Int. J. Production Economics 190 (2017) 146–157

Contents lists available at ScienceDirect

Int. J. Production Economics

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

A tool for solving stochastic dynamic facility layout problems with stochastic demand using either a Genetic Algorithm or modified Backtracking Search Algorithm

Srisatja Vitayasak [a], Pupong Pongcharoen [a,*], Chris Hicks [b]

[a] Centre of Operations Research and Industrial Applications (CORIA), Department of Industrial Engineering, Faculty of Engineering, Naresuan University, Phitsanulok 65000, Thailand

[b] Newcastle University Business School, University of Newcastle upon Tyne, 5 Barrack Road, Newcastle upon Tyne NE1 7RU, UK

ARTICLE INFO

Article history:
Received 30 November 2015
Accepted 14 March 2016
Available online 24 May 2016

Keywords:
Backtracking Search Algorithm
Genetic Algorithm
Dynamic facility layout problem
Stochastic demand

ABSTRACT

Facility layout problems (FLP) involve determining the optimal placement of machines within a fixed space. An effective layout minimises costs. The total material travel distance is a key indicator of the efficiency of internal logistics. Changes in demand and product mix may alter the material flow. The dynamic facilities layout problem (DFLP) takes into account changes in demand and allows for the periodic redesign of facilities. Facility redesign may reduce the material flow cost, but there is a trade-off between material flow improvements and reorganisation costs. There is a limited literature on the re-design of facilities with stochastic demand, heterogeneous-sized resources and rectilinear material flow.

The Backtracking Search Algorithm (BSA) has been used to successfully solve a range of engineering problems, but it has not previously been used to solve operations management problems or the FLP. This paper outlines novel modified Backtracking Search Algorithms (mBSAs) that solved the stochastic DFLP with heterogeneous sized resources. The combination of material flow and redesign costs were minimised. Three mBSA were benchmarked against the classical BSA and a Genetic Algorithm (GA) using 11 benchmark datasets obtained from the literature. The best mBSA generated better solutions than the GA for large-size problems. The total costs for the layouts generated by the best mBSA were significantly lower than for the conventional BSA. The modifications to the BSA increased the diversity of candidate solutions, which increased the amount of exploration. The computational time required by the three mBSAs was up to 70% less than the GA.

1. Introduction

An effective facility layout design reduces manufacturing lead times and increases throughput, overall productivity and efficiency (El-Baz, 2004). The facilities layout problem (FLP) has been defined as "arranging m indivisible departments (each with area a_i) within a given space" (Bozer and Meller, 1997, p. 549). The total distance travelled by materials is a commonly used proxy for measuring the efficiency of layouts, indeed much of the research on the optimisation of layout has sought to minimise some function related to the travel of parts (Drira et al., 2007). The facilities layout problem may be classified as a non-deterministic polynomial time hard problem (Ertay et al., 2006; Pourvaziri and Naderi, 2014) which means that the computational time required to solve problems increases exponentially with problem size. Stochastic search algorithms tend to be most suitable for solving such problems (Nagar et al., 1995). The problems has been solved using Genetic Algorithms (Tam, 1992; Tate and Smith, 1993; Mavridou and Pardalos, 1997; Hicks, 2004), Simulated Annealing (Moslemipour and Lee, 2012), Ant Colony Optimisation (Corry and Kozan, 2004; Lutuksin and Pongcharoen, 2010; Thepphakorn et al., 2014), Bat Algorithm (Dapa et al., 2012), and Biologically-Based Optimisation (Sooncharoen et al., 2015).

Uncertainty in demand may arise due to the actions of competitors, changing consumer preferences, technological innovations, new regulations, unanticipated model changeovers and variable production schedules (Sethi and Sethi, 1990; Chan and Malmborg, 2010). Dynamic facility layout problems (DFLP) take into account anticipated changes in material flow over multiple periods (weeks, months, or years). The layout may be redesigned for each period to minimise costs (Drira et al., 2007). However, redesign costs arise that include labour, equipment and lost production (McKendall et al., 2006; Moslemipour and Lee, 2012). Therefore the layout is only changed if the reduction in material

* Coresponding author.
E-mail addresses: pupongp@nu.ac.th, pupongp@gmail.com (P. Pongcharoen).

http://dx.doi.org/10.1016/j.ijpe.2016.03.019
0925-5273/

flow costs exceeds the redesign cost. The DFLP has been solved using various approaches including: the Quadratic Assignment Problem (Yang and Brett, 1998), Genetic Algorithms (Mazinani et al., 2013), Simulated Annealing (Kia et al., 2013) and Tabu Search (Chang et al., 2013).

A literature search that used the ISI Web of Science database for the period 2001–2014 found that Genetic Algorithms (GA) were the most popular algorithm used for solving the FLP. However, it has been reported that evolutionary algorithms can be sensitive to control parameters and suffer from slow computation and premature convergence (Civicioglu, 2013).

The Backtracking Search Algorithm (BSA) is a new evolutionary algorithm that was designed to be relatively simple with a single control parameter (Civicioglu, 2013). It can quickly solve numerical optimisation problems to produce good solutions. The BSA includes search within the current and previous populations. The BSA uses three genetic operators: selection, mutation and crossover. It has a random mutation strategy that is applied to a single individual (chromosome) and a non-uniform crossover strategy. It generates trial populations and controls the amplitude of the search-direction matrix and search-space boundaries to give powerful exploration and exploitation capacities. It was claimed that it can solve benchmark problems for: numerical function optimisation (Karaboga and Basturk, 2007), real-parameter optimisation (Suganthan et al., 2005), and real world optimisation (Das and Suganthan, 2010) more successfully than six other evolutionary algorithms (Civicioglu, 2013). These included the following: the Adaptive Differential Evolution Algorithm (ADEA) (Brest et al., 2006); the Artificial Bee Colony (Karaboga and Basturk, 2007); the Comprehensive Learning Particle Swarm Optimiser (CLPSO) (Liang et al., 2006); the Covariance Matrix Adaptation Evolutionary Strategy (CMAES) (Igel et al., 2007); Particle Swarm Optimisation (Kennedy and Eberhart, 1995); and the self-adaptive differential evolution algorithm (SADE) (Qin and Suganthan, 2005). The BSA has been used to successfully solve several engineering problems, including: power system optimisation (Kılıç, 2014; Rezaee Jordehi, 2015), the economic dispatch problem (Modiri-Delshad and Abd Rahim, 2014), non-aligned thrust optimisation (Kolawole and Duan, 2014), the localisation problem (De Sá et al., 2014), constrained optimisation problems (Zhao et al., 2014) and nonlinear engineering optimisation problems (Song et al., 2015). The BSA has not previously been used to solve operations management or facilities layout problems.

The aim of the research on which this paper is based, was to design, implement and evaluate a tool for solving stochastic dynamic facility layout problems that included the Backtracking Search Algorithm, a Genetic Algorithm, and three modified BSAs. The performance objective was to minimise the total cost which comprised material flow and redesign costs.

The paper is organised as follows. The literature relating to facilities layout problems with demand uncertainty, the Genetic Algorithm and the Backtracking Search Algorithm is reviewed in Sections 2, 3, and 4 respectively. Section 5 describes the problem formulation. Section 6 outlines the development of the Genetic Algorithm, the Backtracking Search Algorithm and modified BSAs that were used for solving facilities layout problems. The computational experiments are presented in Section 7. Section 8 includes discussion and conclusions and highlights the contributions of the work.

2. Facilities layout problem

Azadivar and Wang (2000, p. 4369) defined the facility layout problem (FLP) as "the determination of the relative locations for, and the allocation of, the available space among a number of workstations". The overall facility layout procedure comprises two phases: (1) the block layout phase that specifies the relative location and size of each department; and (2) the detailed layout phase which determines the exact locations, aisle structures, entry/exit points locations and the layout within each department (Drira et al., 2007). Block layouts normally represent resources as rectangles (Askin and Standridge, 1993). Some methods for solving block layout problems use a grid, whereas others consider a continuous plane where the blocks can be positioned at any point (Lee and Kim, 2000). Block layout methods may consider equally spaced blocks or the size of the blocks may vary (Castillo et al., 2005).

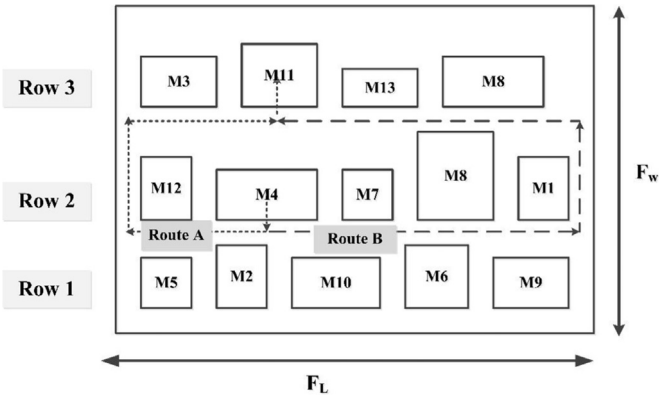


Fig. 1. Example of multiple-row machine layout design (modified from Leechai et al. (2009)).

