A grid-based environment for multiparametric PSE applications: batch plant design case study

Antonin Ponsich[1], Iréa Touche[1], Catherine Azzaro-Pantel[1], Serge Domenech[1], Luc Pibouleau[1], and Michel Daydé[2]

[1]Laboratoire de Génie Chimique, LGC UMR5503 CNRS/INP/UPS, ENSIACET, 5, rue Paulin Talabot, BP1301, 31106 Toulouse cedex 1, France
[2]Institut de Recherche en Informatique de Toulouse, UMR 5505 CNRS/INP/UPS ENSEEIHT, 2, rue Charles Camichel, BP 7122, 31071 Toulouse Cedex 7, France
{Antonin.Ponsich,Irea.Touche,Catherine.AzzaroPantel,Serge.Domenech, Luc.Pibouleau}@ensiacet.fr
Michel.Dayde@enseeiht.fr

Abstract. Complex optimization problems are of high interest for Process Systems Engineering. The selection of the relevant technique for the treatment of a given problem has already been studied for batch plant design issues. Classically, most works reported in the dedicated literature yet considered item sizes as continuous variables. In a view of realism, a similar approach is proposed in this paper, with discrete variables for representing equipment capacities, which leads to a combinatorial problem. For this purpose, a Genetic Algorithm was used, which is multiparametric by nature and a grid approach is perfectly relevant to this case study, since the GA code must be run several times, with different values of some input parameters, to guarantee its stochastic nature. This paper is devoted to the presentation of a grid-oriented GA methodology. Some significant results are highlighted and discussed.

Key words: grid computing, batch plant design, genetic algorithms, mathematical programming

1 Introduction

A great variety of applications from Process Systems Engineering area can be formulated as complex optimization problems. This large number of optimization problems arises from models that have to take into account a truly realistic representation of the system for industrial requirements. Consequently, these models tend to exhibit an increasing sophistication and a high complexity level, basically deriving from the presence of mixed variables, nonlinearities and severe constraints, so that the search for feasible solutions is identified as a difficult process. In order to face these problems, a significant investigation effort has been carried out to develop efficient and robust optimization methods. But if they prove to be well-fitted to the particular case they consider, the performance of these techniques is not constant over all the problems. Actually, a method efficiency for a particular example is hardly predictable. This feature generates a common lack of explanation concerning the use of a method for the solution of a particular example and usually, no relevant justification for its choice is given a priori. A previous work, comparing the performance of three optimization methods for a few instances of some specific batch plant design problems provided some guidelines on the most appropriate methods: it proved the superiority of a Branch & Bound method both on a Genetic Algorithm (GA) and on an Outer Approximation algorithm [1].

In the formulation adopted in the above-mentioned study, the item sizes (volumes for batch stages and treatment capacity for semi-continuous stages) are continuous variables. Yet, it seems obvious that in the industrial practice, the design of unit operation equipment does not require such a level of accuracy, which seems not realistic, since equipment manufacturers propose defined size ranges (volumes or treatment capacity).

In this study, the item sizes adopt a value among a discrete set, defined in the problem data. The objective is to determine if the conclusions resulting from the previous numerical experiments with continuous item sizes are still valid with the GA.

Nevertheless, although the global problem nature does not change, some modifications must be carried out. Basically, a lot of experimental runs are necessary, implying a huge computational time that a single computer will not be able to support. This kind of application is multiparametric by nature and a grid approach is perfectly relevant to this case study, since the GA code must be run several times, with different values of some input parameters to guarantee its stochastic nature [2]. The GA is a serial Fortran code that does not refer to any external library. The experiments have been performed using Grid'5000 infrastructure (see http://www.grid5000.org).

This introduction is followed by 3 sections. Section 2 presents the framework of the batch plant design issue. Then, Section 3 is devoted to the gridification of the GA application and typical results are analysed. Finally, conclusions and perspectives are given in Section 4.

2 Batch plant design problem formulation

The batch plant design problem was extensively presented in our previous work and will not be recalled here (refer to [3] for the presentation). It must be pointed out that a lot of studies previously addressed the batch plant design issue ([4], [5], [6], [7], [8], [9], [10]). Basically, batch plants are composed of items operating in a discontinuous way. Each batch then visits a fixed number of equipment items, as required by a given synthesis sequence, i.e., the so-called production recipe. In this study, only multiproduct plants are taken into account, meaning that all the products follow the same operating steps. Let us note that the operating times may be different from one recipe to another. The production requirements of each product and data related to each item (processing times and cost coefficients) are specified, as well as a fixed global production time.

