A study on meta-heuristics algorithms for solving the intelligent turret assembly system

Tan, Hock Woo

2008

Tan, H. W. (2008). A study on meta-heuristics algorithms for solving the intelligent turret assembly system. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2486>

<https://doi.org/10.32657/10356/2486>

Nanyang Technological University

Downloaded on 26 Apr 2023 23:09:59 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**A STUDY ON META-HEURISTICS
ALGORITHMS FOR SOLVING THE
INTELLIGENT TURRET ASSEMBLY SYSTEM**

**TAN HOCK WOO
SCHOOL OF COMPUTER ENGINEERING
2008**

A Study on Meta-Heuristics Algorithms for Solving the Intelligent Turret Assembly System

Tan Hock Woo

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of Engineering

2008

Abstract

The *Intelligent Turret Assembly System (ITAS)* is a modular, re-configurable system that was researched by *SIMTech* for a flexible assembly system with a small footprint.

Due to the flexibility of the system, the selection of a configuration to maximise its efficiency is often of polynomial time. The aim of the project is to select a suitable algorithm that will aid in the configuration of the *ITAS* and other general classes of optimisation problems.

One of the *forces* may need to wait for the other process to finish in order to avoid collisions between *forces* within the *ITAS*. However, it was found that the ratio of the waiting time with respect to the travelling time decreases with the increases in the number of turrets. When the waiting time component is ignored, the *ITAS* approximates the *Quadratic Assignment Problem (QAP)*.

These were investigated for their suitability to solve the *ITAS* problem: Genetic Algorithm (two implementations, *GA I* and *GA II*), Tabu Search (*TS*), Simulated Annealing (*SA*) and Genetic Algorithm with Ant Colony Optimisation (two hybrids, *ACO I* and *ACO II*). Simulations involving assembly plans for 3 and 4 turrets showed that *SA* and *TS* performed better for cases with 4 turrets while *GA II* and *ACO I* performed better for cases with 3 turrets.

ACO II is a new hybrid algorithm which used a crossover operator based on selection of genes using pheromone information from *ACO*. Overall, this new algorithm was found to perform better for the *ITAS* problem compared to other conventional methods. The improvement in performance was attributed to the use of pheromone information to intensify the *GA* search into station-tool assignments likely to be better, and the algorithm's ability to preserve interdependency relations between station-tool assignments of the parent chromosomes.

Acknowledgement

I would like to thank Assoc. Prof Lee Keok Kee, who has kindly agreed to be my supervisor for the project and for being a very patient and understanding supervisor throughout the project.

I would like to thank Asst. Prof Alex Tay for agreeing to be the co-supervisor for the project. Asst. Prof Tay has also provided valuable advice and helped source for interesting operations research related problems for the project.

I would like to thank Mr. Chung Meng Hee who was the principal researcher for the *ITAS*. It was with Mr. Chung's advice and encouragement that led to my undertaking of this project.

I would also like to thank the staff of Computer Engineering Lab for providing logistics support for the project.

Table of Content

Abstract.....	1
Acknowledgement	v
Table of Content.....	vi
Table of Illustrations.....	ix
List of Symbols	xi
1 Introduction.....	1
1.1 Terminology.....	1
1.2 The Intelligent Turret Assembly System	1
1.3 Uniqueness of the ITAS Problem	7
1.4 Objective	7
1.5 Thesis Outline	8
2 Optimisation	9
2.1 Definition of an Optimisation Problem.....	9
2.2 Estimation of Optimal Input Parameters.....	10
2.3 Search Space Complexity	11
2.4 Neighbourhood of a Solution.....	12
2.5 Search Strategy, Intensification and Diversification	14
2.6 Problems in Operations Research	15
3 Meta-heuristic Algorithms	18
3.1 Genetic Algorithms.....	18
3.2 Simulated Annealing.....	25
3.3 Ant Colony Optimisation.....	28
3.4 Tabu Search	31

3.5	Search Strategies in Meta-heuristic Algorithms	34
3.6	JSSP Experiment.....	36
3.7	Summary	39
4	Domain Knowledge.....	41
4.1	Performance Measures.....	41
4.2	Fitness Landscape Analysis	44
4.3	Observation on Placement of Unique Tools	50
4.4	Discussions	50
5	Genetic Algorithm for <i>ITAS</i>.....	51
5.1	Implementation I.....	51
5.2	Implementation II.....	55
5.3	Production Time versus Mutation Rate	57
5.4	Discussion.....	60
5.5	Conclusion	64
6	Simulated Annealing for <i>ITAS</i>.....	65
6.1	Implementation	65
6.2	Production Time versus Cooling Factor	66
6.3	Production Time versus Number of Fitness Evaluation	67
6.4	Effect of Change in Encoding Format on Simulated Annealing	69
6.5	Conclusion	70
7	Tabu Search for <i>ITAS</i>.....	72
7.1	Implementation	72
7.2	Tabu List and Tenure	72
7.3	Diversification Strategies.....	73
7.4	Tabu Attribute.....	74

7.5	Experiments	75
7.6	Conclusion	80
8	Ant Colony Optimisation	81
8.1	Pheromone	81
8.2	Diversification using Blind Ants	82
8.3	Construction of Trails	83
8.4	Experiments	84
8.5	Hybrid <i>ACO</i> + <i>GA</i>	88
8.6	Summary	93
9	Conclusion and Future Work	94
9.1	Comparison of Algorithms	94
9.2	ACO Pheromone-based Crossover	95
9.3	Future Work	96
	References.....	98
	Appendices.....	102
	A. Production Time Component Distribution.....	103
	B. Experiment Results for Effect of <i>R</i> for Tabu Search.....	113
	C. Publications.....	114

