Math. Prog. Comp. (2016) 8:1–45
DOI 10.1007/s12532-015-0095-1

FULL LENGTH PAPER

A new novel local search integer-programming-based heuristic for PCB assembly on collect-and-place machines

Anupam Seth[1] · **Diego Klabjan**[2] · **Placid M. Ferreira**[1]

Received: 11 April 2011 / Accepted: 24 September 2015 / Published online: 2 November 2015

Abstract This paper presents the development of a novel vehicle-routing-based algorithm for optimizing component pick-up and placement on a collect-and-place type machine in printed circuit board manufacturing. We present a two-phase heuristic that produces solutions of remarkable quality with respect to other known approaches in a reasonable amount of computational time. In the first phase, a construction procedure is used combining greedy aspects and solutions to subproblems modeled as a generalized traveling salesman problem and quadratic assignment problem. In the second phase, this initial solution is refined through an iterative framework requiring an integer programming step. A detailed description of the heuristic is provided and extensive computational results are presented.

Keywords Heuristic · Local search · Integer programming · Printed circuit board · Vehicle-routing · Generalized TSP

Mathematics Subject Classification 90B06 (Primary) · 68T20 · 68T40 · 68W25 · 90-08 · 90B10 · 90C06 · 90C10 · 90C27 · 90C35 · 90C59 · 90C90 · 05C85 · 05C90

1 Introduction

Many manufacturing enterprises in today's world are under severe economic pressure to find ways to satisfy increasingly demanding customers who are seeking higher and

✉ Diego Klabjan
d-klabjan@northwestern.edu

[1] Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA

[2] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA

higher levels of performance on the two conflicting dimensions of cost and time to delivery. These pressures are mounting for organizations whose production processes require expensive, complex automation, and with a broad assortment of products. Manufacturers operating in such conditions must find ways of planning their production, which simultaneously enable both high resource utilization and a quick response to changing demands both in terms of short-term product mix variations and longer-term product range changes as new products are introduced and old products become obsolete.

The electronics industry continues to rank as a very valuable and key industry in this information age at the turn of the century and beyond. Also, most electronic products manufactured today contain printed circuit boards (PCBs) as critical elements [12]. Therefore, PCB manufacturing plays a very important role in today's economy. Global revenues for the PCB industry exceeded $50 billion in 2007 and are expected to reach more than $76 billion in 2012 [35].

PCBs are manufactured in automated assembly lines, where high-speed placement machines place components on the boards. A line can assemble components on multiple types of PCBs and has one or several high speed machines to perform the actual placement of operations [36]. The assembly of PCBs is a complex task involving the placement of hundreds (even up to a few thousand) of electronic components in different shapes and sizes at specific locations on a board.

Because electronics technology can quickly become obsolete, minimizing the time of PCB design and manufacture is crucial and an increasingly significant concern for electronics firms [21]. In order to remain competitive in the PCB market, manufacturers must concentrate their efforts on improving the efficiency of their (now dominantly SMT, or surface-mount technology, as opposed to the older dominantly THT, or through-hole technology) assembly lines. Production planning and control, process planning, and quality control are important activities for achieving this efficiency in the PCB industry. Of these activities, process planning is particularly important due to its direct impact on efficient and responsive PCB assembly operations. With the increasing market pressures, it has become imperative for industries to critically look at novel ways to reduce assembly time, which is one of the largest chunks in the makespan of PCBs, almost to the tune of as much as 40–60 %.

A critical process in electronic manufacturing is the placement of electronic components onto the PCBs using high-precision automated machinery [23]. Not only are these machines frequent bottlenecks in the production, but they are also expensive resources (a typical SMT equipment ranges in price from $250,000 to $1,000,000 [12]). Utilization of these expensive assembly machines is, therefore, an important issue in PCB assembly [47], especially for industrial applications. Modern PCB production, therefore, utilizes computerized machines and highly automated equipment. Efficiency relies on technical capabilities of the placement machines and the underlying workflow. Since further improvements in hardware are both limited and expensive, to improve the performance of the chip placers, operations on these machines need to be scheduled as effectively as possible. Also, as equipment becomes more productive, flexible, and automated (and also more expensive), efficient planning and organization are required to increase production throughput close to the capacity of the equipment [32].

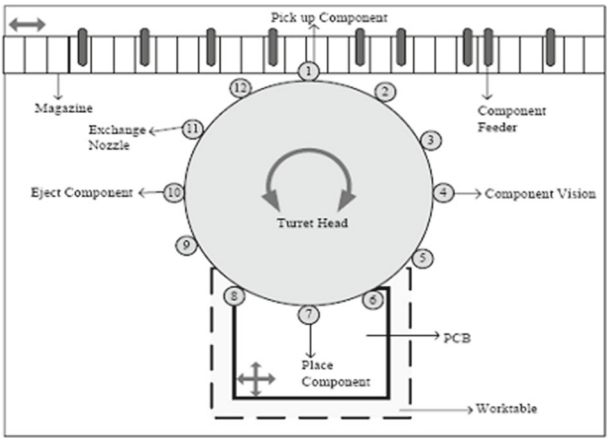

Fig. 1 The chip shooter machine

Today, PCB assembly is carried out by three major types of automated placement equipment, namely, the chip shooter or turret-type placement machines (illustrated in Fig. 1), the pick-and-place-type machines with a single or multiple heads, and the newer rotary collect-and-place-type machines (CAPMs).

Chip shooters have been around for decades and were known for their (once-upon-a-time) high speed. They have low flexibility in terms of handling varieties of components, but were very good at repetitive tasks. They are now almost phased out. Pick-and-place machines are typically slower and, being highly flexible, are used more for placing odd and specialized components. They may have either a single spindle or multiple spindles (see Figs. 2, 3).

CAPMs (shown in Fig. 4) are being increasingly used for meeting the high-volume, high-flexibility production requirements of today. They are becoming increasingly popular in use in industry in the last few years. Research literature is abundant for the chip shooter machines, and also to some degree for the pick-and-place machines, but happens to be extremely limited and scant for the CAPM configuration, inspite of their increasing popularity. Solutions to planning problems arising from the mechanisms involved in such an equipment are provided so far by crude heuristics both in published literature and in shop floor practices. It is noteworthy that this is inspite of the fact that the CAPMs form the bottleneck in high-volume PCB production. Being an extremely versatile equipment often used for high- or very-high-volume production, they are a key resource whose utilization plays a critical role for success as even marginal savings in time can translate to substantial monetary savings.

According to [36], planning problems for these machines are interesting for (at least) two reasons. First, the competitiveness of individual companies in the field directly depends on the cost efficiency of the production, which in turn depends on the throughput of individual assembly lines. Second, the problems themselves are quite challenging due to their inherent difficulty and the fact that they offer the possibility of using several different approaches for formulation and solution.

The placement sequencing problem is a complex scheduling problem with scenario-specific constraints that can blow up in size and complexity for even relatively small

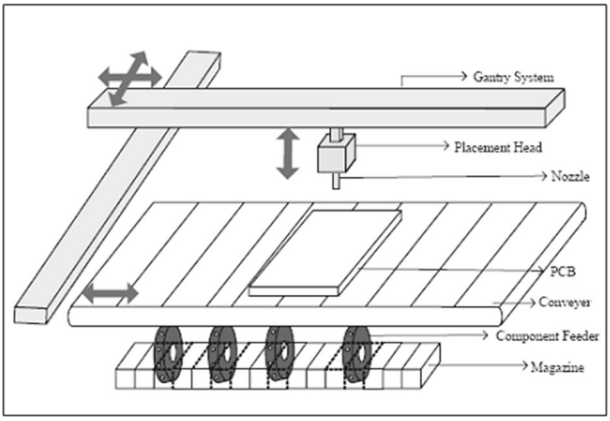

Fig. 2 The single-spindle pick-and-place machine

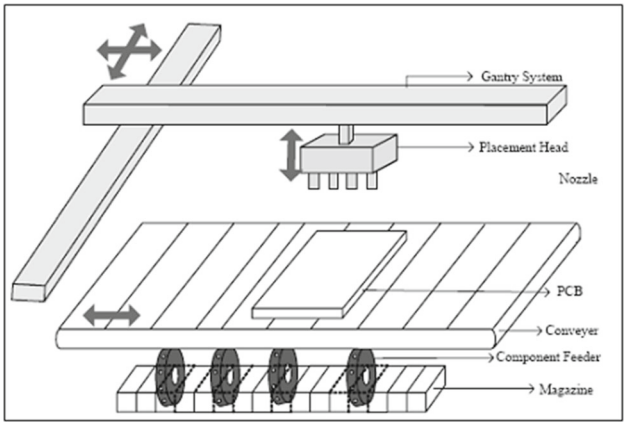

Fig. 3 The multiple-spindle pick-and-place machine

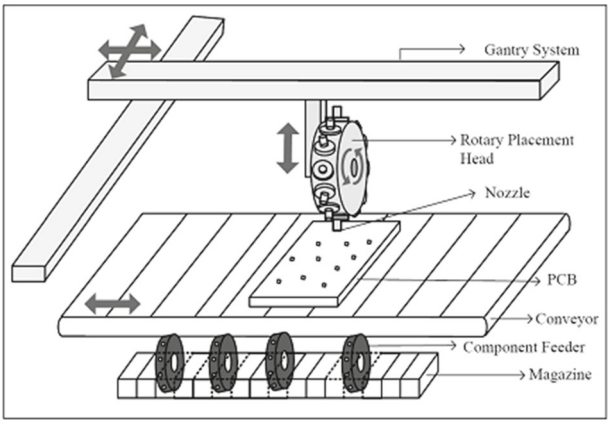

Fig. 4 The collect-and-place machine

cases. Current methodologies are not only limited in terms of the quality of solutions they provide, but also in the size of problems that they can tackle. This is important industry-wide, because not only there are industries (e.g. avionics) that use such large-sized boards (24 in. $\times$ 20 in. boards with several hundreds of components), but also because most manufacturers of small, tiny boards (e.g. in the cellular phone industry) tend to produce their boards by batching or "bundling up" several (tens of) their boards together onto a platter that is later cut out into individual boards. This is to form a virtual large board for the sake of efficient processing.

The CAPM, specifically, offers some very interesting problems and opportunities for application of vehicle routing techniques and algorithms that we uniquely exploit. This cross-domain application has not been explored before. We look at modeling, characterizing, and planning for this machine, which offers enormous automation flexibility due to its modern and highly advanced construction, yet being so, providing greater challenges for efficient planning, routing and scheduling than ever before. In a certain sense, it is like "taming the monster" because even though the machine has been created with ultra-flexible automation capabilities, it is almost impossible to plan for efficient utilization of those capabilities with even the best of modern day optimization, planning tools, and common computational resources available to the industrial users. These factors motivated us to study and present a practical and efficient methodology to tackle optimization of the placement sequence plan for a CAPM.

1.1 A PCB primer

A PCB is a flat board that carries chips and various other electronic components. The board is made up of alternating layers of copper and plastic, with the etching process performed on the copper layers to provide interconnects. These boards are capable of holding several components depending on the required specifications and produce complex interconnections that are of different sizes and varying densities.

Typical end consumers are the computer, communications, automotive, consumer electronics, aerospace and defence industries. In the computer industry, for example, PCBs are used in flat panel displays, ink-jet printers and disc drives. Other applications are RFID systems, engine controls in automobiles and any hand held devices such as personal digital assistants, mobile phones or GPSs.

The injection of boards with chips is executed by placement machines. In addition to a high placement rate and flexibility in terms of handling odd components, high placement accuracy is required of these machines. The principal task of a placement machine is to pick a component from the feeder magazine/tray and place it on the right location on the board. Thereby placement machines can be classified into two groups, namely the pick-and-place and CAPMs; and the turret-styled placement machines. The difference between these two is due to the moving parts. Pick-and-place or CAPMs have a moving head while the circuit board is fixed, and the feeder may move linearly or is fixed. In contrast, a chip shooter machine operates with a rotating turret, which connects through its cruising radius the feeder with the table. Both, the feeder and the table, are moving, too.

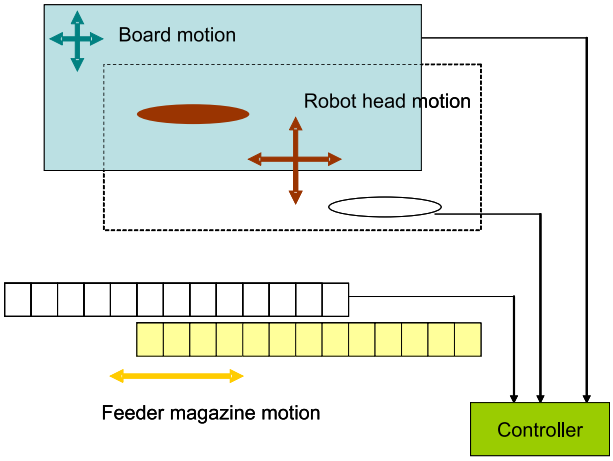

Fig. 5 The basic robot assembly system [41]

The elementary machine is a pick-and-place machine with one head equipped with a single spindle. In every sequence a chip is picked up by the spindle from a feeder tray with multiple slots (each holding a specific type of chip/component), is transported to the board and placed on its intended location (see Fig. 5). Nowadays, the head typically has multiple spindles used to pick-up and place the components. These spindles are loaded with vacuum-suction operated nozzles capable of holding a single component at a time (see Fig. 8). Furthermore, because of the wide range of capabilities of these machines, it makes sense to build an assembly line in which machines complement each other.

Despite such overall complexity, it is still instructive to study the single-board single-machine scenario, as it is done here, since it provides an insight into a subproblem that is at the core of the planning hierarchy (classified into seven sub-problems by Crama et. al. [10]). The single-board single-machine problem in itself involves several factors, which play into the computation and optimization of the process plan, namely:

- robot motion control,
- assembly placement sequence,
- feeder magazine arrangement,
- nozzle set-up and inter-changeability.

1.2 Contributions and scope

The main goal of this study is to develop an algorithm to solve the placement sequence planning problem for the CAPM automated placement equipment. The objective was to develop an algorithm that:

1. uses an underlying model that overcomes some of the limiting and rigid modeling assumptions made in most previous studies (for example, assuming that each component type is carried in a separate tour, fixed feeders, forced return to a fixed "park" location between tours, and single-stepped uni-directional head rotations),

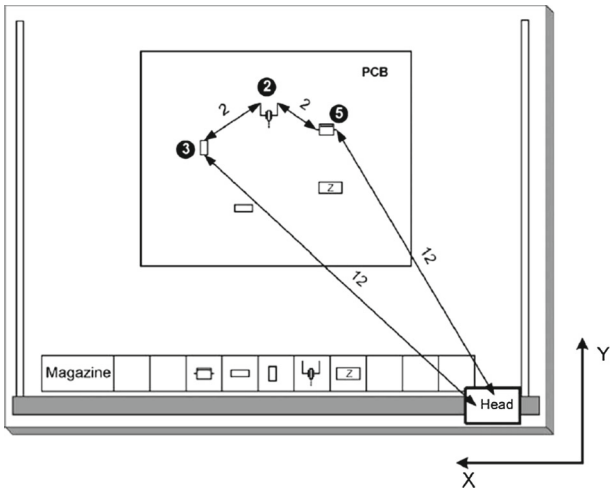

Fig. 6 Schematic of collect-and-place machine [19]

2. is capable of computationally handling the problems arising in the cases of large-scale instances of boards occurring often in industry,
3. uses sophisticated modeling and computational techniques rather than simplistic and approximate ones,
4. is robust in its performance to variations in input parameters and operating conditions such as board designs, machine parameters, set-ups, etc.,
5. and, presents an opportunity for extension and transition to other machine configurations and operating conditions that operate on a TSP or VRP-style routing.