Generally, the objective consists in the minimization of plant investment cost. The model formulation for batch plant design problems, adopted in this paper, is based on Modi's approach [6]. It involves I products treated in J batch stages and K semi-continuous units (pumps, heat exchangers,...). Each (batch or semi-continuous) stage is made up of several out-of phase parallel units of same type and size. For the sake of illustration, a batch plant can be visualized by a series of batch stages (B), semi-continuous stages (SC) and storage tanks (T) as shown in Fig. 1

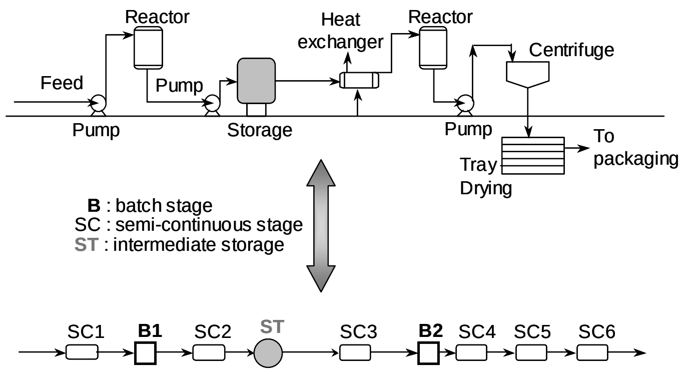


Fig. 1. Typical batch plant and modelling

Typically, the model involves the following optimization variables:

– the size for the batch and semi-continuous stages;
– the item number for each stage.

In the literature dealing with batch plant design (see [3]), the item sizes (volumes for batch stages and treatment capacity for semi-continuous stages) are continuous variables. The resulting optimization problem is then identified as an MINLP (Mixed Integer NonLinear Programming) one, since the other optimization variables (i.e. number of parallel equipment) are inherently integer ones; the equality/ inequatity constraint set, on the one hand, and the objective function, on the other hand, are clearly nonlinear. The aim of batch plant design problems is to find the plant configuration that respects the production requirements within the time horizon, while minimizing the economical criterion. The problem is recognized to be non-convex and NP-hard [8].

Yet, it seems obvious that in the industrial practice, the design of operation unit equipment does not require such a level of accuracy, which seems not realistic. Besides, equipment manufacturers classify items according to defined size ranges (volumes or treatment capacity). This means that an item can only adopt a discrete number of predefined values.

In our previous study [11], the item size of each stage has three possible values, identified as small, medium and large. In this work, three different cases of discretization range are considered, involving different numbers of possible values. The bounds of the continuous variables are initially set as:

- for each j batch stage : $V_{min} = 250, V_{max} = 10000, \forall j$;
- for each k semi-continuous stage :
  $R_{min} = 300L.h^{-1} \leq R_k \leq R_{max} = 10000L.h^{-1}, \forall k.$

Then, three discretization modes are tested :

- *Large-grain Discretization (LD)* : the interval between two possible values is equal to $2500L$ (or $L.h^{-1}$), globally meaning that the five possible values for batch stage size are $250; 2750; 5250; 7750; 10250L$.
  For the semi-continuous stages, the values taken into account are the following ones: $300; 2800; 5300; 7800; 10300$ $L.h^{-1}$.
  Note that the initial upper bound of the variables ($10000L$ or $L.h^{-1}$) is changed.
- *Medium-grain Discretization (MD)* : the interval between two possible values is equal to $500L$ (or $L.h^{-1}$). In this case, the initial continuous upper bound is also slightly modified.
- *Small-grain Discretization (SD)* : the interval between two possible values is equal to $10L$ (or $L.h-1$). In this case, the continuous upper bounds are kept unchanged, since the discretization range enables to reach its initial value.

3 Gridfication of the GA application

3.1 GA overview

The principles of GAs just lie on the analogy between a population of individuals and a set of solutions of any optimization problem. The algorithm presented in this study is adapted from a very classical implementation of a GA. A major difficulty for GAs is concerned with parameters tuning. The quality of this tuning depends strongly on the users experience and problem knowledge. A sensitivity analysis was performed to set parameters such as population size, maximal number of computed generations or survival and mutation rates, to an appropriate value. Two main features of GA implementation are still a challenge for GA performance: variable encoding and constraint handling.

