Int J Adv Manuf Technol (2002) 20:692–700

The International Journal of
Advanced
Manufacturing
Technology

A Tabu-Enhanced Genetic Algorithm Approach to Agile Manufacturing

L. P. Khoo and M. Y. Loi

School of Mechanical and Production Engineering, Nanyang Technological University, Nanyang Avenue, Republic of Singapore

Globalisation and the advent in manufacturing technology have resulted in a more turbulent and rapidly changing market, where more and more unpredictable factors influence market opportunities. In order to maintain a steady share of the market and to survive in a competitive environment, it is necessary to respond rapidly to changes. Agile manufacturing is a manufacturing paradigm developed to meet the challenges which stem from an unpredictable global market. This work attempts to explore the possibility of solving an assembly line balancing problem using a novel tabu-enhanced genetic algorithm approach. An attempt is made to compare the qualities of the optimised solutions produced by genetic algorithms and tabu search. A case study on a tower computer assembly was used to validate the proposed approach. The details of the approach as well as a case study are presented.

Keywords: Agile manufacturing; Genetic algorithms; Tabu search

1. Introduction

Globalisation and the advent of manufacturing technology have resulted in a turbulent and rapidly changing market. In order to seize a steady market share and survive in a competitive environment, it is necessary to respond rapidly to changes [1–3]. Agile manufacturing is a manufacturing paradigm developed by the Iacocca Institute in 1991 to meet the challenges stemming from an unpredictable global market. In this respect, one of the ways to achieve agile manufacturing is to adopt a product modularity approach as proposed by He and Kusiak [4]. Such an approach enables the realisation of product variety and results in the standardisation and interchangeability of component parts. Product modularity is often realised through component swapping/sharing, fabricate-to-fit or sectional modularity [5]. Ulrich [6] further suggested that the implementation of product modularity is dependent on its product architecture. Standardisation allows certain components to be used for different products. This, in turn, allows the resources for product development and capital expenses to be amortised across a large number of units, and the possible exploitation of higher-volume and more efficient production technology that helps in lowering manufacturing cost. Furthermore, the lead time for the production of individual components that form the finished products can be shortened and more resources can be channelled to customise differentiating components [5]. This also implies that the assembly line for a product, which is based on the design concept of product modularity, can be split into two separate subassembly lines, namely a basic operation subassembly line and a variant operation subassembly line [4]. As a result, for a variety of products, all of them would first go through the same basic operation subassembly line. Subsequently, they would be assembled in the variant operation subassembly line according to the respective desired features or functional requirements. As the product mix and options increase, the complexity of the variant operation subassembly line will increase correspondingly. As a result, scheduling of the variant operation subassembly line becomes a critical factor in determining the efficiency of the entire assembly line. By focusing on line-balancing and optimising heuristic scheduling of the variant operation subassembly line, the objective of achieving agility in manufacturing can possibly be realised.

He and Kusiak [4] employed tabu search (TS) for the assembly line design problem. Two different heuristic scheduling rules, namely the shortest total processing time (STPT) [7,8] and the shortest adjusted processing time (SAPT) [9] were experimented with. In their work, the initial solutions required by TS were generated using two different approaches, namely a randomised approach and a so-called sequential assignment procedure (SAP). Essentially, SAP is a line balancing technique aimed at maximising the equality of the average processing time among all the stations in an assembly line [4,10]. Comparisons were subsequently made between the two heuristic scheduling rules. The emphasis on the initial solution is necessary in TS as it plays a significant role in deciding the quality of the final optimised solution [11–13]. Computational results showed that for complex problems, there is no significant difference in the results generated by both rules.

Correspondence and offprint requests to: Dr L. P. Khoo, School of Mechanical and Production Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798.
E-mail: mlpkhoo@ntu.edu.sg

However, SAPT performed significantly better for simple problems, that is, an assembly line with few stations and a small variety of products.

The configuration of an assembly line is also known as an assembly line balancing problem, which is basically NP-hard. Artificial intelligence techniques such as simulated annealing, neural networks, genetic algorithms (GAs) or TS can be used to derive a near optimal solution [14,15]. This paper presents work that leads to the development of a TS enhanced GA approach to agile manufacturing. An attempt is made to compare the qualities of the optimised solutions produced by GAs and TS. A case study on a tower computer assembly was used to validate the proposed approach. The details of the approach as well as a case study are presented.

2. Genetic Algorithms and Tabu Search

2.1 Genetic Algorithms

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine the survival-of-the-fittest string structures or chromosomes with a structured yet randomised information exchange to form a search algorithm with some of the innovative flair of a human search. This implies that GAs use random choices as a tool to guide a highly exploitative search. Through randomisation, they are able to exploit the search space effectively with improved performance [16,17].

