A hybrid genetic algorithm to solve a lot-sizing and scheduling problem

Andrea Toniolo Staggemeier
Alistair R. Clark
Uwe Aickelin
Jim Smith

Internal Research Report MS-2002-4

Intelligent Computer Systems Centre Group
School of Mathematical Sciences
Faculty of Computing, Engineering and Mathematical Sciences
University of the West of England, Bristol, BS16 1QY, England.
Email:Andrea.Staggemeier@uwe.ac.uk.

Presented at 16th triannual Conference of the
International Federation of Operational Research Societies 2002
Edinburgh - July 2002

Abstract

This paper reports a lot-sizing and scheduling problem, which minimizes inventory and backlog costs of multiple products on M parallel machines with sequence-dependent set-up times over T periods. Problem solutions are represented as product subsets (ordered or unordered) for each machine m at each period t. The optimal lot sizes are then determined applying a linear program. A genetic algorithm searches either over ordered or over unordered subsets (which are implicitly ordered using a fast ATSP-type heuristic) to try to identify an optimal solution. Initial computational results are presented, comparing the speed and solution quality of the ordered and unordered genetic algorithm approaches.

Key Words: Lot-sizing and Scheduling Problem, Genetic Algorithm, Linear Programming.

1 Introduction

The production planning and scheduling problem of this paper is concerned with the simultaneous sequencing and sizing of production lots on a set of parallel machines when a sequence-dependent set-up time is incurred to change production between lots of different types.

This paper is organized first to present the mathematical model, which describes the fitness function when the problem is solved applying a Genetic Algorithm (GA) (Hol75). A description of the GAs used is then presented with its operators for ordered and unordered subsets of products. Preliminary computational results are presented after which further work and conclusions are discussed as work-in-progress.

2 Mathematical Model

The production scheduling model aims to satisfy the demand and the capacity constraints for P products over T planning periods by minimizing backlog or inventory carried over from one period to the next. The products are manufactured in lots of varying size on M non-identical parallel machines which may have different production rates and associated set-up times.

A sequence-dependent set-up time is required to change from one product to another during which time the machine cannot process any products. There is no restriction on the number of set-ups in each planning period, but since it makes no sense to produce a product in more than one lot on the same machine in the same period, there will in practice be at most one set-up per product per period on each machine. Therefore the number of set-ups on each machine at each period is at most P products.

This enabled us to represent model solutions as machine-time pair that are a subset of products B_{mt} being produced on each machine m at each time-period t. Here the cardinality

of this subset is at most P products, i.e. $|B_{mt}| \leq P$.

The aggregation of these products in a $B_{m,t}$ enables us to work with multiple changeovers within each time-period as well as representing more than one product on each machine at each time period. This is important since the representation uses *Big Buckets* as well as implicitly *Small Buckets* (BW00).

The objective function to this problem can be formulated as follows:

Minimize

$$\sum_{i,t} \left[h_i I_{i,t}^{+} + g_i I_{i,t}^{-} \right] \tag{1}$$

where h_i is the given unit holding cost for inventory ($I_{i,t}^{+}$) of product i at the end of period t, and g_i is the given unit cost for backlog ($I_{i,t}^{-}$) of product i at the end of period t .

The problem is subject to demand and capacity constraints as follows:

$$I_{i,t-1}^{+} - I_{i,t-1}^{-} + \sum_{m} x_{i,m,t} - I_{i,t}^{+} + I_{i,t}^{-} = d_{i,t} \qquad \forall\, i,t \tag{2}$$

$$\sum_{i \in B_{mt}} u_{i,m} x_{i,m,t} \leq A_{m,t} - \sum_{i \to j \in B_{m,t}} s_{i,j,m} \qquad \forall m,t \tag{3}$$

$$x_{i,m,t} \geq 0;\ I_{i,t}^{+} \geq 0;\ I_{i,t}^{-} \geq 0 \qquad \forall\, i,m,t$$

where $x_{i,m,t}$ is the decision variable representing the lot-size of product i on machine m in time-period t, and $d_{i,t}$ is the given demand for product i at the end of time-period t.