In this paper, we present:

- a mathematical model of the underlying planning problem,
- an underlying framework of a two-phase algorithm,
- a novel vehicle-routing-based solution methodology and its application in the design of our algorithm,
- detailed steps of the execution and flow of our algorithm,
- a successful column generation strategy to manage the enormous number of columns in the model (it is applied in each iteration of the algorithm), and
- a description of the successful implementation, testing, and presentation of computational results.

We limit the problem scope to studying a single-head single board scenario while there now exist machines that can synchronize up to four placement heads (see Fig. 8) simultaneously operating on two boards (for example, the Siemens' SIPLACE HS—see Fig. 7 or Universal Instruments' Quadris S). If several boards or heads are present, the problem can be decomposed, albeit losing optimality, into several single head single board cases.

We study the problem of optimizing the placement sequence of components in a CAPM (see Figs. 6, 7, 8). We allow and exploit more automation flexibility than has ever been explored earlier. We allow interleaving of pick-ups and placements and

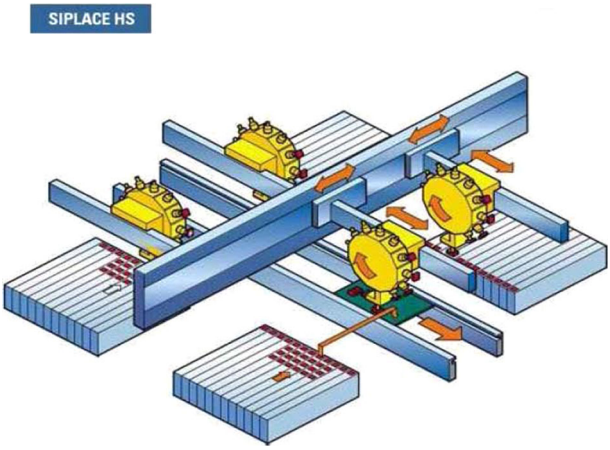

Fig. 7 The Siemens' HS machine dual gantry quad-head configuration [39]

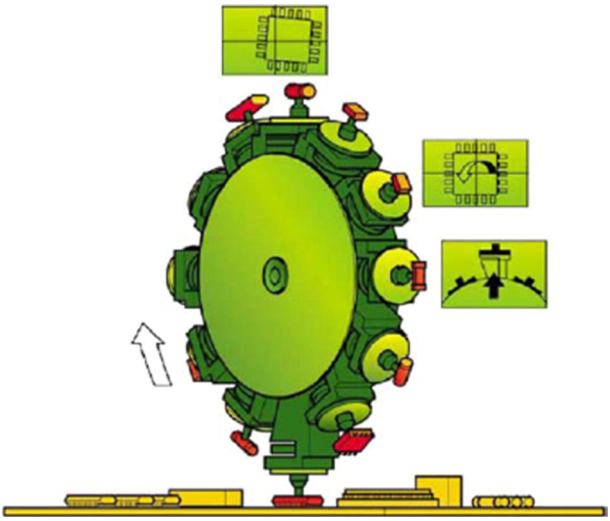

Fig. 8 Detailed view of a rotary collect-and-place head [39]

multi-stepped bidirectional head rotation (this means that we do not have to follow the sequential ordering of spindles to use in successive placements, but can use spindle sequences such as $\{1, 3, 5, 2, 8, \ldots\}$ consecutively in a given route in order to optimize the linear distances to be traveled by the rotary head, see Fig. 8 for an illustration of the head loaded with components on multiple spindles). It allows us to get a solution that truly utilizes the flexibility provided by the system design. More importantly, we also incorporate the effects of allowing the entire feeder magazine to move along the horizontal direction. This allows for greater efficiencies to be achieved, but also causes the problem to become much more complicated. This is the first time this aspect is ever included in the analysis of a modern SMT placer such as a CAPM.

Given these assumptions and specific constraints posed by the scoped configuration of the machine, the placement sequencing problem becomes very similar to a complex, yet striking version of the vehicle routing problem with a single capacitated vehicle (in our case, the rotational robotic head with multiple spindles loaded onto a dual XY gantry) performing multiple trips from the depot (in our case, the feeder magazine) to satisfy customer demand (in our case, the design requirements for the board being populated, stipulating the specific component type to be inserted at each placement location, see Fig. 6).

We tackle this challenging problem by a two phase optimization approach. In the first phase we construct a feasible solution by using greedy principles combined with mathematical programming. In [38], we even provide a worst-case analysis of a single component-type variant.

In the second phase we iteratively improve this solution. In each iteration, we first randomly extract board locations and the corresponding feeder actions, then we solve an integer program (IP) that finds optimal re-insertion points, and finally based on this solution the extracted objects are inserted back to get a new (possibly lower quality) solution. Due to the large number of columns in the insertion IP, it is solved by price-and-branch (where the root node LP relaxation is solved by dynamically generating columns, called also column generation, selecting a subset of columns generated, and applying branch-and-bound only over the selected columns).

The main contributions of this work are:

- a model for CAPM capturing feeder motion, bi-directional and multi-stepped head rotations, and partially loaded heads (interleaving of pick-up and delivery points) allowing optimization as never achieved before up until now,
- a VRP-like local search and mathematical programming heuristic for solving the underlying model, and
- the price-and-branch component of the algorithm.

The solution of this problem holds great promise not only for board assemblers and end-product manufacturers of electronic equipment in reducing cycle assembly time but also is of great value to assembly machine and equipment manufacturers in answering design questions such as:

- Should feeder movement capability be added or should the head index time be reduced?
- Should head capacity be increased or should the XY traverse speed be increased?
- Is it more helpful to have a smaller, faster head vs. a larger, slower head?

One of the interesting features of this problem is the ripe opportunity for extending and expanding the problem scope to make the models and heuristics more and more applicable to real-world scenarios. For instance, going from one to a dual head configuration would introduce problems of load balancing, cycle sequence synchronization, and the geometrically challenging problem of collision avoidance.

In Sect. 2, we survey related work both from the PCB and VRP side. Section 3 formalizes the problem statement and presents the network formulation underlying our solution methodology, which is presented in detail in Sect. 4. Section 5 provides detailed computational results in terms of quality, sensitivity, and scalability of the designed algorithm, and presents a regression model to estimate the performance.

2 Literature review

Given that the problem of picking components from the feeder and placing them onto the board resembles the vehicle routing problem, we first examine some very well-known variants of the VRP and position our work in the context of VRP.

While the problem is essentially similar in nature to the vehicle routing problem, there are a number of interesting differences and anomalies that qualify it to be studied exhaustively in its own right. First of all, the specific construction and operating characteristics of these machines based on the principles of exploiting simultaneous motion along multiple axes call for the formulation and use of the Chebychev distance metric, i.e. the l_∞ norm. This is not common in the vehicle routing literature and many routing algorithms exploit the much more commonly assumed and used Euclidean distance metric. Another difference is that there is a sequencing cost to loading and unloading the vehicle. While some of the vehicle routing literature raises and addresses these concerns, particularly in the context of trucks used both for back-hauling or mixed pick-up and delivery situations, and refrigerated product delivery, it is modeled with additional constraints rather than with a step cost function as is the case with our machine configuration and model. The problem is also interesting because it is a multi-depot problem, but differs in that the depots can 'service' only select customers.

The problem also incorportates the multi-commodity aspect into the VRP setting, which is not very common in published VRP literature. It also simultaneously includes the multiple moving depots aspect. There are multiple possible pick-up and delivery sites for each component that is placed by the head and there can be several feeder slots storing the same component type. There also can be multiple locations on the board requiring the same component type. Our problem also uses multiple routes of a single vehicle, an aspect not prominent in existing literature (most existing VRP research deals with the single-vehicle, single route problem or the multiple-vehicle, multiple route problem). The problem also features the presence of very interesting "free zones," which offer further opportunity for exploiting simultaneous motion and influence the optimal routing. "Free zones" are caused by the minimum rotational indexing time of the placement head causing idle time for the machine over short XY motion spans while picking and placing components. Thus, one could profitably place a farther-off component taking advantage of the idle time to reduce the bottleneck operation time someplace else.

To summarize, the problem turns out to be more involved, complicated, and constrained than the individual standard capacitated vehicle routing problems such as the capacitated VRP, the VRP with backhauls, the multi-depot VRP, the VRP with pick-up and delivery, and the dial-a-ride-problem. For further details, the reader is referred to two comprehensive collections of papers on the VRP and its variants by Toth and Vigo [43] and Golden et al. [18]. Our problem involves a combination of characteristics found in these variants along with some other oddities not found in any of them. Thus, what is required in the present context is a superset of several standard VRP variants.

Recently, Franceschi et al. [17] carried out some remarkable work on developing an integer-linear programming (ILP) based refinement heuristic for the distance-constrained VRP, extending the work of Sarvanov and Doroshoko [37]. Their algorithm extracts a certain number of nodes from an initial starting solution, generates large

number of new sequences through the extracted nodes, and solves a reallocation ILP model to decide reinsertion points for them. They use a generic ILP solver and attain excellent results improving upon best-known objectives from literature for several unsolved VRP instances and solving to optimality other known difficult instances. We extend their work to the more general and complicated VRP arising in PCB setting, and also introduce a column generation approach to handle the reinsertion ILP.

Next, we examine literature in the PCB manufacturing planning domain. Excellent surveys on production planning problems in PCB assembly are given in Crama et al. [10], Smed et al. [40] and Ji and Wan [27]. These reviews establish a framework for the general categorization and classification of an exhaustive list of production planning problems arising in electronics assembly. They also classify research according to the problems addressed for different machine configurations. A very recent survey can be found in Ayob and Kendall [5], wherein the various machine configurations and important optimization algorithms to solve production problems on them are exhaustively reviewed.

While there is an abundance of literature on the turret-type placement machine (see for example, some excellent work carried out by Wilhelm et al. [46], Kumar et al. [31], and Ellis et al. [13]) and both the single-headed (Ho and Ji [24], Ayob and Kendall [4], and Ball and Magazine [6]) and multi-headed pick-and-place machine configurations (Burke et al. [8], Ahmadi et al. [1], and Wilhelm et al. [44,45]), there seems to be a definitive lack of research on the CAPM concept. Part of this might be due to the fact that this machine configuration is fairly recent, being introduced to the commercial market by machine manufacturers only a few years ago as opposed to the decades-long history of usage of the other machine types. The steadily increasing popularity of this machine type led by its unique high-volume, high-flexibility production capabilities in the present highly competitive environment for electronics assembly demands rigorous modeling, analysis, and computational studies on the specific problems arising in planning for production on machines of this configuration.

In relation to the CAPM, Altinkemer et al. [2,28] have solved the integrated feeder location and placement sequencing problem for a single rotary head machine both in the case of a moving and a non-moving feeder. They solve the integrated problem by decomposing it into two sub-problems using Lagrangian relaxation. They also prove an error guarantee of ϵ given an ϵ-approximation algorithm for capacitated VRP. They assume a single feeder location per component type, linear distances, a separate tour for every component type, and that the head returns to the original feeder location of the component after each tour. These assumptions are seriously limiting the capabilities of CAPM. In [28], they solve the same problem, but allowing multiple feeder locations per component type. They use Lagrangian relaxation and a sub-gradient optimization procedure to find an optimal solution given an optimal solution to the multi-depot VRP and prove an ϵ-error guarantee given an ϵ-approximation algorithm for the multi-depot VRP. They, however, continue to assume that the head returns to the original feeder location of the component after each tour and linear distances.

Günther et al. [19] solve the integrated problem of feeder location and placement sequencing using a 3-stage heuristic. The construction heuristic is an adaptation of the Clark and Wright savings heuristic [9] and is supplemented by the use of local search techniques for solution improvement. They relax the assumption of a separate tour

for each component type and achieve solutions within 2–14 % of the kinematically-derived lower bound. They assume no nozzle constraints or nozzle selection problem, a single feeder location per component type, and the head fully loaded except possibly in last tour, placement sequence identical to the pick-up sequence, uni-directional head rotations, and a linear distance function. They extended their work in [30] by using GAs to solve the same problem. They also suggested ways to extend their algorithm to dual-gantry systems. However, their assumptions of uni-directional head rotations and a fixed location to which the head always returns to park between tours remain intact.

Ho et al. [25] have also developed two hybrid GAs to solve the feeder assignment and placement sequencing problems on the CAPM. Their approach also suffers from the limitations of allowing only uni-directional head rotations and fully-loaded tours.

In general, purely evolutionary neighborhood-based approaches such as GAs suffer from the curse of dimensionality and can handle only small cases with a high solution quality when applied to problems of this kind [30]. Kulak et al. [30] make attempts to remedy this, but can still handle only boards of medium complexity and size (the largest board considered has 360 components as opposed to our approach that can easily scale to double that number).

Tirpak et al. [42] describe the development of an optimization software for the Fuji NP-132, a dual-station, dual revolver head, high-speed placement machine. They discuss feeder, nozzle, and placement optimization problems in terms of machine's degrees of freedom and physical constraints. As a solution methodology, adaptive simulated annealing is proposed using cheapest insertion and nearest neighbor path construction heuristics to generate placement sequences, and constraint satisfaction swapping heuristics for generating feeder and nozzle setups. They obtain 6 % improvements over "manually optimized schedules by factory engineers". The paper does not provide details of the mathematical model and the implemented algorithm.

Knuutila et al. [29] formalize only a small subproblem of the scheduling problem of multi-headed SMT machines, i.e., the selection of nozzles, which pick up and place components on PCBs. They aim to minimize the number of component pickups. Given a sequence of component placement commands, they show that a greedy nozzle usage policy can be optimal both in the case when the nozzles are universal, i.e., they can pick up any component, and in the case when only certain component types can be picked up only with certain nozzle types.

Li et al. [34] formulate and solve only the feeder assignment problem for the CAPM. They further make rather simplistic assumptions for the motion of the head. They use GAs to solve the feeder optimization problem and show that their algorithm performs better than the "conventional industry algorithm".

Gyorfi et al. [20] show that the GA by Leu et al. [33] for planning component placement sequences and feeder assignments for pick-and-place PCB assembly tasks is a special case of a more general model that supports multiple-placement nozzles and independent feeder and board link (chromosome) evaluation methods. They also show that independent link evaluation can be used to offset a reduction in the parent link sample space and that these results are better than what can be achieved through link-pair evaluation. These generalizations extend the capabilities of the GA to a broader range of manufacturing scenarios, but do not explicitly address specific concerns in the CAPM scenario.

Thus, it is clear that even in the few scant papers that have been published on the CAPM configuration, none of them addresses or studies the problem with as much rigor, or in as general of a form, or using sophisticated mathematical and computational techniques as we do.

This paper presents the overall methodology, network model and computational experiments. It also provides details regardng ILP and the pricing problem formulation. In [38] we present the theoretical analyses and proofs of the performance bounds of the initial solution generation heuristics.

3 Problem statement and modeling

This section outlines the formal statement of the exact problem. We have developed a network model to represent and solve the problem under consideration. The standard VRP variants that our problem mostly resembles are the static dial-a-ride problem (but, with seating constraints) or VRP with pick-up and delivery (but, where deliveries and pick-ups can be interspersed for optimal capacity utilization).

