Comparison and Application of Metaheuristic Population-Based Optimization Algorithms in Manufacturing Automation

Rhythm Suren Wadhwa[1] and Terje K. Lien[2]

[1,2] Inst. for produksjons- og kvalitetstek.
Trondheim, 7491, Norway

Abstract

The paper presents a comparison and application of metaheuristic population-based optimization algorithms to a flexible manufacturing automation scenario in a metacasting foundry. It presents a novel application and comparison of Bee Colony Algorithm (BCA) with variations of Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) for object recognition problem in a robot material handling system. To enable robust pick and place activity of metalcasted parts by a six axis industrial robot manipulator, it is important that the correct orientation of the parts is input to the manipulator, via the digital image captured by the vision system. This information is then used for orienting the robot gripper to grip the part from a moving conveyor belt. The objective is to find the reference templates on the manufactured parts from the target landscape picture which may contain noise. The Normalized cross-correlation (NCC) function is used as an objection function in the optimization procedure. The ultimate goal is to test improved algorithms that could prove useful in practical manufacturing automation scenarios.

Keywords: *Bee Colony Algorithm, Particle Swarm Optimization, Ant Colony Optimization, Foundry Automation*

1. Introduction

In the 21st century, under the influences of globalization, manufacturing companies are required to meet continuously changing customer demands. Flexible manufacturing systems (FMS) has emerged as a science and industrial practice to bring about solutions for unpredictable and frequently changing market conditions [21]. Existing FMS implementations in manufacturing companies have demonstrated a number of benefits by helping lower production costs, increased factory floor utilization, reduced work-in-process, etc. However, there are a number of problems faced during the life cycle of an FMS, which could be classified into work flow design, production leveling, and control problems [21]. In particular, the production leveling is important owing to the dynamic nature of FMS such as flexible machines, tools and workflow. This work is primarily concerned with production leveling problem. Over the last decade, most research in FMS has been focused on scheduling of FMSs for single or multi objective problems. The present work, however, compares three evolutionary computation techniques Particle Swarm Optimization (PSO), Bee Colony Algorithm (BCA) and Ant Colony Optimization (ACO). The goal of the paper is not to declare one of the techniques as better than the other, but to test their applications after modification to suit the manufacturing scenario discussed, as well as their limitations. The case study is a small-to-medium batch manufacturing foundry and we intend to test the suitability of the algorithms for the purpose of lean workflow and reducing machine starvation in the manufacturing facility.

1.1 Earlier Research

1.1.1 Flexible Manufacturing Systems

During the last two decades much research has been done in this area. The heuristic algorithms developed include enumerative procedures, mathematical programming and approximation techniques, i.e., linear programming, integer programming, goal programming, dynamic programming, network analysis, branch and bound, genetic algorithm (GA), etc.

Shankar and Tzen [39] considered scheduling problems in a random FMS as composite independent tasks. Lee [25] presented a goal-programming model for multiple conflicting objectives in manufacturing. Toker et al. [45] proposed an approximation algorithm for ‘n’ job ‘m’ machine problem. Steeke and Soldverg [43] investigated various operating strategies on a caterpillar FMS by means of deterministic simulation with the number of completed assemblies on a performance criterion manufacturing problem associated with parallel identical machines throughout simulation. Chan and Pak [3] proposed two heuristic algorithms for solving the scheduling problem with the goal of minimizing total cost in a statictically loaded FMS. Shaw and Winston [40] addressed an artificial intelligence approach to the scheduling of FMS. Schultz and Merkens [38] compared the performance of an ES, a GA and priority rules for production systems.

Further, a comprehensive survey on FMS was done by Chan et al. [3].

Many authors have been trying to emphasize the utilization of heuristics in flexible manufacturing automation. In this context, it has been proposed a comparative study on the application of evolutionary algorithms in a specific manufacturing environment i.e. metalcasting foundries.

1.1.2 Object Recognition in Flexible Manufacturing

The challenge of object recognition is to develop the ability to recognize objects even with significant variations in visual appearance. In recent years, a number of metaheuristic algorithms have been proposed. They have been applied to several real world combinatorial problems in manufacturing. For example, Silva, Lopes and Lima [41] as well as Perlin, Lopes and Centeno [36] presented two metaheuristic approaches, one based on compact Genetic Algorithm (CGA) and the other based on Particle Swarm Optimization (PSO). Results show that both methods can be efficiently applied to practical situations with reasonable computational costs.

Some other related works have been presented using variations of metaheurisitc algorithms. Tereshko and Loengarov [44] proposed a collective decision model considering a bee colony as a dynamical system where intelligent decision making arises from an enhanced level of communication among individuals. In their work, they discussed how the information exchange between individuals leads to globally intelligent selection of food sources in an unpredictable environment. Karaboga [19] proposed the Artificial Bee Colony (ABC) algorithm, based on the foraging behavior of real bees, and later compared its performance with other evolutionary and swarm intelligence based algorithms using a large set of numerical functions. Karaboga et al. [19] concluded that the ABC algorithm is a robust optimization algorithm that can be efficiently used in the optimization of multimodal and multi-variable problems. Another version of a bee swarm-based algorithm was proposed by Pham and Zaidi [37], named Bees Algorithm (BA), which can be used for both combinatorial and multi-parameter functional optimization.

More recently, Hackel and Dippold [13] developed an algorithm inspired in bee colony for the vehicle routing problem with time windows. According to Mishra [30], the algorithms mentioned before have an inherent probabilistic nature and thus may not always obtain best solutions with certainty. This paper uses the Matlab toolbox from Karaboga which minimizes or maximizes functions. We have adapted it in order to be able to take 4 templates and landscape image and be able to maximize the NCC value obtained by the equation 1, which is defined as “objective function” for maximization. Plotting commands have been added to the program to represent the matching between both images and so to be able to determine the accuracy of the program. Another command to calculate the time expended in each run has been added as well.