The implication of the equality sign in 2 is that if there is any amount of product in inventory or in backlog from one period to the next, those products will cause an immediately cost at the objective function.

The capacity constraint 3, on the other hand, tells us if the problem is feasible or not. The unit time ($u_{i,m}$) of product i on machine m multiplied by the respective production of product i on machine m at time-period t is added up and must to be less or equal than the available capacity $A_{m,t}$ on machine m at time-period t less the total set-up time used for each exchange of products in a specific subset $B_{m,t}$, where $s_{i,j,m}$ represents the set-up time from product i to product j on machine m.

The model above simplifies the problem from a traditional Mixed Integer Programming formulation (CC00) to a Linear Programme eliminating the combinatorial part of the problem, i.e. the allocation and sequencing of the products.

This problem considers backlogs as the main feature to be optimized since the penalty for not supplying the demand for one or more lot of products implies in large losses. In this scenario minimization of set-up and production costs are not so important. If they need to be optimized they can be incorporated into the objective function 1 by just adding the following term:

$$\sum_{j,t} \left[\sum_{i \to j} SetupCost_{i,j,m} + UnitCost_{j,m} x_{j,m,t} \right] \quad \forall m, t \; and \; i, j \in B_{m,t} \tag{4}$$

Mixed Integer Program formulations for this kind of problem have been the object of research for many years (DK97) and results show that a considerable computational effort is needed. Branch-and-Bound algorithms are generally unable to solve large problem instances.

Therefore, in this report we chose a hybrid metaheuristic approach to solve it rather then using a pure exact method. To find out more about the different mathematical models for the lot-sizing and scheduling problem, consult the following surveys ((DK97) which includes different types of problem such as proportional lot-sizing and scheduling, (BW00) which introduces a classification for Lot-size problems in terms of time-periods model, (SCil) which is an introductory survey including modelling different features such as capacity, presence of backlogs and inventory, multiple stages, multiple machines, sequence-dependent set-up times and costs, and a brief discussion of methods used to solve those problems).

In this report we present a comparison of the optimal solution and the Genetic Algorithm approaches for a small instance of the problem.

3 Solution Approaches

The Lot size and scheduling problem in this report is solved using a hybrid approach, which consists of a metaheuristic to allocate the products sequence and the linear programme 1-3 to calculate the objective function value for each metaheuristic solution, as presented in section 2.

The metaheuristic chosen was a generational GA, which consists of: given a population of individuals (encoding a solution to the problem), at each generation the new population consists entirely of offspring formed by parents in the previous generation (Mit96), but with the use of elitism[1] (normally not a feature of generational GAs).

The very simple GA consists of four elements: population of chromosomes, selection according to fitness evaluation, crossover to produce new offspring, and random mutation of the offspring. The next sections will describe the two alternative representations for the generational genetic algorithm proposed.

3.1 Ordered subsets representation

The first GA approach considers B_{mt} as an ordered set since the product allocation includes the sequence for the products in the individual chromosomes. Only one mechanism can alter the total set-up time involved on each set $B_{m,t}$ in this representation, namely the application of mutation operators.

The chromosome chosen for the problem is considered an indirect representation of the lot-sizing and scheduling problem since the individual represents only part of the solution of the problem. The remaining part is the optimization of lot-sizes by linear programming.

The individual chromosome is composed of a set of product subsets $B_{m,t}$, which each contain the products on machine m in time-period t.

[1] By Elitism we understand only the fittest individuals (in this report 10 percent of the population size) from the current population survives into the next generation.

The initial product allocations are determined randomly so that an ordered subset $B_{m,t}$ receives up to P products. This process was repeated for all sets $B_{m,t}$ after which a chromosome was ready to be evaluated.