– **Variable enccoding** is carried out with the so-called weighting box. This coding procedure is classically used for continuous variables such as presented in a previous study [3]. The variable variation range is discretized according to the desired precision and a technique similar to binary coding is used to represent the resulting reduced variables. In the case of discrete variables, the precision is equivalent to the distance between two possible values of the variables : the higher the number of feasible values is, the closer the variable behaviour gets to a continuous one. This encoding technique allows the use of classical crossover (one cut-point) and mutation techniques (inversion of the bit value) ;

– An elimination method is used for small size examples for **constraint handling** (from example 1 to 3), while a domination-based tournament technique (from example 4 to 7) is adopted for higher-sized, more severely constrained problems [12][13].

The GA performance is tested over a set of seven increasing complexity examples, for which all data are detailed in [3]. The plant structure and stage number corresponding to each example are provided in Fig. 2.

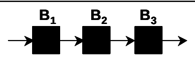
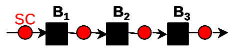
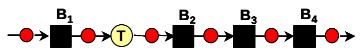
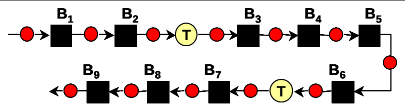
Example	Plant structure	Batch/semicont. stages
1		3 / 0
2		3 / 4
3		4 / 6
4		9 / 12
5	2 x Example 4	18 / 24
6	4 x Example 4	36 / 48
7	5 x Example 4	45 / 60

Fig. 2. Set of increasing complexity examples

3.2 Grid computing approach

A particular drawback of the above-stated methodology must be highlighted : the GA being a stochastic technique, it is always useful to carry out many runs for the same problem, in order to get the best possible result. In this study, each example was treated 100 times by the metaheuristics. Moreover, the considered instances involve problems of size up to 210 variables. As a consequence, these runs involve a huge computational effort, and a classical computing infrastructure may not succeed in managing the amount of necessary computations.

This is why we have considered the use of a computational grid as a suitable alternative. Grid computing is a paradigm well-adapted to two types of applications:

1. parallel applications: an execution of the code uses several nodes to perform the calculation. Usually, the execution time decreases when increasing the number of processors until a certain threshold (the maximum number of processors that can be efficiently exploited, on a given problem, gives some information on the scalability of the code).
2. multiparametric applications: the code is executed several times, but with different values of some input parameters (in series for a single PC operating way). Since the grid often provides a large number of computing nodes, it makes the user able to run the code for all parameter values simultaneously (using one node per parameter value).

Since the GA application considered in this work is multiparametric by nature, a grid approach is perfectly relevant. More precisely, the GA is a serial Fortran code that has no reference to an external library, which greatly simplifies the implementation process in every site of Grid'5000. Practically, the various problem instances (examples 1 to 7) were solved with the GA. Each example was run 100 times.

3.3 Grid'5000

An overview of Grid'5000. Grid is an active research area providing technologies that are now mature enough to be used for real-life applications. Projects such as e-Science, TeraGrid, Grid3, DEISA or NAREGI demonstrate that large scale infrastructures can be deployed to provide scientists with an easy access to resources that are geographically distributed and that belong to different administration domains. Despite its establishment as a viable computing infrastructure, there are still many issues to be solved, mainly in terms of performance, fault tolerance, QoS, security, and fairness. Grid'5000, is a nation-wide infrastructure for research in grid computing (see http://www.grid5000.org, [14]). It is designed to provide a scientific tool for computer scientists similar to the large-scale instruments used by physicists, astronomers, and biologists. Grid'5000 is a large-scale experimental tool, with deep reconfiguration capability, a controlled level of heterogeneity and a strong control and monitoring infrastructure.

The reconfiguration capability allows scientists to deploy, install, boot and run their specific software images (with the possibilty of including all the layers of the software stack). In a typical experiment sequence, a researcher reserves a partition of Grid'5000, deploys its software image, reboots all the machines of the partition, runs the experiment, collects results and relieves the machines. This reconfiguration feature allows all researchers to run their experiments in the software environment corresponding exactly to their needs.

Grid'5000 is used for grid software evaluation and fair comparisons of alternative algorithms, software, protocols, etc. The tools available insure the reproducibility of experiments. The Grid'5000 software includes a reservation tool (OAR2), a deployment tool (Kadeploy2) and several monitoring tools (NAGIOS, a host, service and network monitoring program and GANGLIA, monitoring system for CPU, memory, disc). The Grid'5000 platform is distributed over 9 sites in France : Bordeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia and Toulouse. Every site hosts a cluster and all sites are connected by the RENATER high-speed network (all links will provide 10Gbps). Currently, the platform has around 2,700 processors. The so-called Gridmip Toulouse site is composed of 2 clusters, "Violette" with 57 nodes, dual CPU racks consisting of 2 AMD Opteron running at 2.2Ghz, with 2Gb of memory and "Pastel" with 80 nodes, dual CPU dual core racks consisting of 4 AMD Opteron running at 2.6Ghz, with 8Gb of memory.

