A Variable Neighbourhood Search for Component Pick-and-Place Sequencing in Printed Circuit Board Assembly

Masri Ayob,

School of Computer Science & IT, The National University of Malaysia,
UKM 43600 Bangi, Selangor, Malaysia

Summary
This work presents a heuristic for component pick-and-place sequencing to improve the throughput of a multi-head surface mount device placement machine for assembling printed circuit board. We present a Variable Neighbourhood Monte Carlo Search (*VNMS*), which employs variable neighbourhood search with an Exponential Monte Carlo acceptance criterion. *VNMS* is a descent-ascent heuristic that operates on three sets of neighbourhood structures that are based on three different local searchers. The first two sets use a steepest descent and Exponential Monte Carlo local search, respectively whilst the third set uses a random 3-opt operator. The solution returned by a local search, after exploring a neighbourhood structure, will be accepted based on the *EMCQ* (Exponential Monte Carlo with counter) acceptance criterion. The novelty of the *VNMS* approach (in the context of *VNS*) is the concept of three stages of neighbourhood search, using an *EMCQ* acceptance criterion (at the *VNS* level) and the shaking procedure (which is only applied when the local searchers cannot find an improved solution). Results show that the *VNMS* consistently produces good quality solutions.

Key words:
Heuristic, component placement sequencing, variable neighbourhood search, printed circuit board assembly.

1. Introduction

Electronic components (possibly hundreds or thousands) are assembled onto a PCB (printed circuit board) using SMD (surface mount device) placement machine(s). SMD machines are categorised into five types, based on their specification and their operational methods. These are dual-delivery, multi-station, turret-style, multi-head and sequential machines [1]. In this work, we study a single SMD placement machine with a single PCB to schedule the component pick-and-place operation of a multi-head SMD placement machine. The machine has a fixed feeder carrier, a fixed PCB table and a positioning arm head that is equipped with many pipettes. The positioning arm head (that is movable simultaneously in the X-Y direction) is responsible for transporting components from the component feeder and then mounting them onto the PCB. The pipette (spindle) is located at the end of head and is used to hold a nozzle (tool or gripper). The nozzle is used to grasp the component [2]. The feeder carrier holds several feeder banks. Each feeder bank consists of several feeder slots where the component feeders are located. The PCB table holds the board in a locked position during a pick-place operation.

A sub tour (we refer to a sub tour to differentiate with an overall tour, which is an operation to place all the required components onto a single board) means an operation taken by the robot arm to pick up and place some components (depending on the number of pipette/nozzles per head) in a single trip. A sub tour of the heads begins by picking up some components from the feeders (simultaneously or sequentially depending on the pickup locations). Then, it travels in an X and Y direction (simultaneously) and positions itself at the point where the component will be mounted. Then the head moves down (Z-direction) and mounts the component on the board before returning to its original position and repeating these steps for the next locations on the board that have to be mounted in the same sub tour. After completing a sub tour, the head returns to the feeder location to begin another sub tour. Fig. 1 is an example of a multi-head placement machine.

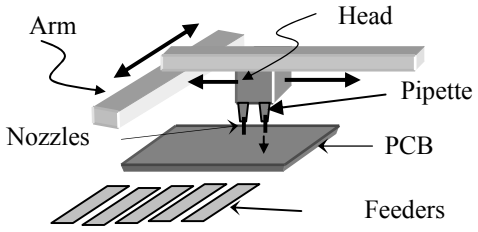

Fig. 1. An example of a pick and place multi-head placement machine.

There are many factors involved in determining the efficiency of the pick-place operation of multi-head placement machines. For example:

1) **Assigning PCB points to a sub tour**. As the robot arm is equipped with many pipette/nozzles, the

Manuscript received July 5, 2007
Manuscript revised July 25, 2007

problem is to determine the sets of PCB points that will be visited by the robot arm (i.e. to place a component) in the same sub tour. For example, if the robot arm is equipped with 8 pipette/nozzles, we may have 8 pickup points and 8 placement points in a sub tour.

2) **Assigning the pickup pipettes.** In this work we assume all components are the same size (we ignore nozzle size selection, i.e. the nozzles are common for all component types and there are no nozzle change operations). The issue is to determine which pipette should be used to pickup a component such that we minimise the robot arm travelling distance.

3) **Sequencing the component pickups**. The problem is to determine the sequence of picking up components in a sub tour to optimise the pickups.

4) **Sequencing the placement operation**. The problem is to determine the sequence of placing components in a sub tour to optimise the placements.

5) **Assigning the placement pipettes**. Again, the issue is to optimise the placement and we must ensure that the PCB points will receive the correct component type. This problem is almost the same as problem (2). However, in this problem, the placement pipette is assigned by aiming to minimise the placement operation (ignoring the cost of the pickup operation). Similarly, in problem (2) the assignment of pickup pipette is done based on minimising the pickup operations (ignoring the cost of the placement operation).

6) **Sequencing the sub tours**. The aim is to optimise the sequence of sub tours in order to minimise the assembly cycle time.

The aim of optimising the pickup and placement operations is to minimise the assembly cycle time. However, there is a trade-off between optimising the pickup and optimising the placement. Indeed, these sub problems are tightly linked. The problem requires various neighbourhood structures in order to carry out an effective search. Generally, local search approaches, such as simulated annealing and tabu search, have difficulty in solving this problem since they are usually dealing with single neighbourhood, whilst the complexity of the problem requires many heuristics for exploring various neighborhood structures. Therefore, heuristics that are capable of exploring multi-neighborhood structures such as Hyper-heuristics [3] and Variable Neighbourhood Search (VNS) [4] are more applicable when solving this type of problem. The hyper-heuristic approach has been successfully studied in our previous work [5],[6]. This suggests that a VNS approach might be suitable as hyper-heuristics allowing various neighbourhoods to be searched, which is the principle behind VNS. Our previous works on improving the SMD placement machine throughput can be found in [7],[8].

