Computers and Chemical Engineering 121 (2019) 212–223

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Fast genetic algorithm approaches to solving discrete-time mixed integer linear programming problems of capacity planning and scheduling of biopharmaceutical manufacture

Karolis Jankauskas[a], Lazaros G. Papageorgiou[b], Suzanne S. Farid[a,*]

[a] Department of Biochemical Engineering, University College London, London WC1E 7JE, UK
[b] Department of Chemical Engineering, University College London, London WC1E 6BT, UK

ARTICLE INFO

Article history:
Received 18 December 2017
Revised 28 May 2018
Accepted 17 September 2018
Available online 21 September 2018

Keywords:
Capacity planning
Scheduling
Genetic algorithm
Biopharmaceutical

ABSTRACT

The previous research work in the literature for capacity planning and scheduling of biopharmaceutical manufacture focused mostly on the use of mixed integer linear programming (MILP). This paper presents fast genetic algorithm (GA) approaches for solving discrete-time MILP problems of capacity planning and scheduling in the biopharmaceutical industry. The proposed approach is validated on two case studies from the literature and compared with MILP models. In case study 1, a medium-term capacity planning problem of a single-site, multi-suite, multi-product biopharmaceutical manufacture is presented. The GA is shown to achieve the global optimum on average 3.6 times faster than a MILP model. In case study 2, a larger long-term planning problem of multi-site, multi-product bio-manufacture is solved. Using the rolling horizon strategy, the GA is demonstrated to achieve near-optimal solutions (1% away from the global optimum) as fast as a MILP model.

1. Introduction

Biopharmaceutical drug development requires substantial investments of capital, human resources, and technological expertise. The cost of development of a single drug entering human trials between 1989 and 2002 was estimated to be in excess of $800M (DiMasi et al., 2003). The development costs have been rising substantially for years. In an analysis by Paul et al. (2010), the cost of a new molecular entity was reported to be $1.8B. The likelihood of a new biopharmaceutical drug product gaining approval for marketing and the rate of approval for new products has been getting lower over the years. According to Kaitin and DiMasi (2010), only one in six new drugs that entered clinical trials in the United States during 1993–98 and the 1999–2004 sub-periods were successfully approved for marketing. Shanley (2014) reported that only 12% of the candidate drugs get approved for use. Given the high costs and the uncertainty of the biopharmaceutical development process, building new capacity for products which may or may not reach the market is not the most desirable option. Therefore, it is essential to optimise the manufacturing capacity in an existing multi-product facility or a network of facilities. Ransohoff (2004) suggested that a typical mammalian cell-culture facility could increase annual revenues by $380M with a 25% increase in plant utilisation.

Biopharmaceutical production planning is a complex combinatorial optimisation problem complicated by the unique features of biomanufacture. Biopharmaceutical companies typically have a portfolio of various products manufactured across a network of owned and contract manufacturing facilities with wide-ranging production capabilities. Biopharmaceutical products tend to be unstable and thus have specialised and costly transportation and storage requirements. Biopharmaceutical companies are also required to meet high-quality standards and prove they can deliver a consistent manufacturing process. The high-quality standards are achieved by rigorous cleaning and sterilisation between individual production campaigns.

The research work in the literature for capacity planning and scheduling of biopharmaceutical manufacture has focused on discrete-time MILP formulations adapted from the pharmaceutical and chemical engineering industries (Papageorgiou et al., 2001). The first medium-term capacity planning model for a multi-product, multi-suite biopharmaceutical facility was presented by Lakhdar et al. (2005). Their approach helped to determine the optimal durations and sequence of production campaigns together

* Corresponding author.
E-mail addresses: karolis.jankauskas.10@ucl.ac.uk (K. Jankauskas), s.farid@ucl.ac.uk (S.S. Farid).

https://doi.org/10.1016/j.compchemeng.2018.09.019

Notation for case study 1

Indices

i	*USP* suites
j	*DSP* suites
p	products
t, θ	time periods

Parameters

H_t	available production time horizon over time period t, [days]
C_p	*USP* storage capacity of product p, [batches]
F_p	*DSP* storage capacity of product p, [batches]
CR_p	*USP* production rate of product p, [batches/day]
FR_p	*DSP* production rate of product p, [batches/day]
CT_p^{min}	min production time for product p in *USP* suite i, [days]
CT_p^{max}	max production time for product p in *USP* suite i, [days]
FT_p^{min}	min production time for product p in *DSP* suite j, [days]
FT_p^{max}	max production time for product p in *DSP* suite j, [days]
α_p	*USP* lead time of product p, [days]
β_p	*DSP* lead time of product p, [days]
ρ_p	*USP* storage cost, [RMU/batch]
ω_p	*DSP* storage cost, [RMU/batch]
ζ_p	*USP* shelf-life of product p, [time periods]
σ_p	*DSP* shelf-life of product p, [time periods]
λ_p	production correspondence factor for *USP* to *DSP* production of product p
η_p	manufacturing cost, [RMU/batch]
ψ_p	changeover cost, [RMU]
T_p	waste disposal cost, [RMU/batch]
ν_p	sales prices, [RMU/batch]
δ_p	lateness penalty, [RMU/batch]
D_{pt}	demand of product p at time period t, [batches]

Integer variables

$product_{it}$	part of the chromosome containing product labels allocated at time period t to *USP* suite i
$product_{jt}$	part of the chromosome containing product labels allocated at time period t to *DSP* suite j
$time_{it}$	part of the chromosome containing the number of production days allocated at time period t to *USP* suite i
$time_{jt}$	part of the chromosome containing the number of production days allocated at time period t to *DSP* suite i
B_{ipt}	number of batches of product p produced at time period t in *USP* suite i
B_{jpt}	number of batches of product p produced at time period t in *DSP* suite j
CI_{pt}	number of batches of *USP* product p stored at time period t
FI_{pt}	number of batches of *DSP* product p stored at time period t
CW_{pt}	number of batches of *USP* product p wasted at time period t
FW_{pt}	number of batches of *DSP* product p wasted at time period t
S_{pt}	number of batches of product p sold at time period t
Δ_{pt}	number of batches of product p in backlog at time period p

Binary variables

Y_{ipt}	1 if product p is produced in *USP* suite i at time period t; 0 otherwise
Y_{jpt}	1 if product p is produced in *DSP* suite j at time period t; 0 otherwise
Z_{ipt}	1 if a new campaign of product p is produced in *USP* suite i at time period t; 0 otherwise
Z_{jpt}	1 if a new campaign of product p is produced in *DSP* suite j at time period t; 0 otherwise

Continuous variables

CT_{ipt}	production time for product p in *USP* suite i during time period t, [days]
FT_{jpt}	production time for product p in *DSP* suite j during time period t, [days]
Profit	total profit – objective function, [RMU]

Notation for case study 2

Indices

i	facilities
p	products
t, θ, ξ	time periods

Sets

PI_i	set of products that can be produced by facility i
TI_i	set of time periods in which facility i is available

Parameters

H_t	available production time horizon over time period t, [days]
C_p	storage capacity of product p, [kg]
T_{ip}^{min}	min production time for product p in facility i, [days]
T_{ip}^{max}	max production time for product p in facility i, [days]
r_{ip}	production rate of product p at facility i, [batches/day]
α_{ip}	lead time for product p at facility i, [days]
ζ_p	shelf-life of product p, [time periods]
η_{ip}	manufacturing cost of product p at facility i, [RMU/batch]
ρ_p	storage cost of product p, [RMU/kg]
ψ_p	changeover cost of product p, [RMU]
ν_p	sales prices, [RMU/kg]
δ_p	lateness penalty, [RMU/kg]
ζ_p	shelf-life of product p, [time periods]
π_p	backlog decay factor of product p
yd_{ip}	yield conversion factor for product p in facility i, [kg/batch]
D_{pt}	demand of product p at time period t, [kg]

Integer variables

$product_{it}$	part of the chromosome containing product labels allocated at time period t to facility i
$time_{it}$	part of the chromosome containing the number of production days allocated at time period t to facility i
B_{ipt}	number of batches of product p produced at time period t in facility i

Binary variables

Y_{ipt}	1 if product p is produced in facility i at time period t; 0 otherwise
Z_{ipt}	1 if a new campaign of product p is produced in facility i at time period t; 0 otherwise

Continuous variables

T_{ipt}	production time for product p at facility i during time period t, [days]
K_{ipt}	amount of product p produced in facility i at time period t, [kg]
I_{pt}	amount of product p stored at time period t, [kg]
W_{pt}	amount of product p wasted in at time period t, [kg]
S_{pt}	amount of product p sold at time period t, [kg]
Δ_{pt}	amount of product p in backlog at time period p, [kg]
Profit	total profit – objective function, [RMU]

with product inventory, sales, and late deliveries profiles. Furthermore, the proposed MILP based optimisation method was shown to find more optimal solutions than the industrial rule-based approach.