Application to the GA working mode. The gridification of the GA application is straightforward. The first step consists in modifying the code in order to be able to schedule the 100 executions over 100 different nodes. This mode is composed of three steps :

GA working mode

```
-Initialization of the parameters
-Loop : do i from 1 to 100

- Random generation of the population
- Simulation
- next i

-Statistic calculations
```

The following phases have to be performed on each node:

– Initialization of the parameters;
– Random generation of the population;
– Simulation.

At the end of the executions on the 100 nodes, statistic calculations can be performed. The modified code has been deployed over five clusters of Grid5000: Lyon, Orsay, Sophia, Bordeaux and Toulouse. A Grid5000's cluster is composed of a frontal machine, which serves to the connexion (where the OAR2 scheduler is accessible), and of the worker nodes, where the jobs are executed. To carry out the computations, the number of nodes required on each cluster has to be chosen. Then, each frontal must be connected and the jobs must be submitted using OAR2. It means that, for each example, 100 jobs have to be submitted with OAR2. To avoid that, a web interface was developed to automatically perform the job submission.

3.4 Computational results

The resolution of all the considered problem instances by the GA, which is computationally intensive, is performed over the computational grid. Getting 100 nodes over five clusters of Grid5000 was not a problem. For the largest instances, each execution required about 4 hours on a single PC, giving a total execution time of 400 hours for the 100 GA runs. This global time was decreased to 4 hours using Grid'5000. This reduction factor represents a great benefit but a more marked effect concerns the time reduction between two runs. It is significantly lower than that can be obtained with traditional large computer infrastructures: a batch system management generally may limit the maximal number of processors to a lower number and impose a long wait on a specific queue. Consequently, the observed time gain can be used more efficiently for carrying out more simulations.

For illustration purpose, Table 1 displays some execution times achieved with GA on Grid'5000 for the two instances involving the highest computational times, i.e. plants 6 and 7 for the Small discretization (SD) mode. These two examples are presented with the number of runs on each site and the time of the fastest and slowest runs. The execution time corresponds to the maximal time considering all the runs.

Example 6				
Site/Nodes	Toulouse/20	Sophia/20	Lyon/20	Orsay/40
Min Time	02:32:57	03:23:15	03:18:16	03:23:18
Max Time	02:41:34	03:57:00	03:43:25	03:57:57
Total Time : 03:57:57				
Example 7				
Site/Nodes	Toulouse/20	Sophia/20	Lyon/20	Orsay/40
Min Time	02:29:37	03:23:05	03:18:20	03:23:17
Max Time	02:40:16	03:57:05	03:43:24	03:57:26
Total Time : 03:57:26				

Table 1. Execution times on Grid'5000

By considering smaller discretization modes, the number of feasible values for the variables becomes greater, so that the GA working mode is gradually reaching continuous variables (with an increasing precision). The GA thus shows an intermediate behaviour, between the two above-considered extreme cases (MINLP and LD). Fig. 3 gives the best solution found by the GA for the example set ; the results for Large discretization are also recorded.

