A Genetic Algorithm to Solve Master Planning Problems in Semiconductor Manufacturing

Conference Paper · January 2009

CITATION
1

READS
279

2 authors:

Thomas Ponsignon
Infineon Technologies
53 PUBLICATIONS **284** CITATIONS
SEE PROFILE

Lars Mönch
FernUniversität in Hagen
223 PUBLICATIONS **3,499** CITATIONS
SEE PROFILE

Proceedings of the 2009 Industrial Engineering Research Conference

A Genetic Algorithm to Solve Master Planning Problems in Semiconductor Manufacturing

Thomas Ponsignon
Infineon Technologies AG
85579 Neubiberg, Germany

Lars Mönch
Department of Mathematics and Computer Science
University of Hagen, 58097 Hagen, Germany

Abstract

We present a master planning problem that arises in semiconductor manufacturing. The problem consists in determining appropriate wafer quantities for several products, facilities, and periods of time. Different demand types, i.e. confirmed orders and forecasts, are considered. We use a combined objective function that takes production costs for in-house locations, subcontracting costs, inventory costs, and costs due to unmet demand into account. Demand fulfillments and capacity constraints are considered. We present a genetic algorithm to solve the master planning problem heuristically. The results of some computational experiments are presented. We compare the results for small size test instances with results obtained from a commercial MIP solver. The genetic algorithm is able to produce solutions with reasonable quality in little time.

Keywords
Master Planning, Genetic Algorithms, Mixed Integer Programming, Semiconductor Manufacturing, Routing

1. Introduction
Our research is related to planning problems found in semiconductor manufacturing. The master planning is one of the major mid-term production planning activities. Its outcome is a weekly-released master plan that depicts the production quantities of different products to be completed within the next half-year in the fabrication network. It transforms aggregated sales and operations plans, considering product families and long time horizons, into disaggregated plans for individual final products [12]. It is also an important entry point for lower production scheduling and control levels within the different fabrication sites. The master plan is sometimes called master production schedule (MPS).
The latest emergence of global production networks, which offer many manufacturing options and more flexibility, emphasizes the importance of accurate master plans. However, master planning problems for semiconductor manufacturing are rarely discussed in the literature. Some papers discuss questions of capacity planning for semiconductor wafer fabrication facilities [1, 2]. But the planning horizon is longer than for master planning (i.e. one or two years) and it is worked on an aggregated level based on product families for only one semiconductor wafer fabrication site. An enterprise-wide semiconductor manufacturing strategic resource planning approach based on mixed integer programming is presented in [11]. An aggregated model is used since the product family level of detail is applied. The suggested model shows some similarities with our model because it explicitly considers the network structure of facilities in semiconductor manufacturing. A more detailed model for one front-end facility is described in [6]. A linear programming formulation and iterative discrete-event simulation are used to determine start rates for wafers of the different products. The support of planning processes in semiconductor manufacturing by SAP APO is discussed in [8]. However, a detailed master planning is not included.
Based on this literature review, to our best knowledge master planning problems are not addressed so far for supply networks in the semiconductor industry, except in [10]. This is surprising because developing master planning procedures is an active field of research for other type of manufacturing systems ([14, 15] amongst others). In a former publication, we presented a MIP formulation to solve master planning problems [10]. But because of the large computational burden of this method, we have to look for efficient heuristics. In this paper, we introduce a genetic algorithm. We compare the results of both methods.

The paper is organized as follows. In Section 2, we describe the problem and we discuss very briefly the MIP formulation. We suggest an appropriate genetic algorithm in Section 3. The results of computational experiments are discussed in Section 4. Finally some ideas for future work are presented in Section 5.

2. Problem Description and Mathematical Model

In this section, we describe the researched problem, and we present a mathematical model formulation already introduced in more detail in a former publication [10].

Semiconductor manufacturing consists of front-end operations, performed in wafer fabrication facilities, and of back-end operations. In this paper, we focus on the first stage, and we want to determine the appropriate wafer quantities for different facilities (in-house sites or sub-contractors) and different periods of time. It is assumed that the weekly demand consists of confirmed orders and additional forecasts. The confirmed orders have to be fulfilled as much as possible to avoid backlog quantities, while the forecasts are fulfilled when capacity is available to generate additional revenues. An inventory is used to store finished goods for later fulfillment of demand.