In this work, we optimize the placement sequence of components for a CAPM. We assume that the arrangement of the feeder is prescribed in a separate pre-optimization step. This is not an unreasonable assumption even though the optimal sequence is dependent on the feeder arrangement as an input since it is possible to arrive at an arrangement by iteratively solving the placement sequencing and feeder arrangement problems within a feedback loop. This has often been done in the literature for these kinds of problems.

We assume for the sake of simplicity that the feeder rack holds each component type in only one slot location. Thus, the component retrieval problem disappears. We also assume for the sake of simplicity that the acceleration and velocity profiles of the robotic head and feeder magazine remain independent of the components that they carry. These assumptions can be easily relaxed because they only impact the cost structure of the arcs. The high speed and precision at which these machines have to operate make it difficult to conceive solutions with multiple stopping points within their acceleration and deceleration ramps, thereby not impacting the number of nodes in the network.

3.1 Formal statement

We aim to find the least cost sequence of component pick-ups from the feeder magazine and then placement on the board subject to the following conditions:

(1) all components required by the design must be placed,
(2) every component that is placed must have been picked up previously (and not yet placed),
(3) every spindle may contain only one component during a "route", and
(4) at no time in a route must the capacity of the head be violated.

We want to fully exploit opportunities offered by simultaneous motion, spindle jumping (a phenomenon caused by allowing bidirectional and multi-stepped indexing

motion of the placement head), and feeder magazine movement. This can be restated in a more generic VRP-like terminology.

We assume

- there are n customers (i.e. board locations in our case),
- each requiring items of different types, i.e., each customer has a demand for a single item of a specific type,
- there are m depots (i.e. feeder slots in our case) to pick up the items from, each depot storing only one type of an item, implying that travel among depots is required, and
- each depot stocks an infinite supply of the kind of the item.

The objective is to find the least cost sequence of item pick-ups from the depots and delivering them to the "customers" subject to the following conditions:

- all items required by the customers must be delivered,
- every item delivered must have been picked up previously (and not yet delivered),
- the capacity limit of the vehicle must not be violated, and
- the restrictions and cost implications of the loading and unloading sequence for the vehicle are respected.

3.2 Network model

We formulate a network model $G(\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ the set of arcs in the model, in order to define the solution methodology. As supporting definitions, first, we define a board node (B_i) corresponding to each physical board insertion location and a feeder node (F_i) corresponding to each feeder slot location capable of holding a number of components of one particular type. From each B_i and F_i, we construct a series of nodes called the greater board nodes (G_j^B) and greater feeder nodes (G_j^F) respectively. The board nodes contain only the physical location and component type information, whereas, the greater board nodes contain a pointer to the board node and the active spindle position on the head at this node. Similarly, the feeder nodes contain the slot location and component type information, whereas the greater feeder nodes contain a pointer to the feeder node, the active spindle position on the head, and the feeder magazine's current position at the time of the head's visit to this node. To define it formally,

$$
\begin{aligned}
B_i &= \text{Board Location } i \text{ with its associated Component Type } j \\
F_i &= \text{Slot Location } i \text{ with its associated Component Type } j \\
G_i^B &= \text{Set of } k \quad \text{nodes for} \quad 0 \le k \le \text{maximum number of spindles} \\
&\qquad \text{with each node of the form } (B_u, \text{Spindle } k) \\
G_i^F &= \text{Set of } k \cdot l \text{ nodes for } 0 \le k \le \text{maximum number of spindles,} \\
&\qquad 0 \le l \le \text{maximum number of feeder magazine positions,} \\
&\qquad \text{with each node of the form } (F_v, \text{Spindle } k, \text{Feeder Magazine Position } l) \\
G^B &= \{G^B\}_i
\end{aligned}
$$

$$G^F = \{G^F\}_i$$
$$\mathcal{N} = G^B \cup G^F$$
$$\mathcal{A} = \mathcal{N} \times \mathcal{N}.$$

These principles are illustrated in Fig. 9. The figure shows an example of the network for a board with four component types at several locations (8 chosen for illustration). The machine has four spindles, a few possible feeder magazine positions (7 chosen for illustration), and four feeder slots each holding one of the four component types. Thus, there are 8 defined B nodes and 4 defined F nodes. We can see how each of the 4 F nodes is replicated 4 (no. of spindles) $\times$ 7 (no. of feeder positions) $=$ 28 times in the G^F network and each of the 8 B nodes replicated 4 (no. of spindles) times in the G^B network.

We next focus on arc costs. In PCB manufacturing, cost is driven by time, and thus all arc costs are measured in time units, i.e., the time to move between two nodes. To this end, let us define:

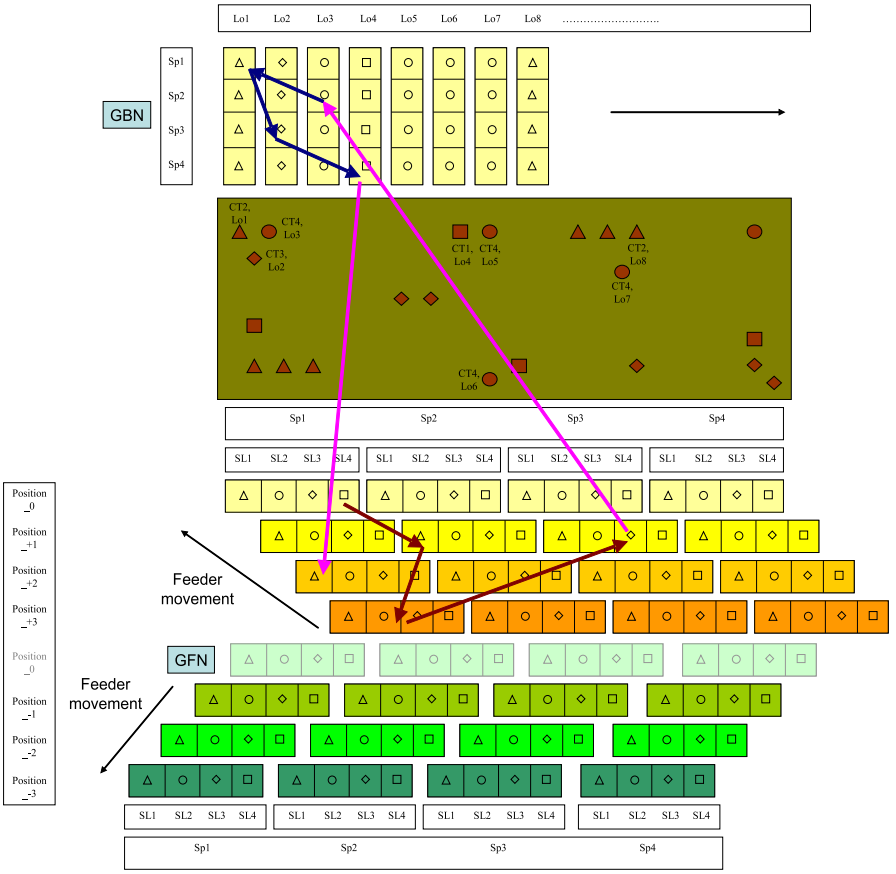

Fig. 9 Extended PCB network model

$$
\begin{aligned}
I_T \text{ (Index Time)} &= \text{Turret/Head rotational index time} \\
W_s \text{ (Slot Width)} &= \text{Width of one slot on the feeder} \\
V_{hx} &= \text{Head traverse velocity in the X direction} \\
V_{hy} &= \text{Head traverse velocity in the Y direction} \\
V_f &= \text{Feeder magazine movement velocity in X direction} \\
h(a) &= \text{Head node of arc } a \\
t(a) &= \text{Tail node of arc } a \\
\Delta_a^S &= \text{Smallest relative non-oriented displacement} \\
&\quad \text{between active spindle positions at nodes } h(a) \text{ and } t(a) \\
&= \min(|\text{Spindle } h(a) - \text{Spindle } t(a)|, \\
&\quad \text{Max No. Of Spindles} - |\text{Spindle } h(a) - \text{Spindle } t(a)|).
\end{aligned}
$$

By Spindle k for $k \in \mathcal{N}$, we denote the spindle position associated with node k. In defining Δ_a^S, we capture the number of positions on the head between two spindle indices. We note that to get from one position to the other one the head can turn either clockwise or in the other direction. For this reason there are two terms.

Given the above definitions, we can define the costs on the arcs in the network, where X_i, Y_i are the x-co-ordinate and y-co-ordinate of node i, respectively. Consider the expression

$$c(a) = \max(|X_{h(a)} - X_{t(a)}|/V_{hx}, |Y_{h(a)} - Y_{t(a)}|/V_{hy}, \Delta_a^S \cdot I_T).$$

We define

$$
\begin{aligned}
&\text{Type (I) arc } (t(a) \in G^B, h(a) \in G^B): \\
&\quad c_b(a) = c(a) \\
&\text{Type (II) arc } (t(a) \in G^F, h(a) \in G^B): \\
&\quad c_{fb}(a) = c(a) \\
&\text{Type (III) arc } (t(a) \in G^B, h(a) \in G^F): \\
&\quad c_{bf}(a) = c(a) \\
&\text{Type (IV) arc } (t(a) \in G^F, h(a) \in G^F): \\
&\quad c_f(a) = \max(|X_{h(a)} - X_{t(a)}|/V_{hx}, \\
&\qquad |\text{Feeder position } h(a) - \text{Feeder position } t(a)| \cdot W_s/V_f, \\
&\qquad \Delta_a^S \cdot I_T)
\end{aligned}
$$

In type (I), (II), (II) arcs, the first two terms capture the fact that the head can move simultaneously both horizontally and vertically. The third term captures the time to

move the head from one spindle to the final one. If this time is longer, the head waits at $h(a)$ until it rotates to the desired spindle.

In type (IV) arcs, the feeder moves concurrently with the head. The first and last terms capture the head operations, while the middle term times the feeder movement.

Using the above stated costs, the triangle inequality holds only for selected arcs, namely:

$$\begin{aligned}
(G_i^B,\ G_k^B) &\le (G_i^B,\ G_j^B) + (G_j^B,\ G_k^B)\\
(G_i^F,\ G_k^F) &\le (G_i^F,\ G_j^F) + (G_j^F,\ G_k^F)\\
(G_i^B,\ G_k^F) &\le (G_i^B,\ G_j^F) + (G_j^F,\ G_k^F)\\
(G_i^B,\ G_k^F) &\le (G_i^B,\ G_j^B) + (G_j^B,\ G_k^F)\\
(G_i^B,\ G_k^B) &\le (G_i^B,\ G_j^F) + (G_j^F, G_k^B).
\end{aligned}$$

On the other hand, due to the special structure of the network costs, the triangle inequality does not hold for arcs:

$$(G_i^F,\ G_k^F) \not\le (G_i^F,\ G_j^B) + (G_j^B,\ G_k^F).$$

We define a route to be the sequence of all arcs between two feeder nodes where the head touches down upon its return from the board, i.e. it starts capturing the head's motion as soon as it hits the feeder, continuing through its journey delivering components on the board, and up until it returns to the feeder to pick up the first component for its next tour out. In terms of the above definitions, a route is a list of arcs preceded by a type III arc, comprising of several type IV arcs, followed by a type II arc, several type I arcs, and ending again with a type III arc. A typical route is illustrated in Fig. 10.

Let $\mathcal{R}$ be the set of arcs contained in route r. Let a_1 be an arc in $\mathcal{R}$ such that $t(a_1) \in G^B$ and $h(a_1) \in G^F$ and a_2 be an arc in $\mathcal{R}$ such that $t(a_2) \in G^F$ and $h(a_2) \in G^B$. Let also a_3 be an arc such that $t(a_3)$ is the 'last' G^F node to be visited by r and $h(a_3)$ is the first G^F node to be visited by the 'next' route (see Fig. 10). Let us also define

$$\begin{aligned}
A &= c_{fb}(a_1) + \sum_{a\in\mathcal{R}:h(a)\in G^B,\ t(a)\in G^B} c_b(a) + c_{bf}(a_2)\\
B &= |X_{h(a_3)} - X_{t(a_3)}|/V_{hx}\\
C &= |\text{Feeder Position } h(a_3) - \text{Feeder Position } t(a_3)| \cdot W_s/V_f\\
D &= \Delta_{a_3}^S \cdot I_T\\
E &= \sum_{a\in\mathcal{R}:h(a)\in G^F,\ t(a)\in G^F} c_f(a).
\end{aligned}$$

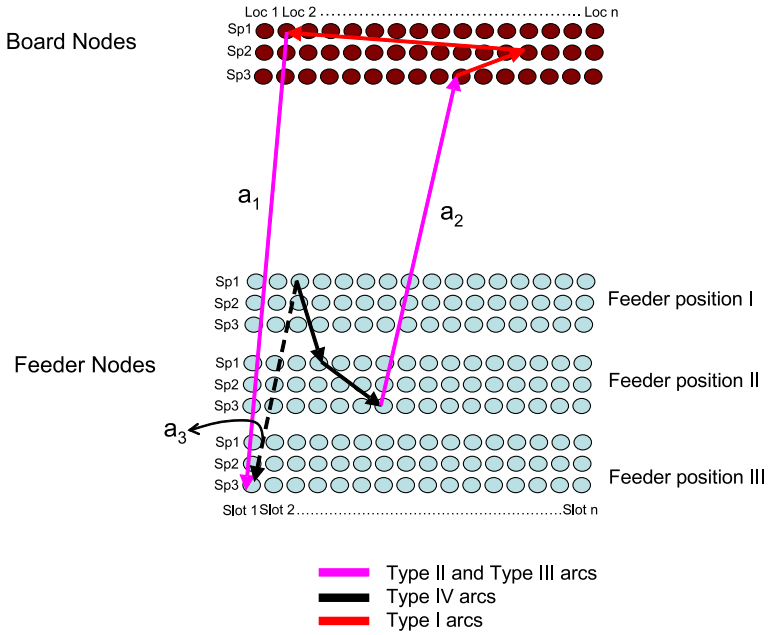

Fig. 10 Example of a typical 'route'

The cost of route r reads:

$$C(r) = \sum_{a \in \mathcal{R}: h(a) \in G^F,\ t(a) \in G^F} c_f(a) + \max\left\{ \left\{ c_{bf}(a_1) + \sum_{a \in \mathcal{R}: h(a) \in G^B,\ t(a) \in G^B} c_b(a) + c_{fb}(a_2) \right\}, c_f(a_3) \right\}. \quad (1)$$

The first part in (1) corresponds to the time it takes the feeder and head during the pick-up operations. At the end of this step, the head goes on the board and spends A units of time before returning back to the feeder. During this time, the feeder needs to move along arc a_3. At node $h(a_3)$, the two meet and have to be synchronized, which is captured by the maximum operator. Cost $c_f(a)$ by definition does not capture mere feeder movement time, but a maximum among the three time-contributing components, and thus next we argue that nevertheless (1) is correct. We have $C(r) = E + \max(A, c_f(a_3))$. The first term E captures the time to pick up the components. At node $t(a_3)$ the feeder and head decouple; the head moves to the board where it stays for A units of time before synchronizing with the feeder at node $h(a_3)$. By definition, it takes the feeder C units of time to move along arc a_3. As a result the true cost/time of route r is $E + \max(A, C)$.