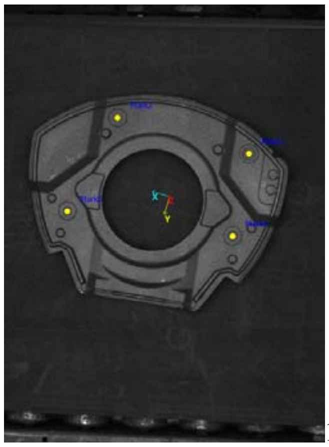
2. Problem Description

One important application of a robot vision system is to recognize whether or not a given part is a member of a particular class of parts. Currently, common examples of object recognition can be found in areas such as industry, engineering, medical diagnosis etc. Generally, recognition of objects in images using traditional search algorithms is computationally expensive. For many industrial applications, these algorithms should normally be executed in real-time. Hence, fast algorithms are essential at all stages of the recognition process in images. This fact suggests the use of fast algorithms based on metaheuristics. Recently, besides the traditional image processing techniques, several methodologies based on computational intelligence have been developed and applied to object recognition problem, so as to reduce computational cost and to improve efficiency. Amongst them, metaheurisitc population-based optimization algorithms, such as those from the Swarm Intelligence area, were successfully applied to the problems.

Recognizing orientation of objects is a challenging task due to constant changes in images in the real world. The most straightforward technique for part orientation recognition is called template matching [2]. Template matching is the process of determining the optimal matching between the same scenes taken at different times or under different conditions and the template known according to some similarity measure. [26]. In other words, the basic idea is to find a match of the pattern in some part of the landscape image. The most common way of finding the matching point between the landscape image and the template is by calculating the correlation function value which indicates the percentage of matching of both images for a specific matching point. The bigger this parameter is, the closer the two images will be.

Normalized Cross Correlation (NCC) is the most robust correlation measure for determining similarity between points in two or more images providing an accurate foundation for motion tracking images [17]. This technique has been used on several works. Cole [6] used this technique to reduce the size of a set of images to which new images were compared. Modegi [31] proposed a structured template matching technique for recognizing small objects in satellite images. There are other methods of tracking that do not use NCC, including Gradient Descent Search (GDS) and Active Contour Matching [1]. The GDS is based on a first order approximation to image motion and has a restriction that the feature translation is small.

The method of template matching loops the template through all the pixels in the captured image and compares the similarity. While this method is simple and easy to implement, it is the slowest one. [48] This speed problem could be reduced by the application of the metaheuristic population-based optimization algorithms.



1a. 1b. 1c. 1d.

Fig. 1 Image of the sample part on the assembly conveyor belt, as seen from the overhead camera image. 1a 1b 1c 1d The templates to be detected on the part to predict its orientation for handling by the robot gripper.

In this work, we want to find a reference image in the target landscape image. When the pattern is found in the target image, its rotation angle is determined. To evaluate a candidate solution, the measure of similarity γ between the reference and target landscape image has been proposed. Several similarity measures have been proposed in the literature, such as mutual information and sum of square of differences between pixels [2][6]. In this work, we used the relation in equation 1, considering the degree of similarity between the images.

$$\gamma = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[F(x+i,y+j)-\bar{F}_{i,j}]\cdot[T(i,j)-\bar{T}]}{\left\{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[F(x+i,y+j)-\bar{F}_{i,j}]^2\cdot\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[T(i,j)-\bar{T}]^2\right\}^{1/2}} \quad (1)$$

In the equation (1), $F(x,y)$ is the landscape image, $\bar{F}_{i,j}$ is the grey-scale average intensity of the captured image in the region coincident with the template image, $T(x,y)$ represents the template image and $\bar{T}$ is the average intensity of the template image. We have to address that the dimensions of the matrix F is MxN and the size of the template T is mxn. The maximum value of γ is 1, will say that the match between the landscape and the template is perfect. [48].

The FMS layout considered in this work, depicted below, consists of a six axis ABB ERB 6400 robot, a vision camera, and a material handling system- a conveyor belt. The Sony XCG-U100E overhead camera (Figure 2b) is used for identifying the orientation of the part lying on a conveyor belt (Figure 2c).

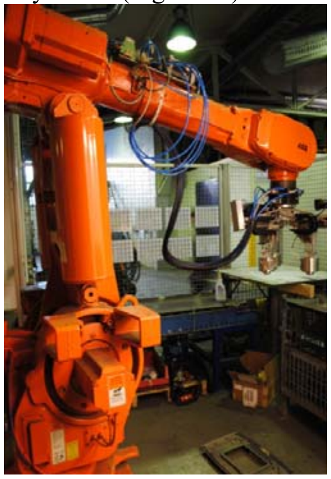
2a.  2b. 2c.

Fig. 2a. The Assembly Robot 2b. Overhead Camera 2c. Conveyor Belt.

The image captured by the camera is transferred via closed network Ethernet connection to the testing PC. The ABB robot tracks the conveyor belt using a conveyor tracking system which is included in the robot controller. The part orientation information is transferred to the robot gripper via the Ethernet, which then orients itself accordingly to pick the part. The object recognition problem is to find the templates on the parts, such as the one chosen in this case, considering the possible position of the images within the required tact time allocated to the robot assembly cell.