The randomness of the biopharmaceutical manufacturing environment can cause significant scheduling and planning difficulties for the biopharmaceutical manufacturing campaigns. To address this, Lakhdar and Papageorgiou (2006) applied MILP and two-stage programming for medium-term planning of biopharmaceutical manufacture under uncertain fermentation titres. The proposed methodology achieved better results than the deterministic MILP model.

The optimisation of biopharmaceutical manufacturing capacity often involves many multiple conflicting criteria and objectives to be considered. George et al. (2007) presented a multi-criteria decision framework for the selection of strategies for acquiring biopharmaceutical manufacturing capacity. Lakhdar et al. (2007) addressed the challenge of making long-term, multi-site capacity planning decisions given multiple strategic criteria such as risk, cost, and customer service levels. Using goal programming, they developed a mathematical model that could optimise multiple objectives.

The vast part of the research on biopharmaceutical manufacture planning has been limited to either batch or fed-batch processes. However, a more recent, large-scale discrete-time MILP model was presented by Siganporia et al. (2014) to optimise long-term capacity plans for a portfolio of biopharmaceutical products, with either batch or perfusion bioprocesses, across multiple facilities to meet quarterly demands.

Genetic algorithms are stochastic, population-based search algorithms that follow the naïve laws of evolution and natural selection. Unlike most other conventional optimisation algorithms, GAs start from a pool of usually randomly generated solutions (also known as chromosomes). Chromosomes with a higher objective function value are selected for reproduction (crossover and mutation) to generate new and hopefully better solutions for each new iteration of the algorithm (also known as a generation). Since a population of solutions is processed in each iteration of the GA, the outcome is also a population of solutions. If an optimisation problem has a global optimum, then all chromosomes can be expected to converge to it. Alternatively, if an optimisation problem has multiple optimal solutions, GAs can capture them in its final population (Deb, 2001). Unlike classical optimisation methods which make assumptions about the relationships between the variables, constraints, and the objective, GAs are flexible optimisers making minimal assumptions about the problem. Therefore, despite the lack of guarantee of finding global optimum and the difficulty of designing the objective function, chromosome structure, and operators, GAs have been used to obtain approximate solutions to a wide range of complex linear and non-linear problem such as training neural networks (Chen and Liao, 1998), finding the optimal number, types, and positions of wireless transmitters (Ting et al., 2009), and creating a program capable of solving planning problems described in Planning Domain Definition Language (PDDL) (Brie and Morignot, 2005).

At the time of writing, the literature on the use of alternative optimisation methods such as genetic algorithms (GAs) in the biopharmaceutical industry was somewhat limited. Most of the publications focused on the management of product portfolios and the optimisation of process design. On the portfolio management front, George and Farid (2008) developed a stochastic, multi-objective optimisation framework based on probabilistic, model-building genetic algorithms for the optimisation of the structure and development pathway of biopharmaceutical drug portfolios. Nie et al. (2012) presented a stochastic, GA-based decision-support tool to address the decisions involved in portfolio management at both the drug development process level and the portfolio level. On the process design front, Simaria et al. (2012) proposed a multi-objective GA-based approach for the selection and optimisation of purification sequences and chromatography column sizing strategies. Allmendinger et al. (2012) presented a GA for the discovery of chromatography equipment sizing strategies for antibody purification processes under uncertainty.

The performance of the GA depends on its hyperparameter values. For example, the rate of crossover controls the capability of the GA in exploiting the known parts of the search space, whereas the mutation rate controls the speed of the GA in exploring of new areas (Lin et al., 2003). The values of these parameters are quite often tuned one by one, i.e. by trial and error. However, this can be a time consuming process leading to suboptimal results, since the interactions between the parameters are ignored this way (Eiben et al., 1999). There has been a number of suggestions and theoretical investigations into the optimal values of crossover, mutation, and population size (Back, 1993; Chipperfield and Fleming, 1995; Goldberg and Deb, 1991; Schaffer and Morishima, 1987). The typical values of crossover and mutation rate have been reported to lie in the range 0.5–1.0 and 0.001–0.05 respectively. However, most investigations were based on simple function optimisation problems with traditional chromosome encoding strategies and genetic operators. Therefore, their applicability for other types of problems and custom genetic operators is quite limited. An alternative to manual parameter tuning is meta-optimisation, i.e. the use of another optimisation algorithm to tune the GA hyperparameters. For example, Grefenstette (1986) applied a meta-GA to optimise the hyperparameters of another GA. In this work, we use a particle swarm optimisation (PSO) algorithm to tune the GA. PSO is an evolutionary, stochastic optimisation technique modelled after swarming and flocking behaviours in animals, developed by James Kennedy and Russel Eberhart in the mid-1990s (James and Russell, 1995; Luke, 2013). PSO is similar to a GA since it is also a population-based algorithm. However, in PSO every solution is also assigned a randomised velocity vector and the potential solutions are called particles. A more detailed description of PSO algorithm can be found in an overview by Poli et al. (2007). A PSO algorithm was chosen due to its simplicity and relatively low computational overhead (compared to using another GA) (Pandey et al., 2010) and suitability for the optimisation of functions with continuous inputs (Hassan et al., 2004).

This paper presents a fast GA-based approach to both medium- and long-term capacity planning and scheduling of single- and multi-site biopharmaceutical manufacture using discrete-time models. The proposed GA is demonstrated as a valid alternative to MILP to obtain near-exact solutions to close to real-world in-

Table 1
The comparison of MILP model complexity between case study 1 and 2.

	Case Study 1	Case Study 2
Single equations	535	19,430
Single variables	457	25,018
Discrete variables	252	9382
Non-zero elements	1750	72,244

dustrial case studies of capacity planning and scheduling of biopharmaceutical manufacture. Other contributions of this paper include the chromosome encoding strategy, the algorithms describing the multi-suite and multi-site biopharmaceutical manufacture, the rolling horizon approach for solving larger, long-term capacity planning problems, and a PSO-based meta-optimisation approach for tuning the GA hyperparameters.

The details of the proposed approach and two case studies are outlined in Section 2 of this paper. Section 3 is dedicated to the discussion of the results and the comparison between the GA and MILP. Conclusions and future work are provided in Section 4.

2. Methods

In this section, the case studies of capacity planning and scheduling of biopharmaceutical manufacture from two different literature sources are described. In case study 1, a medium-term capacity planning and scheduling problem of a multi-suite, multi-product biopharmaceutical manufacture from Lakhdar et al. (2005) is presented. In case study 2, a long-term capacity planning and scheduling problem of multi-site, multi-product bio-manufacture from Lakhdar et al. (2007) is solved. The mathematical models are summarised in the Appendix; however, the reader is advised to refer to the original papers for a more in-depth explanation.

In this paper, the MILP models were recreated in GAMS 23.9.5 and solved with a CPLEX 12.4.0.1 solver. GA and PSO algorithms were implemented using C++ programming language. The implementation of mathematical models using algebraic modelling systems such as GAMS is entirely different compared to the general-purpose programming languages. GAMS allows the mathematical models to be implemented in a way that is very similar to their mathematical description, while the general-purpose programming languages require an explicit definition of every expression. Another critical challenge of developing an efficient GA-based approach was identifying the smallest number of independent variables and the shortest sequence of steps needed to evaluate the candidate solutions for the case studies.

In this section, the structure of the proposed approach and the steps of the algorithms that captured capacity planning objectives for multiple products across multiple suites and facilities are outlined. Most of the proposed GA-based approach is detailed in the methods section for case study 1. In case study 2, the focus is placed on the rolling horizon strategy taken to improve the performance of the standard GA for solving the long-term capacity planning problem. The relative complexity of the optimisation problems is illustrated by the summary of the MILP model statistics shown in Table 1. Both case studies were performed on an Intel i5-6500 based system with 16GB of RAM.

The process of identifying the optimal parameters for an optimization algorithm or a machine learning one is usually costly involves the search of a large, possibly infinite, space of candidate parameter sets, and may not guarantee optimality (Camilleri et al., 2014). In this work, a simple PSO algorithm is implemented as a meta-optimiser to automatically tune the crossover and mutation parameter values in both case studies. Each particle, i.e. a potential solution, is initiated with randomised position and velocity vectors. A position vector holds the parameter values for the crossover and mutation. The fitness of each particle is assessed by running the GA using the parameter values from the position vector for a specified number of independent algorithm runs with fixed population size and measuring the average of the best objective function values achieved. The concept of the meta-optimisation is illustrated in Fig. 1. The parameter values of PSO algorithm (Table 2) were chosen based on the studies performed by Eberhart and Shi (2000) and Trelea (2003).