We know that $A \geq B$ because of the triangle inequality. We also have $A \geq D$ because along the board nodes the spindle must rotate at least as many times as along a_3. We obtain

$$\begin{aligned} E + \max(A, C) &= E + \max(A, B, C, D) \\ &= E + \max(A, \max(B, C, D)) \\ &= E + \max(A, c_f(a_3)), \end{aligned}$$

which matches our definition of $C(r)$.

We have just established that the network model formulated does indeed capture the true cost of the problem scenario accurately.

4 Solution methodology

4.1 Overall approach

The problem as posed in its complete generality is very hard to solve to optimality. This is because of the nature of the problem itself—it is a combination of multiple NP-hard problems. For example, just the sequence determination given no flexibility with spindle jumping or feeder magazine motion is a variant of a VRP, which, by itself, is a hard problem to solve on large data sets.

A pure mathematical programming approach would likely not work due to the reasons described above. For this reason, we resort to a heuristic. Instead of employing a traditional local search strategy or a pure meta-heuristic, we combine very large-scale neighborhood search ideas with mathematical programming, similar in spirit to Franceschi et al. [17]. To this end, we used a two-phase solution methodology. First, we designed a complex heuristic that generates a good initial solution that is bounded by a theoretical worst case performance ratio. The theoretical worst-case analysis is presented in [38]. This solution is then iteratively improved at each subsequent computational iteration by an interchange algorithm so that we are assured to find a no worse solution than the one we begin with.

The improvement phase is based on a recent work by Franceschi et. al. [17] wherein they achieve excellent results using a refinement algorithm following a similar philosophy for the standard VRP. The idea is to select certain nodes to extract from the starting solution, then extract the nodes, and derive sequences or short "strings" of nodes to reinsert into the short-cutted solution, next to formulate an integer program and solve it using an ILP solver to incorporate the results back into the solution. This process is repeated until an iteration limit is reached or the solution is of an acceptable quality (see Fig. 11 for an overview of the methodology). This is a very interesting approach because it uses the x-Opt philosophy, but computes intelligently at each re-insertion the best insertion points of a given sequence of nodes using an integer program that is far more tractable than the original ILP would be. For example, this technique when applied to the TSP leads to a simple assignment problem that needs to be solved in order to determine the best ordering for the refinement phase [37].

A special distinction of this approach is that in the extreme case when all nodes are extracted and all possible sequence combinations of them are reinserted, this heuristic is guaranteed to find the optimal solution. Thus, there is a very explicit control over the tradeoff between efficiency and quality of the solution. Furthermore, due to the

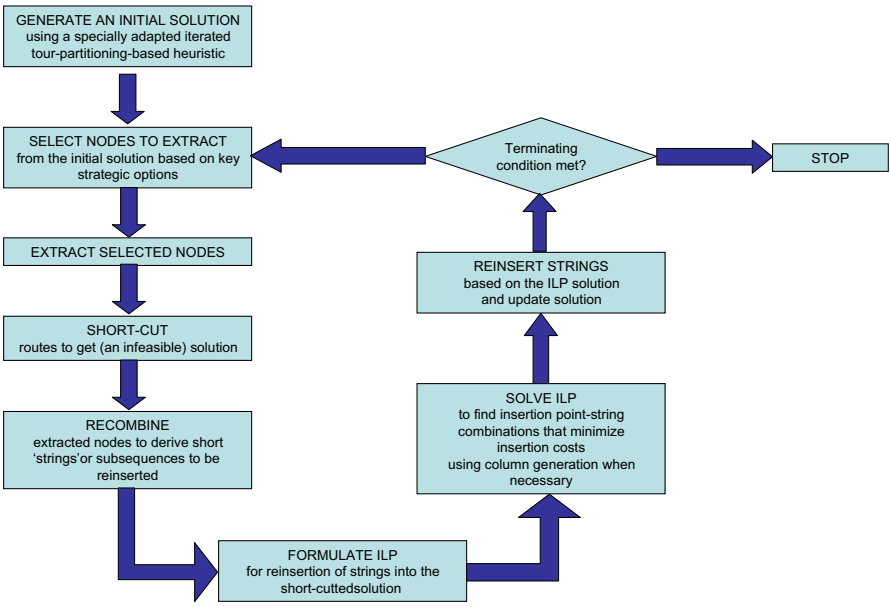

Fig. 11 An overview of the solution methodology

advantageous nature of this approach, it becomes also possible to compare solutions generated by the heuristic vs. an optimal solution for small instances.

4.2 Initial solution generation heuristic design

The initial solution is generated by applying a heuristic designed along the lines of the iterated tour partitioning (ITP) heuristic [22]. In order to bring the problem to a tractable stage, we first neglect the computations on the feeder side and solve the problem on the board side. Taking this solution, the computations on the feeder side are then carried out in order to get a feasible initial solution. In the process of doing this, we encounter the generalized traveling salesman problem (GTSP). The GTSP is a well-known NP-hard problem [16] and is in fact known to be inapproximable within any constant factor [14]. For these reasons, we use the transformation scheme proposed in [15,16] to approximately solve the GTSP-like problems which arise in the course of our initial solution generation heuristic.

4.2.1 Board side computations (BSc)

The standard ITP heuristic works on a TSP tour in order to produce solutions to the capacitated VRP. In our case, however, we cannot use the TSP as we do not need to traverse every single board node in the expanded network formulation. Looking carefully at the structure of the network and the desired solution, we note that we need to visit exactly one out of the set of G^B nodes corresponding to every unique B

node. We also need to use every spindle at most once in every route (note that the TSP solution in the capacitated VRP case corresponds to multiple routes as is also the case for us). As a result, we get two constraints on the generalized shortest cycle problem that makes it resemble a GTSP constrained in two-dimensions. Figure 12 shows how a solution to the standard GTSP is insufficient in satisfying the feasibility requirements of the problem under study. Figure 13 illustrates a sample feasible solution. To the best of our knowledge, this 2-D GTSP problem has never been tackled before. We, therefore, devise, a new heuristic to obtain a solution to this problem.

We first solve the GTSP formed on the board side by neglecting the problem on the feeder side and the spindle assignments (the 2-D aspect). Instead of using a GTSP solver, we transform the underlying GTSP problem to a TSP following [15].

The Fischetti–Salazar–Toth (FST) Transformation to the GTSP is an approximate, yet one of the most efficient algorithms proposed and involves the following steps.

1. First, it requires solving a TSP over a reduced network using approximate edge costs. This reduced network can be formulated using a variety of reduction measures, the relative efficieny and performance of which are detailed in [16]. The reduced network has a node for each cluster in GTSP. The cost of an arc, for example, can be the average cost of arcs going from one cluster to the other. The TSP tour fixes the order of clusters to be visited. In our case, we use the minimum distance between two clusters as the reduction measure.
2. Then we solve a series of shortest path problems to determine the final solution to the GTSP. The shortest path problems select a particular arc between any two consecutive clusters as imposed by the order from the previous step.

The clusters in this GTSP correspond to nodes originating from the same physical board node as a result of the expansions caused by the spindle positions. At this stage, we do not consider the constraint of using a single spindle in every route. Thus, the solution produced by this step does not necessarily satisfy the single-use-of-a-spindle constraint. Next, we disregard the spindle assignment suggested by the GTSP solution and use it only to obtain node sequencing information. Having obtained a tour covering all the requisite nodes exactly once, we partition this tour into groups, each group having the number of nodes equal to the number of spindles, i.e. each

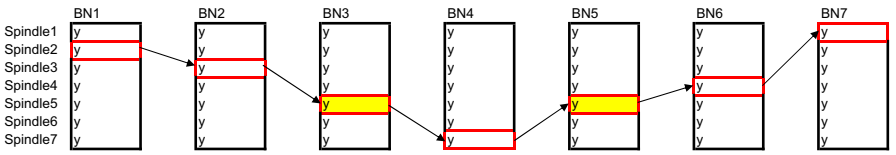

Fig. 12 An example showing how a solution to the standard GTSP is infeasible for the 2-D GTSP

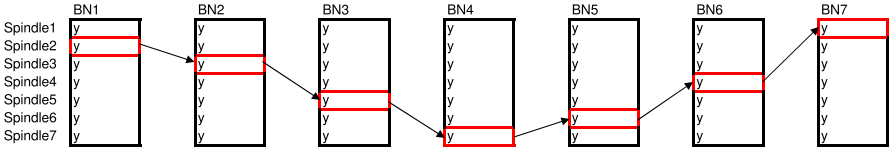

Fig. 13 A feasible solution to the 2-D GTSP

group representing a 'route', similar to the ITP heuristic. The last route might have less nodes than the number of spindles. For each route group, the problem of assigning the spindles to the ordered set of G^B nodes one-to-one becomes the standard quadratic assignment problem (QAP), which is then solved approximately by a greedy strategy.

At the end of the BSc, we have the routing through the G^B nodes completely determined. We address the remainder of the problem through the feeder side computations.

4.2.2 Feeder side computations (FSc)

After solving the BSc, the feeder spindle assignment is implied by the one chosen in the BSc because the spindle assignments have to be identical on the board and feeder side for feasibility. The problem on the feeder side is, however, still not quite simple. For each route group formed in the BSc, the decision as to which feeder positions must be used at the pick-up nodes, as also the sequence of the pick-up nodes themselves, have yet to be determined. This is again a GTSP because we need to visit exactly one out of the G^Fs arising out of each requisite physical pick-up node, i.e. F node, while also choosing a feeder magazine position at which the feeder must be positioned when the head is active at this node. We again make use of the FST transformation to solve this GTSP formed on the feeder side. The clusters in the GTSP this time, however, correspond to nodes coming from the same feeder node as a result of the expansions caused by the feeder magazine positions.

4.3 Improvement heuristic design

The improvement phase heuristic is based upon the algorithm proposed by Franceschi et al. [17] for the standard VRP. We follow the scheme used in the case of the simple VRP as outlined in [17], but extend it to the far more complicated VRP-like problem at hand with domain-specific constraints and distance metrics on our exponentially large and non-uniform network model comprising of both feeder and board nodes. This makes the pricing problem far more complicated, for example, in addition to other complexities that are added to the model. Because of the nature of the problem, additional strategies such as partial vs. full synchonization between board and feeder sides emerge. As a consequence, we also have more design choices that need to be carefully made to achieve a computationally effective and powerful algorithm. Our adopted algorithm is an iterative scheme that begins with an initial solution and carries out the following steps.

4.3.1 Selection

In the first step, nodes from the incumbent solution are selected to be marked for extraction. The selection can be based on multiple schemes, which can be used individually, or in conjunction with each other, to maximally gain benefits from their individual characteristics at each of the different stages of the progression of the execution of the algorithm. Each of the schemes has parameters which can be controlled and tuned to get the best performance of the heuristic.

1. *Random-alternate:* In every route, we choose randomly between extracting all even or all odd nodes (or, in larger examples, every fourth node or every eighth node, etc.).
2. *Scattered:* We decide, for every node, with a predetermined, fixed probability, whether the node be extracted or not. This scheme allows for the extraction of node sequences consisting of consecutive nodes. It is also conceivable to vary the probability of extraction as a function of the number of iterations passed.
3. *Neighborhood:* Here we concentrate on a seed node, say v^*, and remove nodes v with a probability that is inversely proportional to the distance d_{vv*} of v from v^*. A score is assigned to each node v^*, which is proportional to the number of nodes "close" to v^*. This score is used to rank the nodes in descending order, from which potential nodes to be extracted are selected as seed nodes in each iteration.

Franceschi et al. [17] state that schemes random-alternate and scattered appear better suited to improve early solutions, whereas the neighborhood scheme seems more useful to deal with the later iterations.

4.3.2 Extraction

In this step, we extract out the nodes marked in the selection step. Typically, nodes can be pulled with several different strategies. We could apply the selection process to both the board and feeder nodes independently or we could maintain either partial or complete synchronization between the board and feeder nodes, applying the selection process to one of the sets while deriving the selections from the other set by enforcing (partial or complete) feasibility conditions on the resultant routes. Pulling out nodes destroys the routes, which are reconstructed by short-cutting the solution, i.e. creating a new arc between the two nodes neighboring the extracted nodes. Again, three possible schemes can be considered here, however these schemes are more global in that, once chosen, they are adhered to throughout the execution of the algorithm. The idea behind the multiple schemes is the degree of flexibility allowed to the heuristic to destroy feasibility at every iteration. When we pull out nodes randomly from a feasible solution, we run the inevitable risk of destroying feasibility completely. This can be beneficial if the heuristic is able to take advantage of the opportunity for finding better solutions by temporarily traversing infeasible space. Based upon these considerations, we developed three schemes.

1. *Asynchronous Between Board and Feeder:* Under this scheme we pull out both G^B and G^F nodes from the incumbent solution independently of each other without considerations of destroying feasibility of the route. This is the most flexible option likely leading to the most promising results, yet computationally the most challenging.
2. *Synchronous Between Board and Feeder:* We maintain complete feasibility while extracting the nodes. This implies pulling out the corresponding G^F node when a G^B node is pulled out and vice versa. The correspondence is implied by the underlying component type. This approach is the most rigid, but the easiest computationally.

3. *Partial Asynchronous/Alternate Asynchronous:* Considering the above approaches as being two extremes, this scheme aims to get both speed and flexibility with pulling out only G^B nodes in an iteration, only G^F nodes in the next iteration, and so forth, thereby maintaining the feasible structure of the routes on at least one of the board and feeder sides.

4.3.3 Recombination

In this step, we use all the extracted nodes to create a pool of subsequences to be potentially reinserted into the solution, albeit at a better location than they were at earlier. In order to expand the extent of the solution space considered, we form all possible combinations of the extracted nodes using the different spindles on the head and assuming the feeder magazine to be in the different possible positions. Furthermore, to improve the performance and get better solutions in fewer iterations, we also form short sequences of nodes by combinatorially choosing nodes using this expanded node set. We have examined the benefits of including sequences of up to length n, n being a small number (5 in our case). This means that we would have an enormously large number of possible candidates for reinsertion.

Therefore, we use the idea of dynamically pricing these candidate strings based on the solution of the ILP described below to develop 'good' sequences for each insertion point and to find 'good' insertion points for each sequence. The reason we expand the node set is to allow for additional flexibility and to avoid being stuck with rigid choices of the current spindle assignment and feeder magazine placement.

4.3.4 Reallocation

In this step, we form an insertion ILP. This ILP reinserts back the extracted nodes. Note that the model is always feasible since the nodes can always be reinserted back to the original locations. The model is presented in Appendix 1. Essentially, the ILP is an assignment problem, but with many additional constraints. The complete ILP has a large number of columns. We study the effect of two possible strategies to cope with this computational complexity.

Using random strategy We solve the problem by randomly choosing a predetermined, large number of columns from the entire set of columns to add to the problem.

Using column generation strategy We use the dual values of the LP relaxation of the formulated ILP to decide which columns to add to the model. Due to the very large number of total possible columns that can be added to the problem, column generation is employed [7]. The pricing problem, given in Appendix 2, is the problem of finding a constrained lowest cost cycle (in a network with positive and negative cost cycles). This is an extremely hard problem to solve and for this reason, we resort to randomly selecting columns based on the underlying reduced cost. To this end, we first solve the LP relaxation of the model with a reasonable number of columns. Then, using the reduced costs, we select from a large set of randomly generated columns those

columns that have a negative reduced cost. We add these columns to the model, solve the LP relaxation, and re-iterate to find more columns based on updated reduced costs.