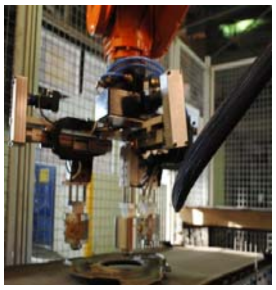
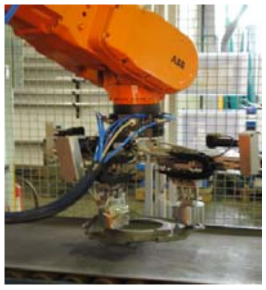
3a.  3b. 

Fig. 3a. Gripper orienting to pick the part 3b. Part lifted from conveyor belt

While the simulations provided in this paper are based on real assembly shop data in a company, the actual part details, the assembly cell rates and the gripper construction details are not revealed due to the proprietary nature of the information.

3. Proposed Methodology

3.1 Bee Colony Algorithm (BCA)

The Bee Colony Algorithm [19] is inspired by the collective behavior of a colony of honeybees working to find food sources around the hive. Although a colony of honeybees has a queen, the control is decentralized rather than hierarchical. The beehive can be understood as a self-organizing system with a multiplicity of agents [24]. A self-organizing system is based on characteristics of positive and negative feedback, random fluctuation as well as the interaction of the system's individuals. The use of preferably good food sources is an emergent property of the beehive.

In BCA algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. A colony of honey bees can move itself over long distances and in multiple directions simultaneously to exploit a large number of food sources. The goal of the colony is to achieve good food sources, which depend on some factors such as the distance to the hive, richness or concentration of nectar and easiness of extracting the nectar.

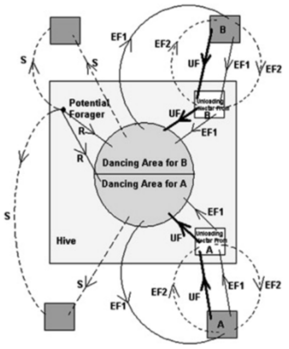


Fig. 4 Behavior of honeybee foraging for nectar (*Adapted from Karaboga et al. 2009*).

A colony of honey bees is classified into three categories; employed bees, onlooker bees and scout bees. All bees that are currently exploiting a food source are known as employed bees. The employed bees exploit the food source and they carry the profitability of the food source back to the hive and share this information with onlooker bees by dancing in the designated dance area inside the hive. Onlooker bees look for a food source to exploit. They watch the dance and choose a food source according to the probability proportional to the quality of that food source. Therefore, good food sources attract more onlooker bees compared to bad ones. Whenever a food source is exploited fully, all the employed bees associated with it abandon the food source, and become scouts. Scout bees will always be searching for new food sources near the hive. The mean number of scouts is about 5–10%. Scout bees can be visualized as performing the job of exploration, whereas employed and onlooker bees can be visualized as performing the job of exploitation.

The main steps of the algorithm are as below: [19]

```
1: Initialize Population
2: repeat
3: Place the employed bees on their food sources
4: Place the onlooker bees on the food sources depending on their nectar amounts
5: Send the scouts to the search area for discovering new food sources
6: Memorize the best food source found so far
7: Until requirements are met
```

In BCA algorithm, each cycle of the search consists of three steps: sending the employed bees onto their food sources and evaluating their nectar amounts; after sharing the nectar information of food sources, the selection of food source regions by the onlookers and evaluating the nectar amount of the food sources; determining the scout bees and then sending them randomly onto possible new food sources. At the initialization stage, a set of food sources is randomly selected by the bees and their nectar amounts are determined. At the first step of the cycle, these bees come into the hive and share the nectar information of the sources with the bees waiting on the dance area. A bee waiting on the dance area for making decision to choose a food source is called onlooker and the bee going to the food source visited by herself just before is named as employed bee.

After sharing their information with onlookers, every employed bee goes to the food source area visited by itself at the previous cycle since that food source exists in her memory, and then chooses a new food source by means of visual information in the neighbourhood of the one in her memory and evaluates its nectar amount. At the second step, an onlooker prefers a food source area depending on the nectar information distributed by the employed bees on the dance area. As the nectar amount of a food source increases, the probability of that food source chosen also increases. After arriving at the selected area, she chooses a new food source in the neighbourhood of the one in the memory depending on visual information as in the case of employed bees. The determination of the new food source is carried out by the bees based on the comparison process of food source positions visually. At the third step of the cycle, when the nectar of a food source is abandoned by the bees, a new food source is randomly determined by a scout bee and replaced with the abandoned one. In our model, at each cycle at most one scout goes outside for searching a new food source and the number of employed and onlooker bees is selected to be equal to each other. These three steps are repeated through a predetermined number of cycles called Maximum Cycle Number MCN or until a termination criterion is satisfied.

An artificial onlooker bee chooses a food source depending on the probability value associated with that food source p_i calculated by Eq. (2):

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{2}$$

where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and SN is the number of food sources which is equal to the number of employed bees or onlooker bees.

In order to produce a candidate food position from the old one in memory, the BCA uses Eq. (3):

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{3}$$

where $k \in \{1,2,\ldots, SN\}$ and $j \in \{1,2,\ldots, C\}$ are randomly chosen indexes. Although k is determined randomly, it has to be different from i. ϕ_{ij} is a random number between [-1,1]. It controls the production of neighbour food sources around x_{ij} and represents the comparison of two food positions visually by a bee. As the difference between the parameters x_{ij} and x_{kj} decreases, the perturbation on the position x_{ij} gets decreased, too. Thus, as the search approaches the optimum solution in the search space, the step length is adaptively reduced.

If a parameter value produced by this operation exceeds its predetermined limit, the parameter can be set to an acceptable value. In this work, the value of the parameter exceeding its limit is set to its limit value.