A special property of GAs is that they maintain a population of chromosomes or potential solutions to be processed, whereas all other methods process a single point in the search space. By working on a population of chromosomes instead of a single point, GAs ensure that the probability of reaching a false peak is reduced. Many search techniques require auxiliary information in order to work properly. For example, gradient techniques require derivatives, which are determined analytically or numerically, in order to obtain a near optimal value. The heavy reliance on such information or restrictive assumptions about the search space limits the capability of these methods when the information is not available or difficult to obtain. By contrast, GAs are blind. To perform an effective search for a better chromosome, they only require pay off value, which is also known as the objective function value or fitness value, associated with an individual chromosome. This makes GAs a more canonical method than many search schemes. In GAs, genetic operators are applied to successive chromosomes to create new chromosomes. These operators are simple and involve nothing more complex than random number generation, string copying, and partial string exchanging. Despite their simplicity, the results obtained are impressive. In comparison to classical methods, GAs belong to a class of general purpose search methods and are domain independent. They employ probabilistic transition rules to guide their search. GAs are different from random algorithms as the elements of directed and stochastic search are inherent within them. They simply use random choice as a tool to guide a search toward regions of the search space having a probable improvement. Because of this, GAs are also more robust than many existing directed search methods.

2.2 Tabu Search

Tabu search (TS) is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. The heuristics may be a high-level procedure or may embody some simple descriptions of available moves for transforming one solution into another, together with an associated evaluation rule [11–13]. TS makes use of a memory structure to keep track of the search history. This memory structure is of two types, namely short-term memory and long-term memory. In this work, only short-term memory is considered. Short-term memory constitutes a form of exploration that seeks to make the best move to attain the highest value. In addition, the candidate solutions satisfy imposed constraints such as assembly or movement constraints. Short-term memory keeps track of the attributes of candidate solutions and possesses the ability of setting the status of *a move* as tabu. It has three main components, namely candidate list strategy also known as the initial solution, tabu tenure, and aspiration criteria.

Candidate List Strategy or Initial Solution

In situations where the neighbourhood is large or its elements are expensive to evaluate, the candidate list strategy is necessary in order to restrict the number of solutions examined in an iteration. It provides a list of good moves that are feasible. This is analogous to selecting a good solution from the neighbourhood for further evaluation.

Tabu Tenure

A tabu list is commonly used to set the tabu status of an attribute during the search process. In this list, tabu tenure is the parameter that monitors and controls the number of iterations so that a move is kept tabu-active. Tabu tenure can vary for different combinations of attributes. It can also change over different intervals of time or stages of a search. A dynamic and robust form of search can be attained through the change.

Mathematically, the tabu-active status of an attribute, A, can be expressed as follows.

$$\text{TabuEnd}\ (A) = \text{TabuStart}\ (A) + \text{TabuTenure}\ (A)$$

Alternatively,

$$\text{Current Iteration} \leq \text{TabuStart Iteration}\ (A) + \text{TabuTenure}\ (A)$$

Aspiration Criteria

During the search process, some moves may be labelled as tabu, and may lead to an inability to obtain good solutions, as restricted moves cannot be performed. In order to initiate the move, its tabu-active status has to be overridden using the aspiration criteria.

Table 1. Methods for generating initial solutions.

	Random	SAP	GAs
1	Unstructured	Unstructured	Structured search
2	No comparison	No comparison	Comparison allowed
3	Feasibility manually checked	Feasibility manually checked	Feasibility checked automatically
4	Heuristic based	Systematic	Heuristic based

3. The Approach

3.1 Integration of Genetic Algorithms and Tabu Search – an Overview

Tabu search (TS) involves the use of a move to perform a search. The evaluated move is the best move selected out of several moves from a candidate list per iteration. Normally, a candidate list consists of five or more recorded moves. It plays the role of identifying the best move among all candidate solutions through evaluating certain attributes. The main idea is to have a good comparison among the moves so as to select the best. TS requires a good initial solution to reduce the search time and improve the quality of the solution [4]. The integration of GAs with TS would allow the near optimal solution generated by GAs to be further exploited by TS. As already mentioned, He and Kusiak [4] have successfully shown that a sequential assignment procedure (SAP) provides a good initial solution for TS compared to that generated randomly. Table 1 shows a comparison of the three techniques. Genetic algorithms allow comparison to be made among the chromosomes or the solutions generated. This property ensures that good initial solutions for TS are generated.

The TS memory structure is employed to keep track of the historical records of the search. The type of memory structure used is dependent upon the application domain. For example, long-term memory structure is effective for global exploration, whereas, short-term memory is good for local exploitation. The application of memory structure is advantageous, as the search for near optimal solutions can be better controlled. It enables a near optimal solution to be obtained relatively fast and easily. However, if the search space is extremely large and noisy, the use of a memory structure requires considerable time for exploration in order to decide a "single move". In addition, the implementation can be complicated. A more reasonable approach is to initiate a quick exploration through the search space using GAs and employ TS to explore certain regions of interest, decided by GAs.

Table 2. Classification of various meta-heuristic algorithms.

Meta-heuristic	Classification
Genetic algorithms	Memoryless/ random sampling/ population based
Tabu search	Adaptive memory/ systematic neighbourhood search / one solution to another

Glover [13] classified a few well-known algorithms based on the following criteria.