Table of Illustrations

Figure 1. A Turret (Unit System).....	2
Figure 2. C-Path.....	3
Figure 3. Encoding Format for Job-Shop Scheduling Problems	37
Figure 4. Performance of SA, GA and Tabu search for the JSSP.....	38
Figure 5. Component Contribution versus Normalized Fitness (2 Turrets)	47
Figure 6. Component Contribution versus Normalized Fitness (3 Turrets)	47
Figure 7. Component Contribution versus Normalized Fitness (4 Turrets)	48
Figure 8. Chromosome Encoding Format for <i>ITAS</i>	53
Figure 9. Numbering of Station	53
Figure 10. Travel Pattern of Tools.....	55
Figure 11. Chromosome Encoding Format for <i>ITAS</i>	56
Figure 12. Distance Matrices for Implementation I.....	57
Figure 13. Distance Matrices for Implementation II.	57
Figure 14. Production Time versus Mutation Rate for 3 Turrets (Implementation I) ..	58
Figure 15. Production Time versus Mutation Rate for 4 Turrets (Implementation I) ..	58
Figure 16. Production Time versus Mutation Rate for 3 Turrets (Implementation II) ..	59
Figure 17. Production Time versus Mutation Rate for 4 Turrets (Implementation II) ..	59
Figure 18. Neighbour of Tool Placement Z (Implementation I).....	62
Figure 19. Neighbour of Tool Placement Z (Implementation II).....	62
Figure 20. Production Time of Simulated Annealing versus Cooling Factor.....	67
Figure 21. Production Time of Simulated Annealing versus Number of Fitness Evaluations.....	68

Figure 22. Production Time of Simulated Annealing versus Number of Fitness Evaluations (Implementation II)	70
Figure 23. Effect of R (3 Turrets)	78
Figure 24. Effect of R (4 Turrets)	78
Figure 25. Effect of α (3 Turrets).....	85
Figure 26. Change in Production Time w.r.t Ratio of Blind Ants (3 Turrets).....	86
Figure 27. Change in Production Time w.r.t Ratio of Blind Ants (4 Turrets).....	86
Figure 28. Production Time versus Ratio of GA to Ants (3 Turrets).....	89
Figure 29. Production Time versus Ratio of GA to Ants (4 Turrets).....	90
Figure 30. Production Time versus Ratio of APX in GA population (3 Turrets).....	92
Figure 31. Production Time versus Ratio of APX in GA population (4 Turrets).....	92
Table 1. Effect of Tabu Attribute and Diversification Strategy.....	75
Table 2. Effect of k	77
Table 3. Effect of Tabu Attribute and Diversification Strategy.....	79
Table 4. Best Production Time Comparison of Meta-heuristic Algorithms for $ITAS$	94

List of Symbols

Chapter 2

$\aleph$	Set of the input parameter
$\Re$	Set of Real Numbers
Z	Set of Integers
X	Input parameters
Y	Output of objective function
N	Number of inputs
$f(X)$	Objective function
$g(X)$	Rank function
$h(X)$	Identification function
$j(X)$	Fitness association function
d	Step size

Job-shop Scheduling Problem (JSSP)

M	Set of machines in JSSP
J	Set of jobs in JSSP
m_k	Set of operations to be processed on machine k
j_i	Set of operations belonging to job i .
S_{mji}	Starting time of operation j of job i on machine m
T_{mji}	Duration of operation j of job i on machine m
$o_n \succ o_{n+1}$	Operation o_n precedes operation o_{n+1}

Quadratic Assignment Problem

N	Number of distinct objects
K	Number of distinct sites
π	Object-site assignment
a_{ij}	Traffic intensity between object i and object j
$b_{\pi(i)\pi(j)}$	Distance between site containing object i and site containing object j

Chapter 3

Schema Theorem

$M(H,t)$	Number of instances from schema H in generation t
$M(H,t+intermediate)$	Intermediate generation arising from the selection of $M(H,t)$, but before crossover or mutation operator is applied (i.e. the mating pool)
$f(H,t)$	Average fitness of Schema H in generation t
$P(H,t)$	Expected percentage of instances in population from Schema H in generation t
$\bar{f}$	Average fitness of population in generation t
p_c	Probability of crossover
N	Population Size
$\Delta(H)$	Defining length of schema H .
L	Chromosome Length

Simulated Annealing

S	Solution
E_i	Energy of solution S_i
$C(S)$	Cost function of solution S
$P_{acc}(\Delta E)$	Probability that a new solution will be accepted
T	Current temperature of the annealing process. Typically 1.0.
k	Scaling factor.
α	Cooling factor, typically between 0.8 and 0.99 [Skiena97]

Ants Colony Optimisation

t	Current cycle
A	Entire population of ants
a	An ant of the colony
e_t^a	Edge where the ant a is at time t
$p(e_{t+1}^a)$	Probability of edge a for next cycle $(t+1)$
N_t^a	Neighborhood of edge a
k	Edge in the neighborhood of N_t^a
τ_t^e	Pheromone of edge e at time t
ρ	Evaporation rate
$f_1(e_t^k)$	Fitness of edge k , i.e. pheromone of edge k
$f_2(e_t^a)$	Change in pheromone for edge e based on fitness of solution generated by ant a

Tabu Search

S	Current tabu solution S
t	Current cycle
$N(S,t)$	Neighborhood set of solutions reachable from current solution S
$T(S,t)$	Tabu set of solutions
$A(S,t)$	Aspirant set of solutions

Chapter 4Performance Measures

U_{tool}	Total Utilization Factor
M	Number of tools in system
$T_p(m)$	Total amount of time tool m was used in the assembly process.
A	Assembly plan for product A
B	Assembly plan for product B
S^J	Number of steps in assembly plan for product J
T_C^J	Time required to produce first unit of product J
T_P^J	Sum of all processing time required by assembly plan J
T_T^J	Traveling time required for first unit of product J
T_t^J	Traveling time required to move a work-piece between adjacent stations

Fitness Landscape Analysis

T_C	Production time
T_P	Processing time
T_T	Traveling time
T_W	Waiting time
N	Number of steps in assembly plan
$PT(i)$	Processing Time of the i^{th} step of assembly plan
S_L	Loading Station
S_U	Unloading Station
S_i	Station for the i^{th} step of assembly plan
$FT_T(S_x, S_y)$	Travel Time/Transfer Time between station for step x and station for step y

Chapter 5

N	Population Size
P_{occ}	Probability of occurrence of each shared tools appearing at least once per station in the population.
P_{not_occ}	$1 - P_{occ}$
T	Number of turrets.
n	Number of different tool placements.
x	A shared tool in the system.
π	A shared station in the system.
S_i	i^{th} station.
S_S	Shared station
S_A	Unique station for assembly process A