A typical block layout with rectangular shaped resources of varying sizes, arranged in multiple rows, is shown in Fig. 1 (Leechai et al., 2009). A typical placement algorithm assigns machines to rows, starting at the bottom left hand side, then moves to the next row when a space constraint is encountered (Hicks, 2006). In the example, the flow paths, shown as dashed lines, represent the rectilinear movement of material handling equipment, e.g. automated guided vehicles, which can move to left or right side of the row and then move up or down to the destination row. A common objective is to minimise the rectilinear distance travelled by the material. For example, in Fig. 1a routing includes movement from M4 to M11 – route A is shorter than route B, so route A would be assumed.

Most of the research on the facilities layout problem has assumed a static model where conditions remain constant over a long period. This is known as the static facility layout problem (SFLP) (Drira et al., 2007). However, changes in demand and product mix can arise from the introduction of new products, the discontinuation of existing products, shorter product life cycles and market fluctuations (Sahin and Turkbey, 2009, p. 6856). These issues can result in changes to the material flow (Sethi and Sethi, 1990; Chan and Malmborg, 2010). The dynamic facility layout problem (DFLP) has been defined as "the minimisation of material flow costs in all periods plus the rearrangement costs for a series of SFLPs. In a DFLP, the rearrangement costs are added whenever an area contains different departments in consecutive time periods. In summary, the total sum of the rearrangement costs and the material flow costs are minimised" (Drira et al., 2007, p. 6856). Rearrangement costs include labour, equipment and lost production (McKendall et al., 2006; Moslemipour and Lee, 2012). The redesign costs for each resource can be either equal (Corry and Kozan, 2004) or unequal (Chen, 2013). The cost may be based on the number of moved machines (Corry and Kozan, 2004) or the total distance that machines are moved (Montreuil and Laforge, 1992). The redesign costs incurred between periods can be accumulated and compared to a budget (Baykasoglu et al., 2006). The

material handling costs (flow costs) may be estimated by multiplying the material flow distance by the cost per distance travelled (Chan and Malmborg, 2010). The demand profiles for each time period can be based upon forecasts (Ertay et al., 2006) or some statistical distribution can be assumed such as the uniform distribution (Krishnan et al. 2009; Jithavech and Krishnan, 2010), normal distribution (Tavakkoli-Moghaddam et al., 2007) or exponential distribution (Chan and Malmborg, 2010). Fuzzy numbers have also been used for representing stochastic flow between facilities (Enea et al., 2005).

3. Genetic Algorithms

The Genetic Algorithm (GA) is a population-based, nature-inspired algorithm (Goldberg, 2002; Gen et al., 2008; Yang, 2008). The approach is based upon an analogy with biological evolution. The probability of an individual surviving or reproducing is determined by its fitness. The GA algorithm starts by encoding the problem to produce a list of genes that may be represented as numeric or alphanumeric characters. The genes are then randomly combined to produce a population of chromosomes, each of which represents a possible solution. Offspring are produced by crossover and mutation genetic operations which are performed on chromosomes that are randomly selected from the population. The fitness of these chromosomes determines their probability of survival (Pongcharoen et al., 2002; Pongcharoen et al., 2004). Their survival is determined by their fitness.

The crossover and mutation operators provide mechanisms for exploitative and explorative search respectively (Gen and Cheng, 1997). GAs have been successfully applied to solve many science and engineering problems especially in production and operations management (Aytug et al., 2003; Chaudhry and Luo, 2005; Pongcharoen et al., 2008; Thapatsuwan et al., 2009). GAs have been used to solve static and dynamic FLPs (Drira et al., 2007). Dunker et al. (2005) presented an algorithm that combined dynamic programming and genetic search for solving a DFLP with departments of unequal size. Jithavech and Krishnan (2010) developed a GA that used a simulation approach to quantify uncertainties in demand. This was used to produce robust configurations that minimised the risks associated with the department layout design. Mazinani et al. (2013) developed an approach for determining the positions of departments for multiple periods that minimised the sum of material handling and rearrangement costs. They assumed deterministic flow between departments and random rearrangement costs. Krishnan et al. (2008) developed a Genetic Algorithm approach for solving the stochastic dynamic facilities layout problem.

4. Backtracking Search Algorithm

The Backtracking Search Algorithm (BSA) developed by Civicioglu (2013) is a population-based iterative evolutionary algorithm that was designed to achieve global optimisation. It can be efficiently used for highly nonlinear, multivariable, and multimodal function optimisation problems (Civicioglu, 2013). The BSA has a simple structure; so that it can be easily adapted to different numerical optimisation problems. The algorithm is robust, easy to implement, and has fewer control parameters to tune than typical evolutionary algorithms and it is not over sensitive to the initial values used (Song et al., 2015).

The data structures used by the BSA and Genetic Algorithms are equivalent, but different terminology is used (see Table 1). The BSA has three parameters: the mix rate for the crossover process; the population size and the maximum number of cycles. In comparison the GA has four parameters (the population size, the number of generations, the probability of crossover, and the probability of mutation). The BSA uses a 'dual-population' algorithm that uses the current and previous populations, which gives it a 'memory' (Lin et al., 2015). The BSA's strategies for generating trial populations and controlling the amplitude of the search-direction matrix and search-space boundaries give it very powerful global exploration and local exploitation capabilities (Civicioglu, 2013).

Table 1
A comparison of Genetic Algorithm and Backtracking Search Algorithm terminology.

Genetic Algorithm	Backtracking Search Algorithm
Gene	Element
Chromosome	Individual
Population	Population
Generation	Iteration
Number of generations	Maximum number of cycles
Probability of crossover/mutation	Mix rate

The BSA has been used to solve engineering problems, but there are no examples of its use for solving operations management problems or layout problems. Modiri-Delshad and Abd Rahim (2014) compared the BSA's performance to several alternative classical and evolutionary methods including Genetic Algorithms, improved evolutionary programming, modified Particle Swarm Optimisation and pattern search for solving four economic dispatch (ED) test cases. The objective of ED problems is to determine how power is shared amongst power system generating units to meet electrical demand, whilst minimising cost and satisfying system constraints. For the ED problems tested, the BSA produced high quality results that converged to a lower cost than the other methods. The BSA has also been used for solving the optimal power flow (OPF) problem, which may be defined "as meeting customers' energy requirements with the minimum cost of energy generation" (Kılıç, 2014, p.1). For the (OPF) problem, the minimum cost obtained by the BSA was better than the other algorithms tested (Genetic Algorithm, Simulated Annealing, Ant Colony Optimisation, and Tabu Search) (Kılıç, 2014). Song et al. (2015) used the Backtracking Search Algorithm for surface wave analysis in geophysics and compared the results with those produced by a Genetic Algorithm (Song et al., 2015). The results produced by the BSA were better than the GA in terms of accuracy and the convergence rate.

Askarzadeh and Coelho (2014) investigated the improvement, evaluation, management, and optimisation of proton exchange membrane fuel cells (PEMFCs). The performance of the cells is dependent upon parameters related to nonlinearities associated with the electrochemical processes. This is a complex problem because the impact of each parameter varies according to various polarisation curves which are different. It was found that the BSA produced results that were better than a wide range of other optimisation approaches including the bird mating optimiser (Askarzadeh, 2013) and differential evolution (Chakraborty et al., 2012; Gong and Cai, 2013).

Duan and Luo (2014) developed an adaptive BSA, which varied the probabilities of crossover and mutation according to the fitness values of solutions. The approach was used to solve induction magnetometer optimisation problems. The adaptive algorithm generated better solutions than the basic BSA, the Differential Evolution Algorithm (Brest et al., 2006), Particle Swarm Optimisation (Kennedy and Eberhart, 1995) and the Artificial Bee Colony (Karaboga and Basturk, 2007). Das et al. (2014) combined the BSA with the Differential Evolution Algorithm to solve interference suppression problems associated with linear antenna arrays. The

approach produced higher quality solutions which converged more quickly than a range of other evolutionary approaches.

5. Problem formulation

The following assumptions were made in order to formulate the problem: (i) the material flow distance between machines was measured using the rectilinear distance between the machines' centroids (Krishnan et al., 2009; Jithavech and Krishnan, 2010; Pillai et al., 2011); (ii) the machines were arranged in multiple parallel rows (Leechai et al., 2009); (iii) there was enough space on the shop floor for the machines to be arranged; (iv) the movement of materials was in rectilinear straight lines; (v) when the layout was redesigned the machines were moved in rectilinear straight lines; (vi) the gap between machines was predefined and constant; (vii) the demand profiles were obtained from empirical data (when the demand in each time period was known) and by using probability distributions (e.g. exponential, normal distribution, or uniform); (viii) the demand patterns for different products were independent; (iv) the redesign cost was determined by the rectilinear distance that machines were moved; (x) material flow costs were calculated by multiplying the material flow distance by the cost per unit distance; and (xi) the processing time and moving time were not taken into consideration.