1. The use of adaptive memory.
2. The kind of neighbourhood exploration used.
3. The number of current solutions carried forward per iteration.

From Table 2, it can be seen that the search mechanism employed by GAs and TS is entirely different. This also shows that their integration would enable them to complement each other. More specifically, owing to the memoryless property of GAs, the optimal solution is achieved by chance rather than through any structural procedure. This implies that the search is blind. In the worst case scenario, GAs may lead to a search cycle with no obvious solution. If the structured memory of TS is used to complement the GAs search after a suitable number of generations, the chances of obtaining a global optimum may be improved significantly.

3.2 Implementation

Genetic Algorithms

Briefly, the implementation of GAs involves the following generic steps.

Initialisation. An initial list of candidate solutions or chromosomes are generated randomly. The feasibility of these chromosomes is checked.

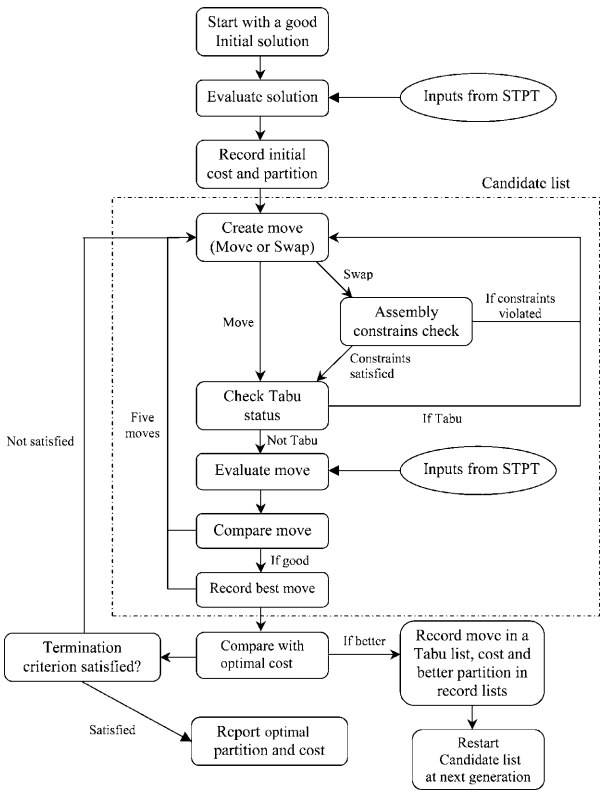

Fig. 1. Working flowchart using tabu search.

Table 3. Assembly constraints for computer tower assembly.

Constraints	Type of rules/constraints	Job index A	Job index B
1	Operation A before operation B	0–1	3–12
2		7–8	9
3		10–11	12
4		13–16	17–28
5		20–21	22
6		20–21	27–28

Notes:
1. Table 4 depicts a table showing configuration type vs. order number.
2. Table 5 shows the details of assembly operations, job index and assembly time.

Fitness evaluation. For ensuring the feasibility of chromosome candidate solutions, the repair algorithm or the penalty approach is frequently used. In this work, the penalty approach was adopted as feasible chromosomes can retain certain characteristic of its parent strings without the loss of any vital information associated with them. Feasible chromosomes are then decoded and subjected to a fitness evaluation. In this case, a feasible chromosome is one that satisfies all the constraints and its penalty value is zero. The fitness function which is also known as the cost function proposed by He and Kusiak [4] is used here.

$$BC[S(p)]=\alpha I^{S(p)} + \beta W^{S(p)}$$

where,

1. Both α (unit idling time cost) and β (unit inventory holding cost) are taken as 1.
2. Only the waiting time *within* and *between* stations, together with the idling time between stations are considered.
3. All waiting and idling time associated with the first product are ignored.

The fitness function is dependent upon the inputs from STPT, which predetermines the ordering sequence of the products to be processed. STPT schedules the products such that the one with the least total operation time will be assembled first.

Genetic operations. Pairs of chromosomes are selected randomly using the roulette wheel approach for crossover and mutation operations [16] to reproduce a new population of chromosomes. The *order crossover (OX)* method from path representation is adopted in this work. The details of the OX method are reported in the work of Michalewicz [17] and will not be described here. Upon completion of the genetic operations, the new population of chromosomes is subject to the same cycle of treatment, that is, fitness evaluation and genetic operations.

Tabu Search

Figure 1 shows a flowchart of the TS implementation. It comprises 6 essential steps.

Step 1. Obtain a good initial solution using GAs. The best feasible solution obtained by GAs is used here as the input to the TS.

Step 2. Derive a candidate list of solutions and perform cost evaluation.

(i) In each TS generation, a list of 5 candidate solutions is gathered by performing a random single-move with respect to the current initial/optimal solution.

(ii) The 5 candidate solutions are then compared and the best solution is, in turn, compared with the current optimal solution.

(iii) If the new candidate solution is better than the current optimal solution, a TS move is initiated and the search for an optimal solution proceeds to the next generation. Otherwise, a new candidate list will be randomly created and Step 2 (ii) is repeated.