The capacity modeling is a crucial point for master planning. In our model, we assume fixed average cycle times of the products. Given the completion time of a wafer, we calculate when a certain wafer will arrive at the bottleneck work centers. And then we accumulate the time that the wafer spends on processing on the machines of the bottlenecks for each period of time. This allows us to take the reentrant flows into account. This method is similar to capacity representation approaches used for capacity planning in semiconductor manufacturing [1, 2].

In our model, we assume to have p_{max} products that can be processed in m_{max} facilities (composed of ih_{max} in-house locations and sc_{max} subcontractor sites). The number of bottleneck work centers associated to facilities is represented by b_{max}. We use one week as length of a period. The quantity t_{max} refers to the number of periods (i.e. 24 weeks). The time index k is used for the capacity consumption, and k_{max} is obtained by reducing the maximum expected cycle time of all products by one period of time. The decision variables and the model parameters are summarized in Tables 1 and 2.

Table 1: Decision Variables

x_{pmt}	number of wafers of products p to be completed at the end of period t in facility m	I_{pt}	inventory level of product p at the end of period t
$s_{pt}^{(o)}$	sales quantity of product p in period t referring to confirmed orders	B_{pt}	backlog of product p at the end of period t referring to confirmed orders
$s_{pt}^{(fc)}$	sales quantity of product p in period t referring to additional forecasted demand	u_{pmt}	binary variable taking the value 1 when product p is processed in facility m in period t and the value 0 otherwise

Table 2: Model Parameters

$d_{pt}^{(o)}$	confirmed orders for product p at the end of period t (in wafers)	mc_{pm}	cost to produce one wafer of product p in facility m
$d_{pt}^{(fc)}$	additional forecasted demand for product p at the end of period t (in wafers)	lc_{pm}	location cost per period when product p is processed in facility m (i.e. fixed costs)
$C_{bt}^{\min}$	minimum utilization of bottleneck b in period t (in hours)	hc_p	inventory cost for holding one wafer of product p during one time period
$C_{bt}^{\max}$	maximum capacity offer at bottleneck b in period t (in hours)	udc_p	cost due to unmet confirmed orders for one wafer of product p
I_{p0}	initial inventory level of product p at the beginning of the first period	rev_{pt}	expected revenue per wafer for satisfying additional demand of product p in period t
B_{p0}	initial backlog of product p at the beginning of the first period	q_{pm}	average cycle time for wafer of product p processed in facility m
x_{pmt}^{i}	number of wafers of product p to be completed at the end of period t in facility m (started before the first period)	cc_{pmbk}	capacity consumption of one wafer of product p at bottleneck b when this product is processed in facility m and the completion period is k periods ahead

We maximize the objective function (1):

$$\sum_{p=1}^{p_{\max}}\sum_{t=1}^{t_{\max}}\left\{rev_{pt}s_{pt}^{(fc)} - hc_p I_{pt} - udc_p B_{pt} - \sum_{m=1}^{m_{\max}} mc_{pm}x_{pmt} - \sum_{m=1}^{m_{\max}} lc_{pm}u_{pmt}\right\}. \quad (1)$$

It is the overall difference between revenues for fulfilling additional forecasted demand and manufacturing, location, inventory holding, and backlog costs.

The objective function is subject to the following constraints. First, we have to model the inventory balance. We obtain:

$$I_{pt} = I_{pt-1} - s_{pt}^{(o)} - s_{pt}^{(fc)} + \sum_{m=1}^{m_{\max}} x_{pmt} + \sum_{m=1}^{m_{\max}} x_{pmt}^{i}, \qquad \forall p = 1,..,p_{\max}, \forall t = 1,..,t_{\max}. \quad (2)$$

We have to relate the sales quantities to the demand and backlog quantities. We get the following constraints:

$$s_{pt}^{(o)} + B_{pt} = d_{pt}^{(o)} + B_{pt-1}, \qquad \forall p = 1,..,p_{\max}, \forall t = 1,..,t_{\max}, \quad (3)$$

$$s_{pt}^{(fc)} \le d_{pt}^{(fc)}, \qquad \forall p = 1,..,p_{\max}, \forall t = 1,..,t_{\max}. \quad (4)$$