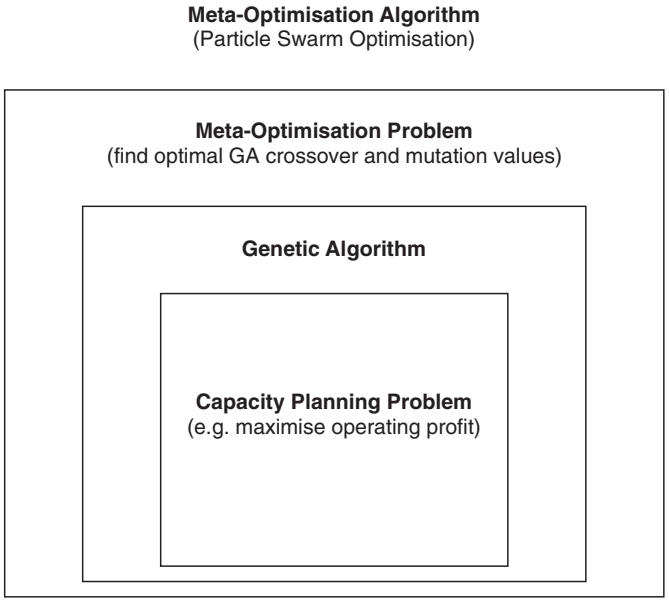

Fig. 1. The meta-optimisation framework. Adapted from Camilleri et al. (2014).

Table 2
Meta-optimisation parameters used in case study 1 and 2 to find the GA optimal crossover and mutation parameter values.

	Case study 1	Case study 2
PSO inertia weight, w	0.729	
PSO local weight, c_1	1.494	
PSO global weight, c_2	1.494	
PSO swarm size (number of particles)	20	
Number of PSO epochs	200	
Number of GA runs	100	50
GA population size	100	200

2.1. Case study 1 – Medium-term capacity planning of a multi-suite, multi-product biopharmaceutical facility

The objective of the planning problem presented here is to generate a yearlong production schedule that would maximise the manufacturing profits. The problem statement adapted from Lakhdar et al. (2005) is as follows:

- Given:
  - Biopharmaceutical products $P = \{p1, p2, p3\}$
  - USP suites $I = \{i1, i2\}$ and DSP suites $J = \{j1, j2\}$
  - A planning horizon of 360 days made of equal time periods $T = \{t1, t2, \ldots, t6\}$
  - Product-dependent production rates, lead times, and production throughputs (correspondence factors)
  - USP and DSP product shelf-life, storage capacities and costs
  - Product demands, sales price and backlog penalty costs
  - Manufacturing and campaign changeover costs
  - Minimum and maximum campaign durations

		t_1	t_2	t_3	t_4	t_5	t_6	…	t_n
USP suites	i_1	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	…	(p, CT_{ipt})
	i_2	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	…	(p, CT_{ipt})
	…	…	…	…	…	…	…	…	…
	i_n	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	(p, CT_{ipt})	…	(p, CT_{ipt})

Fig. 2. Chromosome encoding strategy for case study 1. Each "(p, CT_{ipt})" pair represents a gene with the information regarding which product p and how many days CT_{ipt} have been allocated to each *USP* suite i at a time period t.

- Determine:
  - Duration and sequence of campaigns
  - Production quantities along with inventory profiles
  - Product sales and late deliveries profile
- To:
  - Maximise manufacturing profits

2.1.1. Chromosome structure

In both case study 1 and case study 2, the chromosomes use the direct encoding of the key variables from the mathematical models. Compared to the traditional form of binary encoding, e.g. Holland (1975), real-coded GAs have been shown to be more consistent from run to run and provide faster, more precise performance (Gaffney et al., 2010; Janikow and Michalewicz, 1991). In case study 1, each chromosome is an $|I|$-*by*-$|T|$ array of tuples where I is a set of USP suites, and T represents a set of discrete-time periods (illustrated in Fig. 2). Each tuple comprises a product label p and the length of production in *USP* suite i at time period t, CT_{ipt}, measured in days both of which are randomly generated at the beginning of the GA.

2.1.2. Genetic algorithm

The GA comprises the following critical steps: fitness evaluation, tournament selection, reproduction (i.e. crossover and mutation), and replacement. In case study 1, chromosomes for crossover and mutation are selected using a binary tournament with replacement strategy which favours individuals with a higher objective function value, i.e. schedules with a larger profit value. Tournament selection has been shown to be computationally more efficient and have better or equivalent convergence when compared to other selection methods that are available in the literature (Goldberg and Deb, 1991; Melanie, 1996). A uniform crossover operator with a probability pC (Table 3) is used to exchange the tuples between the chromosomes. Each tuple is also mutated with a small probability pM (Table 3) to avoid premature convergence and improve the quality of the final solution. During mutation, the product label is changed by replacing it with a random value from the set of available products P. The length of production is varied by adding or subtracting a random number of days, ensuring the allocated campaign time is within the constrained range, CT_{ip}^{min} and CT_{ip}^{max}.

In both case studies, the GA is augmented with elitism which is a highly exploitative method of preserving the fittest chromosomes from the previous population (Luke, 2013). In case study 1, a single best chromosome is re-inserted into the population whenever it is lost. Finally, the GA is set to terminate early if the fitness of the best individual has not improved for 100 consecutive generations.

2.1.3. Fitness evaluation

The fitness evaluation procedures of case study 1 and case study 2 are adapted from the MILP models (Lakhdar et al., 2007, 2005) of multi-product biopharmaceutical manufacture. In case study 1, the fitness evaluation procedure generates a complete production schedule which is used to estimate the objective function value, i.e. profit. The pseudo algorithm of the fitness evaluation procedure for case study 1 is presented in Table 4.

Table 3
Case study 1 results and model statistics for MILP and GA models.

	MILP	GA
Max obj. function value	490	490[a]
Solution time (s)	0.22	0.07[b]
Optimality gap	0%	0%[c]
Avg. obj. function value[d]	–	490 ± 0
Population size	–	100
Crossover rate, pC[e]	–	0.710
Mutation rate, pM[e]	–	0.070
Termination[f]	–	100

[a] Max objective function value obtained from 100 independent GA runs.
[b] An average solution time of a single GA run.
[c] An optimality estimate relative to the global optimal obtained using the recreated MILP model, i.e. *1 – obj. function value / global optimum.*
[d] Average of best objective function values from 100 independent GA runs (mean $\pm$ 1 standard deviation).
[e] The parameter values were selected using the PSO algorithm.
[f] If the best objective function value remained the same for 100 consecutive generations, the GA was terminated.

In Table 4, Lines 3 and 4 retrieve the product label p and the number of production days allocated to *USP* suite i at time period t, CT_{ipt}, from the chromosome which is an $|I|$-*by*-$|T|$ array where I is a set of upstream suites and T is a set of time periods. Lines 5 and 6 calculate the number of changeovers and batches produced in *USP* suite i at time period t. In Line 5, the value of the changeover variable Z_{ipt} will be equal to 1 only if product p has not been produced in *USP* suite i at a previous time period t – 1. Line 7 accumulates the production from all *USP* suites. Lines 9 and 10 estimate the amount of product p wasted in *USP* suites at time period t which is equal to the number of batches that was left unprocessed from ζ_p periods ago. The amount of *USP* inventory of product p at time period t is calculated in Line 11 by adding the cumulative value obtained in Line 7 from time period t – 1 and subtracting the amount of waste, CW_{pt}.