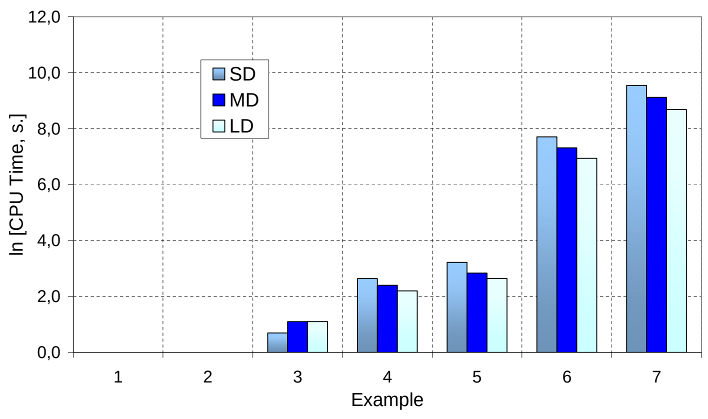


Fig. 3. CPU time of GA for SD, MD, LD problems

4 Conclusions and perspectives

In this work, a variant to the classical batch plant design problem is proposed : the item sizes are no more considered as real variables in the model, but are now represented by discrete variables. The optimization problem was solved with a Genetic Algorithm for seven test examples of incresing complexity. The discretization mode was a parameter of this investigation. The analysis of the behaviour of a stochastic optimization method, on various instances, with several runs for each instance, represents a huge computational effort that would be prohibitive with a classical computational tool. For this purpose, a grid-based approach was implemented, in order to reduce the computing time. Indeed, for the largest instance, the elapsed time for simulation is reduced from 400 hours down to 4 hours, which is an impressive benefit. It must be pointed out that Mathematical Programming techniques (DICOPT ++ and SBB within the GAMS package, [15]) previously tested in our previous work ([3], for instance) are heavily penalized by the combinatorial effect induced by the discretization of the item sizes, which leads to prohibitive computational times. This effect prevented the solvers from converging with fulfilled optimality conditions for the largest instances of the problem. DICOPT++ proved its inefficiency, including for small size examples. However, SBB was able to solve various instances, and when not able to converge, it provided feasible solutions (quite far away from the optimal ones). The GA work is made easier by this discretization, which does not involve any change in the variable encoding, except a great reduction of the chromosome size. So, the GA could solve all the tackled problems, for various discretization modes, showing a good performance, quite similar to that related to the original MINLP problem. Finally, this approach, using discrete variables to represent item sizes, seems appropriate to treat the batch plant design problem, in both industrial and numerical points of view. In that context, the stochastic methods have proven to be efficient to provide good-quality and realistic solutions. This grid-oriented GA could be of great interest for multiobjective applications, which are inherently greedy in computational time.

References

1. Ponsich, A., Azzaro-Pantel, C., Domenech, S., Pibouleau, L.: Some guidelines for Genetic Algorithm implementation in MINLP Batch Plant Design problems. Advances in Metaheuristics for Hard Optimization Series: Natural Computing Series 2008, XVI, 481 ISBN: 978-3-540-72959-4 (Siarry, Patrick; Michalewicz, Zbigniew)
2. Laforenza, D.: Grid programming : some indications where we are headed. Parallel Computing **28** (2002) 1733–1752
3. Ponsich, A., Azzaro-Pantel, C., Domenech, S., Pibouleau, L.: Constraint handling strategies in genetic algorithms application to optimal batch plant design. Chemical Engineering and Processing: Process Intensification **47**(3) (March 2008) 420–434
4. Grossmann, I.E., Sargent, R.W.H.: Optimum design of multipurpose chemical plants. Industrial & Engineering Chemistry Process Design and Development **18**(2) (1979) 343–348
5. Kocis, G.R., Grossmann, I.E.: Global optimization of nonconvex mixed-integer nonlinear programming (minlp) problems in process synthesis. Industrial & Engineering Chemistry Research **27**(8) (1988) 1407–1421
6. Modi, A., Karimi, I.: Design of multiproduct batch processes with finite intermediate storage. Computers & Chemical Engineering **13**(1-2) (1989) 127–139
7. Patel, A.N., Mah, R.S.H., Karimi, I.A.: Preliminary design of multiproduct noncontinuous plants using simulated annealing. Computers & Chemical Engineering **15**(7) (July 1991) 451–469
8. Wang, C., Quan, H., Xu, X.: Optimal design of multiproduct batch chemical process using genetic algorithms. Industrial & Engineering Chemistry Research **35**(10) (1996) 3560–3566
9. Wang, C., Quan, H., Xu, X.: Optimal design of multiproduct batch chemical processes using tabu search. Computers & Chemical Engineering **23**(3) (February 1999) 427–437
10. Chunfeng, W., Xin, Z.: Ants foraging mechanism in the design of multiproduct batch chemical process. Industrial & Engineering Chemistry Research **41**(26) (2002) 6678–6686
11. Dietz, A., Azzaro-Pantel, C., Pibouleau, L., Domenech, S.: A framework for multiproduct batch plant design with environmental consideration: Application to

protein production. Industrial & Engineering Chemistry Research **44**(7) (2005) 2191–2206

12. Deb, K.: An efficient constraint handling method for genetic algorithms. Computers Methods Applied in Mechanical Engineering **186** (2000) 311–338
13. Coello Coello, C., Mezura Montes, E.: Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics **16** (2002) 193–203.
14. R. Bolze, F. Cappello, E. Caron, M. Dayde, F. Desprez, E. Jeannot, Y. Jegou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quetier, O. Richard, E.G. Talbi, and I.Touche Grid'5000: A large scale and highly reconfigurable experimental grid testbed. International Journal of High Performance Computing Applications **20** (2006) 481 – 494
15. : Brooke, A., Kendrick, D., Meeraus, A. and Raman, R., 1998, GAMS Users Guide. GAMS Development Corporation.