S_B	Unique station for assembly process B
Q_i	i^{th} tool.
Q_S	Shared tool
Q_A	Unique tool for assembly plan A
Q_B	Unique tool for assembly plan B
r	Turret number
p_c	Probability of cross-over
p_m	Probability of mutation
π	Object-site assignment in QAP
a_{ij}	Traffic intensity between object i and object j
$b_{\pi(i)\pi(j)}$	Distance between site containing object i and site containing object j
L	Chromosome length
l	Number of genes transmuted from parent chromosome

Chapter 6

T	Number of Turrets
m	Number of Fitness Evaluations (In multiples of population size)
S	Solution
E_i	Energy of solution S_i
$C(S)$	Cost function of solution S
$P_{acc}(\Delta E)$	Probability that a new solution will be accepted
T	Current temperature of the annealing process. Typically 1.0.
k	Scaling factor.
α	Cooling factor, typically between 0.8 and 0.99 [Skiena97]

Chapter 7

T	Number of Turrets
R	Number of non-improving moves before random restarting
k	Number of standard deviation better than the current solution before a move is considered to be in the aspirant set.

Chapter 8

Q	Tool Q
s	Station s
α	Distribution of probability between
ρ_s^Q	Expected production time of tool placement which has tool Q at the station s
σ_v^Q	Expected production time of tool placement which has tool Q located at the next station when the tool located at the current station is v
$\rho_s^{Q'}$	Current value of ρ_s^Q (before update)
$\sigma_v^{Q'}$	Current value of σ_v^Q (before update)

1 Introduction

1.1 Terminology

The following terms shall be used in this thesis:

- A *product* refers to an item that is being manufactured.
- A *unit* refers to a single piece of a product.
- A *process* refers to the production of a specific type of product.
- An *assembly sequence* or *assembly plan* refers to the instructions for the manufacturing of a product.
- An *assembly step* refers to a single instruction in an assembly sequence.
- A *tool* refers to a piece of equipment that can be used for the manufacturing of an unit.

1.2 The Intelligent Turret Assembly System

The *Intelligent Turret Assembly System (ITAS)* is a modular, re-configurable system that was researched by *Singapore Institute of Manufacturing Technology (SIMTech)* to provide the flexibility of change in a small footprint through the use of a modular design.

The modular units of the *ITAS*, known as *turrets*, can be fitted interchangeably with a set of tools. Each turret itself may link with others through the help of a transfer-arm to form a complex assembly system.

1.2.1 Physical Description

A turret (Figure 1) is a cage-like structure that houses a platform, 6 *tool-head assembly units* (*tool-heads*), a video camera and their control equipment.

The 6 tool-heads, which are identical, are designed to allow attachment of a variety of tools. These 6 tool-heads are arranged 3 rows by 2 columns on top of the structure. A tool-head can only be fitted with 1 tool for each assembly process.

At the centre of the structure is the platform where two *forces* are located. These *forces* are driven by electromagnetic force with air-cushioning (air-bearing) and may move to any location on the platform.

Figure 1. A Turret (Unit System)

1.2.2 Operational Behaviour of the ITAS

Due to the wiring attached to each *forces*, the movement of the *forces* follows a C-shaped path (*C-Path*) as shown in Figure 2 to avoid tangling of the wires. That is,

Figure 2. C-Path

forces *A* travels in the path $\{1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5\}$, while forces *B* travels in the path $\{4 \rightarrow 5 \rightarrow 6 \rightarrow 1 \rightarrow 2\}$.

The movement of the forces within the *C-Path* can be controlled independently of the other. However, as they share a common physical path, two forces may not be at the same station simultaneously. It can be observed that each C-Path provides access to only 5 stations within each turret, of which 4 are common to both forces.

As the movement paths are along the locations of the tool-heads that house the equipment necessary to assemble a product, the term *station* will also be used to denote the position of tool-head in the system where an assembly step is to be carried out.

Work-pieces are loaded or removed from the forces by equipment known as the *loader*. These equipment can be located at any of the stations in the turret. In addition, the *loader* can exist together with other assembly tools, i.e. a station can be used to perform an assembly process as well as to load/unload the work-piece, if the *loader* is fitted at the station.

The link-up of turrets is through a transfer-arm that picks and places the work-piece from one forces of a turret to an adjacent turret. A transfer-arm can only be fitted at the four corners of the turret.

1.2.3 Operational Limitations

Besides allowing tools to be placed interchangeably among tool-heads, the *ITAS* also allows a variety of system layouts for linking multiple turrets together.

However, due to the additional complexity that this would bring, the project would continue to assume a fixed layout that places the turrets in a vertical line with a maximum of 2 processes in the whole system at any given time.

Other than simplicity, this configuration was chosen as it was also likely to contain the maximum collision between the two processes. This was because in this configuration, there were no alternative paths for either process to move to in order to avoid collision.

The duplication of tools, i.e. by having more than 1 tool of the same kind to improve the efficiency of the system, will not be considered.

We note that in the absence of collision, the problem is similar to the QAP, with the goal of finding a tool placement that will minimize the waiting time (i.e. the distances between stations), where the cost (distance) between the stations is depicted by the assembly plans.

1.2.4 Data Set

A limitation of the work done in [Chung02] was that the analysis of the *ITAS* was mainly for a single turret and may not apply to multiple turrets. To extend the project to multiple turrets, a pair of assembly plans requiring 2 to 3 turrets were randomly generated based on the following assumptions:

- The number of tools equals the number of tool-heads.
- 95% chance that each tool will be used at least once in the assembly sequence.
- Travelling time is much shorter than processing time (i.e. travelling time is about 10% of processing)

As the assembly sequence is randomly generated, for an assembly sequence with n turrets, the probability of a tool not being picked for a step in the assembly sequence is $\frac{5n-1}{5n}$ because there are 5 tools in each turret that are accessible to each assembly

sequence. Since the selection of tools is random, the probability that a tool will not be used in the subsequent step is the same. Hence, the probability that a tool will not be

used at least once in an assembly process of s steps is $P_{unused} = \left(\frac{5n-1}{5n} \right)^s$.