The food source of which the nectar is abandoned by the bees is replaced with a new food source by the scouts. In BCA, this is simulated by producing a position randomly and replacing it with the abandoned one. If a position cannot be improved further through a predetermined number of cycles, then that food source is assumed to be abandoned. Assume that the abandoned source is x_i and $j \in \{1, 2, \ldots.., D\}$, then the scout discovers a new food source to be replaced with x_i. This operation can be defined as in Eq. (4)

$$x_i^j = x_{\min}^j + rand[0,1](x_{\max}^j - x_{\min}^j) \tag{4}$$

After each candidate source position $v_{i,j}$ is produced and then evaluated by the artificial bee, its performance is compared with that of its old one. If the new food source has an equal or better nectar than the old source, it is replaced with the old one in the memory. Otherwise, the old one is retained in the memory. In other words, a greedy selection mechanism is employed as the selection operation between the old and the candidate one

3.2 Ant Colony Optimization (ACO)

Ant colony optimization was formalized into a metaheuristic for combinatorial optimization problems by Dorigo and co-workers [27], [28]. One can find ACO metaheuristic application to real-world applications mentioned in the literature such as by Price et al. [29], who have applied ACO to an industrial scheduling problem in an aluminum casting center, and by Bautista and Pereira [18], who successfully applied ACO to solve an assembly line balancing problem with multiple objectives and constraints between tasks.

In ACO algorithms a colony of artificial ants iteratively constructs solutions for the problem under consideration using artificial pheromone trails and heuristic information. Its main characteristic is that, at each iteration, the pheromone values are updated by *all* the m ants that have built a solution in the iteration itself. The pheromone τ_{ij}, associated with the edges i and j, is updated as follows:

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{5}$$

where ρ is the evaporation rate, m is the number of ants, $\Delta\tau_{ij}^k$ is the quantity of pheromone laid on the edge (i, j) by ant k.

$$\Delta\tau_{ij}^k = \frac{Q}{L_k} \tag{6}$$

if ant k uses edge (i, j) in its tour, and 0 otherwise. In the equation above, Q is a constant, and L_k is the length of the tour constructed by ant k.

In the construction of a solution, ants select the following city to be visited through a stochastic mechanism. When ant k is in city i and has so far constructed the partial solution s^p ,the probability of going to city j is given by:

$$p_{ij}^k = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{il}\in N(s^p)} \tau_{il}^{\alpha}\eta_{il}^{\beta}} \qquad (7)$$

if $c_{ij} \in N(s^p)$, and 0 otherwise. In the equation above $N(s^p)$ is the set of feasible components; that is, edges (i,l) where l is a city not yet visited by ant k . The parameters α and β control the relative importance of the pheromone versus the heuristic information η_{ij} , which is given by:

$$\eta_{ij} = \frac{1}{d_{ij}} \qquad (8)$$

where d_{ij} is the distance between the cities i and j .

The pheromone trails are modified by ants during the algorithm execution in order to store information about 'good' solutions. We apply the Ant Colony System (ACS) [9,10], a particular ACO algorithm to the problem on hand, which follows the algorithmic scheme given below:

```
1: Set parameters, initialize pheromone trails
2: while (termination condition not met)
3: ConstructSolutions
4: (ApplyLocalSearch)
5: UpdateTrails
6: end while
```

ACO are solution construction algorithms, which, in contrast to local search algorithms, may not find a locally optimal solution. Many of the best performing ACO algorithms improve their solutions by applying a local search algorithm after the solution construction phase. Our primary goal in this work is to analyze the manufacturing related application capabilities of ACO, hence in this first investigation we do not use local search.

3.3 Particle Swarm Optimization (PSO)

The initial ideas on particle swarms of Kennedy and Eberhart were essentially aimed at producing computational intelligence by exploiting simple analogues of social interaction, rather than purely individual cognitive abilities [34]. The first simulations [20] were influenced by Heppner and Grenander's work [16] and involved analogues of bird flocks searching for corn. These soon developed [9][10] into a powerful optimization method— Particle Swarm Optimization (PSO).

PSO is an optimization algorithm that is based on swarm intelligence principle [9], which are widely used in application domains such as function optimization, neural network training, fuzzy system control and so on at present [33]. It has been proved to be very effective for solving global optimization in various engineering application such as image and video analysis and design and optimization of communication networks. However, most applications in this field are using PSO to train ANN. A direct application of PSO variant in maintenance optimization will be shown in this paper.

3.3.1 Basic PSO Algorithm Description

The Particle Swarm Optimization (PSO) algorithm is a heuristic approach motivated by the observation of social behavior of composed organisms such as birds flocking (Fig.5). A number of simple entities – the particles – are placed in the search space of some problem or function, and each evaluates the objective function at its current location. Each individual in the particle swarm is composed of D dimensional vectors, where D is the dimensionality of the search space.

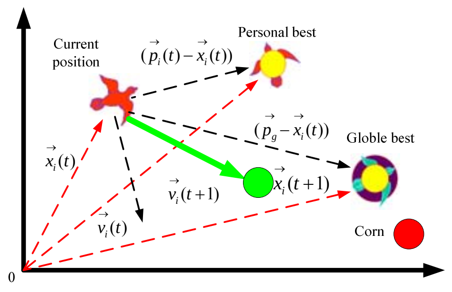


Fig. 5. Bird Flocking of PSO

The current position $\vec{x}_i$ can be considered as a set of coordinates describing a point in space. If the current position is better than any that has been found so far, then the coordinates are stored in the vector $\vec{p}_i$. The value of the best function result so far is stored in a variable that can be called $\vec{p}_g$. The objective, of course, is to keep finding better positions and updating $\vec{p}_i$ and $\vec{p}_g$. New points are chosen by adding $\vec{v}_i$ coordinates to $\vec{v}_i$, and the algorithm operates by adjusting $\vec{v}_i$, which can effectively be seen as a step size. The steps of implementing PSO are shown as follows:

1: Initialize a population array of particles with random positions and velocities on D dimensions in the search space.