Fig. 2 illustrates the dynamic facilities layout problem in which the demand changes for each period D_1.. D_P, where P is the total number of periods. Consider the situation at the beginning of period k, when the layout is initially the same as the last period L_{k-1}. The total estimated costs of maintaining the existing design L_{k-1} with the forecast demand for the forthcoming period D_k is compared with the estimated total cost for the following period if the facility is redesigned. If the total cost of changing the design is greater, the layout is left unchanged i.e. $L_k=L_{k-1}$, otherwise the design is changed. In order to undertake this evaluation a test layout L_T is generated by redesign. The estimated material flow cost F_k (for period k), and the redesign cost C_R associated with L_T that would arise due to machine movement is calculated using Eq. 1 and Eq. 2, respectively. If the savings in material flow i.e. the cost without redesign (F_k with the layout L_{k-1}) minus material flow cost with redesign (F_k' with L_T) is greater than the redesign cost C_R then the layout for period k becomes $L_k=L_T$, otherwise it is left the same $L_k=L_{k-1}$. The redesign cost is calculated by multiplying the total machine movement distance by C_{MD}. In the experiments discussed in Section 7, this was assumed to be 50 currency units per metre as adopted by Vitayasak et al. (2014).

$$\text{Material flow cost } F_k \text{ for period } k = C_{MH} \sum_{g=1}^{N} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{ijk} f_{gij} D_{gk} \quad (1)$$

$$\text{Redesign cost } C_R = C_{MD} \sum_{i=1}^{M} V_{ik} \quad (2)$$

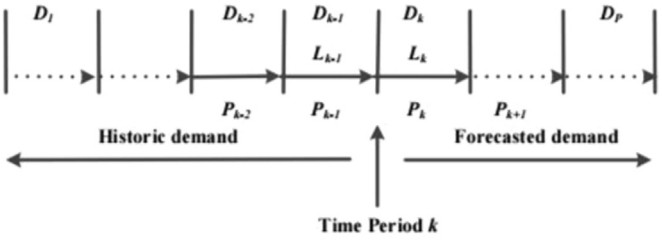


Fig. 2. The dynamic facilities layout problem.

where k is a time period index ($k=1, 2, 3, \ldots, P$), where P is the number of periods; C_{MH} is the material flow cost per distance unit; N is the number of product types; g is a product index ($g=1, 2, 3, \ldots, N$); M is the number of machines; i and j are machine indexes (i and $j=1, 2, 3, \ldots, M$); d_{ijk} is the rectilinear distance from machines i to j ($i \neq j$) for period k; f_{gij} is the frequency of material flow from machines i to j for product g; D_{gk} is the customer demand for product g in period k; C_R is the redesign cost; C_{MD} is the cost per unit distance for moving machines; and V_{ik} is the rectilinear distance moved by machine i in period k.

6. The development of an optimisation tool for solving the stochastic dynamic facilities layout problem

A stochastic dynamic facilities layout tool (SDFLT) was developed for solving the stochastic dynamic facilities layout problem that included a Genetic Algorithm, a Backtracking Search Algorithm and modified Backtracking Search Algorithms. The tool was coded in a modular style using the Tool Command Language and Tool Kit (Tcl/Tk) programming language (Ousterhout, 2010). The tool has the capability to produce layouts with heterogeneous resources and the placement algorithm assumes a continuous plane. The data flow diagram for the layout optimisation tool is shown in Fig. 3. The dynamic facilities layout optimisation tool starts by obtaining input data. Each record of input data comprises: (a) the parameters that specify the stochastic dynamic facilities layout problem characteristics – the number of periods P, the number of machines M, the number of products N; floor length:, F_L floor width: F_W, and the specified gap between the machines G; (b) for each machine ($i=1..M$) machine width M_{Wi} and machine length M_{Li}; (c) for each product–demand profiles $Dg1 \ldots DgN$ for products $g=1 \ldots N$, including the necessary parameters for the exponential, normal and uniform distributions; and the frequency f_{gij} that product g moves between machines i and j; (d) the Genetic Algorithm's parameters – the population size Pop; the number of generations Gen; the probability of crossover P_c; and the probability of mutation P_m; (e) the BSA parameters – population size Pop, maximum number of cycles $MaxCycle$, and the mix rate for the crossover process $Mixrate$.

When the tool has completed its runs, the results are presented. The best-so-far results are reported in text format including a list of the machines in each row, material handling distances, material flow costs, redesign costs, and total costs. The optimised layout can be also shown as a 2D plan.

6.1. Genetic Algorithm

The pseudo-code for the proposed GA for FLD tool is shown in Fig. 4. It follows the following procedure: (i) the problem is encoded to produce a list of genes using numeric strings. Each chromosome contains a number of genes, each representing a

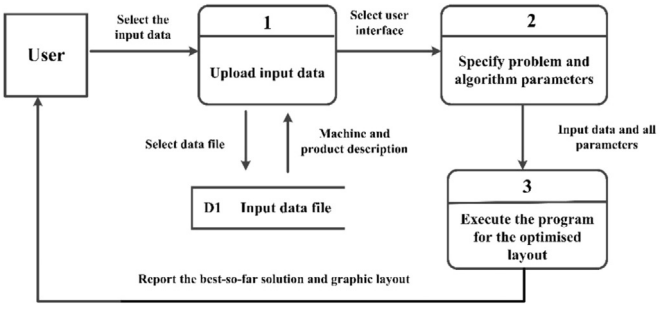


Fig. 3. Data flow diagram for the Stochastic Dynamic Facilities Layout tool.

Input problem dataset
Set parameters (*Pop, Gen,* P_c, P_m, *P*)
Create demand level (D_{gk}) for each product associated with demand distribution
For *k = 1* to *P* {loop to go through periods}
  Randomly create initial population (*Pop*)
  Set *l = 1* (first generation)
  While *l* ≤ *Gen* do {loop to go through the generations}
    For *b = 1* to cross (cross = round ((P_C x *Pop*)/2))), perform crossover operation
    For c = 1 to mute (mute = round(P_m x *Pop*)), perform mutation operation
    Arrange machines row by row based on F_L , F_W and *G* using placement algorithm
    Calculate total of material flow cost (Eq.1)
    Elitist selection
    Chromosome selection using roulette wheel method
    l = l + 1
  End loop while
  Calculate the redesign cost (Eq.2) and compare with the change in material flow cost
  If the reduction in material flow cost due to test redesign is greater than the test redesign cost, adopt the redesign - otherwise leave the layout unchanged.
  k = k+1
 End loop for *k*
Output the best solution

Fig. 4. Genetic Algorithm pseudo-code.

machine number, so that the length of chromosome is equal to the total number of machines that are to be arranged; (ii) in period *k*, an initial population based on the specified population size is randomly generated; (iii) the crossover and mutation operators are applied to generate new offspring in accordance with P_c and P_m; (iv) the machines are arranged row-by-row using a placement algorithm constrained by F_L and F_W; (v) the total material flow cost is evaluated; (vi) the best chromosome that has the minimum total material flow cost is selected using Elitist Selection (Gen and Cheng, 1997); (vii) the chromosomes for the next generation are chosen by using Roulette Wheel selection (Gen and Cheng, 1997); (viii) the GA process is stopped for period *k* after the specified number of generations *G* have been completed. When the GA process is terminated, the best-so-far solution is reported; (ix) the redesign cost is compared with the reduction in material flow cost associated with the redesign. If the reduction in material flow cost due to test redesign is greater than the test redesign cost, the redesign is adopted – otherwise the layout is left unchanged; and (x) the optimisation process is stopped when all of the periods have been considered.

6.2. Backtracking Search Algorithm

The BSA consists of five processes: initialisation, selection-I, mutation, crossover and selection-II. The pseudo-code for the proposed BSA is shown in Fig. 5. The BSA procedure has the following steps for each period *k*:

i) the problem is encoded to randomly produce an initial population *Pop* of individuals (each of which represents a candidate solution that comprises a sequence of machines). In the first iteration an old population *OldPop* is created randomly. For later iterations *OldPop* is randomly copied from a previous iteration;
ii) a layout is created using a placement algorithm and the material flow cost $F_{k,l}$ (for period *k*, iteration *l*) is calculated for all of the individuals within *Pop*;
iii) the Selection I procedure is applied. It has two parts: (1) two uniformly distributed random numbers are generated in the range 0–1. If $a < b$ the old population *OldPop* becomes the current population, otherwise *OldPop* retains its previous value; and (2) the permuting procedure which randomly

Input problem dataset
Set parameters (*MaxPop, Maxcycle, Mixrate*) and create demand level for all periods
For *k = 1* to *P* do {loop to consider all periods}
  Initialisation: Randomly create an initial population *Pop* and an initial old population *OldPop*
  Create a layout using the placement algorithm for all individuals within *Pop*
  Calculate material flow cost for all individuals within *Pop* for period *k*
  Set *l = 1* (first iteration) {loop to go through the iterations}
  While *l* ≤ *Maxcycle* do
    Selection I: if *(a<b |a, b ~ U(0,1))* then *OldPop := Pop*, end
      OldPop := permuting(OldPop)
    Mutation: Mutant = *Pop + F·(OldPop - Pop)*
    Crossover process
    Crossover: $Map_{1:Pop,\ 1:M} = 1$
      Process of creating map values
      if (*c<d |c, d ~ U(0,1)*) then
        for *j = 1* to *Pop* do
          $map_j, u_{(1:\lceil mixrate \bullet rnd \bullet M \rceil)} = 0 \mid u$ = permuting(*1, 2, 3, ..., M*), end
      else for *j = 1* to *Pop* do, $map_{j,\ randj(M)} = 0$, end
      Generation of Trial population (T)
      T := mutant
      for *j = 1* to *Pop* do
        for *m = 1* to *M* do, if $map_{j,m} = 1$ Then $T_{j,m} := Pop_{j,m}$, end
      end
      Create a layout using the placement algorithm for all individuals within T
    Selection II: for *j* = 1 to *Pop* do, Calculate total of material flow cost (Eq.1) for T_j
      if fitnessT_j < fitnessPop_j then fitnessPop_j := fitnessT_j and $Pop_j := T_j$, end
    fitnessPop_{best} = min(fitness*Pop*) | best ϵ (1…*Pop*)
    l = l + 1
  End loop while
  Calculate the redesign cost (Eq.2) and compare with the change in material flow cost
  If the reduction in material flow cost due to test redesign is greater than the test redesign cost, adopt the redesign - otherwise leave the layout unchanged.
  k = k+1
End loop for *k*
Output the best solution

Fig. 5. Pseudo-code of BSA (modified from Civicioglu (2013)).