The printed circuit board problem is easy to describe, but due to the NP [9] characteristics of the sub problems involved, it is practically impossible, for reasonably sized instances, to solve to optimality by exact approaches [10]. De Souza and Lijun [11] discussed that the complexity of concurrent machine operations also causes difficulties in formulating a realistic mathematical programming problem. In practice, heuristic algorithms (such as *VNS*, simulated annealing and tabu search) are used. These approaches can generate good solutions efficiently, at a reasonable computational cost, but without being able to guarantee either feasibility or optimality [12]. As such, the aim of this work is to develop an algorithm that is capable of generating reasonably good solutions in reasonable times for sequencing component pick-and-place operations.

One motivating factor for our work comes from the fact that, as far as we are aware, none of these studies attempt to apply variable neighbourhood search or hyper-heuristic approaches to optimise the SMD placement machine.

2. Related Work

Some studies which have examined the PCB assembly problem include Altinkemer et al. [2], Crama et al. [13], Grunow et al. [14], Ho and Ji [15].

Since the arm and head of a multi-head SMD placement machine can move concurrently in both the X and Y axis, Altinkemer et al. [2] used the chebychev distance (i.e. they calculated the distance as the maximum of the movements in the X or Y direction). They formulate a problem for an SMD placement machine that has a rotating single head that can pick up many components and mount them on the board in a same sub tour. For the case of a moveable feeder carrier (i.e. the feeder of the component type that will be processed next, can move towards the tool magazine while the head is mounting another component type), the distance between the feeder locations and the points on the PCB can be measured from a fixed point next to the tool magazine. The simultaneous movement enables each component type to have the same origin and destination point, and thus allow the formulation to be an independent capacitated vehicle routing problem (VRP). Since the distance between a point on the PCB and feeder is not dependent on where the component is located among feeders, the feeder setup problem does not have to be integrated with the pick-and-place sequencing decisions [2]. For the case of a stationary feeder carrier, they formulate the problem as a combination of assignment-like and vehicle-like problems.

They first solve a VRP for each component type at every possible feeder location, and then use this feasible solution as the cost of assigning the component type to the particular feeder location. They argue that their integrated algorithm provides a feasible solution with an error gap less than or equal to the maximum error gap of the VRP costs.

Crama et al. [13] provides an exhaustive survey of some of the major optimisation problems arising in the area of production planning for the assembly of PCBs. By considering a long-term decision with the *mixed demand* and the *fixed shop layout,* they classified the production planning problems into eight sub problems:

SP1: Assigning PCB types to product families and to machine groups;
SP2: Allocating component feeders to machines;
SP3: Partitioning component locations on the PCB, indicating which components are going to be placed by each machine (for each PCB type);
SP4: Sequencing the PCB types (for each machine group);
SP5: Assigning component feeders to slots on the feeder bank/carrier (for each machine);
SP6: Sequencing the component pick-and-place operation (for each pair consisting of a machine and a PCB type);
SP7: Component retrieving plan. That is the rule indicating from which feeder the component should be retrieved from (if feeder duplications are allowed);
SP8: A motion control specification, indicating where the pick and place device should be located to pickup and place the component.

Normally, the decision as to which of the SP1-SP8 sub problems to be solved, is based on which sub problem will minimise the assembly cycle time [13]. However, the dilemma is that all the sub problems are intertwined and the question arises as to which one should be solved first. As a consequence, some researchers tackled the problem in an iterative manner, instead of a one-pass procedure through each of the sub problems. The technological characteristics of the SMD placement machine can also influence the nature of the problem to be solved and the formulation of the associated models [13].

Recently, Ho and Ji [15] employed a hybrid genetic algorithm (that was proposed in [16]) to simultaneously solve the component pick-and-place sequencing, feeder setup and component retrieval problem for turret-type SMD placement machines. The component retrieval problem is solved using a nearest neighbour heuristic which aims to minimise the movement of the feeder carrier. Results show that the approach is capable of producing a better solution when there are feeder duplications.

3. The Assumptions and Objective Function

The quality of a schedule can be evaluated by a placement time function, known as the assembly cycle time, CT. We model a pick-and-place multi-head placement machine that has a single head equipped with G nozzles/pipettes, fixed PCB (printed circuit board) table and stationary feeder carrier (same as in [5],[6],[7]). This models many real-world machines.

The objective function is to minimise the assembly cycle time, CT, by minimising the travelling distance of the robot to perform pick and place operation:

$$\textbf{Minimise} \quad CT = \sum_{j=1}^{B} \left[P(j) + I(j) \right] \tag{1}$$

Where:

I(j) : *the time taken for the robot arm to travel from feeders to PCB points and place some components in the j^{th} sub tour;*
P(j) : *the time taken for the robot arm to travel from PCB points to feeders and pick some components in the j^{th} sub tour;*
B : *the total number of sub tours;*

The assembly cycle time, CT (equation 1) is a function of the time to pick-and-place all the available/required components onto a PCB. This objective function (equation 1) could be applicable for various types of SMD placement machines such as a dual-delivery, sequential pick-and-place, multi-station and particularly the multi-head. However, the calculation of *P(j)* and *I(j)* are machine dependent due to the machine specifications and operational methods. In this work, *P(j)* and *I(j)* is a summation of the robot travelling distance divided by the robot speed, plus component pickups and placements time, respectively. Since the robot (i.e. the arm and head of SMD placement machine) can move simultaneously in the X-Y axis, the robot travelling distance is dictated by the maximum of the X or Y travelling distance, i.e. a chebychev distance.

