Dynamic Structure Learning of Factor Graphs and Parameter Estimation of a Constrained Nonlinear Predictive Model for Oilfield Optimization

Hyokyeong Lee[1], **Ke-Thia Yao**[2], **and Aiichiro Nakano**[1]
[1]Department of Computer Science, University of Southern California, Los Angeles, CA, USA
(hyokyeol, anakano)@usc.edu
[2]Information Sciences Institute, University of Southern California, Marina del Rey, CA, USA
kyao@isi.edu

Abstract - *Injector-producer relationships (IPRs) are the key knowledge for oilfield optimization, i.e., maximizing oil production at the minimum operational cost. The difficulty associated with the field optimization is that the underlying reservoir structure is unknown and changes continuously over time. Inferring IPRs is a large-scale constrained nonlinear parameter estimation problem. The state-of-the-art hybrid constrained nonlinear optimization (HCNO) method provides excellent accuracy for solving this problem but with prohibitive computational costs for large oilfields. In this paper, we propose a dynamic structure learning and parameter estimation approach based on inference in a probabilistic graphical model named PETROGRAPH Learning (PGL). The learning is initiated by constructing an initial factor graph based on a locality principle and is guided by belief discrepancies, estimation error, and residual correlation analysis. At each iteration, the sum-product algorithm is applied to estimate the parameters, and the factor graph structure is refined as the input for the next round. The iterative learning continues until convergence. Experimental results and analysis show that PGL is scalable to the large scale of real oilfields with much less running time than that of HCNO while providing virtually exact solutions*.

Keywords: structure learning of factor graph, locality principle, belief discrepancies, large-scale constrained nonlinear optimization, factor graph and the sum-product algorithm

1 Introduction

The major task of every petroleum company is to optimize field performance, *i.e.*, to maximize oil production and to reduce operational costs. One of the popular oil recovery techniques is waterflooding, which injects water into injectors to extract oil. Here, the knowledge about injector-producer relationships (IPRs), *i.e.*, which injectors contribute to which producers, is the key for the field optimization. The difficulty associated with the optimization is that the underlying structure of oil reservoirs is unknown and it continuously changes over time. Recently, a predictive model called capacitance-resistive model (CRM) has been proposed to investigate the IPRs [1], [2]. The CRM is a nonlinear predictive model consisting of two sets of parameters: Connectivity and time constants. The connectivity parameters quantify contributions of injectors to producers and each time constant parameter defines the degree of fluid storage between an injector and a producer. Estimating the two sets of parameters is a large-scale constrained parameter estimation problem for the continuous nonlinear system of equations with constraints.

Recently, a hybrid constrained nonlinear optimization (HCNO) approach was developed for the CRM parameter estimation [3]. HCNO is based on sequential quadratic programming (SQP) in a line search framework. In HCNO, the constrained nonlinear time constant parameters are estimated so as to convert the constrained nonlinear system to a constrained linear system, and subsequently the remaining parameters in the constrained linear system are estimated by a constrained linear optimization method. Though HCNO outperforms the conventional SQP in terms of running time and prediction accuracy, there remain two major difficulties. First, the performance of HCNO highly depends on an initial guess. Second is the high computational cost of HCNO. Though not every injector influences every producer in real oilfields (*i.e.*, IPRs are sparse), HCNO examines complete connectivity, and thus its computational cost is unnecessarily high for estimating all the parameters including those for actually non-existing connections.

To achieve further scalability, we propose an approximation method of structure learning and parameter estimation based on inference in a probabilistic graphical model named PETROGRAPH Learning (PGL). In PGL, factor graphs are employed as a graphical language to represent the estimated IPRs, and the CRM parameters are estimated by applying the sum-product algorithm to the factor graphs. PGL initiates the learning of the factor graph structure by constructing an initial factor graph based on an inductive bias, *i.e.*, locality principle. After estimating the CRM parameters using the sum-product algorithm, the factor graph structure is updated through the investigation of belief discrepancies and

estimation errors, augmented with residual correlation analysis. The updated factor graph becomes the input for the next round of learning. The iterative learning process continues until convergence. The experimental results and analysis confirm the scalability of PGL and show that PGL outperforms HCNO in terms of running time for larger number of parameters with comparable (*i.e.* nearly exact) prediction accuracy.