Table 4. Assembly jobs involved in different computer tower configuration.

Type	Assembly jobs (represented by job index)	Order
1	0, 2, 3, 4, 7, 9, 10, 14, 17, 18, 19, 20, 22, 23, 24, 28	7
2	0, 2, 4, 6, 8, 10, 12, 14, 17, 19, 20, 23, 24, 26, 28	6
3	0, 2, 3, 4, 5, 6, 7, 10, 13, 17, 18, 21, 22, 23, 24, 25, 26, 27	8
4	0, 2, 4, 5, 6, 8,9, 10, 12, 13, 17, 19, 21, 22, 23, 24, 25, 26, 27	4
5	0, 2, 3, 5, 6, 7, 9, 10, 13, 17, 18, 19, 21, 22, 23, 25, 26, 27	3
6	0, 2, 5, 8, 10, 12, 14, 17, 20, 23, 25, 28	9
7	1, 2, 3, 4, 7, 11, 16, 17, 18, 19, 20, 23, 24, 26, 28	5
8	1, 2, 4, 6, 8,9, 11, 12, 15, 17, 19, 21, 22, 23, 24, 26, 27	7
9	1, 2, 3, 4, 5, 6, 7, 9, 11, 15, 17, 18, 21, 22, 23, 24, 25, 26, 27	6
10	1, 2, 4, 5, 6, 8, 11, 12, 15, 17, 19, 21, 22, 23, 24, 25, 26, 27	10
11	1, 2, 3, 5, 6, 7, 11, 16, 17, 18, 19, 20, 23, 25, 28	2
12	1, 2, 5, 8, 9, 11, 12, 16, 17, 20, 22, 23, 25, 28	4

Note: Refer to Table 5 for job index.

Table 5. Description of job operation denoted by respective job index.

Job index	Operations	Time
0	Assemble socket type motherboard together with its corresponding tower casing type	4
1	Assemble slot type motherboard together with its corresponding tower casing type	4
2	Insert SD ram onto the ram slot on its respective motherboard	5
3	Insert sound card onto the sound card slot on its respective motherboard	6
4	Insert modem card onto the modem card slot on its respective motherboard	6
5	Insert ethernet card onto the ethernet card slot on its respective motherboard	6
6	Insert scanner card onto the scanner card slot on its respective motherboard	6
7	Insert PCI type VGA card onto the PCI type VGA slot on its respective motherboard	5
8	Insert AGP type VGA card onto the AGP type VGA slot on its respective motherboard	5
9	Insert VGA chipset cooler onto the corresponding VGA card	4
10	Insert socket type CPU onto the socket type CPU slot on its respective motherboard	3
11	Insert slot type CPU onto the slot type CPU slot on its respective motherboard	5
12	Insert CPU cooler onto the corresponding CPU	3
13	Set up vertical tower casing (big) for assembly	5
14	Set up vertical tower casing (small) for assembly	4
15	Set up horizontal tower casing (big) for assembly	5
16	Set up horizontal tower casing (small) for assembly	4
17	Assemble CD rom onto its respective tower casing	6
18	Assemble CD writer onto its respective tower casing	6
19	Assemble DVD rom onto its respective tower casing	6
20	Assemble IDE type hard disk onto its respective tower casing	7
21	Assemble SCSI type hard disk onto its respective tower casing	7
22	Assemble hard disk cooler onto its respective hard disk	5
23	Assemble floppy drive onto its respective tower casing	6
24	Assemble zip 250 drive onto its respective tower casing	5
25	Assemble Jaz 2GB drive onto its respective tower casing	5
26	Assemble magnetic optical drive onto its respective tower casing	5
27	Assemble power box (big) onto its respective tower casing	8
28	Assemble power box (small) onto its respective tower casing	7

(iv) The process will stop once the termination criterion is met.

Step 3. Create a TS move. Typically, there are two different types of TS move, namely a move between two stations and a swap between two operations. The type of TS move is selected randomly:

(i) Move. The move between stations is confined only to operations at the ends of each station. Mathematically,

$$Move = [X,\ S_{from},\ S_{to}]$$

where X is the operation to be moved, S_{from} is the station that the operation previously assigned, and S_{to} is the station that the operation currently assigned.

(ii) Swap. For every swap made, the check for assembly constraint is performed to ensure the feasibility of the swap. An invalid swap is regenerated. Mathematically,

$$Swap = [X,\ fro/to,\ to/fro]$$

where X is the operation to be swapped, *fro* represents the logic "0", *to* represents the logic "1", [x, 0, 1] represents an operation that swaps x to the front.

Step 4. Tabu status of a move. When a move in a generation becomes tabu, a notation, gen_{tabu}, is assigned accordingly. The tabu status was checked using a tabu list by comparing the current generation with respect to the sum of gen_{tabu} and the corresponding tabu tenure (Section 2.2).