Once there are no significant improvements in the objective value of the LP relaxation, we revert back to the ILP formulation with the best columns in terms of their reduced costs, and solve the ILP.

4.3.5 Reinsertion

We incorporate the near-optimal results obtained from the reallocation ILP into the solution using insertion operations. This re-insertion step marks the end of an iteration, and the entire series of steps is repeated in the next major iteration.

We provide the user of our algorithm the complete flexibility to design the computation as best suited to the needs of the underlying particular problem in terms of size, complexity, and availability of computational resources. While we have examined multiple combinations of these strategies on different problem sets, in the numerical experiments that follow, we used the default strategy consisting of the highly flexible scheme of using the scattered selection strategy, the asynchronous scheme for node extraction, and column generation not turned on.

5 Computational experiments

Extensive computational experiments were designed and carried out in order to examine, test, and validate the performance of both the initial solution generation heuristic and the improvement heuristic. The sensitivity of the improvement heuristic was especially tested to its various input and design parameters such as problem size (measured in three dimensions, namely, the total number of components, the number of component types or component diversity, and the number of spindles present on the head), the inital or starting solution provided to the improvement heuristic, the maximum sequence or string length used for recombination, enabling vs. disabling column generation strategy, introduction of additional columns, randomization, extraction probability, and selection and extraction strategies. The performance of the overall heuristic was compared against an implementation of a well-designed state-of-the-art GA for optimizing PCB assembly problems using genetic operators suggested in [30]. We have strived to be as close as possible to their suggested algorithm within the limitation of what has been disclosed in their published work. For the sake of brevity, readers are referred to their original paper for further details on the implementation of the GA. Results presented and discussed in the following sections are representative of results obtained on multiple runs of the heuristic over test cases of varied problem complexity and dimensions. We do not present here the best result among multiple runs on the same test case, but rather present a "representative" result, i.e. one that shows the magnitude of improvements seen in all performed runs (at least 3) even factoring in the random nature of our algorithm. We use the default extraction probability between 0.3 and 0.5 as it represents a good balance between fully exploring the solution space and maintaining integrity of "good" parts of the solution.

The experiments were carried out on a cluster of 52 nodes, each having two 64-bit 3.2GHz Pentium 4 Xeon processors sharing 6GB RAM with InfiniBand, running Red Hat Enterprise Linux 4 and compiled under GCC version 4.1.2. The implementation however, does not currently exploit multiple processors; it is sequential. We used the Concorde Lin-Kernigan TSP Solver [3] to solve TSPs, an in-house implementation of a version of Dijkstra's algorithm [11] to solve shortest path problems, and CPLEX 11.0 MIP solver [26] as the generic ILP solver. The QAP is solved approximately using a greedy heuristic approach. All times are reported in minutes.

The first set of 12 test instances were generated using a randomized model for a board with key parameters such as number of components, component diversity, and number of spindles as inputs. Statistical analysis of the board designs for a major electronics manufacturer partner over recent years (2007–2008) were used to determine the ranges of these parameters. Machine parameters such as velocities, on the other hand, were fixed to those values found most commonly in such configurations. As a reference point, a mid-size instance with 250 board nodes, 10 spindles, 50 component types, 100 feeder slots, and 150 feeder positions yields a complete network with 152,500 nodes.

The second set of test instances comprised of three real-world PCBs used by a partner electronics manufacturer (the only modification being the desensitivization of component serial numbers and names, e.g., part number 132032545443ART58 was replaced with partxxxx001 in the data files). These PCBs belong to representative scales of boards typical in the industry. The smallest instance is a board of low complexity containing 41 components belonging to one of 10 different component types. The second instance is a board of medium complexity containing 185 components belonging to one of 13 different types. The final instance is a board of large complexity containing 540 components belonging to one of 61 types.

These instances were run with our proposed solution methodology to rigorously observe the efficacy of our techniques both against the genetic algorithm implementation and ten variations of "sweep" heuristics often used by practitioners in industry to estimate good assembly sequences for such PCBs. The ten variations arise from the fact that such heuristics vary in detail in the way in which they are used from one manufacturer to another. We do not have access to the source code or the exact proprietory algorithms, but we generated the results of execution of our three real PCBs in consultation with our partner. It is to be noted that these heuristics are not iterative, and hence are independent of CPU time. In each case, the best among them was picked for the purpose of our comparisons.

5.1 Quality of obtained solutions

5.1.1 Initial solution generation heuristic

In order to gauge the quality of the initial solution generation heuristic, we note that the generated initial solution was consistently found to be about 50 % better than the best solution obtained by running the GA for 3 h starting from a completely random solution pool. The running time of the initial solution generation algorithm averages

2–10 min for common board instances. As we shall see in the next section, the quality of the starting solution is critical to the performance of the improvement heuristic.

5.1.2 Improvement heuristic

Two tests were conducted with the heuristic in order to validate and compare its performance. Firstly, the heuristic was compared to an implementation of the state-of-the-art GA methodology as suggested in [30]. It was found that our improvement heuristic clearly outperformed the GA with 25 % of its initial population set to be identical to the solution obtained by using our initial solution generation heuristic. Figure 14 shows a comparison between the two algorithms started from the same initial solution generated using our initial solution generation heuristic for two instances from the first set, one with a large component diversity and even component population density, and the other with a small component diversity and a concentrated component placement grouping. Since the results for the 12 instances are similar in nature, we omit for the sake of brevity, the presentation of graphical results for each one of them. At each time point, the best member in the GA population was selected for comparison. An average objective value difference of 25–50 % was observed over multiple test cases of varying number of components, component diversity, and number of spindles, with the gap being larger for more difficult problems. This was found to be consistent also in the real-world board instances (the second set of test instances).

In the second experiment, our heuristic was started with a randomized initial solution instead of a 'good' solution as provided by the initial solution generation heuristic. It was again found that our heuristic was very effective in improving upon such a random solution (see Fig. 15). The real-world instances were found to perform as well as the randomized examples. It is interesting to note that even after running our heuristic for several hours, we did not get to the value of the starting solution provided by the initial solution generation heuristic. This demonstrates that the quality of the starting solution is extremely important even for the improvement heuristic. Both aspects of our work have their respective crucial importance in the derivation of a good solution to this extremely complex and challenging problem.

We also executed comparisons of the algorithms two individual runs on two different computing environments as listed in Table 1 by starting from a solution generated by our initial solution generation heuristic both in the case of our proposed algorithm and the GA. Table 2 specifically shows the power of our proposed technique in comparison

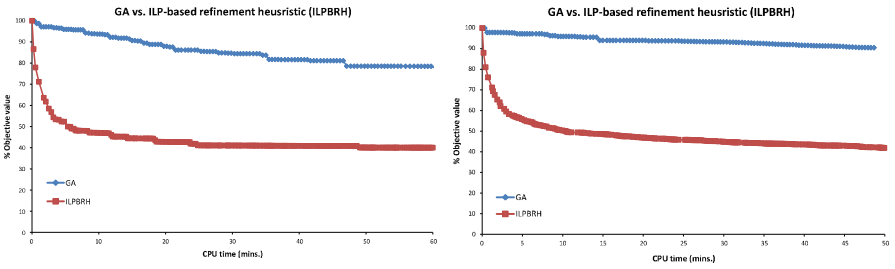

Fig. 14 Comparing heuristic performance with the state-of-the-art GA for two widely different board types

to the GA and the aforementioned sweep heuristics on each of the three real-world test cases wherein we can see that the performance gain improves significantly as the complexity increases. A sample summary iteration log is also presented in Table 3 for the benefit of the interested reader illustrating the progress on the objective function with iterations in executing one of the three real PCBs. The chosen dataset is the real medium complexity PCB run in the environment E1 with execution restarted using the implemented hot-start functionality upon encountering CPLEX exceptions.

We also plot in Figs. 16 and 17 the routes in the starting solution and a solution evolved over time using our heuristic. It is clear that the routes very clearly get 'sorted out', i.e. the number of self-crossings in the routes is dramatically reduced. Going back to the basic TSP/VRP structure underlying the problem, we observe that minimal self-intersection is an essentially desirable characteristic of a good solution. These figures are, thus, a further proof of the efficacy of the developed algorithm.

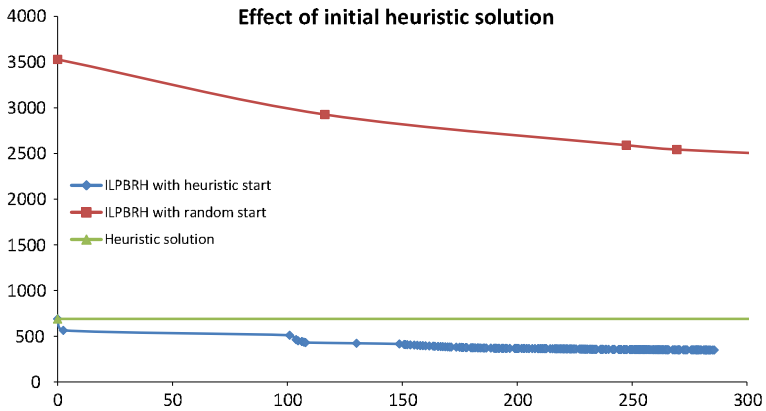

Fig. 15 Effect of starting with random vs. generated initial solution on a test instance from the first set of instances

Table 1 Execution environments for real PCB data evaluation

Rockdude (E1)	
CPLEX	12.5.0.0
Processor	16-core, x86, 64-bit, Intel® Xeon® X5570, 2.93 GHz
RAM	48 GB
OS	CentOS 6.5, Kernel 2.6.32-431.e16.x86_64
GCC	RedHat 4.4.7-4
Euler (E2)	
CPLEX	12.6.2.0
Processor	4-core, x86, 64-bit, Intel® Core™ 2 Quad, 2.4 GHz
RAM	8 GB
OS	CentOS 6.5, Kernel 2.6.32-279.14.1.e16.x86_64
GCC	RedHat 4.4.6-4

Table 2 Comparison of algorithms on real PCB datasets

Algorithm	Environment	Objective value		
		At CPU time = 500 s	At CPU time = 1500 s	At CPU time =5000 s
		Low complexity instance	Medium complexity instance	High complexity instance
ILPBRH	E1	85.50	312.09	978.58
ILPBRH	E2	85.97	309.99	947.78
GA	E1	115.17	556.89	1842.03
GA	E2	112.54	556.98	1890.64
Sweep	N/A	165.67	1016.45	7613.14

Table 3 Summary iteration log for ILPBRH on a real medium complexity PCB dataset

Iteration #	Time (s)	Obj. val.
0	0	601.51
45	72.70	350.54
Restart	–	–
80	121.09	335.55
Restart	–	–
171	273.94	323.69
Restart	–	–
295	467.05	320.23
Restart	–	–
596	961.13	313.99
Restart	–	–
693	1113.21	313.25
Restart	–	–
868	1388.49	312.51
Restart	–	–
938	1479.31	312.46
Restart	–	–
1035	1629.98	311.98
Restart	–	–
1336	2124.50	309.50

5.2 Sensitivity of solution quality to parameter choice

5.2.1 Effect of variation of heuristic design parameters on performance

We found the effect of the choice of the maximum string length parameter for reinsertion to have only a negligible effect on the heuristic's performance, with lower

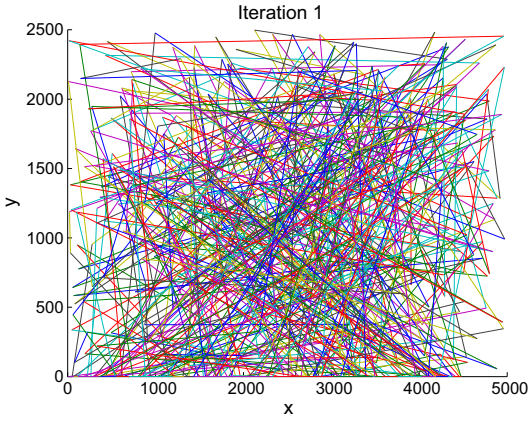

Fig. 16 Initial solution route visualization

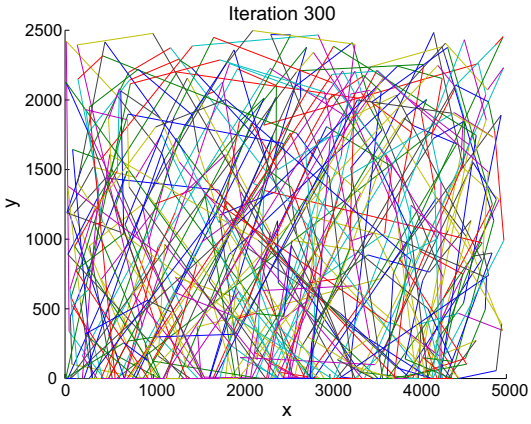

Fig. 17 Optimized solution route visualization

values of the parameter (specifically sequences of size 1, and of size at most 2) often performing the best. These results were consistent across boards of varied sizes and using widely different number of spindles in the head. This can be easily understood, for it is obvious that the heuristic can do better by letting the ILP find which nodes to neighbor rather than having to choose them ineffectively out of a set of millions of possible combinations, albeit using many more iterations to do so. A typical string length of up to a maximum of 3 nodes was used for the sake of the experiments.

We also recorded from numerous experiments the effect of the extraction probability, or the percentage of perturbation in the original solution on the performance of the heuristic, see Fig. 18. The extraction probability is defined as the percentage of nodes in the solution that are extracted out in a given iteration. It is observed that the heuristic performs better in general with lower values of the extraction probability (up to $p = 0.2$ typically). This result too is indicative of the power and value of such an algorithm since using $p = 1$ is equivalent to solving the entire, original integer problem, which obviously is highly intractable as has been discussed. We

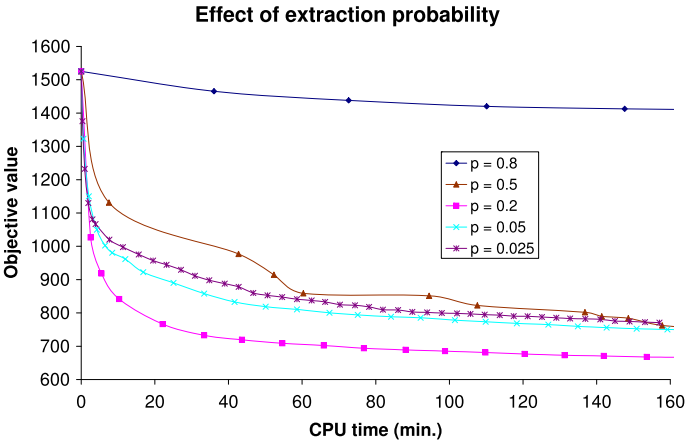

Fig. 18 Effect of different extraction probabilities on heuristic performance

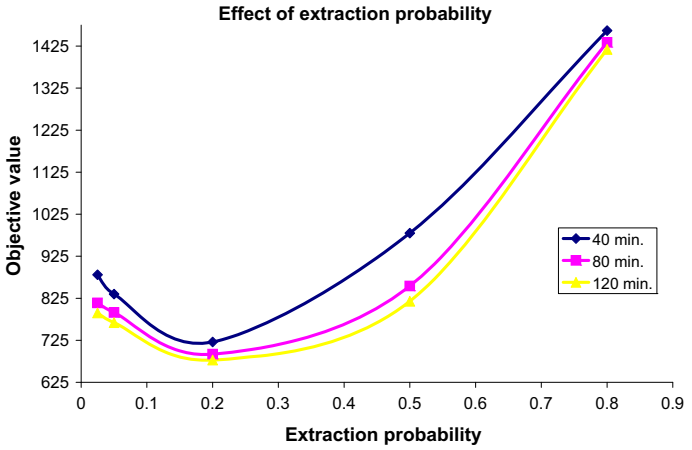

Fig. 19 Section view of effect of extraction probability on heuristic performance

note that even with small perturbations, which lead to smaller ILPs that can be very easily and quickly solved to optimality, we can obtain very sharp decreases in the value of the objective function of routing problems. Beyond a certain point, however, too low of an extraction probability value results in poorer performance, see Fig. 19, where a data point shows how the objective value reached after a certain amount of computation varies with the use of a certain value of extraction probability. An optimal strategy is to continuously adapt the extraction probability with time, starting with lower values and increasing it up to a certain point, or allowing high spikes in between the continual use of low values. Using higher values causes the ILPs to get increasingly difficult and time-consuming to solve, and could, therefore, also be used only intermittently to 'shake out' the solution. With little computational effort, dramatic improvements can be obtained. This is a desirable property of any heuristic.