The main contribution of this paper is the new nonlinear constrained parameter estimation method for oilfield optimization, which embodies a novel dynamic graph-structure learning approach to achieve a significant speedup over the current state-of-the-art while providing virtually exact solutions. This paper thereby provides the first scalable solution applicable to the large problem size of real oilfields. To the best of our knowledge, this is the first time that a constrained nonlinear system is represented by a graphical model, whose structure is dynamically learned, and the constrained nonlinear parameters are inferred by the probabilistic graphical model. The messages are shown to converge to a stable equilibrium over time in the loopy probabilistic graphical model as advocated in Ref. [13], while finding a satisfactory solution.

2 Problem Statement

The capacitance-resistive model (CRM) is a constrained nonlinear model that predicts the production rates for given injection rates. In CRM, the the estimated production rate of producer j at time step t_n is given by

$$\hat{q}_j(t_n) = \sum_{i=1}^{L}\sum_{k=1}^{n}(1-e^{-\Delta t_k/\tau_{ij}})\pi_{ij} i_i(k) e^{-(t_n-t_k)/\tau_{ij}}, \quad (1)$$

where L is the total number of injectors, Δt_k is the time interval between t_k and t_{k-1}, π_{ij} is the connectivity that specifies the fraction of injection rate of injector i flowing into producer j, $i_i(k)$ is the injection rate of injector i at time step k, and τ_{ij} is a time constant. The optimal CRM parameters are the ones that minimize the estimation error of production rates. Thus, the parameter estimation problem is to minimize the objective function,

$$f(x) = \sum_{j=1}^{J}\sum_{n=1}^{N}(q_j(t_n) - \hat{q}_j(t_n))^2, \quad (2)$$

where the parameters are collectively denoted by

$$x = (\pi_{11},\pi_{12},...,\pi_{1J},\pi_{21},...,\pi_{LJ},\tau_{11},\tau_{12},...,\tau_{1J},\tau_{21},...,\tau_{LJ}),$$

$q_j(t_n)$ is the actual production rate of producer j at time t_n, J is the total number of producers, and N is the total number of time steps. The training error is measured using the sum of square errors. The two sets of parameters are subject to the following constraints:

$$\sum_{j=1}^{J}\pi_{ij} = 1, \quad (3)$$

$$\pi_{lb} \le \pi_{ij} \le \pi_{ub}, \quad (4)$$

$$\tau_{lb} \le \tau_{ij} \le \tau_{ub}, \quad (5)$$

where $i = 1, 2, \ldots, L$, $j = 1, 2, \ldots, J$, $\pi_{lb} = 0$ and $\pi_{ub} = 1$. The lower bound τ_{lb} and the upper bound of τ_{ub} are specified by domain experts.

3 Petrograph Learning

3.1 Factor Graph and the Sum-Product Algorithm

A factor graph is a graphical model that visualizes the structure of the factorization of a complicated global function and dependency among variables [10]. When the complicated global function factors into a product of simpler local functions, computational efficiency can be derived by exploiting the factorization of the global function. For example, let $g(x_1, x_2, x_3, x_4, x_5)$ be a function of five variables, and suppose that g can be expressed as a product of five factors.

$$\begin{aligned} &g(x_1, x_2, x_3, x_4, x_5) \\ &\quad = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5) \end{aligned} \quad (6)$$

The factor graph for Eq. (6) is as follows:

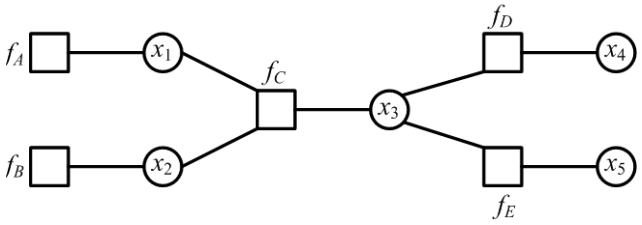


Fig. 1. Factor graph for Eq. (6)