Step 5. Cost function. The same GA cost function is used to evaluate the solution before initiating the move. As already mentioned, it is dependent on the inputs from STPT, which predetermined the ordering sequence of the products to be assembled. A comparison with the previous solution is then made. If a better solution is found, the cost is registered and a move is initiated.

Step 6. Termination criterion. Steps 2–5 are repeated until the termination criterion is met. In this work, when a candidate solution repeats itself seven times in a single generation, the search is terminated. This implies that the search process is terminated when no better solution is found after 35 invalid moves.

4. A Case Study

A computer tower assembly was used to evaluate the effectiveness of the proposed approach. Typically, the assembly process can be quite involved, as different assembly operations must be performed for various port interfaces, for example. Gener-

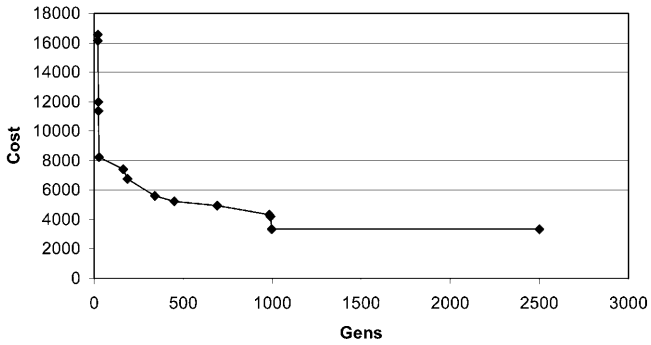

Fig. 2. Search result for computer tower assembly using GAs.

ally, more flexible product architecture has more scope for a good assembly design. The situation is further complicated by mixed configurations for different product variants that are in demand.

Product variants result from assembling different main assemblies such as motherboards and casings according to customer needs. Analysis shows that two assembly operations, namely screwing and inserting, are frequently used. The screwing operation is performed when a part such as a motherboard must to be assembled onto the tower casing. The inserting operation is commonly performed when, for example, a card is to be placed onto the motherboard. Further analysis shows that the computer tower assembly comprises two groups of assembly operations (Tables 3 to 5).

Group I. All components relevant to the motherboard (Job Indices 0~12).

Group II. All components relevant to the tower casing (Job Indices 13~28).

Assembly rules for Group I operations:

1. Motherboard/parts. All the parts to be assembled onto the motherboard should be assembled after the motherboard has been installed onto the tower casing.
2. VGA card/cooler. The VGA chipset cooler can only be assembled onto the VGA card after the card has been installed.
3. CPU/cooler: The CPU cooler can only be assembled onto the CPU after the CPU has been installed.

Assembly rules for Group II operations:

4. Tower casing/parts: All the parts to be assembled onto the tower casing should be assembled only after the tower casing is ready.
5. Hard disk/cooler. The hard disk cooler can only be assembled onto the hard disk after the disk has been installed.
6. Hard disk/power box. The power box should be assembled after the hard disk has been installed.

Table 3 summarises the relationship between two assembly operations according to the above assembly rules/constraints.

5. Results and Discussions

5.1 Results Generated by Genetic Algorithms

Figure 2 shows the results generated by genetic algorithms. Genetic algorithms move swiftly to look for near optimal solutions, that is, minimum cost. This is characterised by a rapid decrease in cost in the first hundred generations, after which, the search for near optimal solutions proceeded at a slower pace. The search process is non-deterministic, which clearly indicates the randomness of GAs. Table 6 shows the near optimal results obtained using 10 different random seed numbers. In this case study, a total of 29 operations were assigned to six work stations in the order of "5, 5, 5, 5, 5, 4". Each search involved 2500 generations. Solution SS10, which is the best among the 10 trial runs, is found in the 997th generation with an optimal cost of 3326. The optimal configuration for the various work stations, stations 1 to 6, is shown in columns 2 to 7 (S1–S6). For example, operations 13, 16, 15, 14, and 1, and operations 0, 11, 21, 19, and 8 are assigned to stations 1 and 2 of solution SS10, respectively.

5.2 Results Generated by Tabu Search

(i) Poor and Good Initial Solutions

Figures 3 and 4 show the TS results based on a poor initial solution (PIS) and a good initial solution (GIS), respectively. In this case, the PIS was generated randomly whereas the GIS was essentially solution SS1 in Table 6. Both solutions were

Table 6. Computational results based on GAs using 10 different seeds.