Each B_{mt} is considered as a gene of the chromosome, which thus has a fixed length of MT genes. The size of each subset varies. It can be at most P products, and with no duplications. This assumption makes it simple to implement crossover and mutation mechanisms in the GA, which will be described later in this report. Table 1 is an example of the chromosome described previously when P=10, M=2, T=5.

m=1					m=2				
t=1	t=2	t=3	t=4	t=5	t=1	t=2	t=3	t=4	t=5
{1}	{},	{0, 2, 5},	{},	{7},	{5},	{},	{1, 6, 7, 5, 2, 8, 9},	{3},	{4}

Table 1: Chromosome Representation

The fitness function of the GA measures the ability of each individual to survive in an environment. In this special case the Linear Programme 1-3 delivers this measure.

In this paper we did not consider any kind of infeasibility treatment, such as, penalty functions, repair mechanisms, decoder processes or even prevention of infeasible solutions (Mic99), although we intend to develop further work on these aspects.

The GA selection process was Binary Tournament Selection attributed for the first time to unpublished work by Wetzel that was studied in Brindle's (1981) dissertation (Bli97).

This particular selection criterion is considered very efficient since it has time complexity O(λ), where λ denotes the number of individuals in the population. When the tournament size is higher the pressure to converge to the best solution is increased. Details and comparison between different types of selection mechanisms can be found in (Mit96) and (Bli97).

For the two GA basic operators, i.e. Crossover and Mutation, we chose the one-point

crossover algorithm (Esh97) considering the crossover point at the level of machines. We guaranteed that the capacity constraint 3 is not violated since the parent individual will already have been checked for this constraint in his evaluation.

For the mutation we implemented four different operators such as Replacement, Insertion, Deletion and Swap products in a randomly chosen subset $B_{m,t}$. Each GA was tested with only one of the four mutation operators running in each simulation.

The mutation operators are special heuristics which act on the selected subset B_{mt} by replacing a product for any other that was not already in B_{mt}, or inserting a product that was not already in B_{mt}, or deleting any product from the B_{mt} , or even swapping the position of two products in the same subset B_{mt}, as shown in the examples in table 2.

Original	{1}	{},	{0, 2, 5},	{},	{7},	{5},	{},	{1, 6, 7, 5, 2, 8, 9},	{3},	{4}
Replacement	{1},	{},	{0, 2,**4**},	{},	{7},	{5},	{},	{1, 6, 7, 5, 2, 8, 9},	{3},	{4}
Insertion	{1},	{},	{0, 2, 5},	{**3**},	{7},	{5},	{},	{1, 6, 7, 5, 2, 8, 9},	{3},	{4}
Deletion	{1},	{},	{0, 2, 5},	{},	{7},	{5},	{},	{1, 6, 5, 2, 8, 9},	{3},	{4}
Swap	{1},	{},	{0, 2, 5},	{},	{7},	{5},	{},	{1, 6,**9**, 5, 2, 8,**7**},	{3},	{4}

Table 2: Mutation Operators

These operators try to diversify the individual chromosome.

3.2 Unordered subset representation

The second GA approach considers each set $B_{m,t}$ as unordered and applies an Asymmetric Traveling Salesman Problem (ATSP) type heuristic to sequence the products inside each $B_{m,t}$ aiming to minimize the total set-up time. The flow chart of the hybrid GA is shown in Figure 1:

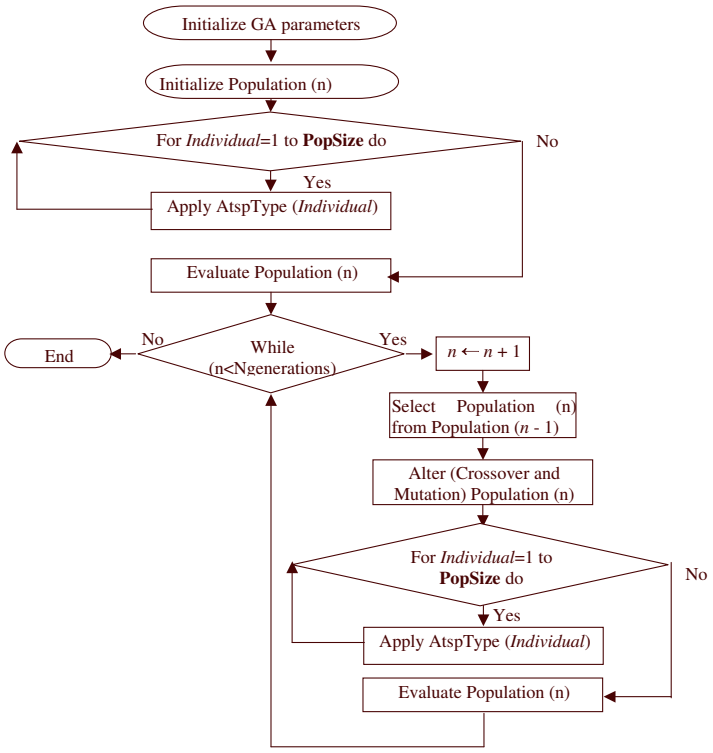


Figure 1: Flowchart of GA+ATSP type heuristic

The ATSP type heuristic used is very simple and also greedy, but the choice was to have a fast heuristic rather than a better quality heuristic, due to the use of a generational GA, i.e. all changes in the current generation, by crossover or mutation must be re-optimized using this heuristic, increasing the GA computational time.

Due to the optimization of the set-up times on each subset B_{mt} the Swap mutation operator does not make sense when applied to this approach, but all these others were kept the same as in subsection 3.1.

4 Computational results

We performed the tests of the GA algorithm on a Sparc Ultra 10 running the Solaris operating system and JAVA compiler 1.3 with Cplex solver 7.5 (ILO02).

The GA was tested with different parameters which generated 25 different possible results for each instance to be analyzed. Each instance has been tested with three different mutation rates and only one mutation operator at any one time.

The parameters of the GA run were population size 50, maximum number of generations 100, with tournament selection size 2. The Crossover probability was set to be 0.90 and Mutation rate was set to be 0.01 (low mutation rate), 0.05 (medium mutation rate) and 0.10 (high mutation rate) in each simulation, where the most suitable on average for all experiments was found to be 0.05. For each simulation we ran the algorithm 10 times.

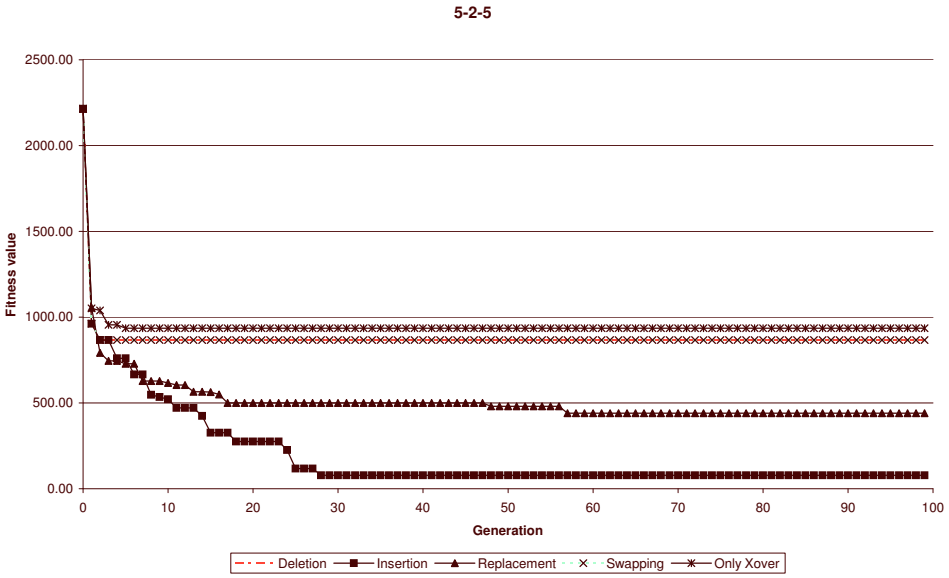
The problems tested used P={5, 10, 25}, M=2 and T=5, and also, when P={50, 100}, M=10, T=5.