Therefore, for every tool to be included in the assembly sequence with probability of P_{used} , the length of the assembly sequence required is:

$$\begin{aligned}
 P_{used} &= 1 - P_{unused} \\
 1 - P_{used} &= \left(\frac{5n-1}{5n} \right)^s \\
 \log(1 - P_{used}) &= \log \left(\frac{5n-1}{5n} \right)^s \\
 \log(1 - P_{used}) &= s \log \left(\frac{5n-1}{5n} \right) \\
 \log(1 - P_{used}) &= s(\log(5n-1) - \log(5n)) \\
 s &= \frac{\log(1 - P_{unused})}{\log(5n-1) - \log(5n)}
 \end{aligned}$$

Rounding up, the number of steps required for every tool to be included in the

assembly sequence with probability P_{used} is $\left\lceil \frac{\log(1 - P_{unused})}{\log(5n-1) - \log(5n)} \right\rceil$.

As the computation of the production time was done through software simulations, a limit of 4 turrets and $P_{used} = 95\%$ (2 standard deviations) was chosen in order to keep the time required by the simulations to a reasonable duration. The 2 standard deviation refers to the confidence interval that 95% of the time, an assembly plan of the number of steps computed will include each tool at least once.

The travelling time was chosen to be small to maximize the effect of the waiting time since it was shown that a smaller travelling time would result in a larger contribution from the waiting time component [Chung02]. This assumption was also made because in real assembly plans, the time spent travelling between adjacent stations is likely to be small compared to the processing time.

1.3 Uniqueness of the ITAS Problem

One of the problems brought about by the modularity of the *ITAS* is that it offers users a number of choices to perform an assembly process. Because the C-Path allows only 5 of 6 stations to be accessed, if an assembly process requires 4 tools, the user would have to choose among 120 ways to fit the tools to the tool-heads in a turret.

Depending on the assembly processes involved, the choice of the tool placement may have an effect on the speed, and hence, the efficiency of the assembly processes. As the number of tools increases, the difficulty in selecting an optimum tool placement increases in non-polynomial time [Chung02].

Thus, the *ITAS* problem differs from conventional NP-hard problems in that the number of possible configuration is exponential with the increasing number of tools and turrets. The problem is complicated with the unknown waiting time and the constraints that tools have to travel in an one-direction C-path.

1.4 Objective

The inclusion of the waiting time and the C-path constraints mean that conventional meta-heuristics algorithm may not be sufficient for dealing with the *ITAS* problem. Thus, the objective of this project is to find a suitable algorithm for the *ITAS* that will help to determine the tool placement that would result in maximizing the efficiency of the system. That is, minimizing the time required to assemble N units of a product (i.e. *completion time* or *production time*) using a particular tool placement.

1.5 Thesis Outline

This thesis is organised into 3 parts. Chapter 1 has introduced the *ITAS* and its operational characteristics.

In chapter 2, the concepts behind Meta-heuristic algorithm will be explained. Chapter 3 looks at the various algorithms of Meta-heuristics and their similarity. A short experiment was conducted to review their performance. Chapter 4 provides the analysis of the *ITAS* problem to understand the characteristics of the *ITAS*. This is followed by chapters 5 to 8, which describe the implementations of various Meta-heuristic algorithms for the *ITAS* problem. Each chapter is accompanied with a summary on the finding of the various algorithms.

2 Optimisation

2.1 Definition of an Optimisation Problem

$$\begin{aligned}
 X &= \{\aleph_0 \times \aleph_1 \times \dots \times \aleph_{N-1}\} \\
 Y &= \{\Re \times \Re \times \dots \times \Re\} \\
 f &: X \rightarrow Y \\
 \forall v \in X, [f(q) \prec f(v)] \vee [f(q) = f(v)]
 \end{aligned}$$

The symbol $a \prec b$ denotes that a is *preferred* over b . An optimisation problem can be defined as a search process for an input condition $q \in X$, such that an objective function f ranks $f(q)$ as being preferred over other values $[f(q) \prec f(v)]$ in the domain of X , or as being equal to other values $[f(q) = f(v)]$ in the domain. Overall, this means $f(q)$ is no worse than all other values $f(v)$. The set of input parameters X is typically specified by a set of feasible solutions that satisfy some constraints. The object function f converts the set of input parameters X into a single real number $[f(X) \in \Re]$ or, for a multi-objective function, into a N -tuple of real numbers $[f(X) \in \{\Re \times \Re \times \dots \times \Re\}]$ in the output domain Y .

For multi-objective functions, each objective in the output is a distinct value which cannot be directly compared to other objectives. That is, different elements in the output N -tuple cannot be compared to each other. For example, a vehicle routing problem may have the time taken (delivery time) and cost required to complete deliveries as its output. The exact relationship between delivery time and cost may not be known. A solution with shorter delivery time and higher cost may not be directly compared to one with longer delivery time and lower cost. In such cases, a pareto-

based comparison (i.e. $\prec$) is used to rank the outputs as being more or less favourable relative to each other. Thus, the optimal output $f(q)$ is the input condition that leads to an output which is no worse than all other outputs.

2.2 Estimation of Optimal Input Parameters

Let $h : X \rightarrow Z$ (Z is the domain of integers) be a transformation function which maps the input parameters into some *unique integer* for each input in the domain of X ; for example, the function $h(X)$ can be the combination or permutation number of a set of input. This flattens the landscape of a function with N inputs into a two dimensional graph of $h(X)$ versus the objective function $f(X)$.

Let $g : X \rightarrow \Re$ be a function which *uniquely ranks* the output $f(X)$ such that if $g(q) < g(v)$, the output of $f(q)$ is no worse than the output of $f(v)$. Thus, these functions would effectively flatten the landscape of a function with N inputs into a two dimensional graph $j(h(X)) = g(X)$.

Since the output of $g(X)$ is unique, it may be possible to derive or estimate an inverse function $j^{-1}(g(X))$. If this function can be found, it would mean that we have a way to relate the outputs to the input parameters. While the method for finding the inverse function has been used for simple problems, finding $j^{-1}(g(X))$ is likely to be a more difficult task than working directly on a search using $f(X)$.