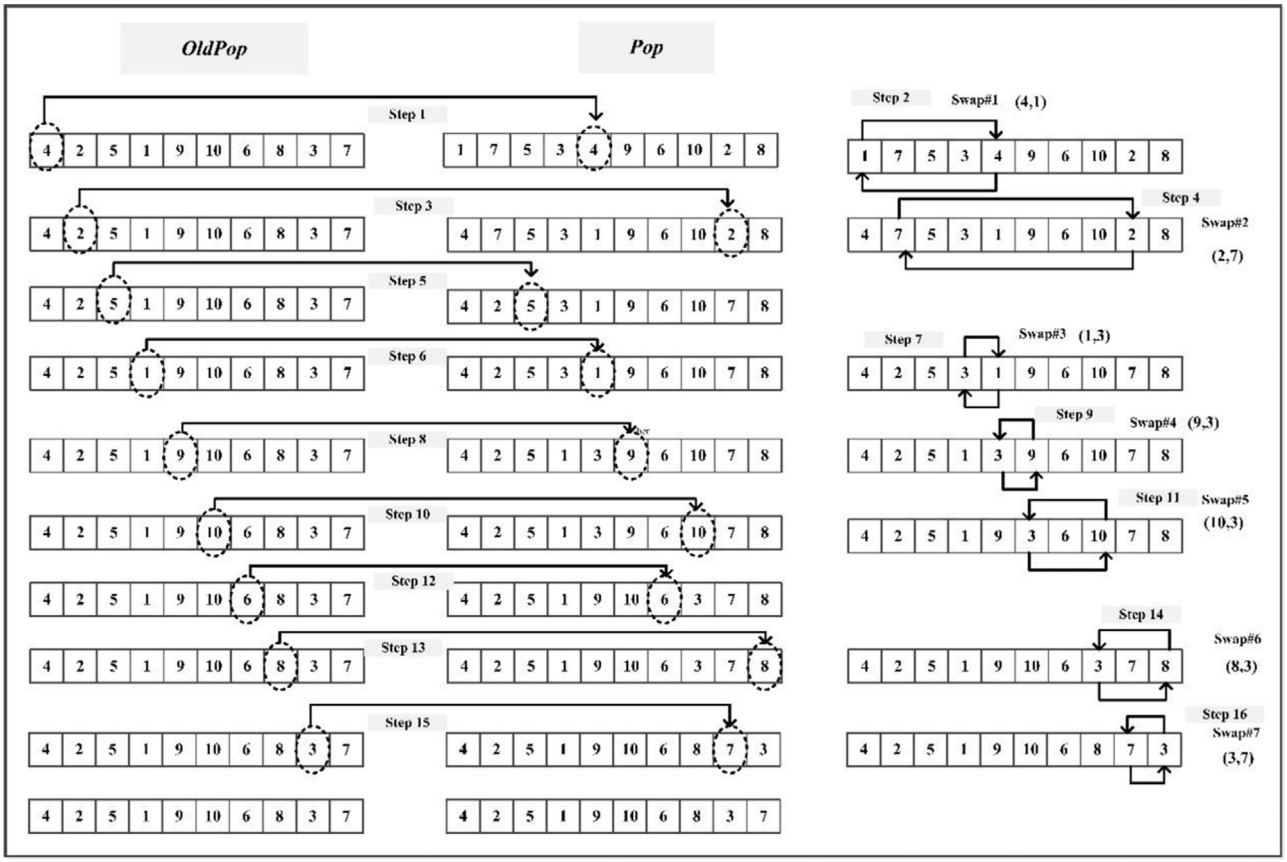


Fig. 6. Swapping operation.

(*OldPop – Pop*) includes the swapping steps: 2, 4, 7, 9, 11, 14, and 16.

Total number of swapping steps is 7

Fig. 7. Swapping steps.

If $F = 0.48$, Mutant = *Pop* + 0.48 *7 swapping steps
Mutant = *Pop* + 3 swapping steps
Mutant = *Pop* + (4,1)(2,7)(1,3)

Mutant

4	2	5	1	3	9	6	10	7	8

Fig. 8. Backtracking Search Algorithm mutation procedure.

changes the order of the individuals within the population; the mutation procedure is illustrated in Fig. 6. The steps are as follows: (1) identify the first element of *OldPop* (in this case with a value of 4). Then locate the position of the value 4 in *Pop*. In this example the value 4 is the fifth element of *Pop*; (2) in this case swap the first (location of value 4 in *OldPop*) and fifth (the location of value 4 in *Pop*) elements within *Pop;* (3) Having processed the first element in *OldPop* move to the second element. In this case containing the value 2. The position of value 2 in *Pop* is then identified. In this case it is the ninth element; (4) the second and ninth elements (the locations of the value 2 in *OldPop* and *Pop*) are then swapped as shown. When the process is undertaken for the third time, the value 5 is positioned in element 3 for both *Pop* and *OldPop*. When this situation occurs there is no swapping of elements. This process is continued until all the elements within *OldPop* have been processed by this procedure; Fig. 7 shows the next step, which is to count how many swaps have taken place. In this case it is 7 i.e. (4,1), (2,7), (1,3), (9,3), (10,3), (8,3) and (3,7). Fig. 8 shows the final step, which is to multiply the number of steps by a uniform random variable F in the range 0–1. In the example, F is 0.48, so the number of swaps is 7*0.48 = 3.36, which rounds to 3. In this case, the first three swaps are undertaken and the remaining swaps are ignored as shown. This process is repeated for all of the individuals within *Pop*. The resultant population is called '*Mutant*';

iv) the crossover process is undertaken. It has two parts: creating mapped values; and generating a trial population. These are explained in (v) and (vi) below.

v) mapped values are created by the following steps: (1) two uniformly distributed random numbers c and d in the range 0–1 are generated. If $c < d$ then for each individual in *Pop* the number of elements to be mapped is calculated. This is illustrated in Fig. 9. The individuals within the population are on the left. The number of elements to be mapped is calculated by multiplying the number of elements in each individual by the mix rate and by a uniformly distributed random number in the range 0–1, as shown in the second column of Fig. 9. The next step is to map the values. If the number of values to be mapped is 2 the first two elements in the mapped value array are set to zero and the other elements are set to 1 as shown in

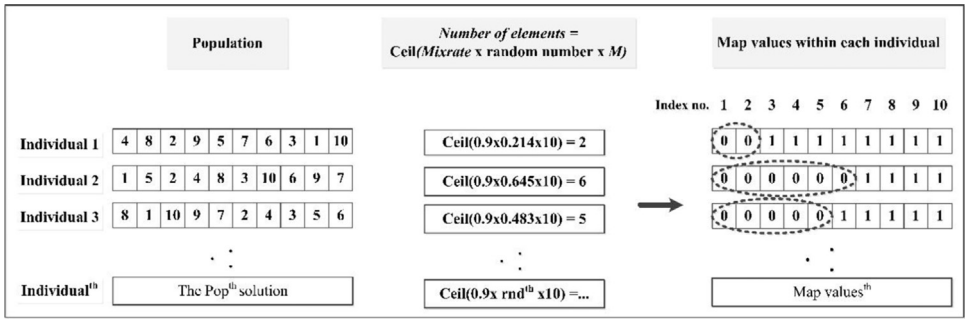


Fig. 9. Process of creating mapped values for $c < d$.

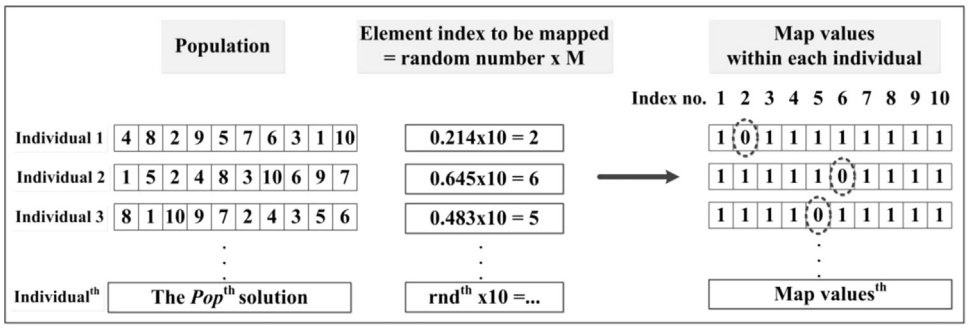


Fig. 10. Processes of creating mapped values for $c \geq d$.