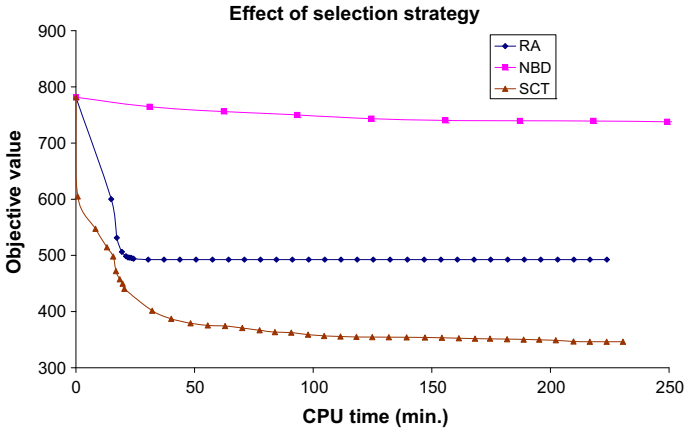

Fig. 20 Effect of different selection strategies on heuristic performance

We also investigated the effects of the three different selection strategies on the performance, see Fig. 20. In this figure, RA corresponds to the random-alternate strategy, NBD corresponds to the neighborhood strategy, and SCT corresponds to the scattered extraction strategy. It was found that better performance of the heuristic is favored by greater randomness in the selection strategy. The scattered strategy performs the best, while the neighborhood stategy, which has the most 'intelligent' structure performs the worst. This can again be explained by leaving the making of intelligent choices to the heuristic by itself rather than us using simplistic and ineffective guesses at them.

The effects of three different extraction strategies were also investigated, see Fig. 21. It was found that in the long run, the fully asynchronous (between the board and the feeder) strategy labeled as 'ASYNC' performs the best because it gives maximum flexibility to the heuristic to hack its way through the search space. In the short run, however, imposing some feasibility constraints through the partially synchronous strategy marked as 'PASYNC' on the heuristic's traversal through the search space can prove to be quite effective. However, over-constraining the problem by rigorous imposition of feasibility as in the fully synchronous strategy denoted by 'SYNC' can have a negative effect, and leads to the heuristic being unable to fully explore the search space. Again, a hybrid approach where the strategy is adapted as the solution of the problem progresses would work best.

The effect of component diversity was not found to be very significant, demonstrating the effectiveness and proving the generality of our solution approach to boards of varied kinds typically found across multiple industry domains (for example, avionics boards typically contain a large and diverse population of component and chip types, whereas certain specialized and consumer electronics boards can sometimes contain only one type of component replicated on hundreds of locations).

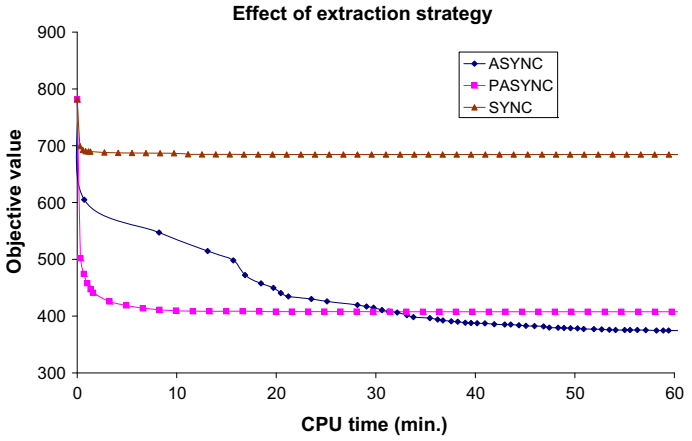

Fig. 21 Effect of different extraction strategies on heuristic performance

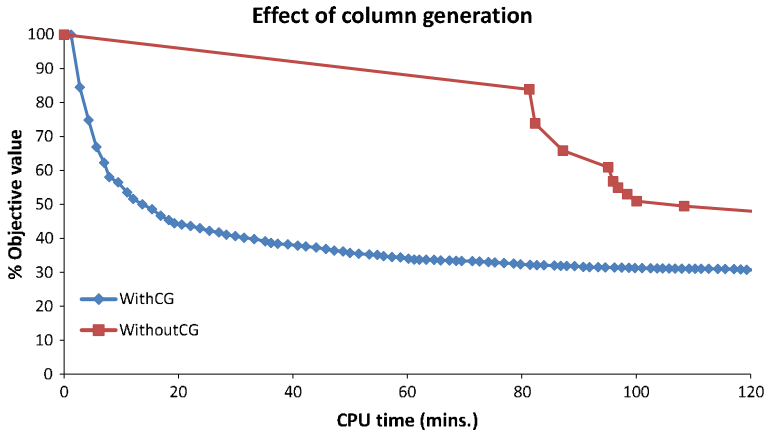

Fig. 22 Effect of column generation strategy on heuristic performance

5.2.2 Effect of using the column generation strategy on performance

We also examined through various computational experiments the influence of using the column generation strategy as opposed to the random large column set selection strategy. A typical run for a 400 component instance is shown in Fig. 22 where 'CG' stands for column generation. It is clear from the results obtained that column generation does help, especially for larger test cases, when it becomes prohibitive to randomly select a good and large enough column set without choking the ILP solver. For smaller test cases, the benefits obtained are to some degree balanced out in many cases by the extra computational effort in generating columns iteratively. Again, it was observed that the maximum string length does not play a significant role in the performance of the heuristic.

5.3 Scalability of performance

5.3.1 Initial solution generation heuristic

The initial solution heuristic was found to consume computational time that only linearly grows with the number of components, while remaining independent of component diversity, and growing in a convex fashion with an increasing number of spindles. In Tables 4, 5, and 6, the column 'Init. soln. time' shows the total running time of this heuristic. All three tables are based on data for runs carried out over 1000 iterations. These trends are in line with our expectations given the nature of the initial solution generation heuristic as an increasing number of components reflects in a near-linearly increasing network size in terms of the number of nodes, whereas an increasing number of spindles corresponds also to an increasing network size for the GTSP, but simultaneously also causes a decrease in the number of routes needed, which at some point becomes the dominating factor and causes the flattening effect. Component diversity does not appear to significantly affect the initial solution generation heuristic's performance.

5.3.2 Improvement heuristic

Tables 4, 5, 6, and 7 give further details on the improvement heuristic. We note that the computational time and memory requirements for the improvement heuristic are reasonable. The average number of columns in the formulated ILP in each iteration grows substantially with the number of components, Table 4 (due to the combinatorial explosion), but grows only linearly with number of spindles, Table 5. On the other hand, the average running time per iteration grows drastically both in the number of

Table 4 Effect of increased number of components, with # of CTs = 10, # of spindles = 6

# of comps	Init. soln. time (min)	Avg. iter. time (min)	# of IP rows	# of IP cols
29	0.53	0.16	161	10,829
67	1.27	0.22	381	15,468
129	3.77	0.71	706	22,239
241	4.81	2.97	1315	55,854
461	10.44	192.17	2498	144,685

Table 5 Effect of increased number of spindles, with # of components = 202, # of CTs = 40

# of spindles	Init. soln. time	Avg. iter. time (min)	# of IP rows (min)	# of IP cols
6	4.59	0.88	1088	38,707
12	14.55	2.55	1007	77,191
18	45.66	16.15	1017	102,262
24	93.53	45.87	1010	120,575

Table 6 Effect of increased component diversity, with # of components = 200, # of spindles = 6

# of comp. types (CTs)	Init. soln. time	Avg. iter. time (min)	# of IP rows (min)	# of IP cols
20	2.62	0.86	1111	35,257
40	2.60	0.86	1111	34,316
80	2.63	0.75	1111	33,201
160	2.65	0.86	1111	32,829

Table 7 Effect of increased no. of components, spindles, and component diversity

# of comps	# of CTs	# of spindles	Time to 10 % impr. (min)	Time to 25 % impr. (min)	Time to 50 % impr. (min)
100	10	5	0.06	0.12	7.73
100	80	5	0.06	1.02	27.82
100	80	20	0.25	0.59	23.75
100	10	20	0.25	0.63	12.20
200	40	5	0.35	4.15	38.05
200	40	15	11	26.10	144.81
400	40	5	3.02	7.50	156.67
400	320	5	17	104.50	981.38

components and the number of spindles on the head. The average number of rows in the formulated integer program at each iteration, meanwhile, grows linearly with the increasing number of components, while actually slightly decreasing with the increasing number of spindles (due to the lower number of resultant routes in the solution). These trends are again more or less aligned with our expectations given the design and nature of the heuristic.

Table 7 (also averaged over 1000 iterations) shows the time to obtain 10, 25, and 50 % improvements in the objective function for increasingly complex instances in terms of number of components, component diversity, and number of spindles. We observe from this table that initial improvements are very quick and further improvements take substantially longer times. Also, the same trends of increasing complexity leading to increased run times can be observed in these numbers.

5.4 Development of a regression equation

Based on extensive experimentation over a typical range of parameter and problem size values found in industrial settings, we developed a regression equation incorporating the influence of the number of components, number of spindles, and the extraction probability with a progression in time in order to be able to provide a mechanism for estimating the value of the objective function as well as for the expected percent decrease in it as a function of time and problem size. The aim of this regression study

Table 8 Regression model summary

Source	Degrees of freedom	Sum of squares	Mean square	F value	Approx. Pr > F
Model	6	2.2053e+9	3.6755e+8	163,449	<.0001
Error	273,192	6.1433e+8	2248.7		
Corrected total	273,198	2.8196e+9			

Table 9 Regression model parameter estimates

Parameter	Estimate	Approx std error	Approx 95 % lower confidence interval	Approx 95 % upper confidence interval
p_0	−66.67	0.980	−68.58	−64.75
p_1	1.340	0.001	1.34	1.34
p_2	954.40	16.637	921.80	987.00
p_3	−371.00	2.240	−375.40	−366.60
p_4	65.90	0.846	64.26	67.58
p_5	−1149.60	60.276	−1267.70	−1031.40
p_6	670.50	5.308	660.10	680.90

is to characterize statistically the expected performance of the proposed algorithm in order to serve as an aid in answering cost-relevant questions (from a model-tuning, resource investment, or hardware-upgrade perspective), e.g. to determine if it is worth expending further computational resources on a given problem in terms of additional expected improvement, or if it would help buying a bigger head on a machine or to establish the performance impact of a 25 % increase in the number of components, etc. We leave out the effect of the selection and extraction strategies, as their choice is dominatingly clearer, and thus not a key issue for prediction in potential future cases. The regression model reads

$$Y = p_0 + p_1 \cdot X_1 + \left(\frac{p_2}{X_2} + \frac{p_5}{X_2^2}\right) + \left(p_3 \cdot X_3 + p_6 \cdot X_3^2\right) + p_4 \cdot \exp(-X_4). \quad (2)$$

In Eq. (2), Y is the predicted variable, namely, the value of the objective function as a function of the predictor variables $X_1, \ldots, X_4$, which refer to the number of components, the number of spindles, the extraction probability, and the CPU time spent. The corresponding parameters $p_0, \ldots, p_6$ are used to fit the curve to the experimental data and for validating the regression model. Table 8 lists the regression coefficients.

This equation has been validated over the collected data and it obtains a good fit with an R^2 value of almost 0.8. Table 9 shows a summary of the fitted model. The model provides a satisfactory explanation of the observed data and contains terms that logically make sense in light of the physical understanding of the system being modeled, e.g., the value decreases negatively exponentially with respect to time and

inversely with respect to the number of spindles used on the machine. On the other hand, the extraction probability turned out to be included as a quadratic polynomial explaining the valley effect (i.e. the performance of lower probabilities being better until a certain minimum threshold probability is hit beyond which the performance starts deteriorating again) over the range in question.

Appendix 1: Detailed integer linear program (ILP) model formulation for improvement phase

We formulate herein the integer linear program allowing feeder magazine motions and spindle jumping by using a bi-complete and total asynchronous extraction strategy.

Parameters

$\mathcal{F}^B$ = set of extracted board nodes
$\mathcal{F}^F$ = set of extracted feeder nodes
$\mathcal{S}^B$ = sequence pool for board node sequences (potentially all board node sequences)
$\mathcal{S}^F$ = sequence pool for feeder node sequences (potentially all feeder node sequences)
$\mathcal{I}^B$ = set of candidate insertion points for board node sequences
$\mathcal{I}^F$ = set of candidate insertion points for feeder node sequences
$\mathcal{S}_P$ = set of spindles
C = capacity of the head (vehicle) = $|\mathcal{S}_P|$
r = route
r^F = sequence of nodes corresponding to part of route r covering the feeder nodes
r^B = sequence of nodes corresponding to part of route r covering the board nodes
$\mathcal{R}$ = ordered list of all routes
$\mathcal{I}(r)$ = set of insertion points (i.e. arcs) associated with route r
$\tilde{q}(s)$ = contribution to capacity usage of sequence s, i.e. number of nodes in node sequence s
$\tilde{c}(s)$ = cost of sequence s
γ_{fi} = additional cost due to insertion of sequence $f \in \mathcal{S}^F$ at insertion point i
γ_{bi} = additional cost due to insertion of sequence $b \in \mathcal{S}^B$ at insertion point i
$l_2(r)$ = insertion point corresponding to the $G^B - G^F$ arc in the route r
$l_1(r)$ = insertion point corresponding to the $G^F - G^B$ arc in the route r
$\mathcal{B}(s)$ = beginning node of sequence s
$\mathcal{L}(s)$ = last node of sequence s

$$\mathcal{T}(s, s_p) = \begin{cases} 1 \text{ if spindle } s_p \text{ is used in sequence } s \\ 0 \text{ otherwise} \end{cases}$$

$\tilde{t}(s, s_p)$ = component type number carried by spindle s_p in the sequence s, returns 0 if none

$F_p(n)$ = feeder position at node $n \in G^F$

γ_{fi} = insertion cost of reinserting a sequence of feeder nodes $f \in \mathcal{S}^F$ into a $G^F - G^F$ type insertion point $i \in \mathcal{I}^F$

$\gamma_{fl_1(r)}^{L_1^F}$ = insertion cost only on the feeder side of reinserting a sequence $f \in \mathcal{S}^F$ of G^F nodes into a $G^F - G^B$ type insertion point $\in \mathcal{I}(r)$