The sum-product algorithm is a generic message passing algorithm operating in factor graphs and computes either exactly or approximately various marginal functions derived from the global function. The *summary* for x_2 by borrowing the notation from [10] is

$$g_2(x_2) = \sum_{\sim\{x_2\}} g(x_1,...,x_5), \quad (7)$$

where g_2 is the marginal function associated with $g(x_1,...,x_5)$ and the summation is over all the variables but x_2. The summary for x_2 can be rewritten as the product of

messages from neighboring nodes and each message consists of sum-of-products

$$g_2(x_2) = \mu_{f_B \to x_2}(x_2)\mu_{f_C \to x_2}(x_2), \quad (8)$$

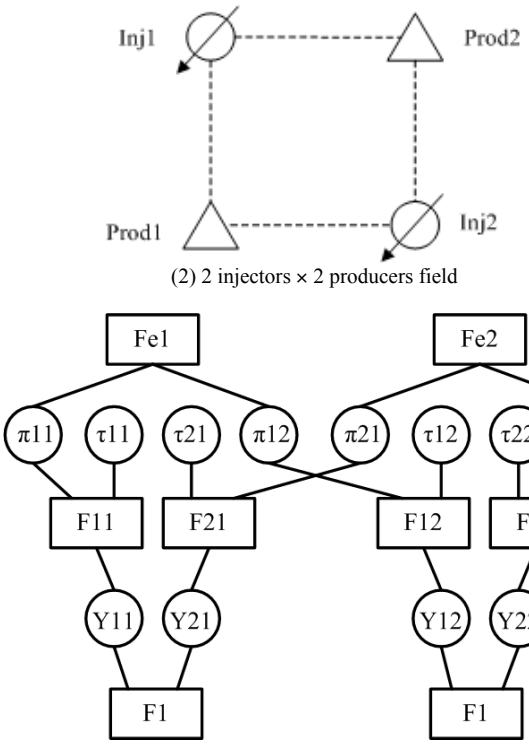
where $\mu_{f_B \to x_2}(x_2) = \sum_{\sim\{x_2\}} f_B(x_2) = f_B(x_2)$ and

$$\mu_{f_C \to x_2}(x_2) = \sum_{\sim\{x_2\}} (f_A(x_1) f_C(x_1, x_2, x_3) \cdot (\sum_{\sim\{x_3\}} f_D(x_3, x_4)) \cdot (\sum_{\sim\{x_3\}} f_E(x_3, x_5))).$$

3.2 Graphical Representation

We represent the objective function, Eq. (2), together with the constraints, Eqs. (3)-(5), by a factor graph. Fig. 2 shows an example of a simple field consisting of 2 injectors and 2 producers (2 × 2 field) along with the corresponding factor graph, where a circle with an arrow represents an injector, a triangle represents a producer, and dotted lines stand for true connectivity.

The factor graph in Fig. 2(b) consists of three types of variable nodes and three types of factor nodes. Fe_i (i = 1, 2) is a factor node for injector i to force the equality constraint to π_{ij} (j = 1, 2) parameters. Fe_i uses a joint probability table of uniform distribution. Y_{ij} is a variable node for the subproduction rate made by injector i and producer j. F_{ij} is a factor node with a joint probability table of the three variable nodes, π_{ij}, τ_{ij}, and Y_{ij}. The joint probability distribution for F_{ij} is uniform. F_j is a factor node with the joint probability table of the subproduction variable nodes for producer j. The joint probability table is computed using Boltzmann distribution that takes root mean squared error (RMSE) between the observed and the estimated production rates as input. Variable nodes for observable data are not presented in the factor graph, and thus variables nodes for injection and production rates are not included in the factor graph. Instead, injection and production rates are implicitly included in F_{ij} and F_j, respectively. The size of the joint probability table for F_j increases as the number of injectors connected to producer j increases, and discretization intervals become smaller. To maintain the table of a reasonable size, we keep only top-k entries with relatively higher probabilities based on the observation that the probabilities of most of the entries are very close to 0.