In this work, we assume that the CT is only dictated by the robot travelling distance by ignoring other optimisation factors such as nozzle changes, component feeder transportation time, simultaneous (gang) pickup, tray feeder reloading time etc. These assumptions are made in order to abstract the problem such that we can model it as a multi-tour travelling salesman problem (which is known to NP-hard). However, as addressed by Kumar and Luo [17], the placement sequencing problem can be viewed as a "*generalised Travelling Salesman Problem (TSP)*" where not only the overall travel time depends on the travel sequence (as in the standard TSP), but the distances between any pair of nodes is sequence

dependent. We also assume that all components are aligned while the robot arm is moving (i.e. on the fly alignment).

4. Variable Neighbourhood Search

Variable neighbourhood search (*VNS*) is a relatively unexplored approach [18]. It was introduced by Mladenović & Hansen [19]. By systematically changing the neighbourhood within a local search algorithm, *VNS* can explore distant neighbourhoods of the current solution, and jump to a new solution if it is superior to the current solution [19]. The shaking procedure in the basic *VNS* approach provides a diversification factor in order to reach a distant neighbourhood whilst the local search will intensify the search to make it converge to local optima. The basic *VNS* algorithm is a descent heuristic but it can be transformed to a descent-ascent or a best improvement method [18].

Recently, there has been increasing interest in the VNS approach. For example, Avanthay et al. [20], developed an adaptation of VNS to solve the graph coloring problem with a Tabucol (a variant of tabu search) algorithm (see [21]) as a local search. They used three neighbourhood structures; these being vertex, class, and non-increasing neighbourhoods. Their VNS however is not superior to the hybrid algorithm proposed by Galinier and Hao [22] that integrates a tabu search and a genetic algorithm. Fleszar and Hindi [23] applied a VNS approach in solving the resource-constrained project scheduling problem. Results show that the quality of solutions and lower bounds achieved by VNS with enhanced moves and precedence augmentation is impressive.

5. Variable Neighbourhood Search for Component Pick-and-Place Sequencing

In this work we develop two types of *VNS* approaches, these being basic *VNS* (*VNS-1*, *VNS-2* and *VNS-3*) and a Variable Neighbourhood Monte Carlo Search (*VNMS*). The first one is based on basic *VNS* [18].[19] that operates on different local searches. The second type (i.e. *VNMS*) has three stages of neighbourhood structures where each stage operates on a different set of local searchers.

Let n_w, $w=1,2...,W$, be a set of predefined neighbourhood structures, and $n_w(x)$ is the set of solutions in the w^{th} neighbourhood of x, $f(x)$ is the quality of solution x. W is the total number of neighbourhood structures to be used in the search.

In this work, we use three acceptance criteria, which are applied at the *VNS* level and/or the local search level. These being:

1) Descent: Only accepts an improved solution.
2) *EMC* (Exponential Monte Carlo): This accepts an improved solution and probabilistically accepts a worse solution. The probability of accepting a worse solution decreases as δ (where $\delta=f(x')-f(x)$) increases. The trial solution, x', is accepted if a generated random number is less than $e^{-\delta}$. The $f(x')$ and $f(x)$ are the qualities of the trial solution and the initial/incumbent solution, respectively
3) *EMCQ* (Exponential Monte Carlo with counter): Similar to *EMC*, this accepts an improved solution. However it may also accept a worse solution with the probability of acceptance increasing as δ decreases and the counter of consecutive none improvement iteration, Q, increases. The probability is computed by $e^{-\theta/\tau}$ where $\theta=\delta*t$, $\tau=\rho(Q)$ and $\delta=f(x')-f(x)$. Computation time, t, is measured as minute being the unit time (in our case). The trial solution, x' is accepted if a generated random number is less than $e^{-\theta/\tau}$. θ and τ are defined such that we ensure that the probability of accepting a worse solution decreases as the time increases and δ increases. The factor of time is included in this formulation as an intensification factor. At the beginning of the search, moderately worse solutions are more likely to be accepted but as the time increases, worse solutions are unlikely to be accepted. However, the probability of accepting a worse solution increases as the counter of consecutive none improvement iterations, Q, increases. This is a diversification factor. $\rho(Q)$ is a function to intelligently control Q. In this work we use $\tau=v*Q$ where $0\leq v\leq 1$, in order to limit the acceptance probability. However, our preliminary experiment on the parameter sensitivity of v shows that the *EMCQ* algorithm with $v=1$ performs the best. Therefore, we set $\tau=Q$. We originally proposed these acceptance criteria (*EMC* and *EMCQ*) in [6].

VNS-1 is absolutely a basic descent heuristic that operates with a random descent local search (*RD-LS*) by using descent acceptance criterion at the local search and *VNS* levels. *VNS-1* only accepts an improved solution returned by the *RD-LS*. We use a 2-opt operator in the local search. A local search heuristic explores a single neighbourhood solution space by performing a sequence of local changes to an initial solution, in order to improve the quality of a solution until a local optimum is found [23]. A neighbouring solution is produced by a move-generation mechanism and will be selected depending on the pre-defined acceptance criteria. In *VNS-1*, the local search starts by generating a solution x' from the w^{th} neighbourhood of x ($x' \in n_w(x)$). Starting from x' as an

initial solution, the *RD-LS* sequentially visits at least [1]P neighbours around the w^{th} neighbourhood of x'. The *RD-LS* accepts the first improving neighbour solution and moves to this solution. Then the search continues until the stopping condition is true. We use two stopping conditions in the local search. These being a number of visited neighbours and the solution's quality. If the current neighbour solution is better than the best obtained solution by the *RD-LS*, then the search continues even after exceeding the limit of visited neighbours.