The capacity restrictions are modeled as follows:

$$C_{bt}^{\min} \le \sum_{p=1}^{p_{\max}}\sum_{m=1}^{m_{\max}}\left(\sum_{j=t}^{\min(t+k_{\max},t_{\max})} cc_{pmbj-t}\left(x_{pmt} + x_{pmt}^{i}\right)\right) \le C_{bt}^{\max}, \qquad \forall b = 1,..,b_{\max}, \forall t = 1,..,t_{\max}. \quad (5)$$

Whenever a unit of product *p* is completed in facility *m*, then location costs appear (6). It makes sure that an additional facility is only used when it is necessary. We introduce a large number α to apply the "Big M" method. We obtain:

$$x_{pmt} \le \alpha \cdot u_{pmt}, \qquad \forall p = 1,..,p_{\max}, \forall m = 1,..,m_{\max}, \forall t = 1,..,t_{\max}. \quad (6)$$

Non-negativity and binary conditions have to be taken into account for the decision variables. We obtain:

$$x_{pmt} \ge 0, s_{pt}^{(o)} \ge 0, s_{pt}^{(fc)} \ge 0, I_{pt} \ge 0, B_{pt} \ge 0, \quad \forall p = 1,..,p_{\max}, \forall m = 1,..,m_{\max}, \forall t = 1,..,t_{\max}, \quad (7)$$

$$u_{pmt} \in \{0,1\}, \qquad \forall p = 1,..,p_{\max}, \forall m = 1,..,m_{\max}, \forall t = 1,..,t_{\max}. \quad (8)$$

3. Genetic Algorithm Based Heuristic

We show in [10] that we are able to solve small size problem instances (i.e. 50 products) within 30 minutes with the presented MIP model, but it is not useable for larger instances. Thus, we have to look for efficient heuristics to reduce the computational effort. Genetic algorithms are widely used to solve large scale combinatorial optimization problems [9], especially in manufacturing. However, the more constraints the problem has, the more difficult it is to ensure the feasibility of obtained solutions. Therefore, it is crucial to design an appropriate chromosome representation, initialization schemes, and genetic operators. In this section, we suggest a suitable genetic algorithm to solve our problem.

3.1 Representation

Each chromosome of the population represents the production quantities x_{pmt} of one feasible solution to a problem instance. The specification of the data structure of chromosome *C* depends on the number of products p_{max}, the number of facilities m_{max}, and the number of time periods t_{max}:

$$C[p,m,t] = x_{pmt}, \qquad \forall p = 1,..,p_{\max}, \forall m = 1,..,m_{\max}, \forall t = 1,..,t_{\max}. \quad (9)$$

All other decision variables (sales quantities, inventory level, backlog, and binary variables) are obtained from the production quantities encoded in chromosome *C* by using the following heuristic procedure derived from (2) to (8):

$$s_{pt}^{(o)} = Min\left(d_{pt}^{(o)} + B_{pt-1}, \sum_{m=1}^{m_{\max}} x_{pmt} + \sum_{m=1}^{m_{\max}} x_{pmt}^{i} + I_{pt-1}\right), \qquad \forall p = 1,..,p_{\max}, \forall t = 1,..,t_{\max}, \quad (10)$$

$$s_{pt}^{(fc)} = Min\left(d_{pt}^{(fc)}, \sum_{m=1}^{m_{\max}} x_{pmt} + \sum_{m=1}^{m_{\max}} x_{pmt}^{i} + I_{pt-1} - s_{pt}^{(o)}\right), \qquad \forall p = 1,..,p_{\max}, \forall t = 1,..,t_{\max}, \quad (11)$$

$$I_{pt} = I_{pt-1} + \sum_{m=1}^{m_{\max}} x_{pmt} + \sum_{m=1}^{m_{\max}} x_{pmt}^{i} - s_{pt}^{(o)} - s_{pt}^{(fc)}, \qquad \forall p = 1,..,p_{\max}, \forall t = 1,..,t_{\max}, \quad (12)$$

$$B_{pt} = d_{pt}^{(o)} + B_{pt-1} - s_{pt}^{(o)}, \qquad \forall p = 1,..,p_{\max}, \forall t = 1,..,t_{\max}, \quad (13)$$

$$x_{pmt} \le \alpha \cdot u_{pmt}, \qquad \forall p = 1,..,\ p_{\max}, \forall m = 1,..,\ m_{\max}, \forall t = 1,..,\ t_{\max}. \tag{14}$$

We can easily prove that the solutions obtained with this procedure are feasible. Note that it is important to determine at first the sales quantities referring, respectively, to confirmed orders and to forecasts in order to ensure the fewer backlogs and the more revenues.