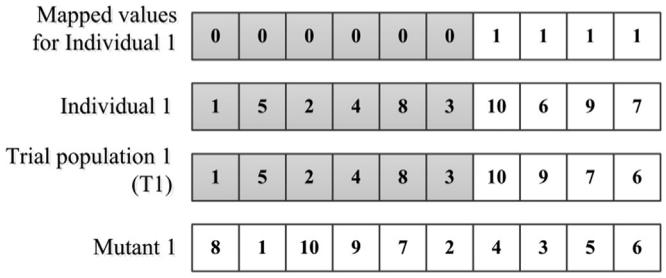


Fig. 11. Operation for generating the trial population in step (vi).

the final column of Fig. 9. If $c \geq d$ only one element is mapped as shown in Fig. 10.

vi) the trial population is generated using the following steps: (1) the trial population is initially the mutant population that was generated in step (iv) above; (2) for each individual within *Pop* the value for each element is selected from *Pop* if the mapped value is 0 otherwise it is selected from *Mutant*. Fig. 11 shows the crossover operation used for generating new individuals for the trial population. The mapped values of the first six indexes equal to 0, so the first six indexes of the new individual are obtained from individual no.1 and the remaining indexes are selected from *Mutant*.

a) Selection II – for each individual within the trial population the material flow cost is calculated (using Eq. (1) above) and compared with the value of the corresponding individual in *Pop*. The set of individuals with the lowest material flow costs are then selected from either *Mutant* or *Pop* to create the new population for the next iteration. The best individual with the lowest material flow cost is selected and a placement algorithm is used to produce a test layout;

b) the redesign cost for the test layout is calculated by considering the distance of machine movements as shown in Eq. (2) above. If the reduction in material flow costs exceeds the redesign costs the layout is changed, otherwise the previous layout is maintained.

6.3. Modified Backtracking Search Algorithm

In order to enhance its exploration and exploitation capacities, the proposed BSA was modified in three ways: (i) the control mechanism was applied after the crossover process as shown in Fig. 12a; (mBSA1); (ii) the process of creating mapped values was changed to create mBSA2 as shown in Fig. 12b; and (iii) the BSA was modified to include both mBSA1 and mBSA2, which was called mBSA3. The mBSA1 can prevent similar solutions being produced in the trial population (T) and the existing population (P) by using the swap operation shown in Fig. 13. The mBSA2 was designed to increase the diversity of solutions. For mBSA2, c and d were randomised in every solution as shown in Fig. 14 which is different to the standard BSA. The c and d values in the BSA were randomised only once, so it was possible for the number of elements to be mapped to be the same as shown in Fig. 10 (in case of $c \geq d$).

```
For j from 1 to Pop do
  if Tj = Pj then Swap times = rnd * (M-1), end
    for j from 1 to Swap times do
      Tj = Tj + two operations random swap, end
  end
end
```

(a)

```
For j from 1 to Pop do
    if c < d then
        map_j, u_(1:⌈mixrate • rnd • M⌉) = 0,
    else map_(j, randj(M)) = 0, end
end
```

(b)

Fig. 12. Pseudo-code (a) Control mechanism for mBSA1; (b) Creation of mapped values in crossover process for mBSA2.

7. Computational experiments

The computational experiments used the eleven datasets shown in Table 2 (Vitayasak et al., 2014). The naming convention 10M5N indicates that the dataset included the production of five products that were processed on ten non-identical rectangular machines. Each type of product had different demand profiles and machine sequences as shown in Table 3 (Vitayasak et al., 2014). Previous research on robust layout design has recommended settings for the BSA parameters as follows: *Pop*=25, *Maxcycle*=100 and *Mixrate*=0.9 (Vitayasak and Pongcharoen, 2014a). Vitayasak and Pongcharoen (2014b) recommended that the best settings for the GA parameters (Pop, Gen, Pc, and Pm) are 25, 100, 0.9 and 0.9, respectively. Vitayasak and Pongcharoen (2011) recommended the Two-point Centre Crossover (2PCX) and the Two Operation Random Swap (2ORS) genetic operators therefore these were adopted.

The computational experiments were performed using five algorithms (GA, BSA, and three modified BSAs (mBSA1, mBSA2, and mBSA3)) as described previously. Each algorithm was tested using the 11 datasets and analysed statistically. For each dataset, each algorithm was replicated 30 times using the recommended parameter settings. (Bluman, 2008) identified that 30 replications is the minimum number that can be used to achieve an approximation to be the normal distribution, which is required for the statistical tests to be valid. The computational results were analysed in terms of the mean, standard deviation (SD) minimum, maximum, and computational time (seconds) as shown in Table 4. The best results are shown in bold.

All of the modified BSAs gave better solutions than the standard BSA except for the 10M10N problem. The quality of solutions measured in terms of the mean total cost for mBSA2 was better than mBSA1 for nine datasets. The new process for creating mapped values generated a higher variety of solutions than the standard approach. The mBSA3 produced the lowest mean total cost except for datasets 10M5N, 10M10N, and 20M20N. The combination of control mechanisms and the improved mapping process helped the algorithm escape from local optima. In terms of SD value, the mBSA1 generated the lowest SD for almost all of the datasets. This indicates that the algorithm produced the lowest diversity of solutions. The mean computational times required for the BSA and modified BSAs were slightly different.

The mean total cost obtained by the GA was lower than mBSA3 for the first seven datasets. The Student's *t*-test showed that there were statistically significant differences in mean total cost (since the *P* values were less than 0.05 for all datasets) for five datasets. The solutions obtained by mBSA3 were better than the GA for datasets with forty and fifty machines. The mBSA3 generated significantly better solutions than the GA for datasets 40M20N and 40M40N.

In the 40M40N case, the convergence of the GA, BSA and modified BSAs were analysed by plotting the average best-so-far

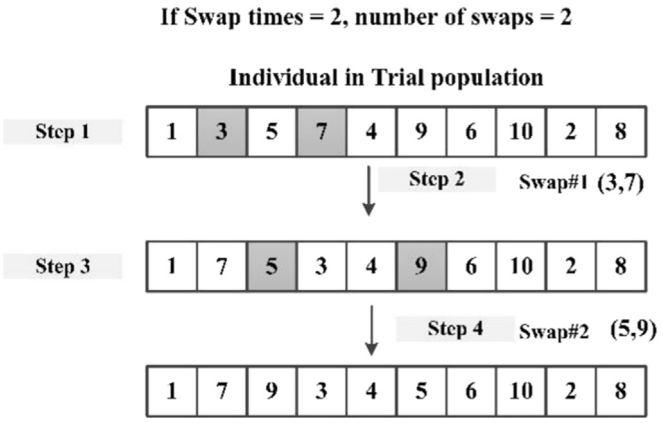


Fig. 13. mBSA1 Swap operation.

Table 2
Datasets for the experimental programme.

Datasets	Number of machine (*M*)	Number of products (*N*)
10M5N	10	5
10M10N	10	10
20M10N	20	10
20M20N	20	20
20M40N	20	40
30M15N	30	15
30M30N	30	30
40M20N	40	20
40M40N	40	40
50M25N	50	25
50M40N	50	40

Table 3
Summary of product demand profiles and machine sequence for 10M5N.

Product	Product demand distribution	Machine sequence
1	Uniform (100, 200)	2–1–6–5–8–9–3–4
2	Uniform (50, 100)	10–8–7–5–9–6–1
3	Normal (180, 50)	9–2–7–4
4	Normal (300, 120)	8–10–5–9–6
5	Exponential (1/200)	2–4–8–10–7

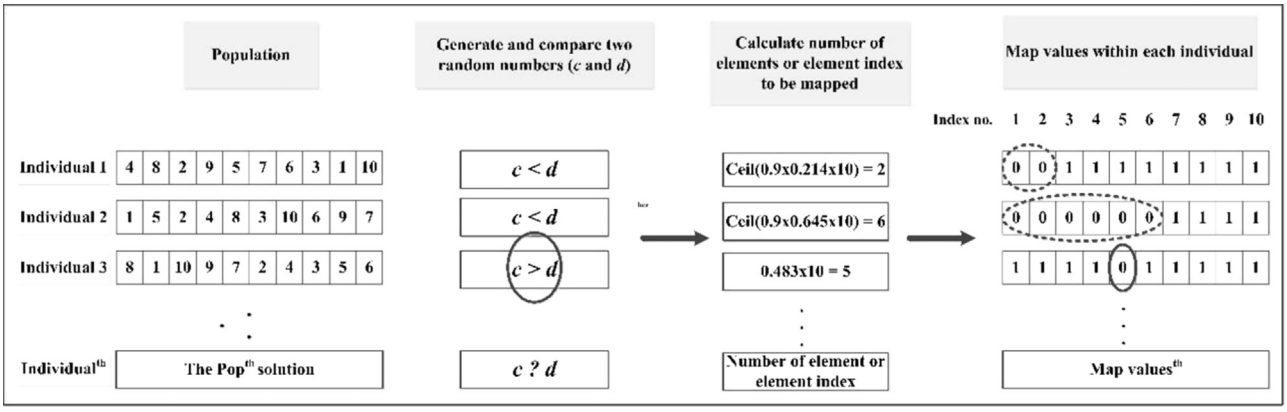


Fig. 14. Creation of mapped values in the crossover process for mBSA2.

Table 4
Comparison of total costs associated with the layouts produced by GA, BSA and three mBSAs Unit: currency unit.