VNS-2 itself is a descent heuristic but the *EMC-LS* local search is a descent-ascent heuristic. The *EMC-LS* also begins by generating a solution x' from the w^{th} neighbourhood of x ($x' \in N_w(x)$). Starting from x' as an initial solution, *EMC-LS* sequentially visits P neighbours around the w^{th} neighbourhood of x'. The search sequentially moves to the new accepted solution even though the new solution is worse (depending on the *EMC* acceptance criterion). The *EMC-LS* exits when it reaches the limit of visited neighbours (P). It returns the best obtained solution to the *VNS* heuristic.

VNS-3 is also a basic *VNS* but it is a descent-ascent *VNS* with *EMCQ* acceptance criteria at *VNS* level that operates with an *EMC-LS* local searchers. Instead of simple descent heuristics, the *VNS-3* transforms the descent *VNS* into a descent-ascent *VNS*. It applies the *EMCQ* acceptance criterion to probabilistically accept a worse solution returned by the local searches. *VNS-3* uses the same *EMC* local search, *EMC-LS* as in (*VNS-2*).

VNMS is a descent-ascent heuristic that accepts an improved solution but probabilistically accepts a worse solution based on the *EMCQ* acceptance criterion that operates at the higher level. *VNMS* has a three stages neighbourhood search, these being a steepest descent, *EMC* (Exponential Monte Carlo) and shaking. The idea is to initially explore the best neighbour in each neighbourhood structure before applying a random element. The search direction changes, when the steepest descent local search that returns the best neighbour is unable to find an improved solution (by applying *EMC* local searchers). The shaking procedure is only applied after the *EMC* local search cannot find an improved solution. The novelty of the VNMS approach (in the context of VNS) is the concept of three stages of neighbourhood search, using an EMCQ acceptance criterion at the VNS level and the shaking procedure (which is only applied when the local searchers cannot find an improved solution). The VNMS algorithm is described in fig. 2.

[1] $P = \sum_{e=1}^{G-1} e$ *where G is the number of pipettes per head.*

Step A: *(Initialisation)*
(1) Select the set of neighbourhood structures n_d, $d=1,2\ldots,D$, n_e, $e=1,2\ldots,E$, n_z, $z=1,2\ldots,Z$ that will be used in the search; find an incumbent solution x; choose a stopping condition;
(2) Record the best obtained solution, $x_{best} \leftarrow x$ and $f(x_{best}) \leftarrow f(x)$;

Step B: *(Steepest Descent Neighbourhood Search)*
(1) Set $d \leftarrow 1$;
(2) Do
a). Exploration of neighbourhood. Find the best neighbour, x' from the d^{th} neighbourhood of x ($x' \in n_d(x)$);
b). Acceptance criteria (Move or not). Apply the EMCQ acceptance criterion. If x' is accepted, then $x \leftarrow x'$.
c). If the new solution is better than the incumbent solution, continue the search with n_d; otherwise, set $d \leftarrow d+1$.
d). If $f(x') < f(x_{best})$ then $x_{best} \leftarrow x'$ and $f(x_{best}) \leftarrow f(x')$;
Until $d=D$ or the stopping condition is met.

Step C: *(EMC Neighbourhood search)*
(1) Set $e \leftarrow 1$;
(2) Do
a). Local search. Systematically generate a neighbour, x'' from the e^{th} neighbourhood of x ($x'' \in n_e(x)$) using EMC local search. Return the best obtained solution x';
b). Apply the same acceptance criterion as in Step B(2b);
c). If the new solution is better than the incumbent, continue the search with n_e; otherwise, set $e \leftarrow e+1$.
d). If $f(x') < f(x_{best})$ then $x_{best} \leftarrow x'$ and $f(x_{best}) \leftarrow f(x')$;
Until $e=E$ or the stopping condition is met.

Step D: *(Shaking)*
(1) Randomly select neighbourhood structures n_z;
(2) Do
a). Generate a point x' at random from the z^{th} neighbourhood of x ($x' \in n_z(x)$);
b). Apply the same acceptance criterion as in Step B(2b);
c). If $f(x') < f(x_{best})$ then $x_{best} \leftarrow x'$ and $f(x_{best}) \leftarrow f(x')$. If x' is accepted, then goto step B;
Until x' is accepted or the stopping condition is met.

Step E: *(Termination)*
Goto Step B if stopping condition=false, otherwise terminate.

Fig. 2. The *VNMS* algorithm for component placement sequencing of multi-head placement machine (minimisation problem).

The steepest descent neighbourhood search has D (six in this work) number of steepest descent local searches. Each steepest descent local search will explore a different neighbourhood structure and returns the best neighbour. The returned solution will be accepted based on the *EMCQ* acceptance criterion at the *VNMS* level.

After exploring the steepest descent neighbourhood structures, *VNMS* will explore the *EMC* neighbourhood

structures at the *EMC* neighbourhood search stage. The *EMC* neighbourhood search is meant to divert the search direction by probabilistically accepting some worse solution. This stage has *E* (six in this work) number of *EMC* local searchers. These *EMC* local searchers are the same as the *EMC* local searches in *VNS-2* and *VNS-3*. Indeed, the *VNS-3* and the *EMC* neighbourhood search stage are the same.

Finally, *VNMS* will apply the shaking stage if the steepest descent neighbourhood search and *EMC* neighbourhood search stages are unable to return an accepted solution. In this work, we use 3-opt operator in the shaking procedure. As the problem domain involves six sub-problems, then we apply 3-opt operator for each of the sub-problem. The three points that are randomly chosen to be swapped are selected from the same sub tour (for generating a neighbour of pickup's pipette, pickup's sequence, placement's pipette and placement's sequence) or a different sub tour (for generating a neighbour of sub tour or sub tour's sequence).