$\gamma_{fl_2(r-1)}^{L_2^F}$ = insertion cost only on the feeder side of reinserting a sequence $f \in \mathcal{S}^F$ of G^F nodes into a $G^B - G^F$ type insertion point $\in \mathcal{I}(r-1)$; note that the cost of the neighboring route of the route containing i is influenced

γ_{bi} = insertion cost of reinserting a sequence $b \in \mathcal{S}^B$ of G^F nodes into a $G^B - G^B$ type insertion point $i \in \mathcal{I}^B$

$\gamma_{bl_2(r)}$ = insertion cost of reinserting a sequence $b \in \mathcal{S}^B$ of G^B nodes into a $G^B - G^F$ type insertion point $i \in \mathcal{I}(r)$

$\gamma_{fl_2(r)}^{L_2^B}$ = insertion cost only on the board side of reinserting a sequence $f \in \mathcal{S}^F$ of G^F nodes into a $G^B - G^F$ type insertion point $i \in \mathcal{I}(r)$

$\gamma_{bl_1(r)}$ = insertion cost of reinserting a sequence $b \in \mathcal{S}^B$ of G^B nodes into a $G^F - G^B$ type insertion point $i \in \mathcal{I}(r)$

$\gamma_{fl_1(r)}^{L_1^B}$ = insertion cost only on the board side of reinserting a sequence $f \in \mathcal{S}^F$ of G^F nodes into a $G^F - G^B$ type insertion point $i \in \mathcal{I}(r)$

Decision variables

$$y_{bi} = \begin{cases} 1 & \text{if board node sequence } b \text{ is allocated to the insertion point } i \in \mathcal{I}^B, \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{fi} = \begin{cases} 1 & \text{if feeder node sequence } f \text{ is allocated to the insertion point } i \in \mathcal{I}^F, \\ 0 & \text{otherwise.} \end{cases}$$

t_r = time for route r

F_r^M = amount the feeder can move when the head is executing route r

Objective function and constraints

The objective function reads

$$\min \sum_{r=1}^{|\mathcal{R}|} t_r.$$

The constraints are as follows.

$$\tilde{c}\left(r^F\right) + \sum_{f \in \mathcal{S}^F} \sum_{i \in \mathcal{I}^F \cap \mathcal{I}(r)} \gamma_{fi} \cdot x_{fi}$$

$$
\begin{aligned}
&+ \sum_{f\in\mathcal{S}^F} \gamma^{L_1^F}_{fl_1(r)} \cdot x_{fl_1(r)} + \sum_{f\in\mathcal{S}^F} \gamma^{L_2^F}_{fl_2(r-1)} \cdot x_{fl_2(r-1)} \\
&+ \max\left\{ \tilde{c}\left(r^B\right) + \sum_{b\in\mathcal{S}^B} \sum_{i\in\mathcal{I}^B\cap\mathcal{I}(r)} \gamma_{bi} \cdot y_{bi} \right. \\
&+ \sum_{b\in\mathcal{S}^B} \gamma_{bl_2(r)} \cdot y_{bl_2(r)} + \sum_{f\in\mathcal{S}^F} \gamma^{L_2^B}_{fl_2(r)} \cdot x_{fl_2(r)} \\
&\left. + \sum_{b\in\mathcal{S}^B} \gamma_{bl_1(r)} \cdot y_{bl_1(r)} + \sum_{f\in\mathcal{S}^F} \gamma^{L_1^B}_{fl_1(r)} \cdot x_{fl_1(r)},\ F_r^M/V_f \right\} = t_r \quad r \in \mathcal{R} \\
&| \sum_{f\in\mathcal{F}^F} x_{fl_2(r)} \cdot F_p(\mathcal{B}(f)) \\
&+ \left(1 - \sum_{f\in\mathcal{F}^F} x_{fl_2(r)}) \cdot F_p(\mathcal{B}((r+1)^F)\right) \\
&- \sum_{f\in\mathcal{F}^F} x_{fl_1(r)} \cdot F_p(\mathcal{L}(f)) \qquad (3)
\end{aligned}
$$

$$
\left(1 - \sum_{f\in\mathcal{F}^F} x_{fl_1(r)}\right) \cdot F_p(\mathcal{L}(r^F))| = F_r^M/W_s \quad r \in \mathcal{R} \qquad (4)
$$

$$
\sum_{f\in\mathcal{S}^F} x_{fi} \le 1 \quad i \in \mathcal{I}^F \qquad (5)
$$

$$
\sum_{b:B(v)\in b} \left(\sum_{r\in\mathcal{R}} (y_{bl_2(r)} + y_{bl_1(r)}) + \sum_{i\in\mathcal{I}^B} y_{bi} \right) = 1 \quad v \in \mathcal{F}^B \qquad (6)
$$

$$
\sum_{b\in\mathcal{S}^B} y_{bi} \le 1 \quad i \in \mathcal{I}^B \qquad (7)
$$

$$
\sum_{f\in\mathcal{S}^F} x_{fl_1(r)} + \sum_{b\in\mathcal{S}^B} y_{bl_1(r)} \le 1 \quad r \in \mathcal{R} \qquad (8)
$$

$$
\sum_{f\in\mathcal{S}^F} x_{fl_2(r)} + \sum_{b\in\mathcal{S}^B} y_{bl_2(r)} \le 1 \quad r \in \mathcal{R} \qquad (9)
$$

$$
\begin{aligned}
&\tilde{t}\left(r^F,\ s_p\right) - \tilde{t}\left(r^B,\ s_p\right) + \sum_{f\in\mathcal{S}^F} \tilde{t}(f, s_p) \cdot \left(\sum_{i\in\mathcal{I}(r)\cap\mathcal{I}^F} x_{fi} + x_{fl_1(r)} + x_{fl_2(r-1)} \right) \\
&- \sum_{b\in\mathcal{S}^B} \tilde{t}(b, s_p) \cdot \left(\sum_{i\in\mathcal{I}(r)\cap\mathcal{I}^B} y_{bi} + y_{bl_1(r)} + y_{bl_2(r)} \right) = 0 \quad r \in \mathcal{R},\ \ s_p \in \mathcal{S}_P
\end{aligned} \qquad (10)
$$

$$\mathcal{T}(r^F, s_p) + \sum_{f \in \mathcal{S}^F} \mathcal{T}(f,\ s_p) \cdot \left(\sum_{i \in \mathcal{I}(r) \cap \mathcal{I}^F} x_{fi} + x_{fl_1(r)} + x_{fl_2(r-1)} \right) \leq 1$$
$$r \in \mathcal{R}, \quad s_p \in \mathcal{S}_P \tag{11}$$

$$\mathcal{T}(r^B, s_p) + \sum_{i \in \mathcal{I}(r) \cap \mathcal{I}^B} \sum_{b \in \mathcal{S}^B} \mathcal{T}(b, s_p) \cdot \left(\sum_{i \in \mathcal{I}(r) \cap \mathcal{I}^B} y_{bi} + y_{bl_1(r)} + y_{bl_2(r)} \right) \leq 1$$
$$r \in \mathcal{R}, \quad s_p \in \mathcal{S}_P \tag{12}$$

$$\tilde{q}\left(r^F\right) + \sum_{f \in \mathcal{S}^F} \sum_{i \in \mathcal{I}^F \cap \mathcal{I}(r)} \tilde{q}(f) \cdot \left(\sum_{i \in \mathcal{I}(r) \cap \mathcal{I}^F} x_{fi} + x_{fl_1(r)} + x_{fl_2(r-1)} \right) \leq C$$
$$r \in \mathcal{R} \tag{13}$$

$$\tilde{q}\left(r^B\right) + \sum_{b \in \mathcal{S}^B} \sum_{i \in \mathcal{I}^B \cap \mathcal{I}(r)} \tilde{q}(b) \cdot \left(\sum_{i \in \mathcal{I}(r) \cap \mathcal{I}^B} y_{bi} + y_{bl_1(r)} + y_{bl_2(r)} \right) \leq C$$
$$r \in \mathcal{R} \tag{14}$$

$$x_{fi} \quad \text{binary} \quad f \in \mathcal{S}_f,\ i \in \mathcal{I}^F \tag{15}$$
$$y_{bi} \quad \text{binary} \quad b \in \mathcal{S}_b,\ i \in \mathcal{I}^B \tag{16}$$
$$t_r \geq 0 \quad r \in \mathcal{R} \tag{17}$$
$$F_r^M \geq 0 \quad r \in \mathcal{R} \tag{18}$$

Constraint (3) models t_r, the time to complete a route. The first term represents the cost on the feeder side of the short-cutted solution. The cost of inserting a G^F node sequence into a $G^F - G^F$ type insertion point is given by the second term, into a $G^F - G^B$ type insertion point by the third term, and into a $G^B - G^F$ type insertion point by the fourth term. The maximum function captures the waiting cost due to the moving feeder and the two terms in it represent the cost of moving the feeder and the cost of the vehicle's trip to the customers before returning to the feeder. The cost incurred by the vehicle's trip is computed as the sum of the cost of the short-cutted solution, the cost of inserting a G^B node sequence into a $G^B - G^B$ type insertion point, and the cost of inserting G^F and G^B node sequences into $G^F - G^B$ and $G^B - G^F$ type insertion points. Constraint (4) models the cost of the feeder movement. The terms capture the two possibilities each for determining the last feeder node to be visited in a given route and the first feeder node to be visited in its subsequent route. Constraints (5) and (7) specify that no more than one sequence can be inserted at an insertion point. Constraint (6) states that every board node can belong to exactly one re-inserted string. Constraints (8) and (9) state that only one of a feeder or board node can be reinserted at route junction points in order to reconstruct a solution. Constraint (10) balances the component types between the board and feeder route segments, whereas constraints (11) and (12) ensure no spindle reusage within the board and feeder route segments. Constraints (13) and (14) enforce capacity limitations on the head on the feeder and board segments, respectively.

Linearization of constraints

We note that (1) and (2) are non linear constraints. Rewriting the constraints in a linearized form and using,

$$\begin{aligned} k_{1fr} &= F_p(\mathcal{L}(r^F)) - F_p(\mathcal{L}(f)) \\ k_{2fr} &= F_p(\mathcal{B}(f)) - F_p(\mathcal{B}((r+1)^F)) \\ k_{3r} &= F_p(\mathcal{B}((r+1)^F)) - F_p(\mathcal{L}(r^F)) \end{aligned}$$

we obtain

$$\begin{aligned} &\tilde{c}\left(r^F\right) + \sum_{f\in\mathcal{S}^F}\sum_{i\in\mathcal{I}^F\cap\mathcal{I}(r)} \gamma_{fi}\cdot x_{fi} + \sum_{f\in\mathcal{S}^F} \gamma^{L_1^F}_{fl_1(r)}\cdot x_{fl_1(r)} + \sum_{f\in\mathcal{S}^F} \gamma^{L_2^F}_{fl_2(r-1)}\cdot x_{fl_2(r-1)} \\ &+\tilde{c}\left(r^B\right) + \sum_{b\in\mathcal{S}^B}\sum_{i\in\mathcal{I}^B\cap\mathcal{I}(r)} \gamma_{bi}\cdot y_{bi} + \sum_{b\in\mathcal{S}^B} \gamma_{bl_2(r)}\cdot y_{bl_2(r)} + \sum_{f\in\mathcal{S}^F} \gamma^{L_2^B}_{fl_2(r)}\cdot x_{fl_2(r)} \\ &+ \sum_{b\in\mathcal{S}^B} \gamma_{bl_1(r)}\cdot y_{bl_1(r)} + \sum_{f\in\mathcal{S}^F} \gamma^{L_1^B}_{fl_1(r)}\cdot x_{fl_1(r)} \le t_r \qquad r\in\mathcal{R} \end{aligned} \tag{19}$$

$$\begin{aligned} &-V_f\cdot\left(t_r - \tilde{c}(r^F) - \sum_{f\in\mathcal{S}^F}\sum_{i\in\mathcal{I}^F\cap\mathcal{I}(r)} \gamma_{fi}\cdot x_{fi}\right. \\ &\left. - \sum_{f\in\mathcal{S}^F} \gamma^{L_1^F}_{fl_1(r)}\cdot x_{fl_1(r)} - \sum_{f\in\mathcal{S}^F} \gamma^{L_2^F}_{fl_2(r-1)}\cdot x_{fl_2(r-1)}\right) \\ &\le \sum_{f\in\mathcal{F}^F} x_{fl_1(r)}\cdot k_{1fr} + \sum_{f\in\mathcal{F}^F} x_{fl_2(r)}\cdot k_{2fr} + k_{3r} \qquad r\in\mathcal{R} \end{aligned} \tag{20}$$

$$\begin{aligned} &\sum_{f\in\mathcal{F}^F} x_{fl_1(r1)}\cdot k_{1fr} + \sum_{f\in\mathcal{F}^F} x_{fl_2(r)}\cdot k_{2fr} + k_{3r} \\ &\le V_f\cdot\left(t_r - \tilde{c}(r^F) - \sum_{f\in\mathcal{S}^F}\sum_{i\in\mathcal{I}^F\cap\mathcal{I}(r)} \gamma_{fi}\cdot x_{fi}\right. \\ &\left. - \sum_{f\in\mathcal{S}^F} \gamma^{L_1^F}_{fl_1(r)}\cdot x_{fl_1(r)} - \sum_{f\in\mathcal{S}^F} \gamma^{L_2^F}_{fl_2(r-1)}\cdot x_{fl_2(r-1)}\right) \qquad r\in\mathcal{R}. \end{aligned} \tag{21}$$

We split constraints (3) and (4) into their linearized forms, thereby getting rid of the troublesome maximum and absolute value functions.