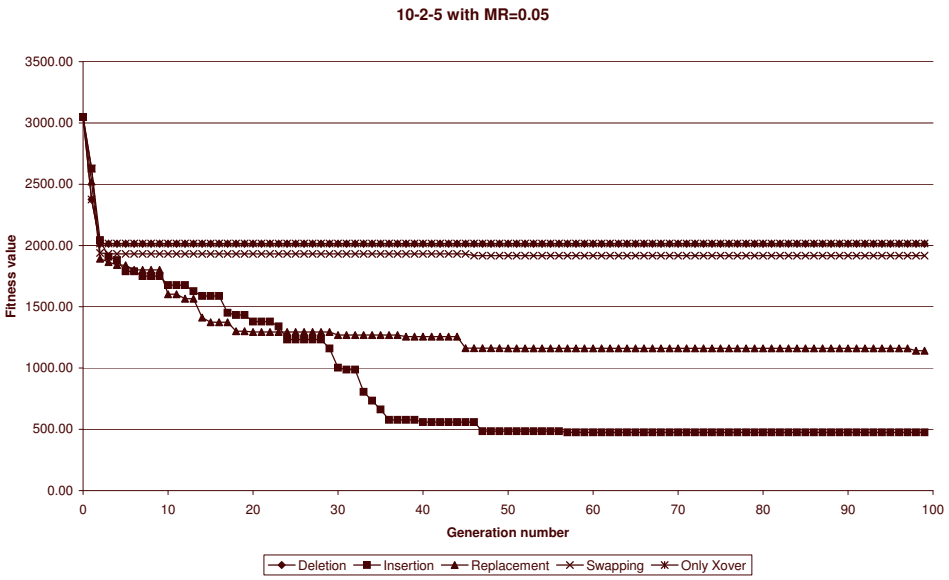
All data generated were randomly and uniformly distributed between the given intervals:

- Demand $d_{i,t}$ [100; 150] units of product i per time-period t;
- Unit production time $u_{i,m}$ [0.005; 0.015] hours of product i on each machine m;
- Setup-time $s_{i,j,m}$ on average of 1 hour per change from product i to product j on machine m. This setup-time is asymmetric, since the time to change from product i to j is not the same as from product j to product i. The setups were also generated observing the triangle inequality so that to setup product i to j and j to k is more time consuming that when the setup is directly from product i to k.
- Holding cost h_i [0; 10] monetary value per each unit of product in inventory;
- Backlog cost g_i is calculated using 100 times of holding cost h_i. The backlog cost is weighted in relation with the inventory cost since it represents in practice a non-supplied order of one or more products to the respective clients.
- For the available capacity we allowed all machines at all time periods to be 24 hours, where for small P, the capacity will be much looser than for large P.

The following charts show the average fitness, for 10 runs, over each of 100 generations, when the GA is applied to the ordered subset encoding, when the crossover operator

plus one of each mutation operator is used, a mutation rate equal to 0.05, and also when only the crossover was the operator used, i.e. zero mutation rate.