Each trial solution obtained by the shaking procedure will be evaluated based on the *VNMS* acceptance criterion. The shaking procedure is repeated until the obtained solution is accepted or the stopping condition is met. Once the obtained solution is accepted, *VNMS* will continue the search by repeating the whole procedure again starting from Step B in fig. 2, steepest descent neighbourhood search.

The local searchers used in the *VNMS* are listed in table 1. H_c denotes the heuristic ID whilst the acceptance criteria determine how to accept the solution in the local search.

As the problem domain requires six different swapping operations, we developed three sets of neighbourhood structures, having six swapping operations in each set to produce six different neighbourhood structures in each set. The three sets are based on the local searchers that are steepest descent (SD-LS), exponential monte carlo (EMC-LS) and random 3-opt operator. The steepest descent neighbourhood search stage uses six local searchers, these being H_c={0,1,2,3,4,5}. The EMC neighbourhood search stage uses the next six local searchers that are H_c={6,7,8,9,10,11} whereas the other six local searchers (i.e. H_c={12,13,14,15,16,17}) that are actually just a simple 3-opt operators, are used in the shaking procedure.

Table 1: A list of local searches used in VNMS

H_c	Local search name	Acceptance Criteria
0	Swap placement sub-tour using SD-LS	choose the best
1	Swap pickup nozzle using SD-LS	choose the best
2	Swap pickup sequence using SD-LS	choose the best
3	Swap placement nozzle using SD-LS	choose the best
4	Swap placement sequence using SD-LS	choose the best
5	Swap sub-tour's sequencing order using SD-LS	choose the best
6	Swap placement sub-tour using EMC-LS	EMC
7	Swap pickup nozzle using EMC-LS	EMC
8	Swap pickup sequence using EMC-LS	EMC
9	Swap placement nozzle using EMC-LS	EMC
10	Swap placement sequence using EMC-LS	EMC
11	Swap sub-tour's sequencing order using EMC-LS	EMC
12	Swap placement sub-tour using random 3-opt	None
13	Swap pickup nozzle using random 3-opt	None
14	Swap pickup sequence using random 3-opt	None
15	Swap placement nozzle using random 3-opt	None
16	Swap placement sequence using random 3-opt	None
17	Swap sub-tour's sequencing order using random 3-opt	None

6. Hyper-heuristic

As the differences among various SMD placement machine specifications and operational methods significantly influence the solution approaches, it is difficult to evaluate the performance of the *VNMS* against other approaches. As such, we have developed hyper-heuristic approaches for comparison purposes.

Hyper-heuristics are (meta-)heuristics that can operate over a set of (meta-) heuristics [3]. The hyper-heuristic framework manages a set of low level heuristics, which operates at a higher level of abstraction without having access to the problem domain-knowledge [3]. In this work, we have developed nine hyper-heuristic approaches. These hyper-heuristics operate on different sets of low-level heuristic and acceptance criteria:

1) *AM-sdEmc3opt* (All Move that operates with *SD-LS*, *EMC-LS* and 3-opt LLH): Randomly selects LLH and accepts any solution returned by the LLH. There are eighteen LLHs (these are the same local searchers used in *VNMS* approach).

2) *OI-sdEmc3opt* (Only Improving that operates with *SD-LS*, *EMC-LS* and 3-opt LLH): Randomly selects LLH and only accepts an improved solution returned by the LLH.

3) *OICF-sdEmc3opt* (Only Improving Choice Function that operates with *SD-LS*, *EMC-LS* and 3-opt LLH): Select LLH based on historical performance (Cowling et al., 2001) and only accepts an improved solution returned by the LLH.

4) $\Lambda(h_k)=max\{\alpha*f_1(h_k)+ \beta*f_2(h_j,h_k)+ \sigma*f_3(h_k)\}$ *Where*

$\Lambda(h_k)$ is a function used by Cowling et al. [24] to choose a heuristic (which has the largest $\Lambda(h_k)$) to be applied at the next iteration. $f_1(h_k)$ is the cumulative performance rate of heuristic h_k, $f_2(h_j,h_k)$ is the cumulative performance rate of consecutive pairs of heuristics (heuristic h_j followed by h_k) and $f_3(h_k)$ is the CPU time which has elapsed since heuristic h_k was

last called. Details of the algorithm can be found in [24]. If the time taken by each LLH to make a swap is too short (approximate to zero milliseconds), then we set the duration as 1.

5) *AMCF-sdEmc3opt* (All Move Choice Function that operates with *SD-LS*, *EMC-LS* and 3-opt LLH): Same as *OICF-sdEmc3opt* but in this case we accept all solutions returned by the LLH's.

6) *EMCQ-sdEmc3opt* (Exponential Monte Carlo with Counter that operates with *SD-LS*, *EMC-LS* and 3-opt LLH): Randomly selects LLH and accepts the returned solution based on the *EMCQ* acceptance criterion.

7) *AM-sd* (All Move that operates with *SD-LS* LLH): Randomly selects LLH and accepts any solution returned by the LLH. There are only six steepest descent LLHs (that are the same steepest descent local searches used in *VNMS*).

8) *OI-2opt* (Only Improving that operates with 2-opt LLH): Randomly selects LLH and only accepts an improved solution returned by the LLH. This is a simple descent heuristic.

9) *OICF-2opt* (Only Improving Choice Function that operates with 2-opt LLH): Select LLH based on historical performance [24] and only accepts an improved solution returned by the LLH. This is a simple descent heuristic with a choice function.