Line 13 ensures that the assignment of product p to *DSP* suite j at time period t is performed only once. Lines 14 and 15 assign product p to *DSP* suite j and calculate how many batches will be produced in that suite at time period t. This is performed by multiplying the *USP* inventory value CI_{pt} by the production correspondence factor λ_p which specifies the respective throughputs from *USP* and *DSP* suites. For example, a factor of 0.5 signifies that for every two *USP* batches one *DSP* batch is produced. Line 15 evaluates the number of changeovers in *DSP* suites similarly to Line 7. Line 16 checks whether the length of production of product p in *DSP* suite j at time period t does not exceed the allowed maximum. If it does, the value of variable B_{jpt} is iteratively decremented until the production time FT_{jpt} is below or equal the value of the constraint, FT_p^{max}. Line 17 updates the value of *USP* inventory of product p at time period t by subtracting the number of batches that

Table 4
Pseudocode for fitness evaluation in case study 1.

```
1  for each time period t
2      for each upstream suite i
3          p = products_it
4          CT_ipt = time_it
5          Z_ipt = 1 − (t > 0 and p == products_i,t-1)
6          B_ipt = Z_ipt + CR_p(CT_ipt − α_p Z_ipt)
7          CI_pt = CI_pt + B_ipt
8          for each product p
9              if t > ζ_p
10                 CW_pt = CI_p,t-ζp-1 − (Σ_j Σ_{θ=t−ζ_p}^{ζ_p} B_jpθ + Σ_{θ=t−ζ_p}^{ζ_p} CW_pθ)
11             CI_pt = CI_pt + CI_p,t-1 − CW_pt
12             for each downstream suite j
13                 if products_jt == 0
14                     B_jpt = λ_p CI_pt
15                     Z_jpt = 1 − (t > 0 and p == products_j,t-1)
16                     while (FT_ipt = β_p Z_jpt + (B_jpt − Z_jpt)/FR_p) > FT_p^max do B_jpt = B_jpt − 1
17                     CI_pt = CI_pt − B_jpt/λ_p
18                     FI_pt = FI_pt + B_jpt
19                     products_jt = p
20                     time_jt = FT_jpt
21             if CI_pt > C_p
22                 CW_pt = CW_pt + CI_pt − C_p
23                 CI_pt = C_p
24     if t > σ_p
25         FW_pt = FI_p,t-σp-1 − (Σ_{θ=t−σ_p}^{σ_p} S_pθ + Σ_{θ=t−σ_p}^{σ_p} FW_pθ)
26     FI_pt = FI_pt + FI_p,t-1 − FW_pt
27     if D_pt > 0
28         if D_pt ≤ FI_pt
29             S_pt = D_pt
30             FI_pt = FI_pt − S_pt
31         else
32             S_pt = FI_pt
33             FI_pt = 0
34             Δ_pt = D_pt − S_pt
35     if Δ_p,t-1 > 0
36         if Δ_p,t-1 ≤ FI_pt
37             S_pt = S_pt + Δ_p,t-1
38             FI_pt = FI_pt − Δ_p,t-1
39         else
40             S_pt = S_pt + FI_pt
41             FI_pt = 0
42             Δ_pt = Δ_pt + Δ_p,t-1 − S_pt
43     if FI_pt > F_p
44         FW_pt = FW_pt + FI_pt − F_p
45         FI_pt = F_p
```

are processed in *DSP* suite *j*. Line 18 accumulates the production from all *DSP* suites. Lines 19 and 20 assign the product p and *DSP* production time FT_{jpt} to the *DSP* part of the chromosome. Lines 21–23 ensure that the levels of *USP* inventory do not exceed the storage limit. Any excess inventory of product p during time period t is calculated as waste, CW_{pt}.

The amount of *DSP* waste, FW_{pt}, and inventory levels of product p at time period t, FI_{pt}, are calculated in Lines 24–26 similarly to Lines 9–11. In Line 27, if there is a demand, D_{pt}, for product p at time period t, then the amount of product sold, S_{pt}, is calculated based on the number of batches stored in *DSP* inventory FI_{pt}. If there are more batches in storage than there are in demand (Line 28), the variable S_{pt} will be equal to the value of demand (Line 29). Otherwise, Line 32 assigns the value of *DSP* storage FI_{pt} to S_{pt}, and the backlog is recorded using variable Δ_{pt} for that time period in Line 34. If the inventory allows it, the backlog from a previous time period $\Delta_{p,t-1}$ is sold in Lines 35–41. Otherwise, it is accumulated in Line 42. Lines 43–45 ensure that *DSP* storage capacity, F_p, is not exceeded by discarding any extra, unsold batches.

The fitness of each chromosome is assessed by calculating the profit achieved by the schedule using the same objective function (1) as presented by Lakhdar et al. (2005). The objective function value is equal to the difference between the total sales and the costs of manufacturing, product changeovers, intermediate and final product storage, waste disposal, and late deliveries.

$$Profit = \sum_p \sum_t \left(\nu_p S_{pt} - \sum_i \eta_p B_{ipt} - \sum_i \psi_p Z_{ipt} - \sum_j \eta_p B_{jpt} - \sum_j \psi_p Z_{jpt} - \rho_p CI_{pt} - \omega_p FI_{pt} - \delta_p \Delta_{pt} - \tau_p (CW_{pt} + FW_{pt}) \right) \quad (1)$$

2.2. Case study 2 – Long-term capacity planning and scheduling problem of multi-site, multi-product bio-manufacture

The goal of the planning problem presented in this case study is to generate a 15-yearlong production schedule to maximise manufacturing profits. The problems presented here is a single-objective problem adapted from Lakhdar et al. (2007). The following is a problem statement:

- Given:
  - A network of multi-product facilities $I = \{i1, i2, \ldots, i15\}$
  - Biopharmaceutical products $P = \{p1, p2, \ldots, p10\}$
  - A planning horizon of 15 years with equal time periods $T = \{t1, t2, \ldots, t60\}$
  - Production rates, yields, and lead times
  - Product lifetimes and storage capacities
  - Product demands and sales prices
  - Backlog decay factor
  - Manufacturing, changeover, storage costs, and late delivery penalties
  - Minimum and maximum campaign durations
- Determine:
  - Campaign durations and sequence of campaigns
  - Production quantities along with inventory profiles
  - Product sales and late deliveries profile
- To:
  - Maximise manufacturing profits

In their paper, Lakhdar et al. (2007) stated that the presented MILP model was an extension of the one already discussed in case study 1 of this paper. The core mathematical formulation for the single-objective problem remained mostly the same with the only most noticeable change being the lack of the explicit model of separate *USP* and *DSP* suites. Nevertheless, the complexity of the problem in case study 2 is much higher compared to case study 1 due to a greater number of products, facilities, and time periods. Each individual time period t is limited to a maximum of 87 days compared to 60 in case study 1, and there are 10 products which need to be allocated to 15 facilities. Additional subsets are introduced to define facility availability and manufacturing capability: PI_i, the set of products that can be manufactured in facility i, and TI_i, the set of time periods during which facility i is available for use. Furthermore, the production yield, as well as manufacturing costs of each product p, also depend on the facility i it is being manufactured in.

2.2.1. Chromosome structure

The increased complexity of the planning problem in case study 2 presented a challenge for the GA-based approach. Encoding the chromosomes as full-scale $|I|$-*by*-$|T|$ arrays was found to be computationally costly. A rolling horizon method was taken to explore the large search space in a more efficient manner by dividing the

Table 5
Pseudocode for the *dynamic* GA in case study 2.

```
1   for each subproblem
2       parents = GeneratePopulation()
3       EvaluateFitness(parents)
4       subproblem_best = max(parents)
5       gen = 0, not_restarted = true
6       while gen < max_gens
7           offspring = Select(parents)
8           Reproduce(offspring, pC, pMutP, pMutT)
9           EvaluateFitness(offspring)
10          Replace(parents, offspring)
11          current best = max(parents)
12          if current_best > subproblem_best
13              subproblem best = current_best
14          if subproblem_best == subproblem_best from 20 generations ago
15              if not_restarted
16                  parents = RestartSubproblem()
17                  not_restarted = false
18              else
19                  break
20          gen = gen + 1
21      ExtendFinalSolution(subproblem_best)
```

15-yearlong planning problem into 15 equal sub-problems which were solved consecutively. In order to accomplish this, each chromosome encoded a sub-problem as an $|I|$-*by*-$|T|$ array of product p and the length of production, T_{ipt}, values where $T \subset T$ and $|T| = 4$. T represents the extent of the rolling horizon, i.e. a dynamic subset of 4 time periods which correspond to the timeline of the sub-problem being solved. For example, $T = \{t1, t2, t3, t4\}$ and $T = \{t57, t58, t59, t60\}$ contain the time periods for the first and last sub-problems, respectively. The best solution from each sub-problem is stored in the final, full-scale $|I|$-*by*-$|T|$ solution, before proceeding to solve the following sub-problem. The values of the variables corresponding to the best solution such as the number of batches of product p produced in each facility i during time period t, B_{ipt}, are fixed so they would not need to be recalculated for the next sub-problem. To distinguish the rolling horizon approach-based GA from the standard one, it will be referred to it as the *dynamic GA*. The concept of the *dynamic* GA is illustrated in Fig. 3. The pseudo-algorithm for the *dynamic GA* is listed in Table 5.

A new parent population is generated for every sub-problem with the values of product p for each facility i selected randomly from the set of allowable products for that facility, PI_i, making sure the facility i is also available for use at time period $t \in TI_i$. A product label with a value of 0 is also included in the set to denote an idle state of the facility i during a time period t when no product is being manufactured.