Dataset	Value	Algorithm					*P* value of *t*-test
		GA	BSA	mBSA1	mBSA2	mBSA3	GA and mBSA3
10M5N	Mean	318,834	324,229	323,751	**317,079**	319,129	0.923
	SD	12,897	11,793	13,547	9,147	10,574	
	Min	301,102	304,850	304,896	304,051	306,305	
	Max	355,462	349,125	346,980	345,216	343,949	
	Time	66.0	26.0	29.0	24.0	29.0	
10M10N	Mean	**786,080**	795,354	801,745	795,632	795,672	0.029
	SD	18,989	21,322	15,107	17,616	13,615	
	Min	762,404	764,078	769,479	767,154	762,404	
	Max	830,284	840,332	828,455	827,320	814,738	
	Time	111.0	50.0	49.0	48.0	47.0	
20M10N	Mean	**1,655,452**	1,706,708	1,684,724	1,667,579	1,672,991	0.014
	SD	27,793	21,633	14,633	28,040	25,571	
	Min	1,588,655	1,670,670	1,646,906	1,613,371	1,620,772	
	Max	1,701,282	1,759,082	1,718,485	1,724,126	1,724,242	
	Time	236.0	88.0	102.0	99.0	105.0	
20M20N	Mean	**5,233,863**	5,418,975	5,325,315	5,341,733	5,313,282	0.000
	SD	51,923	59,050	42,152	51,148	45,789	
	Min	5,127,319	5,290,691	5,249,015	5,216,302	5,230,165	
	Max	5,324,560	5,512,954	5,416,026	5,467,212	5,390,481	
	Time	410.0	162.0	163.0	159.0	165.0	
20M40N	Mean	**10,276,216**	10,602,908	10,432,590	10,466,419	10,395,917	0.000
	SD	74,198	82,151	56,273	65,026	64,450	
	Min	10,124,645	10,434,114	10,280,667	10,352,539	10,238,749	
	Max	10,416,961	10,811,156	10,539,035	10,569,175	10,547,172	
	Time	845.1	209.0	208.0	207.0	214.0	
30M15N	Mean	**3,922,999**	4,119,722	4,046,627	4,008,069	3,982,044	0.000
	SD	59,537	52,352	39,665	53,764	55,367	
	Min	3,825,041	3,995,791	3,977,473	3,924,952	3,857,773	
	Max	4,107,244	4,208,028	4,116,787	4,133,968	4,085,489	
	Time	611.6	153.0	153.0	155.0	157.0	
30M30N	Mean	**8,361,651**	8,712,238	8,470,386	8,452,086	8,396,821	0.131
	SD	89,134	133,221	105,796	124,652	88,876	
	Min	8,201,074	8,428,954	8,300,222	8,142,560	8,211,189	
	Max	8,491,160	9,065,532	8,706,214	8,669,958	8,557,450	
	Time	637.2	272.0	271.0	279.0	273.0	
40M20N	Mean	7,726,750	8,253,328	7,930,640	7,797,841	**7,645,462**	0.012
	SD	139,507	133,813	91,485	117,500	97,005	
	Min	7,539,443	7,990,904	7,729,541	7,546,052	7,389,977	
	Max	8,009,115	8,618,407	8,169,437	8,041,996	7,833,029	
	Time	900.5	218.7	221.4	218.7	229.5	
40M40N	Mean	14,381,572	15,353,287	14,626,948	14,451,439	**14,199,334**	0.005
	SD	257,817	246,172	157,891	232,508	227,414	
	Min	13,848,367	14,785,144	14,401,680	14,074,947	13,702,401	
	Max	14,821,397	15,764,056	14,891,742	14,950,553	14,651,132	
	Time	1,474.2	395.6	348.3	338.9	398.0	
50M25N	Mean	12,443,878	13,424,172	12,829,734	12,562,368	**12,408,188**	0.498
	SD	249,033	226,738	127,294	169,301	141,425	
	Min	11,752,732	12,942,229	12,535,624	12,330,939	12,176,892	
	Max	12,917,746	13,894,422	13,034,445	13,081,449	12,663,321	
	Time	1,043.6	318.0	303.8	294.3	298.4	
50M40N	Mean	18,729,876	19,936,087	19,143,272	18,769,212	**18,618,543**	0.078
	SD	238,603	294,974	184,706	220,927	242,379	
	Min	18,233,882	19,266,096	18,798,687	18,375,822	18,182,190	
	Max	19,451,099	20,512,922	19,560,610	19,346,277	19,202,166	
	Time	1,005.8	365.9	400.3	383.4	417.6	

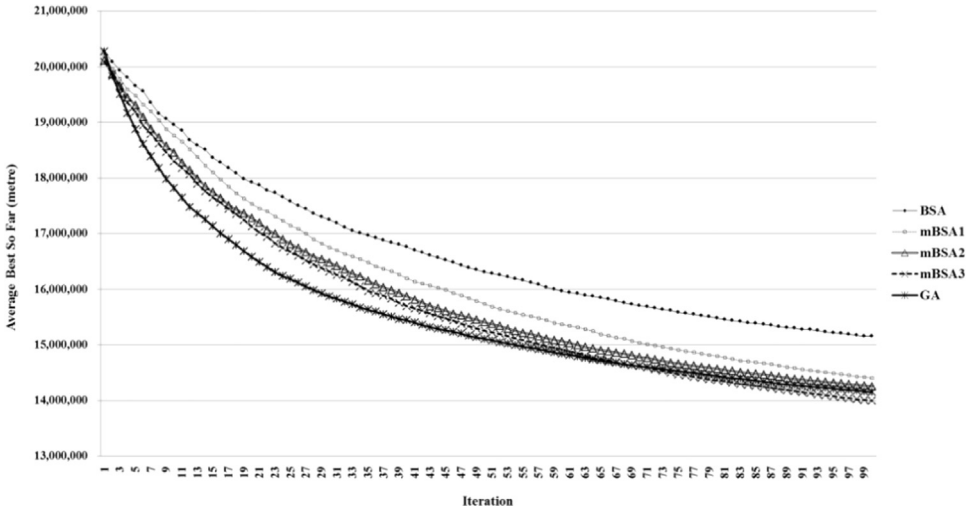
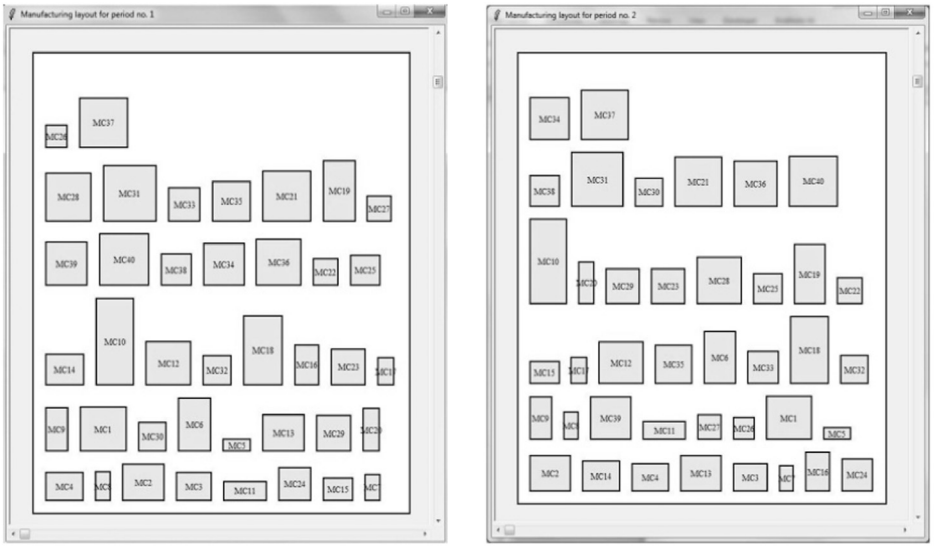


Fig. 15. Comparison of convergence between GA, BSA and mBSAs for 40M40N case.



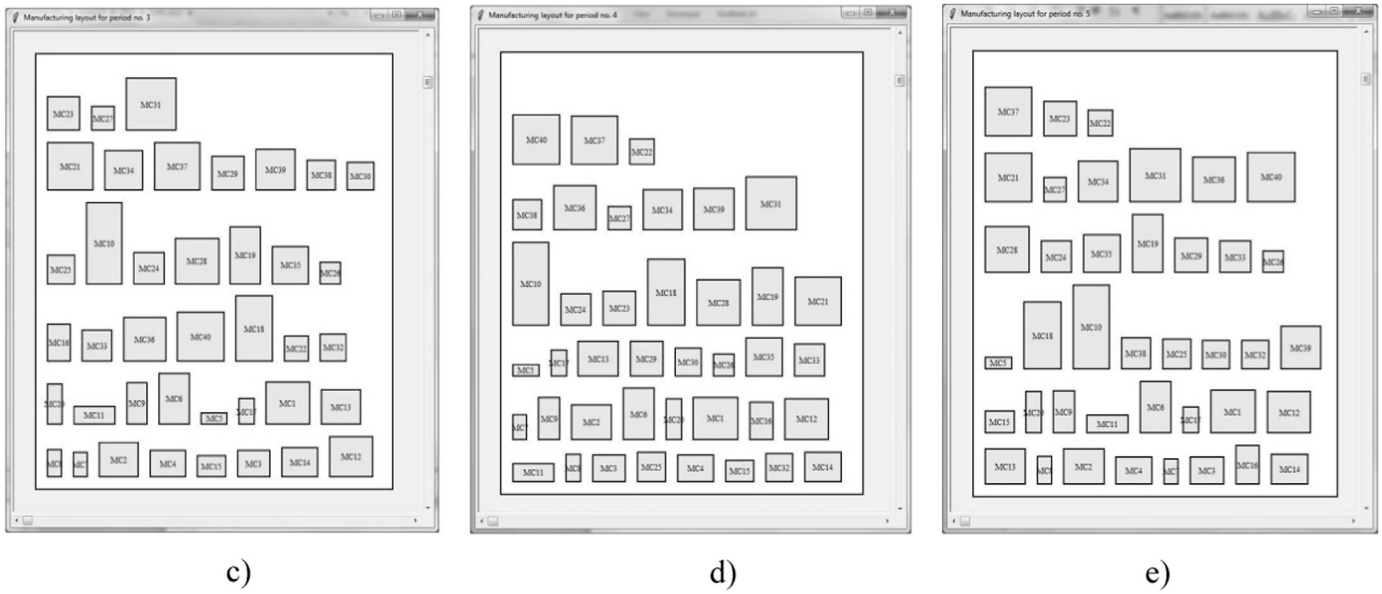


Fig. 16. Graphic layouts reported from the machine relayout design tool for 40M40N case in. (a) Period 1, (b) Period 2, (c) Period 3, (d) Period 4, and (e) Period 5.