10) *EMCQ-2opt* (*EMCQ* that operates with 2-opt LLH): Randomly selects LLH and accepts the returned solution based on the *EMCQ* acceptance criterion.

7. Computational Experiments

An initial solution is obtained by applying either a randomised or an ordered constructive heuristic (i.e. the PCB points are consecutively scheduled based on a sequence of PCB points that are sorted starting with the minimum of maximum (X,Y) coordinate, then with the minimum (X,Y) when there is a duplication of maximum (X,Y)). In the experiment we modelled the multi-head placement machine that has a head equipped with 8 pipettes/nozzles that can pick no more than 8 components in a sub tour. We further assume that the SMD placement machine can only pickup one component at a time (no simultaneous pickup) but the number of components that can be picked up in a sub tour is dependent on the number of pipettes/nozzles per head.

To simulate the pick-place operation of the placement machine, we set the speed of the robot arm, $V=10$ unit distance/unit time, the pickup and placement time, $\lambda = \theta = 0.5$ unit time. For the purpose of generating the random placement points, we set the length (*BL*) and the width (*BW*) of the board, such that the random PCB points fall within the limits. In the experiment we use two datasets (dataset N80K20_A and N240K40_F). The specification of these datasets is shown in table 2. These datasets are randomly generated using our random PCB generator software called PCBgen. PCBgen allows the user to set the required *N* (sum of placement points), *K* (sum of component types), *BW* and *BL*.

Table 2: Experimental datasets.

Dataset	N	K	BL	BW
N80K20_A	80	20	600	200
N240K40_F	240	40	1800	600

We ran the experiments using an Intel® Pentium®4 PC with 1.8GHz speed and 256 MB RAM. In this work we set α=1.0, β=0.01 and σ=0.5 (for OICF and AMCF). The parameters values are chosen based on the results obtained from our preliminary tests on parameter sensitivity. Other approaches are not parameter sensitive. The parameter sensitivity's test showed that the OICF and AMCF performance are very sensitive to their parameters and the best value for the parameters is subject to the problem size [6].

Table 3 shows the experimental results of the average of ten runs on datasets N80K20_A and N240K40_F with each run being given one hour of computation time as a termination criterion. An ANOVA test on the results of dataset N80K20_A and N240K40_F (one hour F runtimes) showed that the CT's averages of all approaches tested in this section are statistically different (at the 99% confidence level with F-ratio=47.87 and 33.27, respectively). Therefore, by just looking at the CT's averages, we can briefly estimate the effectiveness of the approaches (shown in table 4). Based on the results in table 4, we can observe that *VNMS* and *AMCF-sdEmc3opt* show the best performance on dataset N80K20_A and N240K40_F, respectively). Interestingly, *VNMS* and *AMCF-sdEmc3opt* have obtained the best performance of the two heuristics ranked in this experiment (on dataset N80K20_A and N240K40_F). However, based on an ANOVA test carried out on the results of *EMCQ-2opt, AMCF-sdEmc3opt* and *VNMS* (on the results of dataset N80K20_A for one hour runtimes) there is no significant difference in their performance (F-ratio=0.39 for α=0.01) even though they used different local searchers.

Fig. 3 shows a box-and-whisker plot, which represents the results of ten runs (minimum, first quartile, median, third quartile and maximum values of *CT*) on dataset N80K20_A (one hour runtime).

Table 3: An average result of ten runs on each data set (test duration: 1 hour).

Heuristic	Data set N80K20_A (CTo=1061.04)				Data set N240K40_F (CTo=3238.90)			
	CT_{avg}	I(%)	Dev	CT_{min}	CTavg	I(%)	Dev	CT_{min}
VNMS	257.04	75.77	2.49	252.86	2052.12	75.53	0.00	2052.12
VNS-1	311.61	70.63	9.63	300.10	2590.57	69.11	82.64	2477.19
VNS-2	298.65	71.85	10.18	286.40	2617.47	68.79	75.06	2482.07
VNS-3	275.74	74.01	7.53	263.19	2592.18	69.09	63.18	2480.02
AM-sdEmc3opt	272.58	74.31	4.78	264.10	2086.48	75.12	33.16	2045.16
OI-sdEmc3opt	278.26	73.77	9.06	262.15	2063.53	75.39	29.93	2031.71
AMCF-sdEmc3opt	258.63	75.62	6.28	251.85	2047.13	75.59	37.48	1970.74
OICF-sdEmc3opt	273.97	74.18	6.36	264.96	2070.48	75.00	37.84	2026.71
EMCQ-sdEmc3opt	274.47	74.13	9.48	259.00	2063.70	75.39	26.02	2007.75
AM-sd	276.98	73.90	12.19	256.68	2053.88	75.51	22.63	2008.94
EMCQ-2opt	258.77	75.61	4.98	252.14	2264.44	73.00	50.43	2213.25
OICF-2opt	325.42	69.33	18.92	296.95	2781.95	66.83	512.33	2151.95
OI-2opt	279.03	73.70	8.09	264.50	2369.73	71.74	75.24	2276.86

Note:

CT_{avg}: Average assembly cycle time(unit time); *CT_{min} : Minimum assembly cycle time(unit time);*
Dev : Standard deviation of CT; *CT_o : Initial CT;*
I : CT's Improvement=$(CT_o-CT_{avg})*100/ CT_o$

Table 4: A heuristic ranking based on CT_{avg} of ten runs starting with the most effective heuristic (smallest CT_{avg}) (test duratio:one hour).

