruehwirth, and H. Meuss. On confluence of constraint handling rules. *Lecture Notes in Computer Science*, 1118:1–15, 1996.

[2] Henning Christiansen, Christian Theil Have, Ole Torp Lassen, and Matthieu Petit. A constraint model for constrained hidden markov models: a first biological application. In *Proc. of the International Workshop on Constraint Based Methods for Bioinformatics*, pages 19–26, Lisbon, Portugal, September 2009.

[3] Henning Christiansen, Christian Theil Have, Ole Torp Lassen, and Matthieu Petit. Inference with constrained hidden markov models in PRISM. *TPLP*, 10(4-6):449–464, 2010.

[4] Henning Christiansen, Christian Theil Have, Ole Torp Lassen, and Matthieu Petit. Bayesian Annotation Networks for Complex Sequence Analysis. In John Gallagher and Michael Gelfond, editors, *Technical Communications of the 27th International Conference on Logic Programming (ICLP'11)*, volume 11 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 220–230, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[5] Arthur L. Delcher, Kirsten A. Bratke, Edwin C. Powers, and Steven L. Salzberg. Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics*, 23:673–679, 2007.

[6] James W. Fickett and Chang-Shung Tung. Assessment of protein coding measures. *Nucl. Acids Res.*, 20(24):6441–6450, 1992.

[7] Thom W. Frühwirth. Constraint handling rules. In Andreas Podelski, editor, *Constraint Programming*, volume 910 of *Lecture Notes in Computer Science*, pages 90–107. Springer, 1994.

[8] Yoko Fukuda, Yoichi Nakayama, and Masaru Tomita. On dynamics of overlapping genes in bacterial genomes. *Gene*, 323:181 – 187, 2003.

[9] Anders Krogh, I. Saira Mian, and David Haussler. A hidden Markov model that finds genes in E.coli DNA. *Nucl. Acids Res.*, 22(22):4768–4778, 1994.

[10] Shaun Mahony, James O. McInerney, Terry J. Smith, and Aaron Golden. Gene prediction using the self-organizing map: automatic generation of multiple gene models. *BMC Bioinformatics*, 5:23, 2004.

[11] Anton M.Shmatkov, Arik A.Melikyan, Felix L.Chernousko, and Mark Borodovsky. Finding prokaryotic genes by the frame-by-frame' algorithm: targeting gene starts and overlapping genes. *Bioinformatics*, 15(11):874–886, 1999.

[12] S. Normark, S. Bergstrom, T. Edlund, T. Grundstrom, B. Jaurin, F. P. Lindberg, and O. Olsson. Overlapping genes. *Annual Review of Genetics*, 17:499–525, 1983.

[13] Taisuke Sato. Generative Modeling by PRISM. *Proceedings of the International Conference on Logic Programming*, LNCS 5649:24–35, 2009.

[14] Taisuke Sato and Yoshitaka Kameya. A viterbi-like algorithm and EM learning for statistical abduction, June 16 2000.

[15] Jon Sneyers, Wannes Meert, Joost Vennekens, Yoshitaka Kameya, and Taisuke Sato. CHR(PRISM)-based probabilistic logic learning. *TPLP*, 10(4-6):433–447, 2010.

[16] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.