(BSF) solutions achieved in each generation (obtained from the 30 replicated runs) as shown in Fig. 15. The GA had faster convergence than both the BSA and mBSAs especially in the fifth generation. However, the mBSA3 algorithm produced better results than the GA after the seventy-fifth generation, Both the modified control mechanism (mBSA1) and the revised mapping creation process (mBSA2) were useful for increasing the diversity of chromosomes during the search process and helped the search escape from local optima Examples of two dimensional plans that illustrate the BSF solutions produced by the tool for five periods using the mBSA3 are presented in Fig. 16a–e. The plans changed between the periods because the sum of redesign cost and material handling cost of the new layouts were lower than material handling cost of the previous layouts.

In order to achieve a fair comparison of the performance of the GA and BSA parameters were selected that achieved the same amount of search for both algorithms. For the GA, the amount of search is governed by the combination of *Pop* and *Gen*; The corresponding parameters for the BSA are *Pop* and *Maxcycle*. In each generation, the GA process consists of three loops including crossover, mutation, and selection. In each cycle of the BSA there are four loops: mutation, two crossover loops, and selection II. However, each loop with the GA mechanism is more complex than with the BSA. The computational time taken by the BSA and the modified BSAs was at least 55% less than the GA for all of the datasets.

8. Discussion and conclusions

This paper has presented a tool that effectively solves stochastic dynamic facilities layout problems taking into account demand over several time periods using Genetic Algorithms and the Backtracking Search Algorithm (BSA). The BSA has been used for solving a range of engineering problems, but no previous research has used it for solving the stochastic dynamic facilities layout problem. Further, the BSA was successfully modified to improve its search capability. The algorithms aimed to minimise the combination of material flow and redesign costs. The computational experiments were based on eleven datasets obtained from the literature. The experimental results indicated that the GA's performance was better than the conventional BSA in terms of minimising total cost. However, the BSA produced solutions much more quickly for all datasets. The performance of the conventional BSA was improved by applying three modifications: (i) applying the control mechanism after the crossover process (mBSA1); (ii) changing the process of creating mapped values (mBSA2); and (iii) a combination of both mBSA1 and mBSA2 (mBSA3). The solution quality obtained by mBSA3 was better than the other BSAs. The mBSA3 algorithm generated significantly better solutions than the GA especially for the forty-machine datasets. The BSA3 mechanism that included a control mechanism a process for creating mapped values led to increased diversification and exploration of solutions. This allowed the BSA to escape from local optima and improve the efficiency of the search process. The performance of the modified BSA (mBSA3) and the GA were similar for four datasets, but the average computational time required by mBSA3 was at least 55% less than the GA. This would suggest that the BSA would be particularly suitable for solving large, computationally intensive optimisation problems especially in the area of engineering for production an operation management such as lot sizing, resource allocation, and bottleneck allocation in manufacturing system.

Acknowledgements

This work was part of the research project supported by the Naresuan University Research Fund under Grant number R2559C230.

References

Askarzadeh, A., 2013. Parameter estimation of fuel cell polarization curve using BMO algorithm. Int. J. Hydrog. Energy 38 (35), 15405–15413.
Askarzadeh, A., Coelho, L.D.S., 2014. A backtracking search algorithm combined with Burger's chaotic map for parameter estimation of PEMFC electrochemical model. Int. Hydrog. Energy 39 (21), 11165–11174.
Askin, R.G., Standridge, C.R., 1993. Modeling and Analysis of Manufacturing Systems. Wiley, New York.
Aytug, H., Knouja, M.J., Vergara, E.F., 2003. Use of Genetic Algorithms to solve production and operations management problems: a review. Int. J. Prod. Res. 41 (17), 3957–3979.
Azadivar, F., Wang, J., 2000. Facility layout optimization using simulation and genetic algorithms. Int. J. Prod. Res. 38 (17 SPEC), 4369–4383.
Baykasoglu, A., Dereli, T., Sabuncu, I., 2006. An Ant Colony Algorithm for solving budget constrained and unconstrained dynamic facility layout problems. Omega—Int. J. Manag. Sci. 34 (4), 387.
Bluman, A.G., 2008. Elementary Statistics: A Step by Step Approach, A Brief Version. McGraw Hill, New York.
Bozer, Y.A., Meller, R.D., 1997. A reexamination of the distance-based facility layout problem. IIE Trans. (Inst. Ind. Eng.) 29 (7), 549–560.
Brest, J., Greiner, S., Bošković, B., Mernik, M., Zumer, V., 2006. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evolut. Comput. 10 (6), 646–657.
Castillo, I., Westerlund, J., Emet, S., Westerlund, T., 2005. Optimization of block layout design problems with unequal areas: a comparison of MILP and MINLP optimization methods. Comput. Chem. Eng. 30 (1), 54–69.
Chakraborty, U.K., Abbott, T.E., Das, S.K., 2012. PEM fuel cell modeling using differential evolution. Energy 40 (1), 387–399.
Chan, W.K., Malmborg, C.J., 2010. A Monte Carlo simulation based heuristic procedure for solving dynamic line layout problems for facilities using conventional material handling devices. Int. J. Prod. Res. 48 (10), 2937–2956.
Chang, C.C., Wu, T.H., Wu, C.W., 2013. An efficient approach to determine cell formation, cell layout and intracellular machine sequence in cellular manufacturing systems. Comput. Ind. Eng. 66 (2), 438–450.
Chaudhry, S.S., Luo, W., 2005. Application of Genetic Algorithms in production and operations management: a review. Int. J. Prod. Res. 43 (19), 4083–4101.
Chen, G.Y., 2013. A new data structure of solution representation in hybrid ant colony optimization for large dynamic facility layout problems. Int. J Prod. Econ. 142 (2), 362–371.
Civicioglu, P., 2013. Backtracking Search Optimization Algorithm for numerical optimization problems. Appl. Math. Comput. 219, 8121–8144.
Corry, P., Kozan, E., 2004. Ant colony optimisation for machine layout problems. Comput. Optim. Appl. 28 (3), 287–310.
Dapa, K., Loreungthup, P., Vitayasak, S., Pongcharoen, P., 2012. Bat Algorithm, Genetic Algorithm and Shuffled Frog Leaping Algorithm for designing non-identical rectangular machine layout. Lect. Notes Artif. Intell. 8271, 59–68.
Das, S., Mandal, D., Kar, R. and Ghoshal, S.P. 2014. Interference suppression of linear antenna arrays with combined Backtracking Search Algorithm and Differential Evolution. In: Proceedings of the International Conference on Communication and Signal Processing, ICCSP 2014.
Das, S., Suganthan, P., 2010. Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems. Jadavpur University/Nanyang Technological University, Kolkata.
De Sá, A.O., Nedjah, N., De Macedo Mourelle, L., 2014. Distributed efficient node localization in wireless sensor networks using the backtracking search algorithm. Lect. Notes Comput. Sci. (Incl. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.) 8630, 794–808.
Drira, A., Pierreval, H., Hajri-Gabouj, S., 2007. Facility layout problems: a survey. Annu. Rev. Control. 31 (2), 255–267.
Duan, H., Luo, Q., 2014. Adaptive backtracking search algorithm for induction magnotmeter optimization. IEEE Trans. Magn. 50 (12), 1–6.
Dunker, T., Radons, G., Westkamper, E., 2005. Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem - Discrete optimization. Eur. J. Oper. Res. 165 (1), 55–69.
El-Baz, M.A., 2004. A genetic algorithm for facility layout problems of different manufacturing environments. Comput. Ind. Eng. 47 (2-3), 233–246.
Enea, M., Galante, G.M., Panascia, E., 2005. The facility layout problem approached using a fuzzy model and a genetic search. J. Intell. Manuf. 16 (3), 303–316.
Ertay, T., Ruan, D., Tuzkaya, U.R., 2006. Integrating data envelopment analysis and analytic hierarchy for the facility layout design in manufacturing systems. Inf. Sci. 176 (3), 237–262.
Gen, M., Cheng, R., 1997. Genetic Algorithms and Engineering Design. John Wiley and Sons, New York.
Gen, M., Cheng, R., Lin, L., 2008. Network models and optimization: Multiobjective