	Dataset N80K20_A		Dataset N240K40_F	
	Heuristic rank	CTavg	Heuristic rank	CTavg
1	VNMS	257.04	AMCF-sdEmc3opt	2047.13
2	AMCF-sdEmc3opt	258.63	VNMS	2052.12
3	EMCQ-2opt	258.77	AM-sd	2053.88
4	AM-sdEmc3opt	272.58	OI-sdEmc3opt	2063.53
5	OICF-sdEmc3opt	273.97	EMCQ-sdEmc3opt	2063.70
6	EMCQ-sdEmc3opt	274.47	OICF-sdEmc3opt	2070.48
7	VNS-3	275.74	AM-sdEmc3opt	2086.48
8	AM-sd	276.98	EMCQ-2opt	2264.44
9	OI-sdEmc3opt	278.26	OI-2opt	2369.73
10	OI-2opt	279.03	VNS-1	2590.57
11	VNS-2	298.65	VNS-3	2592.18
12	VNS-1	311.61	VNS-2	2617.47
13	OICF-2opt	325.42	OICF-2opt	2781.95

By referring to fig. 3, we can observe that the *VNMS* slightly outperformed the other approaches with the smallest median and variation of the *CT* values. Fig. 3 also shows that the *AMCF-sdEmc3opt* and *EMCQ-2opt* have fairly equal performance; and the *AM-sdEmc3opt, OICF-sdEmc3opt, EMCQ-sdEmc3opt, VNS-3, AM-sd, OI-sdEmc3opt* and *OI-2opt* also have fairly equal performance. These results also show that our basic *VNS* approaches are not performing well compared to the other approaches across all the datasets tested in this work. This may due to the local searches used in the basic *VNS* and the basic *VNS* approaches itself. Generally, the quality of the obtained solution by basic *VNS* is dependent on the local search used (as argued by Hansen & Mladenović [19]). Unfortunately, the nature of our problem presents difficulties when applying other established local search such as tabu search and simulated annealing since this scheduling problem involves some interrelated sub problems, which should not be solved independently. There is also the issue of how far to solve each of the sub problems before solving the others and in which order should we solve the sub problem. Optimising one factor (sub problem) may increase the cost of another factor(s). However, we can conclude that the *VNS* may perform better by integrating an acceptance criterion at the *VNS*

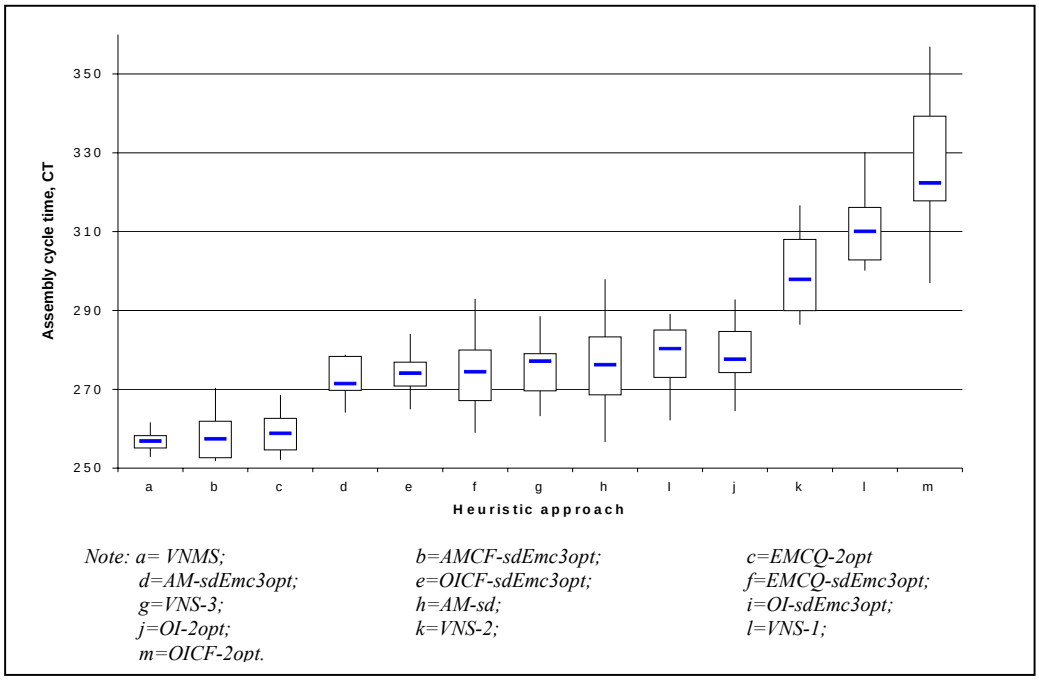

Fig. 3 A comparison of VNMS, VNS and hyper-heuristic approaches using box-and-whisker plot on the results of ten runs on dataset N80K20_A (test duration: 1 hour).

level. In addition, the strategy to initially explore the best neighbour in each neighbourhood structure before applying a random element and shaking procedures (which will only be applied after the previous local search group cannot find an improved solution) might be effective. The drawback is that the steepest descent local search is very time consuming.

8. Conclusions

A Variable Neighbourhood Search for solving the component pick-and-place sequencing of a multi-head SMD placement machine has been proposed. A basic version of a *VNS*, that is *VNS-1, VNS-2* and *VNS-3* have been presented. *VNS-1* is a descent heuristic that operates with random descent local searchers. *VNS-2* is also a descent heuristic but operates with *EMC* (Exponential Monte Carlo) local searchers. The *EMC* local search always accepts an improved solution and probabilistically accepts a worse solution depending on the degree of worsening (δ). The *VNS-3* is a descent-ascent heuristic with *EMCQ* (Exponential Monte Carlo with counter) acceptance criterion that operates with *EMC* local searchers. Our experimental results show that the *VNS-1, VNS-2* and *VNS-3* do not perform well compared to some hyper-heuristics approaches presented in this work. However, *VNS-3* is superior to *VNS-1* and *VNS-2*, which might indicate that the use of an acceptance criterion at the local search and VNS level can guide the search in order to obtain better solutions.