Another observation is that it is possible to use $g(X)$ to replace $f(X)$ as the objective function for meta-heuristic operations, if $g(X)$ can be more easily computed. This possibility exists because the optimisation process is concerned with the preference of input parameters and not with the exact value of the output.

2.3 Search Space Complexity

While optimisation necessitates the ability to compute the output of $f(X)$, the mapping function can be complex with non-linearly separable input terms, time-varying and difficult to understand with no exact methods for computing $f(X)$. The estimated complexity of a problem having N inputs is $\prod_{i=0}^{N-1} |\mathbb{S}_i|$ with $|\mathbb{S}_i|$ being the cardinality of the $(i+1)^{\text{th}}$ input domain. For inputs which is a permutation set, the complexity is $_{|X|}P_N$, with $|X|$ being the number of items in the permutation set. As the complexity of these problems is non-polynomial time, an exhaustive search is infeasible for large instances of these problems.

The existence of a non-random mapping function in these problems is important as it means there is some kind of pattern that relates the output to the input. This means that the landscape of the function is not random, but is made up of regions with certain properties which may be exploited in order to speed up the search. If the landscape is truly random, the performance of any search would be no better than that of an exhaustive search since the entire search space needs to be explored.

As the response towards changes in one input parameter may differ from another depending on the values of the other parameters; the coupling, or how the change in output is affected by other input parameters, determines the landscape of the function. This in turn determines the difficulty in which the characteristics of the landscape may be recognized.

The difficulty of the search space is different from the complexity of the search space estimated earlier. A problem can have an exponential search space but with little coupling between its parameters. For example, the function $f(x_0 \dots x_{N-1}) = \sum_{i=0}^{N-1} x_i$ has a search space that increases exponentially with increase in the number of inputs, but has zero coupling between the input parameters since the change in the output can be defined solely in terms of the change in each input parameter.

2.4 Neighbourhood of a Solution

The neighbourhood of a solution is defined as the solutions which can be generated from that solution by a small change in its parameters. This is important as meta-heuristics often sample or perform searches within the neighbourhood of a solution, as observed for Tabu Search, Simulated Annealing and Ant Colony Optimisation.

For a problem with N unique input parameters, there are Nd neighbours that can be derived from changing each of the N values by up to $d/2$ steps away from the current value.

For a permutation-based problem, the neighbourhood can be defined by

- replacing one of the N permutation values with a value that was previously not selected,
- exchanging one pair of values in the permutation so that the graphical distance between the current and neighbour solution is 1.
- exchanging one adjacent pair of values in the permutation so that in addition to having a graphical distance of one, the positional distance between the swapped pair is also 1.

For each of these cases, the complexity of the neighbourhood is $N(|X| - N)$,

$\frac{N}{2}(N - 1)$ and $N - 1$ respectively. In the first case, there are $|X| - N$ values that were not present in the existing solution. Each of these values may replace one of the N elements to form a neighbour solution. For the second case, each of the N elements in the permutation can be swapped with $N - 1$ other elements. However, since a swap is symmetric, the total number of unique swaps is divided by 2. For the third case, the number of adjacent pairs of element in the permutation is $N - 1$.

Interestingly, most meta-heuristics were observed to perform some kind of implicit estimation of $h(X)$ in the form of adaptation to the results found as the search progresses. That is, the meta-heuristic learns to predict the fitness of a region based on its past experiences.

2.5 Search Strategy, Intensification and Diversification

Since an exhaustive search is often infeasible, meta-heuristics must make some kind of prediction whether a region of the search space would potentially hold the optimal solution based on a few points of that region. This prediction forms the basis for the search strategy of a meta-heuristic and differentiates the different algorithms from one another.

The number of points in the region sampled affects the accuracy of the prediction because a larger number of points gives a better reflection of the average fitness of the region. Many meta-heuristic algorithms use an adaptive approach towards the sampling of regions. If a single sample from the region shows good fitness, the search will intensify into the region. How the region is being explored forms the intensification strategy of the search.

Eventually, as more points from the region are available, a better estimation of the region fitness can be obtained. That is, the perceived average fitness of a region changes over time as the meta-heuristic explores the region. This process helps to reduce the effect of noise in the sampling process.

Since each sample is reflective of the region it is in, the fewer numbers of samples, the larger area a single sample needs to be representative of. To provide an accurate overview of the fitness landscape for identifying regions for intensification, as many points from diverse areas of the landscape should be sampled. This forms the basis for the need of a diversification strategy for a meta-heuristic algorithm.

2.6 Problems in Operations Research

2.6.1 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is probably one of the most researched combinatorial problems.

Given that a list of cities and the distances between the cities is stipulated, the goal of the TSP is to find an itinerary that will allow the salesman to travel for the shortest distance to visit all the cities in the list exactly once and return to the starting point.

Like the TSP, one objective of the *ITAS* is to minimize the travelling time. However, unlike in the TSP, the order in which the stations are to be visited is restricted and what we are interested in is to adjust the tool to station assignment.

2.6.2 Job-shop Scheduling Problem

Like the TSP, the Job-shop Scheduling Problem (JSSP) is another famous combinatorial problem. The objective of the JSSP is to find the optimal allocation of operations to M machines from J jobs according to some criteria such as to minimize the make-span (i.e. the time required for all jobs to complete processing), or the tardiness (i.e. the amount of time from a machine being available to the time that a job was assigned to it).

$$M = \{m_1 \dots m_M\}$$

$$J = \{j_1 \dots j_J\}$$

For make-span,

$$f = \max(\forall i, j \{S_{mji} + T_{mji}\})$$

- m_k Set of operations to be processed on machine k
- j_i Set of operations to which belongs to job i .
- S_{mji} Starting time of operation j of job i on machine m
- T_{mji} Duration of operation j of job i on machine m
- f Fitness function, i.e. make-span of the JSSP

Each job consists of a set of operations $\{o_1, o_2, \dots, o_m\}$ that must be executed sequentially (i.e. $o_n \succ o_{n+1}$). Each machine may only perform one operation at a time and once an operation is started, it may not be pre-empted. Each operation o_n takes place at an assigned machine.