2.2.2. Genetic algorithm

A uniform crossover operator with a probability *pC* (Table 6) is used to create two offspring from two parent chromosomes. T noise into the system, the mutation of the values of p and T is made independent, i.e. the probabilities *pMutP* and *pMutT* (Table 6) associated with each step are independent of one another. Provided that the facility i is available for use at time period t, the value of product label p is mutated by assigning 0 or a random value from the subset PI_i. The value of T_{ipt} is mutated by adding or subtracting a random number of days, ensuring the allowable range for T_{ipt} is not exceeded.

The parent population of $gen + 1$ is made up of the recombined and mutated offspring. If no better solutions are found, then the top 5% of the previous parent population replace the worst performers of the latest generation. A completely new parent population is generated when the best fitness value remains unchanged for a specified number of consecutive generations (Lines 14–16, Table 5) and the previous top 5% chromosomes are added. When

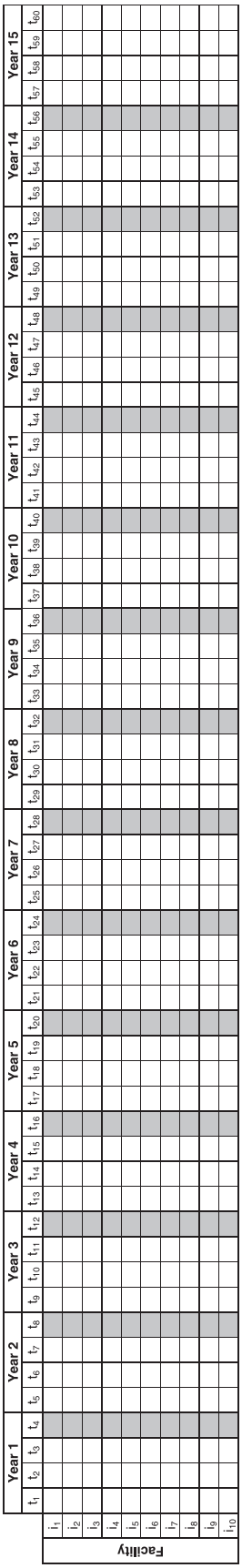

Fig. 3. An illustration of how the long-term capacity planning problem from case study 2 can be divided into smaller sub-problems. The full solution and each sub-problem *(Year n)* are $|I|$-*by*-$|T|$ and $|I|$-*by*-$|T|$ arrays respectively. The sub-problems overlap with one another on the parts that are shaded in grey.

Table 6
Case study 2 results and model statistics for MILP and GA models.

	MILP		GA	
	Global optimum	Relaxed	Dynamic	Standard
Max obj. function value	66,360	65,940	65,849[a]	61,880[a]
Time (s)	1000.36	8.77	8.09[b]	21.56[b]
Optimality gap	0%	0.6%	0.8%[c]	6.8%[c]
Avg. obj. function value[d]	–		$65{,}652 \pm 112$	$61{,}186 \pm 437$
Population size	–		200	200
Crossover rate, pC[e]	–		0.686	0.597
Mutation rate, $pMutP$[e]	–		0.004	0.001
Mutation rate, $pMutT$[e]	–		0.295	0.295
Elitism	–		5%	5%
Termination	–		25[f]	1000[g]

[a] Max obj. function value obtained from 50 independent GA runs.
[b] An average solution time of a single GA run.
[c] An optimality estimate relative to the global optimum obtained using the recreated MILP model, i.e. *1 – obj. function value / global optimum.*
[d] Average of best objective function values from 50 independent GA runs (mean ± 1 standard deviation)
[e] The parameter values were selected using the PSO algorithm.
[f] If the best objective function value remained unchanged for 25 consecutive generations, the GA was restarted with a new parent population. The second time the best objective function value stayed the same for the same number of generations, the GA was terminated.
[g] Terminated after 1000 generations had elapsed irrespective of the best objective function value achieved.

Table 7
Case study 2 results and model statistics for the dynamic GA model using different early stopping values and population sizes.

Avg. obj. function value[a]	Max obj. function value[b]	Avg. solution time	Population size	Termination[c]
$65{,}399 \pm 131$	65,653	3.91 s	100	25
$65{,}518 \pm 144$	65,799	6.11 s	100	50
$65{,}543 \pm 144$	65,818	8.30 s	100	75
$65{,}652 \pm 112$	**65,849**	**8.09 s**	**200**	**25**
$65{,}755 \pm 105$	65,934	12.87 s	200	50
$65{,}797 \pm 92$	65,987	17.20 s	200	75
$65{,}806 \pm 66$	65,921	12.66 s	300	25
$65{,}855 \pm 86$	65,997	19.86 s	300	50
$65{,}883 \pm 92$	66,068	26.85 s	300	75

[a] Average of best objective function values from 50 independent GA runs (mean ± 1 standard deviation).
[b] Max objective function value obtained from 50 independent GA runs.
[c] If the best objective function value remains unchanged for a given number of consecutive generations, the GA is restarted with a new parent population. The second time the best objective function value stays the same for the same number of generations, the GA is terminated.

Table 8
Pseudocode for fitness evaluation in case study 2.

```
1   ξ = 0
2   for each time period t in sub_problem
3       for each facility i
4           p = products_iξ
5           T_ipt = time_iξ
6           Z_ipt = 1 − (t > 0 and B_ip,t-1 == 0)
7           B_ipt = Z_ipt + r_ip(T_ipt − α_p Z_ipt)
8           K_ipt = B_ipt yd_ip
9           I_pt = I_pt + K_ipt
10      ξ = ξ + 1
11      for each product p
12          if t > ζ_p
13              W_pt = I_p,t-ζp-1 − ( Σ_{θ=t−ζ_p}^{ζ_p} S_pθ + Σ_{θ=t−ζ_p}^{ζ_p} W_pθ )
14          I_pt = I_pt + I_ip,t−1 − W_pt
15      if D_pt > 0
16          if D_pt ≤ I_pt
17              S_pt = D_pt
18              I_pt = I_pt − S_pt
19          else
20              S_pt = I_pt
21              I_pt = 0
22              Δ_pt = D_pt − S_pt
23      if Δ_p,t-1 ≥ 0
24          if Δ_p,t-1 ≤ I_pt
25              S_pt = S_pt + Δ_p,t-1
26              I_pt = I_pt − Δ_p,t−1
27          else
28              S_pt = S_pt + I_pt
29              I_pt = 0
30              Δ_pt = Δ_pt + π Δ_p,t−1 − S_pt
```

this repeats, the best sub-problem is added to the full solution, and a new sub-problem is started (Lines 14, 17–20, Table 5). We tested terminating the GA after 25, 50, and 75 generations with population sizes of 100, 200, and 300 chromosomes (Table 7).

2.2.3. Fitness evaluation

In Table 8, the variable ξ is used to iterate through the values of the $|I|$-*by*-$|T|$ array contained within each chromosome. The product label p and production length T_{ipt} are retrieved from the chromosomes in Lines 4 and 5. The value of the binary changeover variable Z_{ipt} is set to 1 in Line 6 if variable $B_{ip,t\text{-}1}$, the number of batches of product p produced in facility i in the previous time

period slot, is 0. The value of the number of batches variable B_{ipt} during time period t is calculated in Line 7 and converted into kilograms K_{ipt} using the yield conversion factor yd_{ip} in Line 8. The value of yd_{ip} depends the facility i which the product p is being manufactured in. Line 9 accumulates the value of K_{ipt} into the variable I_{pt} – the amount of product p in kilograms stored at time period t. The amount of product waste W_{pt} is estimated in Line 12 and 13. The value of this variable is equal to the amount of product p that was not sold and remained in storage for more than ζ_p time periods. The rest of the pseudocode in Table 8 (from Line 14 and onwards) is nearly identical to the Lines 27-45 from the pseudocode for the fitness evaluation in case study 1 (Table 3). The only notable differences are the lack of storage capacity constraint and the addition of backlog decay factor π which diminishes the importance of the backlogged orders during the fitness evaluation in case study 2.