3.2 Initialization

The chromosomes of the initial population are formed by random values taking into account the capacity restriction (5). All other constraints are not strict quantity limitations, as inventory level and backlogs are not subject to a maximum bound. As a result, all initial chromosomes obtained are feasible solutions of the problem instances.

To ensure a heterogeneous initial population, we use three different schemes. The first scheme is a simple allocation algorithm generating rather good individuals. First of all, the product with the highest backlog cost or the highest revenue is selected. Then, the facilities with the most resting capacity or the fewest manufacturing costs are chosen. Afterwards, the quantity referring to the demand of the selected product for the current time period is assigned to as less facilities as possible in order to minimize the location cost. Note that confirmed orders are allocated with a higher priority than forecasts. Moreover, minimum and maximum capacity limits are strictly respected. Finally, we repeat this procedure until all products and time periods have been considered.

The second and third schemes are used to generate more average solutions. The second and third schemes, respectively, assign random quantities to products and facilities arbitrarily chosen until the minimum or maximum capacity limit of each bottleneck is reached. The distribution of the initial chromosomes generated by the first, the second and the third schemes is: 10%, 45% and 45%, respectively. Furthermore, we use a criterion based on a distance function to ensure that individuals are widely spread in the solution space.

Note that our initialization is based on constructive algorithms. Hence it is faster than trial-and-error schemes, which check only constraint violations and produce new samples if necessary.

3.3 Genetic Operators

From two given chromosomes C_1 and C_2, we obtain two offspring chromosomes C_3 and C_4 by applying the crossover operator. We use the scheme suggest in [5] to carry out the crossover:

Step 1: Choose $\lambda \sim U(0,1)$.

Step 2: $C_3[p,m,t] = \lambda C_1[p,m,t] + (1-\lambda) C_2[p,m,t],$ (15)

$$C_4[p,m,t] = (1-\lambda) C_1[p,m,t] + \lambda C_2[p,m,t], \forall p = 1,..,\ p_{\max}, \forall m = 1,..,\ m_{\max}, \forall t = 1,..,\ t_{\max}. \tag{16}$$

The genes of the offspring chromosomes are convex combinations of the corresponding genes in the parent chromosomes. As the set determined by (5) is convex, an arbitrary convex combination of feasible solutions is also a feasible solution.

Due to the nature of this crossover operator, it is obvious that widely spread individuals allow investigating a larger search space and lead to an enhanced optimization rather than with homogenous chromosomes. That is the reason why we implement different initializing schemes generating extreme cases.

We chose the dynamic mutation operator suggested in [9], which allows us to continuously decrease the impact of the mutation with an increasing number of generations. It starts with determining a gene of chromosome *C* that has to be mutated. In order to ensure the feasibility of mutated solution, an upper bound *ub*, representing the maximum remaining capacity, is calculated for the value of the gene. Furthermore, as a product may go through several bottleneck work centers, we have to check all capacity utilizations in order to find the most restrictive limit.

3.4 Fitness and Local Search

The fitness of chromosomes is evaluated based on the objective function (1). Moreover, sigma truncation scaling described in [13] is applied to normalize the fitness of individuals. Note that we chose this scaling function because it allows negative fitness that occurs when the sum of all costs is higher than the revenues.

In order to enhance the optimization of solutions, we implement a simple local search algorithm. It takes place every hundred generations, only for the ten best chromosomes. First of all, a product p_1 with high revenue and a product p_2 with low backlog cost are chosen. For all time periods, we interchange small amounts of production quantities x_{pmt} from p_1 to p_2, and we check if the modification improves the fitness of the chromosome. If the fitness increased, the modification is kept; otherwise the chromosome is not changed. Of course, modifications violating the capacity restrictions are discarded. The procedure is repeated until all products have been considered.