(2) 2 injectors × 2 producers field

(b) Factor graph for (a)

Fig. 2. 2 × 2 field and its factor graph

3.3 Structure Learning and Parameter estimation

In PGL, the construction of a factor graph and parameter estimation is dynamic and it consists of two phases: Belief propagation over the factor graph using the sum-product algorithm; and update of the factor graph by inserting nodes and edges. Algorithms 1 and 2 show the procedures. The initial factor graph is constructed based on a locality principle. The locality principle in the IPRs problem is that a producer is most likely influenced by the immediately neighboring injectors. In the initial factor graph, each producer is thus only connected to the closest injector based on the physical distance. Euclidean distance is used to calculate the distance between injectors and producers. The initial factor graph thus constructed could contain actually non-existing edges, but those edges are eventually identified by PGL. The learning starts with the initial factor graph. At the 1st round of learning, the sum-product algorithm is applied to compute the marginal probability distributions of the variable nodes. Once the messages converge after the 1st round, the injector-producer pairs missing actual edges need to be identified. A naive approach would be the exhaustive combinatorial search and to perform the message passing repeatedly until a stopping criterion is satisfied. However, this method does not scale to a large number of injectors. We instead employ a greedy forward search to avoid the very expensive computations. The search consists of following three steps:

- Step1: For every injector i and producers connected to the injector, determine whether the most probable value for π_{ij} identified by the message from Fe_i to π_{ij} is not the same as that identified by the message from F_{ij} to π_{ij}.
- Step2: For every producer j, determine whether there are producers whose estimation errors are not less than a prescribed error threshold.
- Step3: Among injectors selected in Step1 and producers selected in Step2, determine whether there are injector-producer pairs whose injection rates are positively and significantly correlated with the difference between the observed and the estimated production rates.

When there is any missing edge between injector i and producers, a belief discrepancy is observed such that the most probable value π_{ij} identified by the message from Fe_i to π_{ij} is different from that identified by the message from F_{ij} to π_{ij}, where j represents the indices of all producers that are connected to injector i. Algorithm 1 is the pseudocode that identifies those injectors.

Algorithm 1. findFreeInjectors($\pi Discrete$, $msgFe_i To\pi_{ij}$, $msgF_{ij}To\pi_{ij}$, L, J)
1. out = { }
2. for i = 1 to L
3. for each p connected to injector i
4. [$maxFe_i$, $idx1$] = max($msgFe_i To\pi_{ij}$ (i, p, :)
5. [$maxF_{ij}$, $idx2$] = max($msgF_{ij}To\pi_{ij}$ i, p, :)
6. if $\pi Discrete$($idx1$) > $\pi Discrete$($idx2$)
7. $out \leftarrow out \cup \{i\}$
8. return

A high estimation error of a producer implies that the producer has missed contributions from injectors that are supposed to be connected to it. Thus, producers with missing connections with injectors can be detected by investigating the estimation errors of those producers. Now we have found injectors and producers with missing edges, but we still do not know which injectors are supposed to be connected to which producers. The last observation is that producers with high estimation errors have residuals between the observed and the estimated production rates. The idea is that the residual indicates the lack of contributions from injectors for the producer, thus a positive correlation (*e.g.* ≥ 0.3) between the set of injection rates of injector i and the residual is used to determine the injector-producer pairs. Algorithm 2 shows the pseudocode for the update of the factor graph structure and parameter estimation.

Algorithm 2. PGL
1. **Input**:
2. $injData$, q, JPs
3. P: all producer ids
4. $\pi Discrete$, $\tau Discrete$: discrete values
5. **Output**:
6. $\pi Guess$, $\tau Guess$, G
7. **Begin**
8. $freeInjectors \leftarrow \{\}$, $freeProducers \leftarrow \{\}$
9. $numRound \leftarrow 0$, $done \leftarrow$ FALSE
10. [$connInjIds$, G] $\leftarrow$ getClosestInjectorInitG(P)
11. while ~$done$
12. if $numRound > 0$
13. $changed \leftarrow$ FALSE
14. for each pp in $freeProducers$
15. if ~isempty($freeInjectors$)
16. [$maxCorr$, $injId$] = max(corr($freeInjectors$, $injData$, $q(pp) - \hat{q}(pp)$))
17. if $maxCorr \geq$ TOL_CORR
18. $connInjIds(pp) \leftarrow connInjIds(pp) \cup \{injId\}$
19. $changed \leftarrow$TRUE
20. initializeMessages($connInjIds$, pp)
21. if ~$changed$
22. break
23. [$mp\pi_{ij}$, $mp\tau_{ij}$, $msgFe_i To\pi_{ij}$, $msgF_{ij}To\pi_{ij}$] $\leftarrow$ SumProductFlooding(JPs, G, TOL_DIFF)
24. [$\pi Guess$, $\tau Guess$] $\leftarrow$ computeExpectation($\pi Discrete$, $\tau Discrete$, $mp\pi_{ij}$, $mp\tau_{ij}$)
25. $freeInjectors \leftarrow$ findFreeInjectors($\pi Discrete$, $msgFe_i To\pi_{ij}$, $msgF_{ij}To\pi_{ij}$, L, P)
26. $freeProducers \leftarrow$ findFreeProducers($\pi Guess$, $\tau Guess$, $injData$, q, TOL_ERR)
27. update G by $\pi Guess$ and $\tau Guess$
28. $\hat{q} \leftarrow$ estimateProduction($\pi Guess$, $\tau Guess$, $injData$)
29. $numRound \leftarrow numRound + 1$
30. if max(calcTrainingError($\hat{q}$, q)) < TOL_ERR
31. $done \leftarrow$ TRUE