2: Loop

3: For each particle, evaluate the desired optimization fitness function in D variables.

4: Compare particle's fitness evaluation with that of its $\vec{p}_i$. If current value is better than that of $\vec{p}_i$, then set $\vec{p}_i$ equal to the current coordinates.

5: Identify the particle in the neighborhood with the best success so far, and assign it to the variable $\vec{p}_g$.

6: Change the velocity and position of the particle according to the following equation:

$$\vec{v}_i(t+1) = \omega \cdot \vec{v}_i(t) + c_1 \cdot r_1\left(\vec{p}_i - \vec{x}_i(t)\right) + c_2 \cdot r_2(\vec{p}_g - \vec{x}_i(t)) \quad (9)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (10)$$

Where: ω is the inertia weighting; c_1 and c_2 are acceleration coefficients, positive constraint; r_1 and r_2 are the random numbers deferring uniform distribution on [0, 1]; i represents i^{th} iteration.

7: If a criterion is met (usually a sufficiently good fitness or a maximum number of iterations), exit loop.

8: End loop

In PSO, every particle remembers its own previous best value as well as the neighborhood best; therefore it has a more effective memory capability than an algorithm such as the GA. In addition, PSO is easier to implement and there are fewer parameters to adjust compared with GA [8].

3.3.2 Discrete PSO (DPSO) Algorithm Description

The general concepts behind optimization techniques initially developed for problems defined over real-valued vector spaces, such as PSO, can also be applied to discrete valued search spaces where either binary or integer variables have to be arranged into particles [8]. When integer solutions (not necessarily 0 or 1) are needed, the optimal solution can be determined by rounding off the real optimum values to the nearest integer. DPSO has been developed specifically for solving discrete problems. The new velocity and position for each is determined according to the velocity and position update equations given by (8) and (9).

$$\vec{v}_i(t+1) = round(\omega \cdot \vec{v}_i(t) + c_1 \cdot r_1\left(\vec{p}_i - \vec{x}_i(t)\right) + c_2 \cdot r_2(\vec{p}_g - \vec{x}_i(t))) \quad (11)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (12)$$

In equation (11), the value of velocity is binary or integer because *round ()* function can round off the value.

3.3.3 Improved DPSO (IDPSO) Algorithm Description

DPSO or PSO performs well in the early iterations, but they have problems approaching a near-optimal solution. If a particle's current position accords with the global best and its inertia weight multiply previous velocity is close to zero, the particle will only fall into a specific position. If their previous velocities are very close to zero, all the particles will stop moving around the near-optimal solution, which may lead to premature convergence of algorithm. All the particles have converged to the best position discovered so far which may be not the optimal solution. So, an improved DPSO is proposed here.

In IDPSO, before updating the velocities and positions in every iteration, the particles are ranked according to their fitness values in descending order. Select the first part of particles (suppose mutation rate is α, fist part is (1-α)) and put them into the next iteration directly. Regenerate the rest part of particles (α) randomly. In this project, we can regenerate the positions and velocities according to the following equation:

$$x_{id} = round(rand \cdot (S^{\max}(j) - S^{\min}(j)) + S^{\min}(j)) \quad (13)$$

$$v_{id}(t) = v_{max} - round \cdot (rand \times 2v_{max}) \qquad v_{id}(t) \in [-v_{max}, v_{max}] \quad (14)$$

Because of the characteristics of the flexible manufacturing environment, PSO needs to be discretized. The PSO was modified in order to improve the optimization effect. Therefore, an improved discrete PSO (IDPSO) was applied in this case.

4. Results and Comparison

4.1 Bee Colony Algorithm

There exist in literature [36] many ways to implement a BCA algorithm. In this paper the bee colony algorithm was implemented in Matlab. For inital tests, we defined the number of employed bees or initial solutions as 100, the maximum number of cycles as 300, and the scout bees as 10% of employed bees. During the search, the stagnation criterion was the non-improvement of the solutions for 10% of the cycles. When stagnation occurred, explosion was performed. All experiments were run to evaluate the object recognition task in digital color and grey images.

The objective of the experiment is to identify the strategies that maximize the average fitness and the number of best solutions that have fitness values greater than 0.95. This value was empirically found and indicates that the object is identified by the algorithm with almost correct coordinates, except by a small tolerance.

The number of food sources in the program can affect to the precision and the velocity of the program. The variation of running time of the program with different number of food sources is shown below to appreciate the differences. In both runs the rule of the same number of employed and onlooker bees have been kept.

4.1.1 With 500 Food Sources

For this experiment, the limit of iterations has been eliminated. This is to avoid be many errors due to the fact that with less food sources there should be more iterations. The Y axis shows the Fitness Values in all plots. With 500 food sources the following was observed:

- The average running time is of 14.70 seconds.
- There has not been any error in the detection of the correct coordinates, all of the templates have reached an NCC higher than 0.52 before 500 iterations.

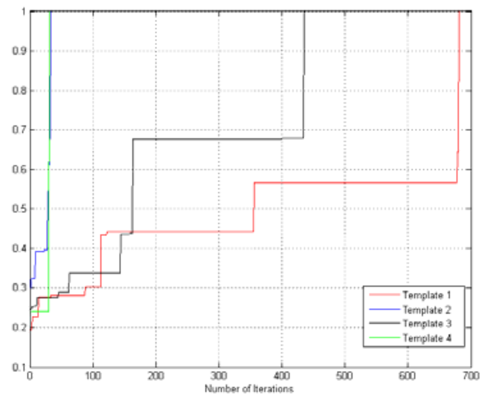


Fig. 6. Testing with 500 food sources

4.1.2 With 100 Food Sources

- The average running time is of 9.19 seconds.
- All of the templates reached NCC value higher than 0.52 before 500 iterations.