Differences between the JSSP and *ITAS* are:

- a. JSSP is concerned with the assignment of operations to machines; the *ITAS* is concerned with the placements of tools.
- b. JSSP allows jobs to be moved between machines with no constraint on the flow between them (unlike the C-path in *ITAS*). The time required to move a job from one machine to another (travelling time) is also ignored.
- c. There are no repetitions, i.e. the jobs do not repeat themselves in JSSP unlike in the case of the *ITAS*, where more than one unit of a product may be required.

2.6.3 Quadratic Assignment Problem

Like the *ITAS*, the QAP is concerned with the assignment of a set of N distinct objects to a set of K or more sites where each site may only be assigned one object. The cost of an assignment π is [Tate92]

$$f(\pi) = \sum_{i=1}^N \sum_{j=1}^N a_{ij} b_{\pi(i)\pi(j)}$$

Where:

a_{ij} Traffic intensity between object i and object j

$b_{\pi(i)\pi(j)}$ Distance between site containing object i and site containing object j

$f(\pi)$ Cost of the assignment π .

The *ITAS* is similar to a QAP with a dynamic a_{ij} , whose value is affected by the collision that occurs during the assembly process and changes in the tool placements.

3 Meta-heuristic Algorithms

Meta-heuristics are “high level concepts for exploring search space by using different strategies in such a way that a dynamic balance is given between the exploitation of accumulated search experience (which is commonly called intensification) and the exploration of the search space (which is commonly called diversification)” [Blum01]. Four such strategies along with factors which are important to their implementation are presented in the following sections.

3.1 Genetic Algorithms

Genetic Algorithm (*GA*) is a group of algorithms that make use of the principles of Darwin’s evolution theory in the search process.

The principles employed by *GA* are [Koza99]:

- Better individuals are more likely to be selected than inferior individuals;
- Reselection is allowed. That is, better individuals can be selected for breeding more than once; and
- Selection is probabilistic.

3.1.1 Algorithm Overview

In Genetic Algorithms, problem parameters (genes) are joined together to form a string of values (known as a chromosome) [Beasley93]. By simulating natural behaviour in evolving organisms to handle complex problems, *GA* puts possible

solutions (chromosomes) through a series of selection (survival of the fittest), crossover (reproduction) and mutation to generate better solutions.

The algorithm for a simple *GA* approach is as follows:

```

Create initial population  $P_0$ 
 $P = P_0$ 
repeat
  Compute fitness of all individuals in  $P$ 
  for  $i=1$  to  $N/2$  do
    select two individuals (chromosome)  $C_M, C_F$ , from  $P$ 
      biased by the individual's fitness
    recombine (crossover)  $C_M, C_F$  to  $C_S, C_D$ 
    place  $C_S, C_D$  into  $P_{t+1}$ 
  Replace  $P$  with  $P_{i+1}$ 
until (termination condition met)

```

An initial population P_0 of N size is created, which is made up of randomly generated chromosome strings. The fitness of individual members of the population is evaluated and used in a biased selection process of $N/2$ trials.

The probability that an individual (chromosome) gets selected as C_F or C_M is based on the selection process used and the individual's fitness. In a simple, maximizing *GA*, this approximates to:

$$P_{sel}(C) = \frac{Fitness(C)}{\sum_{i=1}^n Fitness(C_i)}$$

A scaling function may be applied to normalize the fitness of the population to prevent over-dominance by fit chromosome, or to improve the selection probability of fitter chromosomes when the *GA* tends towards convergence. The general principle is that chromosomes with higher fitness values get a higher probability of being selected into the next population, with the chromosome with median fitness having the probability of being selected once for the crossover process.

Following the selection process is the crossover operation where genes from the selected parent chromosomes are swapped to create two new child chromosomes. Thus, each child chromosome will contain part of the genes of the parent chromosomes and hopefully, part of the better characteristic of its parent. Mutation, in which genes in the new chromosome are randomly changed, may be applied.

After the crossover and mutation process, the child chromosomes are placed into a new population that will replace the current population. This process is repeated until a termination condition is met, which may be a specific number of generations or a population convergence ratio.

3.1.2 Schema Theorem and the Basic Building Block Hypothesis

3.1.2.1 Schema

Schemas are string templates containing wild-characters (i.e. ‘*’) used to represent a group of strings with similar characteristics. The wild-characters in a schema may take on any alphabet (symbol) of the string. For example, for a 4-bit binary string, a schema of 1**0 would match these strings: {1000, 1010, 1100, 1110} as well as schemas {1*00, 1*10, 10*0, 11*0}.

Therefore, a single member of the *GA* population of length l would contain $(\alpha + 1)^l$ schema, where α is the number of alphabets in the chromosome. This is believed to allow *GA* to sample between $(\alpha + 1)^l$ to $N(\alpha + 1)^l$ schemas in a population of size N .

3.1.2.2 Schema Theorem

The schema theorem (ST), also called the “fundamental theorem of genetic algorithm” by Goldberg [Goldberg89], explains how *GA* processes the schemata through examining the expected number of occurrences of a schema in the next generation.

The theory states that the number of individuals that belong to a particular schema in a subsequent population depends on the schema fitness in the current one, i.e. a fitter schema would have more chromosomes in the subsequent population.