	S1	S2	S3	S4	S5	S6	Generations	Cost
SS1	16 15 14 0 13	1 11 21 7 20	2 22 27 8 23	17 19 24 10 6	3 18 28 25 26	9 5 4 12	2157	4092
SS2	14 13 16 15 21	0 1 11 20 18	7 2 3 4 28	23 27 25 10 17	22 12 8 5 6	9 24 26 19	1706	3948
SS3	16 14 0 15 13	1 8 11 10 4	20 19 21 7 17	23 6 12 5 9	27 28 25 3 22	2 26 18 24	2403	4108
SS4	15 14 16 0 1	10 7 8 13 19	4 11 9 18 20	23 21 24 22 12	27 5 17 25 26	6 28 2 3	1764	4118
SS5	15 16 14 13 1	0 20 10 18 19	11 5 21 17 28	2 4 22 3 8	12 23 7 9 24	27 26 6 25	465	4018
SS6	16 13 0 15 1	6 14 4 21 20	26 8 11 17 27	23 28 22 10 2	19 18 25 7 5	3 12 24 9	337	4139
SS7	13 15 16 14 26	1 19 0 25 20	3 8 21 2 4	17 18 7 10 24	27 23 28 11 22	9 5 6 12	1818	3776
SS8	14 16 15 1 0	11 13 26 6 20	4 8 25 21 23	17 18 7 9 3	19 5 10 28 27	22 12 2 24	1682	4332
SS9	14 13 16 15 26	1 0 18 17 11	10 25 20 23 24	4 21 7 2 8	22 3 5 12 27	28 6 9 19	385	4240
SS10	13 16 15 14 1	0 11 21 19 8	10 20 7 17 24	23 12 5 3 28	9 27 2 4 25	26 22 18 6	997	3326

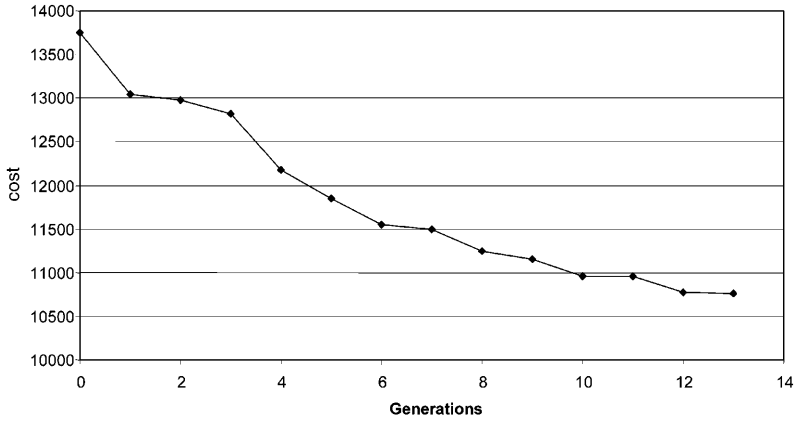

Fig. 3. TS results derived using a poor initial solution.

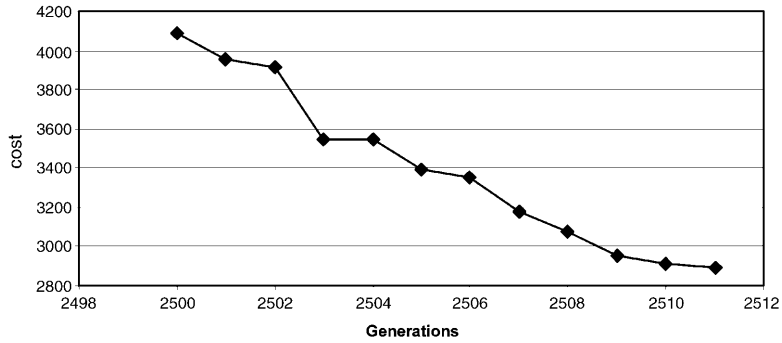

Fig. 4. TS results derived using a good initial solution computed by GAs (Table 8, Seed 3).

further optimised by a TS using 10 different random seed numbers. The results are shown in Tables 7 and 8 respectively. The best results obtained are shown in Figs 3 and 4. The outcome was expected, as it shows that the quality of the near optimal solution is dependent upon the initial solution used. Furthermore, in terms of the duration of search, GIS appears to converge faster than to PIS. This is logical as an inferior initial solution has more room for improvement.

(ii) Further Optimisation of GAs Results Using TS

The results shown in Fig. 4 can also be used to compare the performance between GAs and the TS enhanced approach. In this case, using TS, a further 29% improvement in cost can be obtained, compared to that of GAs. From Table 8, it can be seen that the improvement is fairly consistent for different random seed numbers. This also demonstrates the robustness of the TS enhanced approach. Figure 5 shows the relative improvement attained by different random seed numbers. Further investigations were carried out to verify the above observation using the rest of the near optimal solutions (SS2 – SS10) generated by GAs. The results obtained are summarised in Fig. 6. It is observed that TS, in general, improves the GA results by 900 on average for cost. In the case of SS10, which is the best result generated by GAs, the TS enhanced approach manages to reduce the cost from 3326 to 2534. Table 9

Table 7. TS results derived using a poor initial solution.