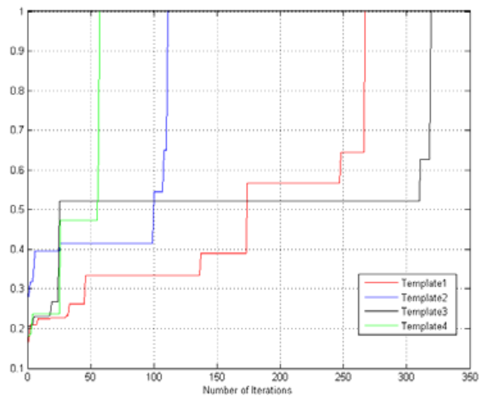


Fig. 7. Testing with 100 food sources

4.1.3 With 10 Food Sources

It was observed that the average number of iterations per template is significantly bigger (4,500) than with more food sources.

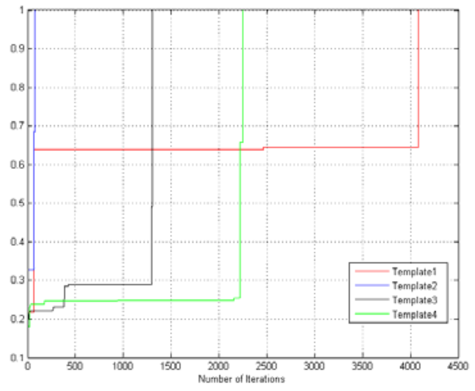


Fig. 8. Testing with 10 food sources

We also noticed that a minimum number of food sources of 10 was the bottom limit required to obtain any resolution with the Bee Colony Algorithm application to the discussed problem.

4.2 Improved Discrete Particle Swarm Optimization (IDPSO)

4.2.1 With 150 particles

To implement IDPSO, a population size of 150 particles was chosen to provide sufficient diversity into the population taking into account the dimensionality and complexity of the problem. This population size ensured that the domain was examined in full but at the expense of an increase in execution time. The other parameters of DPSO and IDPSO were: c1 = c2 = 2.0, ω = 1.2 - 0.8 with linearly decreasing, total iteration = 300 and V ∈ [-3, 3].

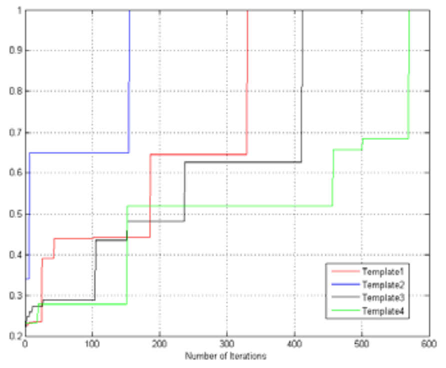


Fig. 9. Testing with 150 particles

The average NCC value of the templates obtained in the experiments was .998 or greater in less than 1000 iterations.

4.2.2 With 500 particles

The average NCC value of the templates obtained in the experiments was .998 or greater in less than 1000 iterations.

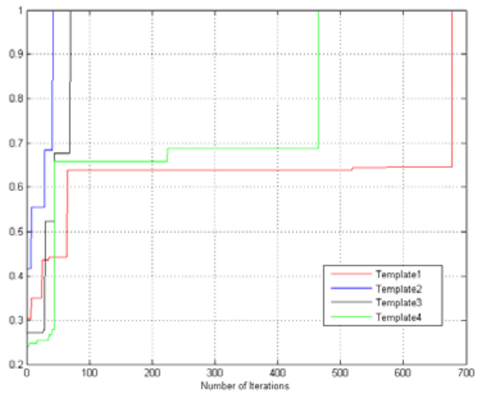


Fig. 10. Testing with 500 particles

4.3 Ant Colony Optimization (ACO)

With ACO we chose the following settings $m = 10, \beta = 2, q_0 = 0.98, \alpha = \rho = 0.1$ and $\Delta\tau_{ij}^k = \frac{Q}{L_k}$.[27][28]

The location of the four templates/markers (Figure 11) by the three algorithms is shown in Figure 12. It took an average 8.86 seconds for ACO to find the four templates, and hence the fastest of the three algorithms. Results showed the limits of robustness of the Bee Colony Algorithm, for different food sources. When compared with the results obtained by a particle swarm algorithm [36] for the same problem, they are generally equivalent. The average time taken by ACO, was the closest match to the robot assembly cell takt time of 9 seconds that would be required to establish a lean workflow and reduce machine starvation at the manufacturing facility.