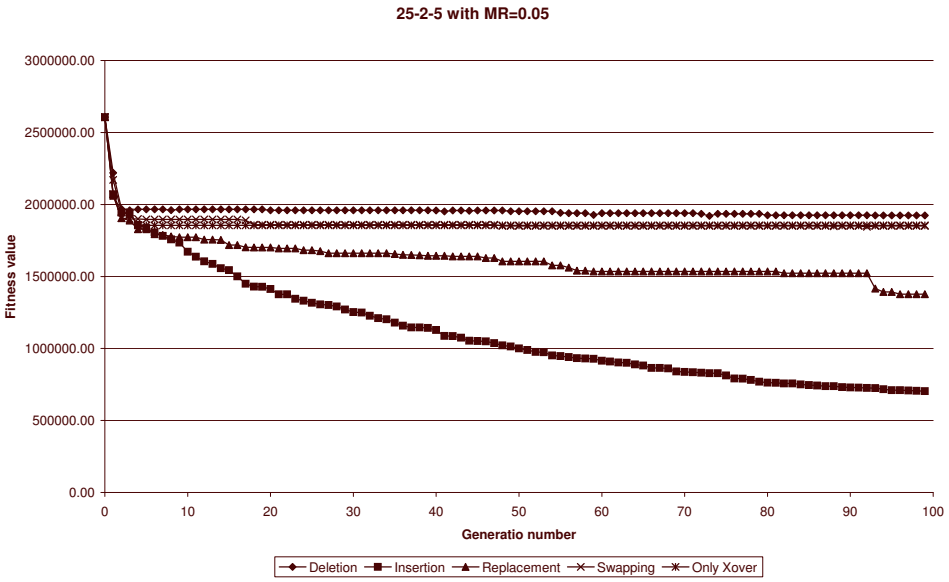


Figure 2: GA using crossover and mutation operators, with medium mutation rate

We note that the insertion mutation operator works better in loose capacity environment and when production capacity starts to be tight then the replacement mutation operator, i.e. deletion followed by insertion, is the second better operator.

Deletion and Swap mutation operators do not work well in loose capacity since by deleting a product in a subset $B_{m,t}$ we were increasing the backlog cost and also by swapping two products we were just exchanging product positions and not helping the algorithm enough to decrease the objective function.

Observing the results on fig. 2 we also note the limitation of the crossover developed for the problem representation. It was well above the average best fitness in two of the instances tested and only in the biggest one were the Deletion and Swap mutation operators better.

The GA algorithm was unable to solve the instance cases where P={50, 100} and M=10 with the same horizon time, since presented, in section 3.1, the algorithm was not able to build the first generation of solutions within 3 hours after running.

In order to compare the quality of the solution found we solved the problem using

Cplex on a Mixed Integer Programming formulation from (Cla00) for the instance $P = 5$. The other instances are too large to be solved optimally by Cplex.

Table 3 presents, for each instance, the optimal value and the CPU time used to find the solution, comparing MIP and the ordered subset representation when the GA is applied with crossover and each one of the four mutation operators at once. We also included in this table the results when only the crossover operator is performed.

Algorithm	5-2-5	10-2-5	25-2-5
MIP	(0.00)1.2	(0.00)900.00	(No-solution found)1800.00
GA(X+Insert)	(0.00)243.98	(0.00)410.29	(1280625.79)1059.11
GA(X+Delete)	(0.00)346.53	(0.00)603.91	(253493.43)1656.12
GA(X+Replace)	(0.00)348.46	(0.00)698.66	(508408.48)1690.41
GA(X+Swap)	(0.00)351.27	(0.00)605.77	(1280625.79)1641.59
GA(X)	(0.00)237.60	(0.00)387.79	(1280625.79)1062.33

Table 3: Best Fitness value and average CPU Time (in seconds) for each instance tested

Table 3 shows that the MIP zero-valued solution is the incumbent when the Branch-and-Bound search was stopped after 900 seconds in the case of 10 products, 2 machines, and 5 time-periods, and no incumbent solution had even been identified after 1800 seconds in the case where the instance has 25 products, 2 machines, and 5 time-periods.

The GA zero-valued optimal solution for instances where P=$\{5$ and $10\}$ were found in 80 and 50 percent, respectively, of the number of runs when we applied crossover plus insertion mutation operators in less than 3 generations.