The factor graph is updated by adding new π_{ij}, τ_{ij}, Y_{ij} and F_{ij}, and updating Fe_i and F_j at each round. The partially updated factor graph is the input for the message passing at the next round. The messages for the updated factor graph are initialized for the next round as well. findFreeProducers() finds producers whose estimation errors are not less than TOL_ERR. The convergence criterion of the sum-product algorithm is that the maximum difference between two consecutive marginal probabilities of all π_{ij} and those of all τ_{ij} is not larger than the difference threshold (TOL_DIFF). The exit condition is any of the two criteria: (1) the maximum estimation error among producers is less than the error threshold (TOL_ERR); (2) there is no change between two consecutive sets of selected injectors. The goodness-of-fit of the estimated factor graph and the parameters is determined by the estimation error. Since the factor graph is a connected graph, flooding message passing scheme is used.

4 Experimental Results

4.1 Experimental Setting

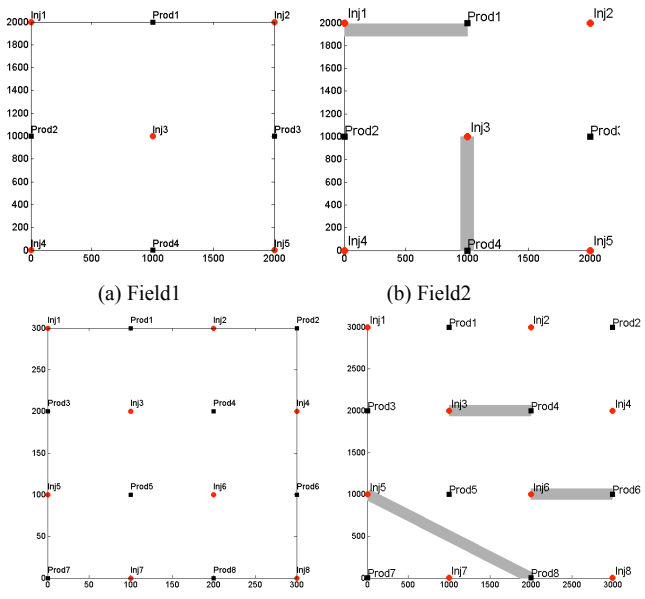
The performance of PGL is compared with HCNO (hybrid constrained nonlinear optimization) in Ref. [3]. Synthetic data are used to verify the accuracy of the solution

using the ground truth, *i.e.*, location of higher permeability. In this way, we can eliminate the uncertainty in deriving π_{ij} and τ_{ij} parameters from either real oilfield or reservoir simulation data. The prediction accuracy is measured by RMSE for the test data, running time, and accuracy of locating injector-producer connectivity. 80% of data is used for training, while the remaining 20% is used as the test data. The joint probability tables for the factor nodes are generated using the training data. Table I shows the field name, field type, field size (number of injectors × number of producers), the number of time steps, and the number of parameters to be estimated.