The fitness of each chromosome is evaluated using the objective function of profit maximisation (2) defined by Lakhdar et al. (2007). The objective function value is equal to the difference between the total sales, with each batch sold at a price v_p, and the total operating costs which include the costs of manufacturing, product changeovers, storage, and late deliveries.

$$Profit = \sum_{p} \sum_{t \in TI_i} \left(v_p S_{pt} - \rho_p I_{pt} - \delta_p \Delta_{pt} - \sum_{i \in IP_i} \left(\eta_p B_{ipt} + \psi_p Z_{ipt} \right) \right) \quad (2)$$

3. Results and Discussion

In this section, the results to the case studies of capacity planning and scheduling of biomanufacture from the literature are presented. In case study 1, the problem consists of a multi-suite facility, with 2 USP ($I = \{i_1, i_2\}$) and 2 DSP ($J = \{j_1, j_2\}$) suites to produce 3 products ($P = \{p_1, p_2, p_3\}$) with multiple intermediate demand dates due during a 360-day long production time horizon. The horizon is discretised into 6 time periods ($T = \{t_1, t_2, \ldots, t_6\}$) of 60 days. In case study 2, the problem consists of 10 facilities ($I = \{i_1, i_2, \ldots, i_{10}\}$) with different manufacturing capabilities PI_i (subset of facilities capable of producing product p) and availability TI_i (subset of facilities available at time period t) to produce 15 products ($P = \{p_1, p_2, \ldots, p_{15}\}$) due annually during a 15-yearlong production time horizon. The horizon consists of 60 discrete time periods ($T = \{t_1, t_2, \ldots, t_{60}\}$) of 87 days.

The GAs discussed in the previous sections for case study 1 and case study 2 are used to solve the respective scheduling problems, and the results are compared with the recreated MILP models (see Tables 3 and 6). A comparison between the production schedules generated using MILP, and a GA is also provided in Figs. 4 and 5. The reader is advised to refer to the original papers (Lakhdar et al., 2007, 2005) for the input data used.

3.1. Case study 1 results

The proposed GA developed in this research was first applied to case study 1 on medium-term capacity planning for a single-site, multi-suite, multi-product biopharmaceutical facility. Initially, a MILP model was developed for the problem as a benchmark for comparison with the GA performance. In their original MILP work, Lakhdar et al. (2005) reported an objective function value of 487 relative monetary units (RMU) with a 5% optimality gap for this problem, while using the reproduced MILP model an objective function value of 490 RMU was achieved with 0% optimality gap indicating a global optimum.

		Time periods (t_n = 60 days)					
		t_1	t_2	t_3	t_4	t_5	t_6
Suites	USP 1	4 (60)	4 (50)	2 (50)	3 (60)	2 (40)	3 (60)
	USP 2	2 (54)	2 (44)	2 (44)	4 (60)	4 (50)	2 (50)
	DSP 1	3 (55)	5 (50)	2 (50)	3 (30)	2 (20)	5 (50)
	DSP 2	2 (52)	2 (20)	2 (20)	3 (55)	5 (50)	

Product 1 Product 2 Product 3

Fig. 4. Production schedule for case study 1 with an objective function value of 490 RMU and 0% optimality gap. Both the MILP model and the proposed GA generated the same schedule. The first number in each cell denotes the number of batches produced which is followed by the production time [days] in brackets. The shading of the box indicates which product is being manufactured.

The proposed PSO-based meta-optimisation approach was used to tune the crossover and mutation parameter values, *pC* and *pM*. Using this approach, the optimal values of crossover rate ($pC = 0.710$) and mutation rate ($pM = 0.070$) were identified, and the GA achieved the global optimum of 490 RMU for 100 consecutive, independent algorithm runs. The GA also generated a production schedule with the product allocation pattern identical to the one from the recreated MILP model (Fig. 4). The average solution time of the GA was 0.07 s compared to MILP which took an average of 0.22 s to find the global optimum (the MILP model was run 10 times to evaluate the running time).

Given the fast performance of the proposed GA-based method and the optimality of the results, it can be considered as a viable alternative for addressing medium-term capacity planning and scheduling problems similar in structure and complexity to case study 1.

3.2. Case study 2 results

Having tackled medium-term, single-site facility scheduling, the GA was then extended to address long-term planning across multi-site, multi-product biopharmaceutical manufacturing facilities in case study 2. To set the benchmark for the GA, the recreated single-objective MILP model was used to achieve an objective function value of 66,360 RMU with a 0% optimality gap for this problem. It took approximately 16.7 min to find the global optimum. With the optimality gap increased to 1%, the MILP model achieved an objective function value of 65,940 RMU in 8.77 s.

As discussed earlier, two versions of a GA (*standard* and *dynamic*) were applied to solve the long-term capacity planning problem presented in case study 2. Using the *standard* version, each chromosome encoded the full-scale problem as an $|I|$-*by*-$|T|$ array (where $|I| = 10$ and $|T| = 60$), and the GA was set to terminate after 1000 generations had elapsed. In the *dynamic* version, a rolling horizon approach was utilised to break down the full-scale 15-yearlong scheduling problem into 15 sub-problems. Each chromosome encoded only a part of the full schedule as an $|I|$-*by*-$|T|$ array (where $T \in T$ and $|T| = 4$) corresponding to the sub-problem being solved. Additionally, the *dynamic* GA was restarted once the fitness value remained unchanged for a set number of consecutive generations. The crossover, mutation and elitism steps were identical in both *standard* and *dynamic* versions.

The comparison of the results between the MILP and GA is summarised in Table 6. For a fair comparison, the PSO-based meta-optimisation was applied to tune both GA versions. After 50 runs, the average best objective function value using the *standard* GA was $61{,}186 \pm 437$ while the *dynamic* GA achieved $65{,}652 \pm 112$. Additionally, the solution time of the *dynamic* GA was approximately 2.7 times faster than that of the standard version (8.09 s vs 21.56 s) and slightly faster than that of the relaxed MILP model (8.09 s vs

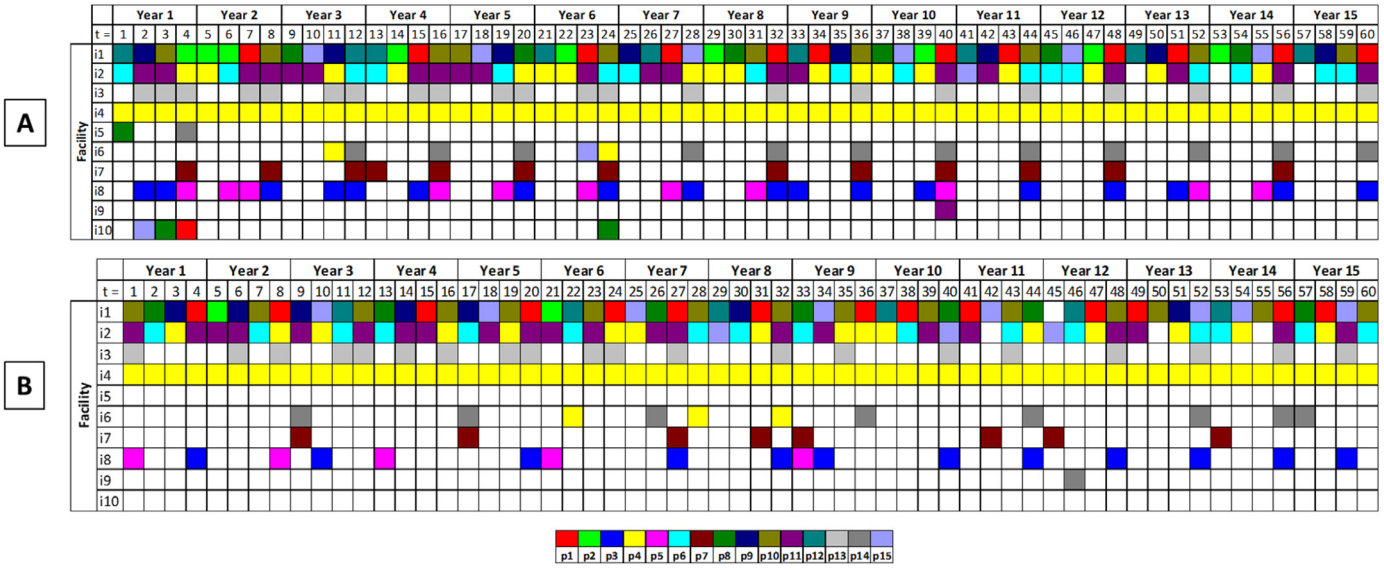

Fig. 5. Production schedules for case study 2. Each product p_n (where $n = 1, 2, \ldots, 15$) is denoted by a colour label displayed in the legend below the schedules. The numbers of batches of each product produced have been removed for clarity purposes.
A) Generated using the MILP model. An objective function value of 65,940 RMU was obtained with 0.6% optimality margin (based on the known global optimum as the upper bound).
B) Generated using the dynamic GA. An objective function value of 65,849 RMU was obtained (0.8% estimated optimality margin).