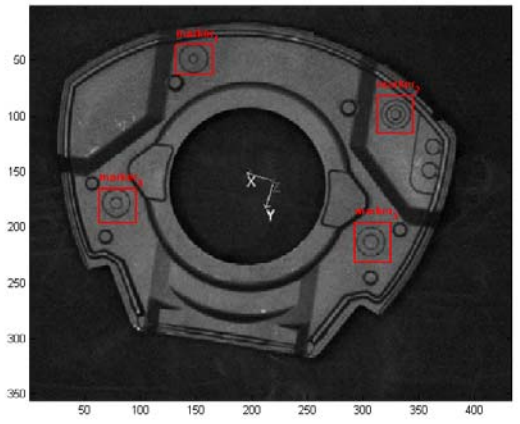


Fig. 11. Templates detected by ACO algorithm

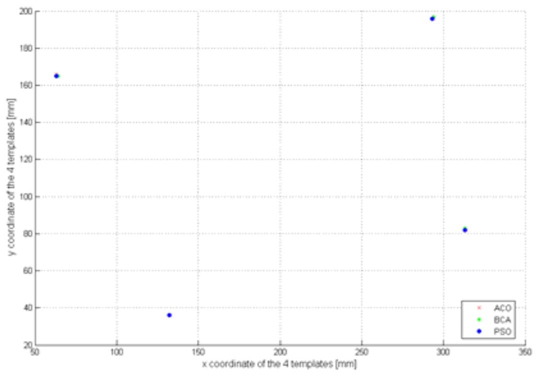


Fig. 12. Coordinates of the 4 templates as solved by ACO, BCA and modified PSO

5. Conclusion

In this paper, the BCA algorithm was tested with variants of Particle Swarm Optimization and Ant Colony Algorithm, and a combination of different strategies, such as generation of scout bees, varying the number of food sources, and explosion of stagnated population. The performance of the Bee Colony Algorithm is good when dealing with images without scaling factor, but this wasn't necessary for our particular manufacturing case study scenario. The choice of algorithms for a manufacturing assembly scenario could vary with the required tact times in the assembly cell, and the production environment such as vibration, dust etc. With real world images, the performance degrades to certain limits, but still finds optimal solution in more than 75% of the cases, and with greater than 10 food sources. It is observed that the computational cost effectiveness of the BCA varies according to the number of food sources chosen. The algorithm can still offer good solutions in the presence of noise within reasonable ranges. Future work will focus on improving the robustness of the algorithm in such situations.

We plan to test other approaches such as comparing the performance of modification such as conventional weight aggregation (CWA) and dynamic weight aggregation (DWA) in multi-objective optimization problems [35], and also compare with other competing evolutionary algorithms, like Genetic Algorithm.

Acknowledgments

Financial support from the AutoCast Consortium and the Norwegian Research Council is gratefully acknowledged.

First Author Rhythm Suren Wadhwa is a PhD student at the department of production and quality engineering, NTNU. She has worked in the Manufacturing Automation industry for five years. Current research interests include assembly automation, optimization techniques, assembly simulation and industrial robotics. She was the president of Society of Women Engineers at the University of Michigan. She has a Masters Degree in Mechanical Engineering and Bachelors degree in Manufacturing Processes Automation Engineering.

Second Author Terje Kristoffer Lien is a Professor in Manufacturing Automation and Robotics at the department of Production and Quality Engineering, NTNU. He has been active in the deveopment of cellular manufacturing systems. His work has attracted international interest, in particular the use of force feedback as a programming tool, and as an enhancment of the control of robots used for grinding operations.
.

References

[1] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," presented at International Joint Conference on Artificial Intelligence, Vancouver, 1981.

[2] Brunelli, R.Template Matching Techniques in Computer Vision: Theory and Practice. New York: John Wiley & Sons, 2009.

[3] Chan, T.S., Pak, H.A., Heuristic job allocation in a flexible manufacturing system. Int J Adv Manuf Technol 1(2):69-90, 1986.

[4] Chidambaran, C. and Lopes, H. S., (2010) An Improved Artificial Bee Colony Algorithm for the Object Recognition Problem in Complex Digital Images Using Template Matching, International Journal of Natural Computing Research, Vol1, Issue2, pp.54-70.

[5] Chisman, J.A. Manufacturing cell: analytical set up times and part sequencing. Int J Adv Manuf Technol 1(5):55-60, 1986.

[6] Cole, L., Austin, D., & Cole, L., Visual object recognition using template matching. In Proceedings of the Australasian Conference on Robotics and Automation. 2009

[7] Curkovic, P. and Jerbic, B. (2007) Honey-Bees Optimization Algorithm applied to path planning problem, International Journal of Simulation Modeling, Vol 3, pp. 154-164

[8] Del Valle, Y., Venayagamoorthy G. K., Mohagheghi, S., Hernandez. J., and Harley, R. G., (2008) Particle swarm optimization: basic concepts, variants and applications in power system, IEEE Trans. Evol. Comput., Vol. 2, pp. 171–195.

[9] Eberhart, R. C., and Kennedy, J., (1995) A new optimizer using particles swarm theory. Proceedings of Sixth International Symposium on Micro Machine and Human Science, pp. 39-43.

[10] Eberhart, R. C., Simpson, P. K., and Dobbins, R. W., (1996) Computational intelligence PC tools, Boston: Academic Press.

[11] Evans, H., & Zhang, M. Particle Swarm Optimization for Object Classification. In Proceedings of the 23rd International Conference on Image and Vision Computing (pp. 1-6),2007.

[12] Greenberg, H.H. A branch and bound solution to the general scheduling problem. Int J Oper Res 16:353-361.

[13] Hackel, S. Dippold, P. The bee colony inspired algorithm (BCiA): a two-stage approach for solving the vehicle routing problem with time windows. In Proceedings of the 11th Genetic and Evolutionary Computation Conference (pp. 25-32), 2009

[14] Hambecker, F., Lopes, H.S., & Godoy, W. Jr. Particle Swarm Optimization for Multidimensional Knapsack Problem (pp. 358-365),2007.

[15] Hoitomt, D.J., Luh P.B., Pattipati, K.R., A practical approach to job shop scheduling problems. IEEE Trans Robot Autom 9(1):1-13.

[16] Heppner, H., and Grenander, U., (1990) A stochastic non-linear model for coordinated bird flocks, The ubiquity of chaos, pp. 233–238. Washington: AAAS.