genetic algorithm approach. Springer Science & Business Media.
Goldberg, D., 2002. The Design of Innovation (Genetic Algorithms and Evolutionary Computation). Springer, London.
Gong, W., Cai, Z., 2013. Accelerating parameter identification of proton exchange membrane fuel cell model with ranking-based differential evolution. Energy 59, 356–364.
Hicks, C., 2004. A genetic algorithm tool for designing manufacturing facilities in the capital goods industry. Int. J. Prod. Econ. 90 (2), 211.
Hicks, C., 2006. A Genetic Algorithm tool for optimising cellular or functional layouts in the capital goods industry. Int. J. Prod. Econ. 104 (2), 598–614.
Igel, C., Hansen, N., Roth, S., 2007. Covariance matrix adaptation for multi-objective optimization. Evolut. Comput. 15 (1), 1–28.
Jithavech, I., Krishnan, K.K., 2010. A simulation-based approach for risk assessment of facility layout designs under stochastic product demands. Int. J. Adv. Manuf. Technol. 49 (1–4), 27–40.
Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) Algorithm. J. Glob. Optim. 39 (3), 459–471.
Kennedy, J., Eberhart, R., 1995. Particle Swarm Optimisation. In: Proceedings of the IEEE International Conference on Neural Networks. Perth, Australia.
Kia, R., Javadian, N., Paydar, M.M., Saidi-Mehrabad, M., 2013. A simulated annealing for intra-cell layout design of dynamic cellular manufacturing systems with route selection, purchasing machines and cell reconfiguration. Asia–Pac. J. Oper. Res. 30 (4), 1350004.
Kılıç, U., 2014. Backtracking search algorithm-based optimal power flow with valve point effect and prohibited zones. Electr. Eng. 97 (2), 101–110.
Kolawole, S.O., Duan, H., 2014. Backtracking search algorithm for non-aligned thrust optimization for satellite formation. In: Proceedings of the IEEE International Conference on Control and Automation, ICCA.
Krishnan, K.K., Cheraghi, S.H., Nayak, C.N., 2008. Facility layout design for multiple production scenarios in a dynamic environment. Int. J. Ind. Syst. Eng. 3 (2), 105–133.
Krishnan, K.K., Jithavech, I., Liao, H., 2009. Mitigation of risk in facility layout design for single and multi-period problems. Int. J. Prod. Res. 47 (21), 5911–5940.
Lee, G.C., Kim, Y.D., 2000. Algorithms for adjusting shapes of departments in block layouts on the grid-based plane. Omega 28 (1), 111–122.
Leechai, N., Iamtan, T., Pongcharoen, P., 2009. Comparison on rank-based ant system and shuffled frog leaping for design multiple row machine layout. SWU Eng. J. 4 (2), 102–115.
Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S., 2006. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evolut. Comput. 10 (3), 281–295.
Lin, Q., Gao, L., Li, X., Zhang, C., 2015. A hybrid backtracking search algorithm for permutation flow-shop scheduling problem. Comput. Ind. Eng. 85, 437–446.
Lutuksin, T., Pongcharoen, P., 2010. Best-worst ant colony system parameter investigation by using experimental design and analysis for course timetabling problem. In: Proceedings of the 2nd International Conference on Computer and Network Technology, ICCNT 2010.
Mavridou, T.D., Pardalos, P.M., 1997. Simulated annealing and Genetic Algorithms for the facility layout problem: a survey. Comput. Optim. Appl. 7 (1), 111–126.
Mazinani, M., Abedzadeh, M., Mohebali, N., 2013. Dynamic facility layout problem based on flexible bay structure and solving by genetic algorithm. Int. Journal. Adv. Manuf. Technol. 65 (5–8), 929–943.
McKendall Jr., A.R., Shang, J., Kuppusamy, S., 2006. Simulated annealing heuristics for the dynamic facility layout problem. Comput. Oper. Res. 33 (8), 2431–2444.
Modiri-Delshad, M., Abd Rahim, N., 2014. Solving non-convex economic dispatch problem via backtracking search algorithm. Energy 77, 372–381.
Montreuil, B., Laforge, A., 1992. Dynamic layout design given a scenario tree of probable futures. Eur. J. Oper. Res. 63 (2), 271–286.
Moslemipour, G., Lee, T.S., 2012. Intelligent design of a dynamic machine layout in uncertain environment of flexible manufacturing systems. J. Intell. Manuf. 23 (5), 1849–1860.
Nagar, A., Haddock, J., Heragu, S., 1995. Multiple and bicriteria scheduling: a literature survey. Eur. J. Oper. Res. 81 (1), 88–104.
Ousterhout, J.K., 2010. Tcl and Tk tookit. Addison Wesley.
Pillai, M.V., Hunagunda, I.B., Krishnan, K.K., 2011. Design of robust layout for dynamic plant layout problems. Comput. Ind. Eng. 61 (3), 813–823.
Pongcharoen, P., Chainate, W., Pongcharoen, S., 2008. Improving artificial immune system performance: Inductive bias and alternative mutations. Lect. Notes Comput. Sci. (Incl. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.) 5132, 220–231.
Pongcharoen, P., Hicks, C., Braiden, P.M., 2004. The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure. Eur. J. Oper. Res. 152 (1), 215–225.
Pongcharoen, P., Hicks, C., Braiden, P.M., Stewardson, D.J., 2002. Determining optimum Genetic Algorithm parameters for scheduling the manufacturing and assembly of complex products. Int. J. Prod. Econ. 78 (3), 311–322.
Pourvaziri, H., Naderi, B., 2014. A hybrid multi-population genetic algorithm for the dynamic facility layout problem. Appl. Soft Comput. 24, 457–469.
Qin, A.K., Suganthan, P.N., 2005. Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005.
Rezaee Jordehi, A., 2015. Brainstorm optimisation algorithm (BSOA): an efficient algorithm for finding optimal location and setting of FACTS devices in electric power systems. Int. J. Electr. Power Energy Syst. 69, 48–57.
Sahin, R., Turkbey, O., 2009. A new hybrid tabu-simulated annealing heuristic for the dynamic facility layout problem. Int. J. Prod. Res. 47 (24), 6857.
Sethi, A., Sethi, S., 1990. Flexibility in manufacturing: a survey. Int. J. Flex. Manuf. Syst. 2 (4), 289–328.
Song, X., Zhang, X., Zhao, S., Li, L., 2015. Backtracking search algorithm for effective and efficient surface wave analysis. J. Appl. Geophys. 114, 19–31.
Sooncharoen, S., Vitayasak, S., Pongcharoen, P., 2015. Application of biogeography-based optimisation for machine layout design problem. Int. J. Mech. Eng. Robot. Res. 4 (3), 251–254.
Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S., 2005. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL Rep., 2005005.
Tam, K.Y., 1992. Genetic algorithms, function optimization, and facility layout design. Eur. Journal. Oper. Res. 63 (2), 322–346.
Tate, D. M. and Smith, A. E. 1993. Genetic algorithm optimization applied to variations of the unequal area facilities layout problem. *Proceedings of the 2nd Industrial Engineering Research Conference, May 26-28 1993*, Los Angeles, CA, USA, Publ by IIE, Norcross, GA, USA.
Tavakkoli-Moghaddam, R.S., Javadian, N., Javadi, B., Safaei, N., 2007. Design of a facility layout problem in cellular manufacturing systems with stochastic demands. Appl. Math. Comput. 184 (2), 721–728.
Thapatsuwan, P., Sepsirisuk, J., Chainate, W. and Pongcharoen, P. 2009. Modifying particle swarm optimisation and genetic algorithm for solving multiple container packing problems. *Proceedings – 2009 International Conference on Computer and Automation Engineering, ICCAE 2009*.
Thepphakorn, T., Pongcharoen, P., Hicks, C., 2014. An ant colony based timetabling tool. Int. Journal. Prod. Econ. 149, 131–144.
Vitayasak, S. and Pongcharoen, P. 2011. Interaction of crossover and mutation operations for designing non-rotatable machine layout. *Proceedings of the Operations Research Network Conference*, Bangkok, Thailand.
Vitayasak, S., Pongcharoen, P., 2014a. Backtracking Search Algorithm for designing robust machine layout. WIT Trans. Eng. Sci. 95, 411–420.
Vitayasak, S., Pongcharoen, P., 2014b. Identifying Optimum Parameter Setting for Layout Design Via Experimental Design and Analysis. Adv. Mater. Res. 931-932, 1626–1630.
Vitayasak, S., Pongcharoen, P. and Hicks, C. 2014. A tool for generating optimum facilities layouts under demand uncertainty with/without preventive and breakdown maintenance. *The 18th International Working Seminar on Production Economics*, Innsbruck, Austria.
Yang, T., Brett, A.P., 1998. Flexible machine layout design for dynamic and uncertain production environments. Eur. Journal. Oper. Res. 108, 49–64.
Yang, X.-S. 2008. Nature-Inspired Metaheuristic Algorithm, Luniver.
Zhao, W., Wang, L., Yin, Y., Wang, B., Wei, Y., 2014. An improved backtracking search algorithm for constrained optimization problems. Lect. Notes Comput. Sci. (Incl. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.) 8793, 222–233.