	S1	S2	S3	S4	S5	S6	Cost
PIS	0 1 2 3 4	5 6 7 8 9	10 11 12 13 14	15 16 17 18 19	20 21 22 23 24	25 26 27 28	13757
Seed 1	1 0 2	4 3 5 6 8	7 11 10 9 12	14 13 15 16 17 18 19	21 20 22 23	24 25 26 28 27	10808
Seed 2	0 1 2	4 3 5 6 8 7 9	11 10 12 13 15 14	16 17 18 19 20 21	22 23	24 26 25 27 28	11530
Seed 3	0 1 2	3 4 5 6 8 7	9 10 11 12 13 14 16	15 17 18 19 21	20 22 24	23 25 26 28 27	10763
Seed 4	0 1 2	3 4 5 6 8 7	9 10 11 12 13 14 16	15 17 18 19 21	20 22 24	23 25 26 28 27	10778
Seed 5	0 1 2 3	4 5 6 8 7 9	11 10 12 13 15	14 16 17 18 19 21	20 22 24 26	23 25 28 27	10986
Seed 6	0 1 2	4 3 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 21	20 22 24 23 26	25 28 27	11000
Seed 7	0 1 2	3 4 5 6 7 8 9	11 10 12 13 15 16	14 17 19 18 21 20	22 24	23 26 25 28 27	11300
Seed 8	0 1 2	4 3 5 6 8 7	9 10 11 12 13 14 15	16 17 18 19 20 21	22 23	24 25 26 28 27	11007
Seed 9	0 1 4 2	3 5 6 8 7	9 10 11 12 13 14 15 16	17 18 19 21	20 23	22 24 25 27 26 28	11864
Seed 10	0 1 2	3 4 5 6 8 7	9 10 11 12	13 14 15 16 17 18 19 21	20 23 22 24	25 26 28 27	11071

Table 8. TS results derived using a good initial solution computed by GAs (Table 6, SS1).

	S1	S2	S3	S4	S5	S6	Cost
GIS (SS1)	16 15 14 0 13	1 11 21 7 20	2 22 27 8 23	17 19 24 10 6	3 18 28 25 26	9 5 4 12	4092
Seed 1	16 15 14 0 13	1 11 21 7 20 2	22 27 8 23	17 19 24 6 10	3 18 28 25 9 26	5 4 12	3702
Seed 2	16 15 14 0 13	1 11 21 7 20 2	22 27 8 23	17 19 24 6 10	3 18 28 25 9 26	5 4 12	3536
Seed 3	15 16 14 0 13	1 11 21 7	20 2 22 27	8 17 23 19 10 24	3 18 6 28 9 25	26 5 4 12	2891
Seed 4	16 15 14 0 13	1 11 7 21 20	2 22 27 8 23	17 19 24 10 6	3 18 25 28 9 26	5 4 12	3730
Seed 5	15 16 14 0 13	1 11 21 20 7	2 22 27 8 23	17 19 24 10	3 6 18 28 25	26 9 5 4 12	3051
Seed 6	15 16 14 0 13	1 11 20 21 7	22 2 27	8 17 23 19 24	10 6 3 18 28	25 26 4 9 5 12	2911
Seed 7	16 14 0 15 13	1 11 21 20 7	2 22 27 8 23	17 19 24 10	6 3 18 28 25	26 9 5 4 12	3467
Seed 8	16 15 14 0 13	1 11 21 7 20 2	22 27 8 23	17 19 24 10 6	3 18 28 25 9	26 5 4 12	3717
Seed 9	16 15 14 0 13	1 11 21 7 2	20 22 27 8 23	17 19 24 10 6	3 18 28 25 26	9 5 4 12	3933
Seed 10	16 15 14 0 13	1 11 21 2 7	20 22 27 8 23	17 19 24 10 6	3 18 28 25 9 26	5 4 12	3858

Table 9. Ordered and number of operations to be assigned to each station.

	Optimum assignment of operations	Generation	Cost
GA (SS10)	13 16 15 14 1 | 0 11 21 19 8 | 10 20 7 17 24 | 23 12 5 3 28 | 9 27 2 4 25 | 26 22 18 6	997	3326
GA/TS	13 16 15 14 1 | 0 11 21 19 8 | 10 20 7 17 24 23 12 5 | 28 3 27 | 2 9 25 4 | 26 22 18 6	1005	2534

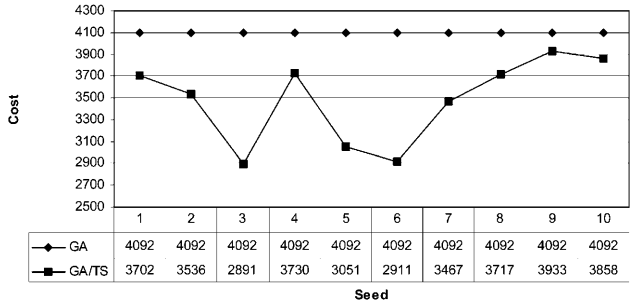

Fig. 5. A comparison between GA and GA/TS using SS1.