Mathematically, the ST (for a Simple *GA*) is explained as [Bäck00]:

$$\begin{aligned}
 M(H, t + immediate) &= M(H, t) \frac{f(H, t)}{\bar{f}} \\
 M(H, t + 1) &= (1 - p_c) M(H, t) \frac{f(H, t)}{\bar{f}} + p_c \left[M(H, t) \frac{f(H, t)}{\bar{f}} (1 - losses + gain) \right] \\
 M(H, t + 1) &\geq (1 - p_c) M(H, t) \frac{f(H, t)}{\bar{f}} + p_c M(H, t) \frac{f(H, t)}{\bar{f}} (1 - disruptions) \\
 P(H, t) &= \frac{M(H, t)}{N} \\
 disruptions &= \frac{\Delta(H)}{L - 1} (1 - P(H, t)) \\
 P(H, t + 1) &\geq P(H, t) \frac{f(H, t)}{\bar{f}} \left[1 - p_c \frac{\Delta(H)}{L - 1} (1 - P(H, t)) \right]
 \end{aligned}$$

Where:

$M(H,t)$	Number of instances from schema H in generation t
$M(H,t+intermediate)$	Intermediate generation arising from the selection of $M(H,t)$, but before crossover or mutation operator is applied (i.e. the mating pool).
$f(H,t)$	Average fitness of Schema H in generation t
$P(H,t)$	Expected percentage of instances in population from Schema H in generation t
$\bar{f}$	Average fitness of population in generation t.
p_c	Probability of crossover
N	Population Size
$\Delta(H)$	Defining length of schema H. The defining length of a schema is defined as the distance between the position of the first and last fixed symbol of a string.
L	Chromosome Length

Assumptions:

- Gains arising from crossover of other schemas are ignored; and
- Losses in the schema H occur largely as a result of crossover (i.e. the effect of other operators, such as mutation, are ignored). Therefore, losses are approximated by disruptions to the schema.

That is, *disruption* is assumed to be the probability that a crossover operation will result in a chromosome that does not belong to the parent schema.

3.1.2.3 Basic Building Block Hypothesis

While the ST explains the selection pressure in *GA*, it does not explain how *GA* is able to generate better solutions [Fang94]. The explanation of the ability of *GA* to evolve better solutions is given in the Basic Building Block Hypothesis (BBH).

The BBH explains this by stating, “[A] genetic algorithm seek[s] near optimal performance through the juxtaposition of short, low-order, high performance schemata, or building blocks” [Goldberg89].

What this suggests is, if the encoded problem parameters can be broken into small independent blocks, the optimal solution would be the combination of the best of each independent block.

3.1.3 Factors affecting *GA*

3.1.3.1 Objective Function

The objective function in a *GA* should provide a deterministic mapping of input parameters to a fitness value. In short, it defines what attribute of the problem the *GA* is trying to optimise for. However, since the objective function is dependent on the problem itself, there may be limited choice in its selection.

As the objective function is invoked frequently in *GA*, it would also be advantageous for it to require as little processing time as possible.

3.1.3.2 Coding

The importance of the coding format is due to the BBH. To allow the BBH to function, chromosomes similar to each other should be mapped into problem parameters that are similar to each other.

In some problems, we may not have control over the objective function (or evaluation criteria), such as in the case of the TSP where the fitness of a tour is rigidly defined. However, we do have control over the coding format of the problem.

In a way, the coding format of the problem is like the $g(X)$ function described in Section 2.2 in that it determines which solutions are neighbours of each others.

3.1.3.3 Population Size

As explained in Section 3.1.2.1, GA is believed to search between $(\alpha+1)^l$ to $n(\alpha+1)^l$ schemas in a population size of n . However, other than the initial population, which is randomly generated, each subsequent population is a function of its previous population.

For example, consider a GA that has 16-bit chromosomes and a population size of 4. We note that the GA is likely to converge in a few generations and does not fully explore all the 4×3^{16} schemas believed to be contained in the population.

A possible reason is that there are only 2 pairs of parents that will be selected to create the 4 child chromosomes. If any of the selection trials favours one of the chromosomes more than others, thus selecting a chromosome more than once as a

parent, the result would be that chromosome having 50% contribution towards the genes of the new generation. This quickly causes the *GA* to become saturated.

In addition, even without the bias associated with the schema fitness as explained by the Schema Theorem, the probability that a random number generator repeating a value is higher due to the small number of choices that can be made. It should be noted that while a uniform random number generator has a uniform probability distribution, this is only true in the long run.

3.1.3.4 GA Operators

In [Galinier99], Galinier noted that the performance of standard *GA* operators in solving optimisation problems is generally poor; and that the design of the crossover operator should identify properties that are meaningful for the problem and provides a recombination mechanism in such a way that these properties are transmitted from parents to offspring.

3.2 Simulated Annealing

Simulated Annealing (*SA*) is based on the metallurgical technique that uses a disciplined cooling schedule to efficiently bring steel to a low-energy, optimal state [Skiena97].

It is based on the concept of Markov Chain Monte Carlo (MCMC), in which the probability of the outcome of an event is independent of previous trials. Since an

ergodic MCMC chain is recurrent, as the number of trials approaches infinity, all points in the chain will eventually be reached.

3.2.1 Algorithm Overview

The algorithm for SA is as follows:

```

Create initial solution  $S$ 
Initialize temperature  $T$ 
 $S^* = S$ 
repeat
  for  $i=1$  to  $iteration-length$  do
    Generate a random transition from  $S^*$  to  $S_i$ 
     $E^* = C(S^*)$ 
     $E_i = C(S_i)$ 
    if ( $Rand[0,1] < P_{acc}(\Delta E)$ ) then  $S^* = S_i$ 
     $T = \alpha T_i$ 
until (no change in  $C(S^*)$ )
return  $S^*$ 

```

For maximization problems using the Boltzmann distribution,

$$P_{acc}(\Delta E) = \begin{cases} \exp\left(\frac{E_i - E^*}{kT}\right) & E_i < E^* \\ 1 & E_i \geq E^* \end{cases}$$

Where:

$C(S)$	Cost function of solution S
E_i	Energy of solution S_i
$P_{acc}(\Delta E)$	Probability that a new solution will be accepted
T	Current temperature of the annealing process. Typically 1.0.
k	Scaling factor.
α	Cooling factor, typically between 0.8 and 0.99 [Skiena97]

Better solutions are always accepted, whereas poorer solutions are accepted with a probability of P_{acc} , which is proportional to the fitness of the new solution and the current temperature T .

T is initially large to allow SA to search a diversified range of solutions. As the search progress, T is gradually decreased to narrow down the search.

3.2.2 Cooling Schedule

The temperature schedule of a SA system is determined through α and k . The temperature schedule of the system affects the length of the Markov Chain and thus the ability of SA to reach all points within the solution space.

Ingber noted that having a “long time of execution of standard Boltzmann-type SA has many times driven these projects to utilize a temperature schedule too fast to satisfy the sufficiency conditions required to establish a true (weak) ergodic search.” [Ingber93]

The selection of the optimum convergence rate is problem-dependent and typically determined through trial and error [Skiena97].