3.5 Implementation

The suggested genetic algorithm is implemented by means of the object-oriented framework GALib 2.47 [13] using the C++ programming language. The presented chromosome representation is coded by using the template class GA3DArrayGenome. The genetic algorithm stops whenever the maximum number of generations is reached (i.e. 1500 generations) or when the diversity of population falls below a defined threshold (i.e. 0.001%). We accomplished extensive computational experiments in combination with a trial-and-error strategy to select the parameters of the genetic algorithm (population size 300 individuals, crossover rate 0.75). In order to assess the performance of the suggested genetic algorithm, we used the MIP solver ILOG CPLEX 11.1 as presented in [10].

4. Results of Computational Experiments

In this section, we assess the performance of the suggested genetic algorithm by comparing its outcomes with the results of the MIP solver for different stochastically generated test instances.

4.1 Design of Experiments

We used a factorial design with three factors to generate problem instances. The number of products p_{max}, the number of in-house facilities ih_{max} and the number of sub-contractors sc_{max} are selected as factors. Each of these factors varies in two levels. We use $p_{\max} \in \{50,100\}$, $ih_{\max} \in \{2,4\}$, and $sc_{\max} \in \{6,8\}$.

The definition of other model parameters is detailed in [10]. We obtain eight possible factor combinations. For each of these combinations, four instances are generated and solved by the genetic algorithm and the MIP solver. The 32 test instances are solved on a Pentium M Processor (1.7 GHz, 1.0 GB RAM).

4.2 Results of Computational Experiments

The solution quality is measured as the ratio of the objective function value given by the genetic algorithm and the corresponding objective function value obtained with the MIP solver. This measure is called GA/MIP ratio and it represents the decrease in percents of the objective function value by using a heuristic algorithm instead of the exact solution procedure. All result ratios are grouped according to the factor levels of the design of experiments. The average value is shown Table 3.

The average MIP gap given by the MIP solver is also shown in Table 3. It indicates the ratio between the best feasible solution found during the branch-and-bound procedure and the best node amongst the remaining solutions (not necessary feasible) not explored at the end of the given computational time. This percentage gives an indication on how far the solution given by the MIP solver is from a hypothetical optimal solution. The computational time for the MIP solver was set to 30 minutes, while the genetic algorithm requires between 2 and 9 minutes (including the time for the initialization) to converge to a solution.

Table 3: Average GA/MIP Ratio, Average MIP Gap and Computation Time for Different Factor Levels

Number of Products p_{max}	**Number of Locations**		**Average GA/MIP Ratio**	**Average MIP Gap**	**Computation Time**	
	ih_{max}	**sc_{max}**			**MIP Solver**	**GA**
50	6	2	89.04%	3.71%	30 min	2 min
50	6	4	89.77%	2.86%	30 min	2 min
50	8	2	91.24%	2.23%	30 min	3 min
50	8	4	88.41%	1.96%	30 min	4 min
100	6	2	85.45%	14.35%	30 min	6 min
100	6	4	75.49%	9.89%	30 min	8 min
100	8	2	89.23%	8.98%	30 min	9 min
100	8	4	82.75%	6.59%	30 min	9 min
Overall			**86.42%**	**6.32%**	**30 min**	**5 min**

The overall average GA/MIP ratio shows that the genetic algorithm provides an objective function value inferior by nearly 13.5% to the value obtained with the exact solution procedure in only on sixth of the time needed by the MIP solver. Some additional experiments for very small size problem instances (i.e. 5 products) lead to a GA/MIP ratio of around 98%. This indicates that the designed genetic algorithm is able to find near-to-optimal solutions.

As indicated by the MIP gap and by [10], the difficulty to find near-to-optimal solutions or to prove optimality seems to be mainly affected by the number of products (all experiments with 50 products have MIP gap of less than 4%, while all experiments with 100 products have a MIP gap of at least 6.5%). Thus, it is interesting to investigate

the behavior of the genetic algorithm for large scale problems (several hundreds of products). Even if the results provided by the genetic algorithm are not optimal for small scale problems, it does make sense to use it rather than the MIP solver due to the gain of time.

5. Conclusions and Outlook to Future Research

We presented a genetic algorithm for master planning in semiconductor wafer fabrication networks. The suggested problem formulation considers demand in form of confirmed orders and forecasts. The objective function is the difference between revenues obtained for fulfilling forecasted demand and overall costs. Furthermore, we consider location costs that are basically fixed production costs for facilities where a product is processed. We take the finite capacity of the semiconductor manufacturing system into account during the calculation of master plans.