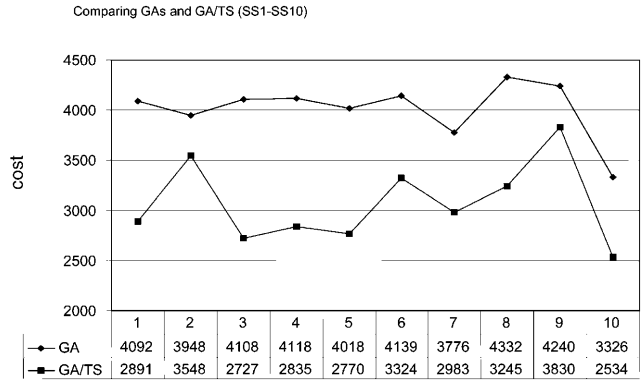

Fig. 6. Comparison of GAs and GA/TS (SS1–SS10).

summarises the operation sequences obtained using GAs and the TS enhanced approach. From the table, it can be deduced that all the assembly constraints have been satisfied. For example, Table 10 (*a*) shows that stations 1 and 2 comprise operations 13, 16, 15, 14 and 1, and operations 0, 11, 21, 19, and 8, respectively. It is noted that operations 13–16 refer to the type of casing to be used, whereas operations 0–1 concern

Table 10. (*a*) Operations assigned to the respective stations. (*b*) The respective operations to be performed on product type 1 in each station.

{*a*)	
Station 1	13 16 15 14 1
Station 2	0 11 21 19 8
Station 3	10 20 7 17 24
Station 4	23 12 5 3 28
Station 5	9 27 2 4 25
Station 6	26 22 18 6
(*b*)	
Station 1	14
Station 2	0 19
Station 3	10 20 7 17 24
Station 4	23 3 28
Station 5	9 2 4
Station 6	22 18

the assembly of the motherboard onto the casings, as indicated in Table 5. This shows that the assembly process starts with the setting up of a casing, and then follows by assembling the motherboard. With these two main components assembled, subsequent assembly operations simply involve securing of parts either into the casing or onto the motherboard at other stations. Further analysis reveals that none of the assembly constraints listed in Section 4 has been violated. This implies that the assembly sequence obtained using the TS enhanced GA approach is feasible. Table 10(*b*) shows an example of a typical product (type 1) that can be assembled using the configuration established in this work.

6. Conclusion

The work yields a novel approach that combines the strengths of GAs and TS. A case study on the assembly of a computer tower assembly was used to ascertain the performance of the

proposed approach. The assembly problem was first solved using GAs with 10 different random number seeds. It has been shown that GAs were able to generate near optimal solutions after several hundred generations. The importance of the initial solution to TS was then established. This is obtained by using one of the near optimal solutions obtained by the GAs as a good initial solution, and a randomly generated one as a poor initial solution. From a comparative study, it appears that a good initial solution is crucial for an effective tabu search. The 10 near optimal solutions obtained by GAs were then evaluated using TS. It has been ascertained that the results obtained were superior to those of GAs. This also demonstrates the robustness of the TS enhanced GA approach. The assembly sequence suggested by the proposed approach was then investigated. It is clear that the configuration derived is for a typical product (type 1) that can be assembled using the configuration.

References

1. H. Sharifi and Z. Dann, "Agile manufacturing – a structured perspective", Responsiveness in Manufacturing (Digest No. 1998/213), IEE, pp. 5/1–5/4, 1998.
2. P. Gould, "What is Agility", Manufacturing Engineer, 76(1), pp. 28–31, 1997.
3. P. T. Kidd, "Agile manufacturing: a strategy for the 21st century". IEE Colloquium on Agile Manufacturing. Pp. 1/1–1/6, 1996.
4. W. D. He and A. Kusiak, "Design of assembly system for modular products", IEEE Transactions on Robotics and Automation, 13(5), pp. 646–655, 1997.
5. K. Ulrich and K. Tung, "Fundamental of product modularity", in A. Sharon, R. Behun, F. Prinz, and L. Young (eds), Issues in Design Manufacture/ Integration, ASME, pp. 73–79, 1991.
6. K. Ulrich, "The role of product architecture in the manufacturing firms", Research Policy, 24(3), pp. 419–440, 1995.
7. S. French, Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop", Ellis Horwood, 1982.
8. K. R. Baker, Introduction to Sequencing and Scheduling", Wiley, 1974.
9. C. Rajendran, "Heuristic algorithm for scheduling in a flowshop to minimize total flowtime", International Journal of Production Economics, 29(1), pp. 65–73, 1993.
10. W. Chow, Assembly Line Design: Methodology and Applications, Marcel Dekker, 1990.
11. F. Glover, "Tabu search: a tutorial", Interfaces, 20(4), pp. 74–94, 1990.
12. J. A. Bland and G. P. Dawson, "Tabu search and design optimization", IEEE Computer-Aided Design, 23, pp. 195–201, 1991.
13. F. Glover, Tabu Search, Kluwer, 1997.
14. S. Ghosh and R. J. Gagnon, "A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems", International Journal of Production Research, 27(4), pp. 637–670, 1989.
15. F. Glover and H. J. Greenberg, " New approaches for heuristic search: a bilateral linkage with artificial intelligence", European Journal of Operational Research, 39(1), pp. 119–130, 1989.
16. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
17. Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 2nd edn, Springer-Verlag, 1994.