Table I. Field name, field type, size, number of time steps, and number of parameters

Field	Type	Size	TS	Param
Field1	Homogeneous	5 × 4	160	40
Field2	Heterogeneous	5 × 4	160	40
Field3	Homogeneous	8 × 8	160	128
Field4	Heterogeneous	8 × 8	160	128

Fig. 3 shows the field configuration of the four fields. The (red) circles stand for injectors and the (black) squares do for producers. The grey lines in Field2 and Field4 indicate the higher permeability, whereas the permeability in Field1 and Field3 is uniform over the field. In the homogeneous fields (Field1 and Field3), the contribution (*i.e.* connectivity) of an injector is evenly distributed for all the neighboring producers and the injector has almost no influence on distant producers. The connectivity between the injector-producer pairs with the higher permeability is stronger than that of any other pairs. π_{lb} is 0 and π_{ub} is 1, and the discretization interval for π_{ij} is 0.1 and that for τ_{ij} is set to 4. τ_{lb} and τ_{ub} are set to 0.0001 and 20, respectively. For HCNO, randomly generated 10 initial guesses are used. A set of initial guesses is used for Field1 and Field2, and the other set of initial guesses is used for Field3 and Field4.



(a) Field1 (b) Field2

(c) Field3 (d) Field4

Fig. 3. Well locations and true higher permeability (grey lines) of four fields

4.2 Results

The performance of the two algorithms, HCNO and PGL, is compared in terms of running time and test error. The running time of HCNO is the search time until convergence, and that of PGL is the message passing and edge search time until an exit condition is satisfied. As a comparison, we also include some results for the most commonly used base algorithm, conventional sequential quadratic programming (SQP) in line search framework, which is the baseline of HCNO as described in Ref. [3].

Table II shows the running time of HCNO and PGL, where HCNO(mean) is the average running time of the 10 initial guesses.

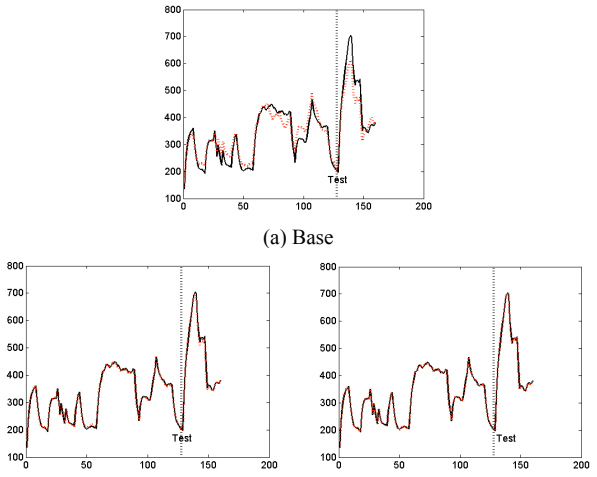
Table II Running times (Sec) of HCNO and PGL

Field	HCNO (Mean)	PGL
Field1	3.26	10.16
Field2	2.60	2.68
Field3	802.32	261.07
Field4	867.79	69.54

We observe that the running time of PGL is much less than that of HCNO for larger number of parameters. For even larger fields, the scalability of HCNO and PGL can be analyzed as follows. The complexity of HCNO is determined by the convergence rate, the number of quadratic programming (QP) subproblem iterations, and the number of constrained linear least squares iterations. Let n be the number of injectors and m be the number of producers. In practical situations, m is on the order of n, and we set $m = n$ in the asymptotic analysis. The number of QP subproblem iterations per SQP iteration is mainly determined by the number of inequality constraints because an active-set method is used to solve the QP subproblem. Thus, the number of QP subproblem iterations per SQP iteration is $2n^2$. The number of constrained linear least squares iterations is $2n^2 + n$. The number of SQP iterations is mainly determined by the convergence rate. quasi-Newton methods typically converge Q-superlinearly, thus the number of SQP iterations is determined by the convergence rate. The worst case complexity of HCNO is thus ln^2, where l is the number of SQP iterations determined by the convergence rate. Therefore, the complexity of HCNO is $O(ln^2)$. On the other hand, PGL begins with disjoint $O(n)$ subgraphs, and for practical sparse graphs in which the number of edges is $O(n)$, the iterative learning of the factor graph structure should terminate after $O(k)$ iterations assuming that a new edge per injector is added per iteration, where k is the maximum degree of any node. For real physical large reservoirs, we expect k to be a small constant $k << n$. Consequently, the complexity of PGL can be made $O(kn)$. We thus expect that PGL is scalable to the large problem size of real oilfields. It should also be noted that PGL based on a locality principle

and message passing is readily amenable to parallel computing.