8.77 s). Using the known global optimum of 66,360 RMU as an upper bound, the average and the lowest optimality gaps achieved using the *dynamic* GA were estimated to be 1.0% and 0.8%, respectively. In comparison, the relaxed MILP model returned an objective function value of 65,940 with a 0.6% optimality gap

The pattern of the most profitable production schedule produced by the dynamic GA (Fig. 5B) holds a close resemblance to the MILP-generated one (Fig. 5A), for example:

- Facilities i_1 and i_2 run with little to no idle time and with a variety of different products allocated to them.
- Facility i_3 is busier in the first half of the scheduling table with more product allocations.
- Product p_4 is almost exclusively produced in the facility i_4.
- Facility i_4 has no idle time periods.
- Certain facilities such as i_5 and i_{10} are completely idle.

Nevertheless, the production schedule generated with MILP is more systematic and has more products allocated overall. For example, in Fig. 5A, the product allocation in the facility i_8 is more consistent than in Fig. 5B, in which the production is scheduled as late as possible thus saving storage costs.

4. Conclusions

In this work, novel GA-based optimisation approaches were developed for medium and long-term, discrete-time, mixed-integer capacity planning models of biopharmaceutical manufacture. The key enabling features of the GA-based approaches included a chromosome encoding strategy, a dynamic, rolling horizon approach to improving the performance of the GA for tackling the long-term capacity planning and scheduling problem, and algorithms that captured capacity planning objectives for multiple products across multiple suites and facilities. A PSO-based meta-optimisation method was also presented for automatic tuning of crossover and mutation parameter values as an alternative to manual tuning. The feasibility of the novel GA-based optimisation algorithms was demonstrated on two case studies from the literature.

In case study 1, a medium-term capacity planning problem of a single-site, multi-suite biopharmaceutical facility was solved. The proposed GA obtained the global optimum with less CPU time than a MILP model.

In case study 2, a more computationally complex, long-term capacity planning problem of a multi-site biopharmaceutical manufacture was solved. Using the rolling horizon approach, the full-scale problem was divided into 15 sub-problems which were solved consecutively. Based on the known global optimum, the average optimality gap of the solution generated using the dynamic GA was 1.0%.

In this paper, the GA-based approaches were demonstrated to be capable of achieving the exact or very similar performance compared to MILP in terms of the objective function optimality and speed. This paper serves as a starting point for tackling even more complex capacity planning and scheduling problems of biopharmaceutical manufacture using GA-based approaches. In the future, the work presented in this paper could be extended to address stochastic planning problems with uncertain parameters, e.g. fermentation titres or product demand, as well as multi-objective problems.

Acknowledgements

Financial support from the UK Engineering and Physical Sciences Research Council (EPSRC) and Eli Lilly & Co. is gratefully acknowledged. UCL Biochemical Engineering hosts the Future Targeted Healthcare Manufacturing Hub in collaboration with UK universities and with funding from the UK Engineering & Physical Sciences Research Council (EPSRC) and a consortium of industrial users and sector organisations. This is funded by an Industrial Case Award supported by EPSRC and Eli Lilly. The grant code is EP/L505717/1.

Appendix A

This appendix summarises the mathematical model presented by Lakhdar et al. (2005).

Production constraints

Constraints 1 and 2 represent the manufacture of product in *USP* and *DSP* suites. Upstream production, B_{ipt}, and downstream

production, B_{jpt}, are represented by continuous rates of production, CR_p and FR_p, which are combined with their respective *USP* and *DSP* lead times, α_p and β_p, and *USP* and *DSP* production times, CT_{ipt} and FT_{jpt}. Constraints 3 and 4 activate lead time in *USP* suite i and *DSP* suite j if the same product p has not been manufactured in the preceding time period, t – 1. Constraints 5 and 6 ensures that only one product p is produced in any *USP* suite i and *DSP* suite j at any time period t.

$$B_{ipt} = Z_{ipt} + CR_p\left(CT_{ipt} - \alpha_p Z_{ipt}\right) \ \forall i, p, t \tag{1}$$

$$B_{jpt} = Z_{jpt} + FR_p\left(FT_{jpt} - \beta_p Z_{jpt}\right) \ \forall j, p, t \tag{2}$$

$$Z_{ipt} \geq Y_{ipt} - Y_{ip,t-1} \ \forall i, p, t \tag{3}$$

$$Z_{jpt} \geq Y_{jpt} - Y_{jp,t-1} \ \forall j, p, t \tag{4}$$

$$\sum\nolimits_p Y_{ipt} \leq 1 \ \forall i, t \tag{5}$$

$$\sum\nolimits_p Y_{jpt} \leq 1 \ \forall j, t \tag{6}$$

Timing constraints

Constraints 7 and 8 represent the appropriate minimum and maximum production times for *USP* and *DSP* suites, which are only activated when Y_{ipt} and Y_{jpt} are equal to 1. Constraints 9 and 10 ensure that the total *USP* or *DSP* time does not exceed the specified production time horizon, H_t.

$$CT_p^{min} Y_{ipt} \leq CT_{ipt} \leq CT_p^{max} Y_{ipt} \ \forall i, p, t \tag{7}$$

$$FT_p^{min} Y_{jpt} \leq FT_{jpt} \leq FT_p^{max} Y_{jpt} \ \forall j, p, t \tag{8}$$

$$\sum\nolimits_p CT_{ipt} \leq H_t \ \forall i, t \tag{9}$$

$$\sum\nolimits_p FT_{jpt} \leq H_t \ \forall j, t \tag{10}$$

Storage constraints

Constraints 11 and 12 enforce an inventory balance in upstream and downstream production and force the total downstream production to meet the product demand. Constraints 13 and 14 ensure that the amount of upstream and downstream product stored over timer period t is positive and below the maximum available storage capacities, C_p and F_p. Both upstream and downstream product inventory is constrained by the limited product shelf-life. Constraints 15 and 16 ensure the total amount of stored upstream product and downstream product is used after the next ζ_p or σ_p time periods, respectively.

$$CI_{pt} = CI_{p,t-1} + \sum\nolimits_i B_{ipt} - \frac{1}{\lambda_p}\sum\nolimits_j B_{jpt} - CW_{pt} \ \forall p, t \tag{11}$$

$$FI_{pt} = FI_{p,t-1} + \sum\nolimits_j B_{jpt} - S_{pt} - FW_{pt} \ \forall p, t \tag{12}$$

$$0 \leq CI_{pt} \leq C_p \ \forall p, t \tag{13}$$

$$0 \leq FI_{pt} \leq F_p \ \forall p, t \tag{14}$$

$$CI_{pt} \leq \sum\nolimits_j \sum\nolimits_{\theta=t+1}^{t+\zeta_p} B_{jp\theta} \ \forall p, t \tag{15}$$

$$FI_{pt} \leq \sum\nolimits_{\theta=t+1}^{t+\sigma_p} S_{p\theta} \ \forall p, t \tag{16}$$

Backlog constraints

Constraint 17 penalises the amount of product p that was late for delivery at time period t, Δ_{pt}.

$$\Delta_{pt} = \Delta_{p,\ t-1} + D_{pt} - S_{pt} \ \forall p, t \tag{17}$$

Objective function

The objective function is to maximise profit which is equal to the difference between total sales and total operating costs. All costs and prices are in relative monetary units (RMU).

$$\begin{aligned} \max Profit = \ & \sum\nolimits_p \sum\nolimits_t (\nu_p S_{pt} - \sum\nolimits_i \eta_p B_{ipt} - \sum\nolimits_i \psi_p Z_{ipt} - \sum\nolimits_j \eta_p B_{jpt} \\ & - \sum\nolimits_j \psi_p Z_{jpt} - \rho_p CI_{pt} - \ \omega_p FI_{pt} - \ \delta_p \Delta_{pt} \\ & - \ \tau_p (CW_{pt} + FW_{pt})) \end{aligned} \tag{18}$$

Appendix B

This appendix summarises the mathematical model presented by Lakhdar et al. (2007).