5 Preliminary conclusions and further works

In this paper we presented two different hybrid approaches for a generational Genetic Algorithm in terms of representation of a complex Lot-sizing and Scheduling problem.

In both cases we used the concept of product subsets to build the individual chromosome, i.e. the production planning solution, one using ordered subsets and the other unordered subsets.

We also decided to use a Linear Program as the fitness function to the algorithm since it gives the optimal solution to the lot-size problem for a particular production sequence.

The allocation and the sequence of those products then became the most important feature of the genetic algorithm to try to escape from local optima through its operators. As described in this paper the operators used do not have any knowledge of the problem itself. They are standard ones such as the one-point-crossover operator.

By using the definition of a gene in our problem such as each subset, the mutation operators presented are if not standard, very straightforward.

The combination of those operators, i.e. crossover and mutation, with the respective parameters presented in section 4, found the optimal solution for the two small instances as well as traditional Mixed Integer Programming. If we take into consideration the number of times when GA found it we could say, for instance, those solutions were found even faster than the MIP formulation. For the large instance, Cplex was not even able to identify an initial solution for the MIP in the time taken by the GA.

Due to the work-in-progress nature of this paper we would like to continue working on the GA+ATSP approach and do the same experiments in order to identify if the ATSP setup times optimization will improve the benefit/cost for the algorithm proposed.

Analyzing the results in section 4 we also suggest the idea of using multiple mutation operators at the same iteration, first by using a random choice of Insertion and Deletion operators, and secondly by using a smarter procedure such as choosing the operator according to the utilisation of capacity on each machine-period pair $B_{m,t}$.

Because of the inability of the algorithm to build a feasible first generation when the instance tested were bigger, i.e. P={50, 100}, M={10} and T={5}, we intend to use repair

mechanisms on the infeasible individuals at this stage in order to test the scalability of the GA proposed.

References

[Bli97] T. Blickle, *Handbook of evolutionary computation*, ch. Tournament Selection, pp. 1–4, 1997.

[BW00] G. Belvaux and Laurence A. Wolsey, *Lot-sizing problems: Modelling issues and a specialized branch-and-cut system bc-prod*, Management Science (2000), no. 46, 724–738.

[CC00] Alistair R. Clark and Simon J. Clark, *Rolling-horizon lot-sizing when setup times are sequence-dependent*, International Journal of Production Research **38** (2000), no. 10, 2287–2308.

[Cla00] Alistair R. Clark, *A local search approach to lot sequencing and sizing*, FIP WC5.7 (2000), 10, Special interest group on advanced techniques in production planning and control.

[DK97] Andreas Drexl and Alf Kimms, *Lot sizing and scheduling - survey and extensions*, European Journal of Operational Research (1997), no. 99, 221–235.

[Esh97] L. J. Eshelman, *Handbook of evolutionary computation*, ch. Genetic Algorithms, pp. 1–11, 1997.

[Hol75] J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[ILO02] ILOG, *Cplex 7.5 user's manual*, ILOG S.A, BP 85, 9 Rue de Verdun, 94253 Gentilly Cedex, France, 2002, http://www.cplex.com/.

[Mic99] Zbigniew Michalewicz, *Genetic algorithms + data structures = evolution programs*, Springer-Verlag, Berlin, 1999.

[Mit96] M. Mitchel, *An introduction to genetic algorithms*, The MIT Press, Massachusetts, 1996.

[SCil] Andrea T. Staggemeier and Alistair R. Clark, *A survey of lot-sizing and scheduling models*, Annals of the 23rd Annual Symposium (Simpsio Brasileiro de Pesquisa Operacional) of the Brazilian Operational Research Society (SOBRAPO), (6-9 November 2001, Campos do Jordo SP, Brazil), 938–947, available at http://www.sobrapo.org.br/simposios/xxxiii/artigos/120-ST014.pdf.