Appendix 2: Pricing problem formulation for improvement phase

The dual of the above linear program has the objective

$$\begin{aligned}
\max \Bigg\{ & \sum_{r\in\mathcal{R}} \left(\tilde{c}\left(r^F\right) + \sum_{r\in\mathcal{R}} \tilde{c}\left(r^B\right) \right) \cdot w_r^A + \sum_{r\in\mathcal{R}} \left(V_f \tilde{c}\left(r^F\right) - k_{3r} \right) \cdot w_r^B \\
& + \sum_{r\in\mathcal{R}} \left(V_f \tilde{c}\left(r^F\right) + k_{3r} \right) \cdot w_r^C \\
& + \sum_{i\in\mathcal{I}^F} w_i^D + \sum_{v\in\mathcal{F}^B} w_v^E + \sum_{i\in\mathcal{I}^B} w_i^F + \sum_{r\in\mathcal{R}} w_r^G + \sum_{r\in\mathcal{R}} w_r^H \\
& + \sum_{r\in\mathcal{R}} \sum_{s\in\mathcal{S}_P} \left(\tilde{t}\left(r^F, s\right) - \tilde{t}\left(r^B, s\right) \right) \cdot w_{rs}^I \\
& + \sum_{r\in\mathcal{R}} \sum_{s\in\mathcal{S}_P} \left(1 - \mathcal{T}\left(r^F, s\right) \right) \cdot w_{rs}^J + \sum_{r\in\mathcal{R}} \sum_{s\in\mathcal{S}_P} \left(1 - \mathcal{T}\left(r^B, s\right) \right) \cdot w_{rs}^K \\
& + \sum_{r\in\mathcal{R}} \left(C - \tilde{q}\left(r^F\right) \right) \cdot w_r^L + \sum_{r\in\mathcal{R}} \left(C - \tilde{q}\left(r^B\right) \right) \cdot w_r^M \\
& + \sum_{i\in\mathcal{I}^F} w_i^E + \sum_{v\in\mathcal{F}^B} w_v^F + \sum_{i\in\mathcal{I}^B} w_i^G \Bigg\} \qquad (22)
\end{aligned}$$

subject to constraints

$$w_r^A + V_f \cdot w_r^B + V_f \cdot w_r^C \le 1 \qquad r \in \mathcal{R} \qquad (23)$$

$$\begin{aligned}
\sum_{r\in\mathcal{R}} \Bigg\{ & -\gamma_{fi} \cdot w_r^A - (V_f \cdot \gamma_{fi} - k_{1fr}) \cdot w_r^B - \sum_{s\in\mathcal{S}_P} w_{rs}^J \cdot \mathcal{T}(f,s) \\
& (-V_f \cdot \gamma_{fi} + k_{1fr}) \cdot w_r^C - w_i^D - \sum_{s\in\mathcal{S}_P} w_{rs}^I \cdot \tilde{t}(f,s) - w_r^L \cdot \tilde{q}(f) \Bigg\} \le 0 \\
& f \in \mathcal{S}^F, \quad i \in \mathcal{I}^F \cap \mathcal{I}(r) \qquad (24)
\end{aligned}$$

$$\begin{aligned}
\sum_{r\in\mathcal{R}} \Bigg\{ & -\left(\gamma_{fi}^{L_1^F} + \gamma_{fi}^{L_1^B} \right) \cdot w_r^A - V_f \cdot \gamma_{fi}^{L_1^F} \cdot w_r^B - \sum_{s\in\mathcal{S}_P} w_{rs}^J \cdot \mathcal{T}(f,s) \\
& -V_f \cdot \gamma_{fi}^{L_1^F} \cdot w_r^C - w_r^G - \sum_{s\in\mathcal{S}_P} w_{rs}^I \cdot \tilde{t}(f,s) - w_r^L \cdot \tilde{q}(f) \Bigg\} \le 0 \\
& f \in \mathcal{S}^F, \quad i = l_1(r) \qquad (25)
\end{aligned}$$

$$\sum_{r\in\mathcal{R}} \Bigg\{ -(\gamma_{fi}^{L_2^F} - V_f \cdot \gamma_{fi}^{L_2^F} \cdot w_r^B - \sum_{s\in\mathcal{S}_P} w_{rs}^J \cdot \mathcal{T}(f,s)$$

$$\left.-V_f \cdot \gamma_{fi}^{L_2^F} \cdot w_r^C - \sum_{s \in \mathcal{S}_P} w_{rs}^I \cdot \tilde{t}(f,s) - w_r^L \cdot \tilde{q}(f)\right\} \leq 0$$
$$f \in \mathcal{S}^F, \quad i = l_2(r-1) \tag{26}$$
$$\sum_{r \in \mathcal{R}} \left\{-k_{2fr} \cdot w_r^B - k_{2fr} \cdot w_r^C - \gamma_{fi}^{L_2^B} \cdot w_r^A - w_r^H\right\} \leq 0 \quad f \in \mathcal{S}^F, \quad i = l_2(r) \tag{27}$$
$$\sum_{r \in \mathcal{R}} \left\{-\gamma_{bi} \cdot w_r^A - \sum_{s \in \mathcal{S}_P} w_{rs}^K \cdot \mathcal{T}(b,s) \right.$$
$$\left. -w_i^F - \sum_{s \in \mathcal{S}_P} w_{rs}^I \cdot \tilde{t}(b,s) - w_r^M \cdot \tilde{q}(b)\right\} \leq 0 \quad b \in \mathcal{S}^B,$$
$$i \in \mathcal{I}^B \cap \mathcal{I}(r) \tag{28}$$
$$\sum_{r \in \mathcal{R}} \left\{-\gamma_{bi} \cdot w_r^A - \sum_{s \in \mathcal{S}_P} w_{rs}^K \cdot \mathcal{T}(b,s) \right.$$
$$\left. -w_i^F - w_r^G - w_r^H - \sum_{s \in \mathcal{S}_P} w_{rs}^I \cdot \tilde{t}(b,s) - w_r^M \cdot \tilde{q}(b)\right\} \leq 0 \quad b \in \mathcal{S}^B,$$
$$i \in \{l_1(r), l_2(r)\} \tag{29}$$

$$w_r^A, w_r^B, w_r^C, w_r^G, w_r^H, w_r^L, w_r^M \geq 0 \quad r \in \mathcal{R} \tag{30}$$
$$w_v^E \text{ unrestricted} \quad v \in \mathcal{F}^F \cup \mathcal{F}^B \tag{31}$$
$$w_{rs}^I \text{ unrestricted} \quad r \in \mathcal{R}, \quad s \in \mathcal{S}_P \tag{32}$$
$$w_{rs}^J, w_{rs}^K \geq 0 \quad r \in \mathcal{R}, \quad s \in \mathcal{S}_P \tag{33}$$
$$w_i^D, w_i^F \leq 0 \quad i \in \mathcal{I}^F \cup \mathcal{I}^B \tag{34}$$
$$w_r^I, w_r^J \leq 0 \quad r \in \mathcal{R} \tag{35}$$

where $w^A, w^B, \ldots, w^M$ are dual variables corresponding to the 13 constraint sets labelled (19)–(21) and (5)–(14). The column generation subproblem (i.e. finding the column with the minimum reduced cost) reduces to one with the following form for a given insertion point i:

$$\max_{r \in R} \left\{ \max_{f \in \mathcal{S}^{\mathcal{F}}} \left\{ -\beta_1 \cdot (\text{TSP cost for } f \text{ at } i) + \beta_2 \cdot \sum_{v \in f} \alpha_v + \beta_3 \right\} \right\}$$

for x_{fi} variables, and, similarly, for y_{bi} variables we have

$$\max_{r \in R} \left\{ \max_{b \in \mathcal{S}^{\mathcal{B}}} \left\{ -\beta_1 \cdot (\text{TSP cost for } b \text{ at } i) + \beta_2 \cdot \sum_{v \in b} \alpha_v + \beta_3 \right\} \right\}.$$

In both cases, we have to solve $|\mathcal{R}|$ maximum weight circuit problems.

References

1. Ahmadi, J., Ahmadi, R., Matuso, H., Tirupati, D.: Component fixture positioning/sequencing for printed circuit board assembly with concurrent operations. Oper. Res. **43**(3), 444–457 (1995)
2. Altinkemer, K., Kazaz, B., Köksalan, M., Moskowitz, H.: Optimization of printed circuit board manufacturing: integrated modeling and algorithms. Eur. J. Oper. Res. **124**(2), 409–421 (2000)
3. Applegate, D., Bixby, R., Chvatal, V., Cook, W.: CONCORDE TSP solver. Website http://www.tsp.gatech.edu/concorde.html (2009)
4. Ayob, M., Kendall, G.: A triple objective function with a Chebychev dynamic pick-and-place point specification approach to optimise the surface mount placement machine. Eur. J. Oper. Res. **164**(3), 609–626 (2005)
5. Ayob, M., Kendall, G.: A survey of surface mount device placement machine optimisation: machine classification. Eur. J. Oper. Res. **186**(3), 893–914 (2008)
6. Ball, M.O., Magazine, M.J.: Sequencing of insertions in printed circuit board assembly. Oper. Res. **36**(2), 192–201 (1988)
7. Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P.: Branch-and-price: column generation for solving huge integer programs. Oper. Res. **46**(3), 316–329 (1998)
8. Burke, E.K., Cowling, P.I., Keuthen, R.: Effective heuristic and metaheuristic approaches to optimize component placement in printed circuit board assembly. In: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 1 (2000)
9. Clarke, G., Wright, J.: Scheduling of vehicles from a central depot to a number of delivery points. Oper. Res. **12**(4), 568–581 (1964)
10. Crama, Y., van de Klundert, J., Spieksma, F.C.R.: Production planning problems in printed circuit board assembly. Discrete Appl. Math. **123**(1–3), 339–361 (2002)
11. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
12. Ellis, K.P., Kobza, J.E., Vittes, F.J.: Development of a placement time estimator function for a turret style surface mount placement machine. Robot. Comput. Integr. Manuf. **18**(3–4), 241–254 (2002)
13. Ellis, K.P., Vittes, F.J., Kobza, J.E.: Optimizing the performance of a surface mount placement machine. IEEE Trans. Electron. Packag. Manuf. [see also IEEE Transactions on Components, Packaging and Manufacturing Technology, Part C: Manufacturing] **24**(3), 160–170 (2001)
14. Feremans, C., Grigoriev, A.: Approximation schemes for the generalized geometric problems with geographic clustering. METEOR, Maastricht research school of Economics of TEchnology and ORganizations; University Library, Universiteit Maastricht (2004)
15. Fischetti, M., Salazar Gonzalez, J.J., Toth, P.: The symmetric generalized traveling salesman polytope. Networks **26**(2), 113–123 (1995)
16. Fischetti, M., Toth, P.: The generalized traveling salesman and orienteering problems. Comb. Optim. **12**, 609–662 (2002)
17. Franceschi, R.D., Fischetti, M., Toth, P.: A new ILP-based refinement heuristic for vehicle routing problems. Math. Program. **105**(2), 471–499 (2006)
18. Golden, B.L., Raghavan, S., Wasil, E.A.: The vehicle routing problem: latest advances and new challenges, vol. 43 of Operations Research/Computer Science Interfaces Series. Springer (2008)
19. Grunow, M., Günther, H.O., Schleusener, M., Yilmaz, I.O.: Operations planning for collect-and-place machines in PCB assembly. Comput. Ind. Eng. **47**(4), 409–429 (2004)
20. Gyorfi, J.S., Wu, C.H.: An efficient algorithm for placement sequence and feeder assignment problems with multiple placement-nozzles and independent link evaluation. IEEE Trans. Syst. Man Cybern.: Part A: Syst. Hum. **38**(2), 437 (2008)
21. Haberle, K.R., Graves, R.J.: Cycle time estimation for printed circuit board assemblies. IEEE Trans. Electron. Packag. Manuf. [see also IEEE Transactions on Components, Packaging and Manufacturing Technology, Part C: Manufacturing], **24**(3), 188–194 (2001)

22. Haimovich, M., Kan, A.H.G.R., Stougie, L.: Vehicle Routing: Methods and Studies. North-Holland, 1988, ch. Analysis of heuristics for vehicle routing problems, pp. 47–61
23. Hirvikorpi, M., Knuutila, T., Johnsson, M., Nevalainen, O.: A general approach to grouping of PCB assembly jobs. Int. J. Comput. Integr. Manuf. **18**(8), 710–720 (2005)
24. Ho, W., Ji, P.: Component scheduling for chip shooter machines: a hybrid genetic algorithm approach. Comput. Oper. Res. **30**(14), 2175–2189 (2003)
25. Ho, W., Ji, P., Dey, P.K.: Optimization of PCB component placements for the collect-and-place machines. Int. J. Adv. Manuf. Technol. **37**(7), 828–836 (2008)
26. ILOG Inc.: CPLEX linear optimizer and mixed integer optimizer v. 11.0. Website http://www.ilog.com (2009)
27. Ji, P., Wan, Y.F.: Planning for printed circuit board assembly: the state-of-the-art review. Int. J. Comput. Appl. Technol. **14**(4), 136–144 (2001)
28. Kazaz, B., Altınkemer, K.: Optimization of multi-feeder (depot) printed circuit board manufacturing with error guarantees. Eur. J. Oper. Res. **150**(2), 370–394 (2003)
29. Knuutila, T., Pyottiala, S., Nevalainen, O.S.: Minimizing the number of pickups on a multi-head placement machine. J. Oper. Res. Soc. **58**(1), 115 (2007)
30. Kulak, O., Yilmaz, I.O., Günther, H.O.: PCB assembly scheduling for collect-and-place machines using genetic algorithms. Int. J. Prod. Res. **45**(17), 3949–3969 (2007)
31. Kumar, R., Luo, Z.: Optimizing the operation sequence of a chip placement machine using TSP model. IEEE Trans. Electron. Packag. Manuf. **26**(1), 14–21 (2003)
32. Lapierre, S.D., Debargis, L., Soumis, F.: Balancing printed circuit board assembly line systems. Int. J. Prod. Res. **38**(16), 3899–3911 (2000)
33. Leu, M.C., Wong, H., Ji, Z.: Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm. J. Electron. Packag. **115**(4), 424 (1993)
34. Li, S., Hu, C., Tian, F.: Enhancing optimal feeder assignment of the multi-head surface mounting machine using genetic algorithms. Appl. Soft Comput. J. **8**(1), 522–529 (2008)
35. Parkhi, K.: Printed circuit board: world outlook. Tech. Rep. 844-F2, Visant Strategies, Inc. (2007)
36. Salonen, K., Smed, J., Johnsson, M., Nevalainen, O.: Grouping and sequencing PCB assembly jobs with minimum feeder setups. Robot. Comput. Integr. Manuf. **22**(4), 297–305 (2006)
37. Sarvanov, V.I., Doroshoko, N.N.: The approximate solution of the travelling salesman problem by a local search algorithm with scanning neighborhoods of factorial cardinality in cubic time. Softw.: Algorithms Programs **31**, 11–13 (1981)
38. Seth, A., Klabjan, D., Ferreira, P.M.: Analyses of advanced iterated tour partitioning heuristics for generalized vehicle routing problems. Tech. rep., Northwestern University. http://www.klabjan.dynresmanagement.com (2009)
39. SIPLACE-Americas. Siemens automation electronic assembly systems. Siemens' official website http://ea.automation.siemens.com (2009)
40. Smed, J., Johnsson, M., Johtela, T., Nevalainen, O.: Techniques and applications of production planning in electronics manufacturing systems. Tech. Rep. 320, Turku Centre for Computer Science (1999)
41. Su, C., Ho, L., Fu, H.: A novel tabu search approach to find the best placement sequence and magazine assignment in dynamic robotics assembly. Integr. Manuf. Syst. **9**(6), 366–376 (1998)
42. Tirpak, T.M., Nelson, P.C., Asmani, A.J.: Optimization of revolver head SMT machines using adaptive simulated annealing (ASA). In: Proceedings of the Twenty-Sixth IEEE/CPMT International Electronics Manufacturing Technology Symposium, pp. 214–220 (2000)
43. Toth, P., Vigo, D.: The Vehicle Routing Problem. Monographs on discrete mathematics and applications. Society for Industrial and Applied Mathematics (2002)
44. Wilhelm, W.E., Arambula, I., Choudhry, N.N.D.: Optimizing picking operations on dual-head placement machines. IEEE Trans. Autom. Sci. Eng. [see also IEEE Transactions on Robotics and Automation] **3**(1), 1–15 (2006)
45. Wilhelm, W.E., Choudhry, N.D., Damodaran, P.: A model to optimize placement operations on dual-head placement machines. Discrete Optim. **4**(2), 232–256 (2007)
46. Wilhelm, W.E., Tarmy, P.K.: Circuit card assembly on tandem turret-type placement machines. IIE Trans. **35**(7), 627–645 (2003)
47. Yilmaz, I., Günther, H.-O.: A group setup strategy for PCB assembly on a single automated placement machine. In: Haasis, H.-D., Kopfer, H., Schönberger, J. (eds.) Operations Research. Springer, Berlin, Heidelberg pp. 143–148 (2005)