To demonstrate the accuracy of the proposed method, Fig. 4 shows the observed and estimated production curves by the three methods: Base algorithm, HCNO, and PGL.



(a) Base (b) HCNO (c) PGL

Fig. 4. Observed and estimated production curves of Producer7 in Field4

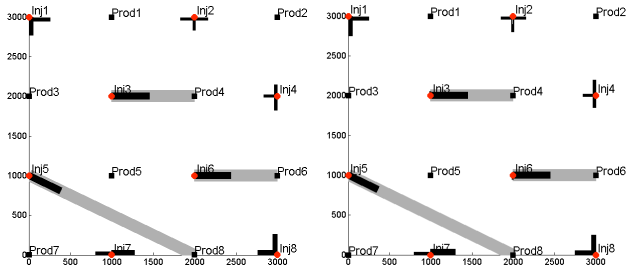
In Fig. 4, the (black) solid curves stand for the observed production rates and the (red) dotted curves stand for the estimated production rates. The vertical lines indicate the separation between training and prediction curves. The portions of the curves to the left of the vertical lines are estimations by training and those to the right are prediction curves. Each figure represents the overall training and prediction curves across all producers by each method. The three figures clearly show that the training and test errors of the base algorithm are significantly higher than those of HCNO and PGL, while there is no significant difference in the test errors of HCNO and PGL, *i.e.*, both PGL and HCNO predictions are virtually indistinguishable from the ground-truth.

Table III shows that the test errors of HCNO and PGL are comparable and are orders-of-magnitude smaller than the most commonly used base SQP algorithm.

Table III Test Error (RMSE)

Field	Base (Mean)	HCNO (Mean)	PGL
Field1	8308.90	24.64	18.76
Field2	9208.90	44.70	82.93
Field3	1402.56	44.01	43.92
Field4	552.57	147.64	33.86

Fig. 5 shows that true higher permeability and the estimated permeability for Field4.



(a) HCNO (b) PGL

Fig. 5. True higher permeability and estimated permeability

The grey lines in Fig. 5 stand for the true higher permeability and the black lines stand for the estimated permeability. The length and width of the black lines indicate the magnitude of the connectivity parameters. As shown in the figure, the true permeability is also well estimated by both HCNO and PGL methods. The contribution of the locally connected injectors is evenly distributed among the neighboring producers and the higher permeability across wells is also well estimated.

5 Related Work

Much research has been done in structure learning of Markov network (MN) or Markov random field (MRF) and Bayesian network (BN). We here focus on the structure learning directly related to factor graphs and parameter estimation. Factor graphs subsume both MNs and BNs in the sense that every MN or BN can be written as a factor graph of the same size [4]. Empirical entropy estimates were used to select an approximate Markov blanket (MB) for each variable, and then the parameter estimation algorithm was used to estimate parameters and identify which factors were likely to be irrelevant. It was shown that the class of factor graphs with bounded factor size and bounded connectivity could be learned in polynomial time and polynomial number of samples with the assumption that the data was generated by a network in the class. Mirowski and LeCun proposed a method for training dynamic factor graphs (DFG) with continuous latent state variables [12]. A DFG included factors modeling joint probabilities between hidden and observed variables, and factors modeling dynamical constraints on hidden variables. Given a training observation sequence, the optimal state sequence was found by minimizing the energy using a gradient-based minimization method. Second, the parameters of the model were updated using a gradient-based procedure so as to decrease the energy. The two steps were repeated over all training sequences.