Production constraints

Constraint 1 represents biopharmaceutical production. The number of batches produced in facility i of product p at time period t, B_{ipt}, is represented by a continuous production rate, r_{ip}, production lead time, α_{ip}, and production time T_{ipt}. Constraint 2 converts the integer number of batches, B_{ipt}, into kilograms, K_{ipt}, using a yield conversion factor, yd_{ip}. Constraint 3 activates lead time in facility i if the same product p has not been manufactured in the preceding time period, $t-1$. Constraint 4 ensures that only one product p is produced in any facility i at any time period t.

$$B_{ipt} = Z_{ipt} + r_{pt}\left(T_{ipt} - \alpha_{ip} Z_{ipt}\right) \ \forall i, p \in PI_i, \ t \in TI_i \tag{1}$$

$$K_{ipt} = B_{ipt} yd_{ip} \ \forall i, p \in PI_i, t \in TI_i \tag{2}$$

$$Z_{ipt} \geq Y_{ipt} - Y_{ip,t-1} \ \forall i, p \in PI_i, t \in TI_i \tag{3}$$

$$\sum\nolimits_{p \in PI_i} Y_{ipt} \leq 1 \ \forall i, t \in TI_i \tag{4}$$

Timing constraints

Constraints 5 and 6 represent the appropriate minimum and maximum campaign durations, T_{ip}^{min} and T_{ip}^{max}, which are only activated when Y_{ipt} is equal to 1.

$$T_{ip}^{min} Y_{ipt} \leq T_{ipt} \ \forall i, p \in PI_i, t \in TI_i \tag{5}$$

$$T_{ipt} \leq \min\left\{T_{ip}^{max}, \ H_t\right\} Y_{ipt} \ \forall i, p \in PI_i, t \in TI_i \tag{6}$$

Storage xonstraints

Constraint 7 enforces inventory balance for production and forces the total production to meet the product demand. Constraint 9 enforces that the amount of product p in inventory at time period t is below the maximum storage capacity, C_p, while the constraint 10 ensures that the global storage capacity, C_p^{tot}, is not exceeded. The duration a product can be is stored in inventory is limited by the constraint 10.

$$I_{pt} = I_{p,t-1} + \sum\nolimits_i K_{ipt} - S_{pt} - W_{pt} \ \forall p \in PI_i, t \in TI_i \tag{7}$$

$$0 \leq I_{pt} \leq C_p \ \forall p, t \tag{8}$$

$$0 \leq \sum_p I_{pt} \leq C_p^{tot} \ \forall t \tag{9}$$

$$I_{pt} \leq \sum_{\theta=t+1}^{t+\zeta_p} S_{p\theta} \ \forall p, t \tag{10}$$

Backlog constraints

Constraint 11 penalises the amount of product p that was late for delivery at time period t, Δ_{pt}.

$$\Delta_{pt} = \pi_p \Delta_{p,\ t-1} + D_{pt} - S_{pt} \ \forall p, t \tag{11}$$

Objective function

The objective function is to maximise profit which is equal to the difference between total sales and total operating costs. All costs and prices are in relative monetary units (RMU).

$$\max Profit = \sum_p \sum_{t \in TI_i} (\nu_p S_{pt} - \rho_p I_{pt} - \delta_p \Delta_{pt} - \sum_{i \in IP_p} \left(\eta_{ip} B_{ipt} + \psi_{ip} Z_{ipt}\right)) \tag{12}$$

References

Allmendinger, R., Simaria, A.S., Farid, S.S., 2012. Efficient discovery of chromatography equipment sizing strategies for antibody purification processes using evolutionary computing. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 468–477.

Back, T., 1993. Optimal mutation rates in genetic search. In: Proceedings of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp. 2–8.

Brie, A.H., Morignot, P., 2005. Genetic planning using variable length chromosomes. In: ICAPS, pp. 320–329.

Camilleri, M., Neri, F., Papoutsidakis, M., 2014. An algorithmic approach to parameter selection in machine learning using meta-optimization techniques. WSEAS Trans. Syst. 13, 202–213.

Chen, M.-S., Liao, F.H., 1998. Neural networks training using genetic algorithms. In: Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on, 3. IEEE, pp. 2436–2441.

Chipperfield, A., Fleming, P., 1995. The MATLAB genetic algorithm toolbox. In: Applied Control Techniques Using MATLAB, IEE Colloquium on. IET, pp. 10/11–10/14.

Deb, K., 2001. Multi-Objective Optimization Using Evolutionary Algorithms, 16. John Wiley & Sons.

DiMasi, J.A., Hansen, R.W., Grabowski, H.G., 2003. The price of innovation: new estimates of drug development costs. J. Health Econ. 22, 151–185.

Eberhart, R.C., Shi, Y., 2000. Comparing inertia weights and constriction factors in particle swarm optimization. In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, 1. IEEE, pp. 84–88.

Eiben, Á.E., Hinterding, R., Michalewicz, Z., 1999. Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. 3, 124–141.

Gaffney, J., Pearce, C., Green, D., 2010. Binary versus real coding for genetic algorithms: a false dichotomy? ANZIAM J. 51, 347–359.

George, E., Titchener-Hooker, N.J., Farid, S.S., 2007. A multi-criteria decision-making framework for the selection of strategies for acquiring biopharmaceutical manufacturing capacity. Comput. Chem. Eng. 31, 889–901.

George, E.D., Farid, S.S., 2008. Strategic biopharmaceutical portfolio development: an analysis of constraint-induced implications. Biotechnol. Prog. 24, 698–713.

Goldberg, D.E., Deb, K., 1991. A comparative analysis of selection schemes used in genetic algorithms. Found. Genet. Algorithms 1, 69–93.

Grefenstette, J.J., 1986. Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man Cybern. 16, 122–128.

Hassan, R., Cohanim, B., de Weck, O., Venter, G., 2004. A Comparison of Particle Swarm Optimization and the Genetic Algorithm. American Institute of Aeronautics and Astronautics.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. U Michigan Press.

James, K., Russell, E., 1995. Particle swarm optimization. In: Proceedings of 1995 IEEE International Conference on Neural Networks, pp. 1942–1948.

Janikow, C.Z., Michalewicz, Z., 1991. An experimental comparison of binary and floating point representations in genetic algorithms. In: ICGA, pp. 31–36.

Kaitin, K., DiMasi, J., 2010. Pharmaceutical innovation in the 21st century: new drug approvals in the first decade, 2000–2009. Clin. Pharmacol. Ther. 89, 183–188.

Lakhdar, K., Papageorgiou, L.G., 2006. An iterative mixed integer optimisation approach for medium term planning of biopharmaceutical manufacture under uncertainty. Chem. Eng. Res. Des. 86, 259–267.

Lakhdar, K., Savery, J., Papageorgiou, L., Farid, S., 2007. Multiobjective long-term planning of biopharmaceutical manufacturing facilities. Biotechnol. Prog. 23, 1383–1393.

Lakhdar, K., Zhou, Y., Savery, J., Titchener-Hooker, N.J., Papageorgiou, L.G., 2005. Medium term planning of biopharmaceutical manufacture using mathematical programming. Biotechnol. Prog. 21, 1478–1489.

Lin, W.-Y., Lee, W.-Y., Hong, T.-P., 2003. Adapting crossover and mutation rates in genetic algorithms. J. Inf. Sci. Eng. 19, 889–903.

Luke, S., 2013. Essentials of Metaheuristics. Lulu Com.

Melanie, M., 1996. An Introduction to Genetic Algorithms.

Nie, W., Zhou, Y., Simaria, A.S., Farid, S.S., 2012. Biopharmaceutical portfolio management optimization under uncertainty. In: Symposium on Computer Aided Process Engineering, 17, p. 20.

Pandey, S., Wu, L., Guru, S.M., Buyya, R., 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE, pp. 400–407.

Papageorgiou, L.G., Rotstein, G.E., Shah, N., 2001. Strategic supply chain optimization for the pharmaceutical industries. Ind. Eng. Chem. Res. 40, 275–286.

Paul, S.M., Mytelka, D.S., Dunwiddie, C.T., Persinger, C.C., Munos, B.H., Lindborg, S.R., Schacht, A.L., 2010. How to improve R&D productivity: the pharmaceutical industry's grand challenge. Nat. Rev. Drug Discovery 9, 203–214.

Poli, R., Kennedy, J., Blackwell, T., 2007. Particle swarm optimization. Swarm Intell. 1, 33–57.

Ransohoff, T.C., 2004. Considerations impacting the make vs. buy decision. Am. Pharm. Outsourcing 5, 52–63.

Schaffer, J.D., Morishima, A., 1987. An adaptive crossover distribution mechanism for genetic algorithms. In: Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, NJ. Lawrence Erlbaum Associates, Inc, pp. 36–40.

Shanley, A., 2014. Tufts' New Figures on Drug-Development Costs Spark Debate, 2016. BioPharm International BioPharm International.com.

Siganporia, C.C., Ghosh, S., Daszkowski, T., Papageorgiou, L.G., Farid, S.S., 2014. Capacity planning for batch and perfusion bioprocesses across multiple biopharmaceutical facilities. Biotechnol. Prog..

Simaria, A.S., Turner, R., Farid, S.S., 2012. A multi-level meta-heuristic algorithm for the optimisation of antibody purification processes. Biochem. Eng. J. 69, 144–154.

Ting, C.-K., Lee, C.-N., Chang, H.-C., Wu, J.-S., 2009. Wireless heterogeneous transmitter placement using multiobjective variable-length genetic algorithm. IEEE Trans. Syst. Man Cybern. Part B 39, 945–958.

Trelea, I.C., 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf. Process. Lett. 85, 317–325.