We further developed another Variable Neighbourhood Search with an Exponential Monte Carlo (*VNMS*) acceptance criterion. The *VNMS* is a descent-ascent heuristic that operates on three sets of neighbourhood structures that are grouped together based on three different local search/operator approaches. Each group contains six neighbourhood structures. The first two sets use a steepest descent and *EMC* local search whilst the third set uses a random 3-opt operator. The solution returned by a local search, after exploring a neighbourhood structure, will be accepted based on the *EMCQ* acceptance criterion at the VNS level. The *EMCQ* acceptance criterion always accepts an improved solution. However, the probability of accepting a worse solution increases as δ decreases and the counter of consecutive none improvement iterations, Q, increases.

Results show that the VNMS is capable of producing good quality and stable results (for smaller dataset) in solving the component pick-and-place sequence of multi-head SMD placement machine. Therefore, the VNMS is more reliable compared to the hyper-heuristic approaches tested in this work. The proposed framework of VNMS might be suitable for solving other types of SMD placement machines or even other problem domains. However, the local searchers are problem specific, which often has to be the case.

References

[1] M. Ayob and G. Kendall, "A survey of surface mount device placement machine optimisation: machine classification." European Journal of Operational Research, in press, 2007.

[2] K. Altinkemer, B. Kazaz, M. Koksalan and H. Moskowitz, "Optimization of printed circuit board manufacturing: Integrated modeling and algorithms. "European Journal of Operational Research, 124, 409-421, 2000.

[3] E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross and S. Schulenburg. "Hyper-Heuristics: An emerging direction in modern search technology," ch. 16. In: Glover, F. & Kochenberger, G., Handbook of Meta-Heuristics , 457-474, Kluwer, 2003.

[4] P. Hansen and N. Mladenović. "Variable neighborhood search*." European Journal of Operational Research,* 130, 449-467, 2001.

[5] M. Ayob and G. Kendall, "An investigation of an adaptive scheduling for multi headed placement machines." Proc. of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA 2003 (363-380). Nottingham, UK, 2003.

[6] M. Ayob and G. Kendall. "A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine." Proc. of the International Conference on Intelligent Technologies, InTech'03 (pp. 132-141). Chiang Mai, Thailand, 2003.

[7] M. Ayob and G. Kendall, "Real-time scheduling for multi headed placement machine." Proc. of the 5th IEEE International Symposium on Assembly and Task Planning, ISATP'03 (pp. 128-133). Besançom, France, 2003.

[8] M. Ayob and G. Kendall, "A triple objective function with a chebychev dynamic point specification approach to optimise the SMD placement machine." *European Journal of Operational Research*, 164, 609-626, 2005.

[9] M.R. Garey and D.S. Johnson, "Computers and intractability-a guide to the theory of NP-completeness." Freeman, 1979.

[10] L.K. Moyer and S.M. Gupta, "Simultaneous component sequencing and feeder assignment for high speed chip shooter machines." Journal of Electronics Manufacturing, 6, 271-305, 1996.

[11] R. De Souza and W. Lijun, « Intelligent optimization of component insertion in multi-head concurrent operation PCBA machines." Journal of Intelligent Manufacturing, 6, 235-243, 1995.

[12] Aarts, E. & Lenstra, J.K. (2003). *Local search in combinatorial optimization*. Princeton University Press, 2003.

[13] Y. Crama, J.van de Klundert and F.C.R. Spieksma, "Production planning problems in printed circuit board assembly." Discrete Applied Mathematics, 123 , 339-361, 2002.

[14] M. Grunow, H-O. Günther, M. Schleusener and I.O. Yilmaz, „Operations planning for collect-and-place machines in PCB assembly." Computers & Industrial Engineering, 47(4), 409-429, 2004.

[15] W. Ho and P. Ji, "A genetic algorithm approach to optimising component placement and retrieval sequence for chip shooter machines." Int. J. of Adv. Manufacturing Technology, 28, 556-560, 2006.

[16] W. Ho and P. Ji, "Component scheduling for chip shooter machines: a hybrid genetic algorithm approach, *Computers and operations research*, 30, 2175-2189, 2003.

[17] R. Kumar and Z. Luo, "Optimizing the operation sequence of a chip placement machine using TSP model." IEEE Transactions on Electronics Packaging Manufacturing, 26(1), 14-21, 2003.

[18] P. Hansen and N. Mladenović, "Variable neighborhood search." European Journal of Operational Research, 130, 449-467, 2001.

[19] P. Hansen and N. Mladenović, "Variable neighborhood search for the P-median. Location Science, 5(4), 207-226, 1997.

[20] C. Avanthay, A. Hertz and N. Zufferey, "A variable neighborhood search for graph coloring." European Journal of Operational Research, 151, 379-388, 2003.

[21] A. Hertz and D. de Werra, "Using tabu search techniques for graph coloring." Computing, 39, 345-351, 1987.

[22] P. Galinier and J.-K. Hao, "Hybrid evolutionary algorithms for graph coloring." Journal of Combinatorial Optimization, 3, 379-397, 1999.

[23] K. Fleszar and K.H. Hindi, "Solving the resource-constrained project scheduling problem by a variable neighbourhood search." European Journal of Operational Research, 155(2), 402-413. 2004.

[24] P. Cowling, G. Kendall and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit." In: Burke, E. and Erben, W., editors, Selected Papers Of The Third International Conference On The Practice And Theory Of Automated Timetabling, PATAT'2000,176-190. LNCS, 2001.