[17] Hii, A. J. H., Hann, C. E., Chase, J. G. and Van Houten, W. E. W., (2006) Fast normalized cross correlation for motion tracking using basis functions. Computer Methods andPrograms in Biomedicine. Vol. 82, No. 2, pp. 144-156.

[18] J. Bautista and J. Pereira, “Ant algorithms for assembly line balancing,” in Proc. ANTS2002, ser. LNCS, M. Dorigo et al., Eds., Springer Verlag, vol. 2463, pp. 65–75, 2002.

[19] Karaboga, D., Akay, B., A comparative study of artificial bee colony algorithm, Applied Mathematics and Computation, 214 (1), 108-132, 2009.

[20] Kennedy, J., and Eberhart, R. C. (1995) Particle swarm optimization. Proceedings of the IEEE international conference on neural networks IV, pp. 1942–1948.

[21] Koren, Y.U.A., Reconfigurable manufacturing systems: Key to future manufacturing. Journal of
Intelligent Manufacturing, 2000. 11(4).

[22] L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. In Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC’96), pages 622–627. IEEE Press, Piscataway, NJ, 1996.

[23] LaDou, J., (2006) Printed circuit board industry. International Journal of Hygiene and Environmental Health. Vol. 209, No. 3, pp. 211-219.

[24] Lemmens, N., (2006) To bee or not to bee: A comparative study in swarm intelligence. Master’s thesis, Maastricht University, Faculty of Humanities and Sciences. MICCIKAT 06-12.

[25] Lee, S.M., Jung H.J. A multi objective production planning model in flexible manufacturing environment. Int J Prod Res 27 (11): 1981-1992

[26] Lin, Y. H. and Chen, C. H., (2008) Template matching using the parametric template vector with translation, rotation and scale invariance. Pattern Recognition. Vol. 41, No. 7, pp.2413-2421.

[27] M. Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1):53–66, 1997.

[28] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B, 26(1):29–41, 1996.

[29] M. Gravel, W.L. Price, and C. Gagn´e, “Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic,” European Journal of OperationalResearch, vol. 143, pp. 218–229, 2002.

[30] Mishra, S.K., Performance of Differential Evolution and Particle Swarm Methods on Some Relatively Harder Multi-modal Benchmark Functions. Munich personal RePec (No.1743)

[31] Modegi, T. Small object recognition techniques based on structured template matching for high-resolution satellite images. SICE Annual Conference, 2168-2173

[32] Monkman G., Hesse S., Steinmann R., and Schunk, H. Robot Grippers, (2007).

[33] Pan Hongxia, and Wei Xiuye, (2009) Particle Swarm Optimization Algorithm with Adaptive Velocity and its Application to Fault Diagnosis, 2009 IEEE Congress on Evolutionary Computation, pp. 3075-3079.

[34] Riccardo Poli, James Kennedy, and Tim Blackwell (2007) Particle swarm optimization, Swarm Intelligence, Vol. 1, pp. 33-57.

[35] Parsopoulous, K.E., Vrahatis, M.N. Particle Swarm Optimization Method in Multi objective Problems, SAC 2002, Madrid

[36] Perlin, H.A., Lopes, H.S., & Centeno, T.M. Particle Swarm Optimization for object recognition in computer vision, 2008.

[37] Pham, D.T., Soroka, A.J.,Ghanbarzadeh, A., & Koc, E. Optimizing neural networks for identification of wood defects using bees algorithm. In Proceedings of the International Conference on Industrial Informatics (pp. 1346-1351),2006.

[38] Schulz, J., Mertens P., A comparison between an expert system, a GA and priority for production scheduling. In: Proceedings of the 1st international conference on operations and quantitative management, Jaipur, India, 2.506-513, 1997.

[39] Shankar K,Tzen Y.J. A loading and dispatching problem in a random flexible manufacturing system. Int J Prod Res 23: 579-595. 1985.

[40] Shaw, M.J., Whinston, A.B., An artificial intelligence approach to the scheduling of flexible manufacturing systems. IEEE Trans 21:170-182,1989.

[41] Silva, R.R., Lopes, H.S., & Lima, C.R.E. A compact genetic algorithm with elitism and mutation applied to image recognition (LNCS 5227, pp.1109-1116),2008.

[42] Singh, A. An artificial bee colony algorithm for leaf-constrained spanning tree problem. Applied Soft Computing, 9(2), 625-631. 2009.

[43] Steeke, K.E., Soldberg, J.J., Loading and control policies for a flexible manufacturing system. Int J Prod Res 19(5):481 -490, 1982.

[44] Tereshko, V. & Loengarov, A. collective decision-making in honey bee foraging dynamics. Computing and Information Systems, 9(3), 1-7. 2005.

[45] Toker, A., Kondacki, S., Erkip, N., Job shop scheduling under a non-renewable resource constraint. J Oper Res Soc 45(8): 942-947, 1994.

[46] Wang, K., (2005) Applied Computational Intelligence in Intelligent Manufacturing Systems, Advanced Knowledge International Pty Ltd, Australia.

[47] Wang, K., (2010) Swarm Intelligence in Manufacturing Systems: Principles, Applications and Future Trends, IWAMA (2010)
[48] Wu, C.-H., D.-Z. Wang, et al. (2009). "A particle swarm optimization approach for components placement inspection on printed circuit boards." Journal of Intelligent Manufacturing 20(5): 551-551.

[49] Zhao,X., Lee, M.E., & Kim, S.H. Improved Image Thresholding using Ant Colony Optimization Algorithm. In Proceedings of International Conference on Advanced Language Processing and Web Information Technology (pp. 201-215) 2008.