We introduced a genetic algorithm representing the production quantities, and we discussed how to derive the other decision variables. Then, we emphasized the necessity to design appropriate initialization schemes that generate feasible heterogeneous individuals. We described the crossover and mutation operators.

Some computational experiments were performed with the MIP solver and the genetic algorithm. We compared the results in terms of solution quality and of computation time. We demonstrated that the heuristic algorithm converges faster than ILOG, but it provides results with slightly lower quality. We think that the performance of the genetic algorithm can be increased by changing the chromosome representation and incorporating the sales quantities into the genetic algorithm. It allows for avoiding the determination of other main decision variables based on x_{pmt}. Furthermore, it seems possible to provide other heuristics based on decomposition schemes (product-, time- or location-based decomposition in several easier sub-problems). In addition, it is necessary to replace our assumption of fixed average cycle times of products in the problem formulation by a more sophisticated method. In several publications, an iterative simulation scheme for production planning problems was discussed [6, 7]. We expect to achieve more realistic cycle times with this method. Finally, it is interesting to consider the interaction of stochastically defined forecasts with the master planning approach embedded into a rolling horizon simulation model.

References

1. Barahona, F.; Bermon, S.; Günlük, O.; Hood, S. 2005. "Robust Capacity Planning in Semiconductor Manufacturing". Naval Research Logistics, 52, 459-468.
2. Bermon, S.; Hood, S. J. 1999. "Capacity Optimization Planning System (CAPS)". Interfaces, 29(5), 31-50.
3. Habla, C., Drießel, R., Mönch, L., Ponsignon, T., Ehm, H. 2007. "A Short-Term Forecast Method for Sales Quantities in Semiconductor Manufacturing". Proceedings IEEE Conference on Automation Science and Engineering, 94 – 99.
4. Habla, C., Mönch, L. 2008. "Solving Volume and Capacity Planning Problems in Semiconductor Manufacturing: A Computational Study". Proceedings Winter Simulation Conference, 2260-2266.
5. Hornung, A., Mönch, L. 2008. "Heuristic Approaches for Determining Minimum Cost Delivery Quantities in Supply Chains". European Journal of Industrial Engineering, Vol. 2, No. 4, 377-400.
6. Hung, Y.-F., Leachman, R.C. 1996. "A Production Planning Methodology for Semiconductor Manufacturing based on Iterative Simulation and Linear Programming Calculations". IEEE Transactions on Semiconductor Manufacturing, 9(2), 257-269.
7. Irdem, D. F., Kacar, N. B., Uzsoy, R. 2008. "An Experimental Study of an Iterative Simulation-Optimization Algorithm for Production Planning". Proceedings of the 2008 Winter Simulation Conference, 2176-2184.
8. Kallrath, J., Maindl, T. I. "Planning in Semiconductor Manufacturing". Handbook of Real Optimization with SAP APO, Springer, 107-118.
9. Michalewicz, Z. 1996. "Genetic Algorithms + Data Structures = Evolution Programs", 3rd ed., Berlin, Heidelberg, New York, Springer.
10. Ponsignon, T., Habla, C., Mönch, L. 2008. "A Model for Master Planning in Semiconductor Manufacturing". Proceedings of the 2008 Industrial Engineering Research Conference.
11. Stray, J., Fowler, J. W., Carlyle, M., Rastogi, A. P. 2006. "Enterprise-wide Semiconductor Resource Planning". IEEE Transactions on Semiconductor Manufacturing, 19(2), 259-268.
12. Vieira, G. E. 2006. "Understanding Master Production Scheduling from a Practical Perspective: Fundamentals, Heuristics, and Implementations". Handbook of Production Scheduling, J. Hermann (ed.), Springer, 149-176.
13. Wall, M. 2006. "GALib – a C++ library of genetic algorithm components". http://lancet.mit.edu/ga/.
14. Xie, J.; Lee, T. S.; Zhao, X. 2004. "Impact of Forecasting Errors on the Performance of Capacitated Multi-Item Production Systems". Computers & Industrial Engineering 46, 205-219.
15. Zobolas, G. I., Tarantilis, C. D.; Ioannou, G. 2008. "Extending Capacity Planning by Positive Lead Time and Optional Overtime, Earliness and Tardiness for Effective Master Production Scheduling". International Journal of Production Research, 46, 3359-86.