Steepest descent was used in estimating parameters in factor graphs [5]. By showing that steepest descent can be combined with the sum-product algorithm, they demonstrated that steepest descent would be elegantly combined with other summary propagation algorithms. Korl and Loeliger estimated the autoregressive parameters using the sum-product algorithm and factor graphs [9]. A state-space form was derived for the AR model, and then the messages were propagated over the corresponding factor graph. A parallel

inference algorithm on large factor graphs was proposed in the distributed memory setting of computer clusters [6], [7]. Graph-partitioning was incorporated to distribute the work. The vertex with highest residual was updated because updating a vertex with no residual wastes process cycles. A parallel implementation was designed for the loopy belief propagation algorithm for factor graphs [11]. They found three main parameters susceptible to be defined by the user; scheduling policies, stopping criteria, and initial message values. The scheduling policies are a rule-based scheduling such that a node sends messages to the neighbor nodes after receiving a fixed number of messages or receiving messages from all the nodes identified in *SndSet*. The stopping criteria are that the algorithm stops after certain number of iterations or a maximum given number of messages. The user can manually define the function values for each factor node. Electromyographic (EMG) signals were decomposed into its components using factor graph and the sum-product algorithm [8]. The discrete-time model of EMG signals were represented by a factor graph and the sum-product algorithm operated by passing messages forward and backward in the loopy factor graph.

6 Conclusions

We have developed an approach for dynamic construction of graphs and parameter estimation using factor graphs and the sum-product algorithm. Instead of starting with a fully connected graph for the given constrained nonlinear system, the graph structure was built incrementally by learning from belief discrepancies, estimation error, and correlation analysis. As the number of wells increased, the running time of PGL became less than that of the state-of-the-art HCNO method. The prediction accuracy of PGL was comparable to that of HCNO and was virtually indistinguishable from the ground-truth. The three regularizations prevented the greedy forward search from including actually non-existing edges in the factor graph. We have demonstrated that the approach of the probabilistic inference in the loopy graph structure was successful in estimating parameters of the constrained nonlinear model.

7 Acknowledgment

This work was partially supported by the Center for Interactive Smart Oilfield Technologies at the University of Southern California.

8 References

[1] M. Sayarpour, E. Zuluaga, C. S. Kabir, L. W. Lake, “The Use of Capacitance-Resistive Models for Rapid Estimation of Waterflood Performance and Optimization”. SPE Annual Technical Conference and Exhibition, 2007.

[2] M. Sayarpour, “Development and Application of Capacitance-Resistive Models to Water/CO2 Floods”. PhD. Thesis.

[3] H. Lee, K. Yao, O. Okpani, A. Nakano, I. Ershaghi, “Hybrid Constrained Nonlinear Optimization to Infer Injector-Producer Relationships in Oil Fields”. Int’l J. Computational Science, vol. 4, no. 1, pp. 1-22, 2010.

[4] P. Abbeel, D. Koller, A. Y. Ng, “Learning Factor Graph in Polynomial Time & Sample Complexity”. Uncertainty in Artificial Intelligenc, 2005.

[5] J. Dauwels, S. Korl, and H.-A. Loeliger, “Steepest Descent on Factor Graphs”. IEEE ITSOC Information Theory Workshop on Coding and Complexity, 2005.

[6] J. E. Gonzalez, Y. Low, C. Guestrin, D. O'Hallaron, “Distributed Parallel Inference on Large Factor Graphs”. Uncertainty in Artificial Intelligence, 2009a.

[7] J. Gonzalez, Y. Low, C. Guestrin, “Residual Splash for Optimally Parallelizing Belief Propagation”. J. Machine Learning Research, vol. 5, pp. 177-184, 2009b.

[8] V. M. Koch and H.-A. Loeliger, “Decomposition of Electromyographic Signals by Iterative Message Passing in a Graphical Model”. Conf. IEEE Engineering in Medicine and Biology Society, 2004.

[9] S. Korl, H.-A. Loeliger, “AR Model Parameter Estimation: From Factor Graphs to Algorithms”. Int'l Conf. Acoustics, Speech, and Signal Processing, 2004.

[10] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor Graphs and the Sum-Product Algorithm”. IEEE Trans. Information Theory, vol. 47, no. 2, pp. 498-519, 2001.

[11] A. Mendiburu, R. Santana, J. A. Lozano, E. Bengoetxea, “A Parallel Framework for Loopy Belief Propagation”. Workshop on Parallel Bioinspired Algorithms in conjunction with Genetic and Evolutionary Computation Conference, 2007.

[12] P. Mirowski, Y LeCun, “Dynamic Factor Graphs for Time Series Modeling”. European Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2009.

[13] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, 1988.