3.2.2.1 Solution Generation

According to *Mathematical Optimisation* [CSEP91], a solution generator should:

- (i) introduce small random changes; and
- (ii) allow all possible solutions to be reached.

Condition (i) can be observed when *SA* generates a new solution in each iteration by making a random ‘transition’ from the current best solution (S^*). A transition, in this case, means that the new solution is one that can be ‘reached’, or share similarity with the original solution.

Condition (ii) is desirable as we would want to be able to explore all possible solutions to the problem.

3.3 Ant Colony Optimisation

Ant Colony Optimisation (*ACO*) is a fairly new population-based meta-heuristic search algorithm that is modelled after the behaviour of ants in their search for food. Basically, ants deposit chemicals (pheromones) on the trail that they take. The higher the amount of deposited pheromone a trail contains, the more likely it is to be taken by other ants that are visiting the area.

Essentially, the shortest path that leads to food is likely to be used more times as it takes a shorter time for an ant to make the trip. This, in turn, leads to a higher amount of pheromone being deposited, which then guides the other ants towards using that path.

The idea is adapted for optimisation by using artificial ants in the search space and updating the search path with a pheromone proportional to the fitness of the solution. We observed that over time, the pheromone is likely to take on the Markov variable of the edge.

3.3.1 Algorithm Overview

An outline of a possible implementation of an *ACO* is as follows:

```

t = 0
Reset pheromone deposited on all edges to 0
Set global best to some initial solution

for each ant a
    place a at some random location in graph

repeat
    // pheromone evaporation
    for each edge e
        remove a certain amount of pheromone from e

    // schedule ant movements
    for each ant a
        find list of edges  $N_t^a$  eligible to be visited
        compute next edge  $e_{t+1}^a$  to take in the
            neighbourhood  $N_t^a$  based on pheromone  $\tau_t^e$ 
            deposited on the edge  $e_t^a$ 

    // pheromone update
    for each ant a
        move a to new node via  $e_{t+1}^a$ 
        Add  $e_a$  to the list of edges traversed  $Y^a$ 

        if (solution is complete)
            deposit pheromone to the entire path  $Y^a$ 
                proportional to fitness of solution found
            update global best  $G$ 
        else
            optionally deposit a small amount of pheromone
                to edge  $e_{t+1}^a$ 

    t = t + 1

until (termination condition met)

```

3.3.2 Selection Probability and Pheromone Updates

The probability of the edge selection and pheromone update rule is problem-specific.

Generally, the probability of selecting edge e_{t+1}^a from the neighborhood N_t^a of ant a at time t is

$$p(e_{t+1}^a) = \frac{f_1(e_t^a)}{\forall k \in N_t^a \sum f_1(e_t^k)}$$

and the pheromone update rule is

$$\tau_{t+1}^e = (1 - \rho)\tau_t^e + \rho \sum_{a \in A} f_2(e_t^a)$$

where A refers to the entire population of ants.

ρ is the evaporation rate, or how fast the previously deposited pheromone disappear. This is used to prevent infinite accumulation of pheromone and to allow the *ACO* to unlearn bad moves.

f_1 is typically a function of the pheromone at edge e_t^k , but could include some estimate of the perceived fitness of the edge e_t^k . f_2 is some user-defined function and determines how the change in the fitness affects the pheromone. The change in pheromone ($\Delta\tau_t^{e_a}$) is computed based on the fitness of the current solution taken by the ant (e_t^a). For example, the pheromone of an edge may be given a positive increment whenever it forms part of a trail that leads to a good solution and a deduction whenever it leads to a bad solution.

While pheromones may be updated after every move, it may also be performed after a trail is completed. In such cases, the pheromones of the entire trail may be updated to reflect either a favorable trail or non-favorable trail depending on where a solution was found at the end of the trail.

3.4 Tabu Search

While Tabu search may be superficially viewed as a class of search algorithm that uses memory to inhibit certain moves from recurring, a better understanding of the ideas behind Tabu Search can be gained from the quotes:

“Humans use insight to overcome the limitation of sequential information processing. Tabu search treats insight as a function of memory together with associated strategies to manipulate information stored during the search for the best solution.” [LagunaW1]

“In a system that uses memory, a bad choice based on strategy can provide useful clues about how the strategy may profitably be changed. (Even in space with significant randomness)” [Karaboga00]

An important keyword in both quotes is *strategy*. That is, it is believed that the true power of the Tabu search lies not in the algorithm, but in the associated strategies that can be used to improve the algorithm.

3.4.1 Algorithm Overview

The basic structure of a tabu search algorithm is as follows:

```

Create initial solution S
t=0
repeat
  identify  $N(S, t)$ 
  identify  $T(S, t)$ 
  identify  $A(S, t)$ 
  choose best  $S' \in N(S, t) - T(S, t) + A(S, t)$ 
  mark move/attribute of  $S'$  tabu
  if  $(f(S') > f(S))$  then  $S = S'$ 
   $t = t+1$ 
until (termination condition met)

```

t is the current tabu tenure (i.e. the current time state) of the system; $f(S)$ is the cost function of solution S .

The Neighbour set $N(S, t)$ contains the set of moves that can be generated from the current solution S at time t . The $T(S, t)$ contains the set of tabu moves at time t . Moves that are in $T(S, t)$ are not to be used to generate the new solution unless they are also in the aspirant set $A(S, t)$, which contains moves that may be useful towards generating better solutions although such moves may currently be considered tabu.

The selection of the best move among the non-tabu set may be based on criteria such as the greatest amount of change in fitness, the largest structural differences between the new solution and the current solution, instead of the best fitness. After a move is taken, information about the move is kept in the tabu set $T(s)$ to prevent them from recurring until a number of cycles has passed (*tabu tenure*). This is done to prevent the new move from retracing the steps used [Glover93]. If the new move is better than the current best solution, it replaces the latter. This process is repeated until a user-defined stop criterion is met.

3.4.2 Tabu Strategies

The following sections list some of the concepts and strategies discussed in the book *Tabu Search* [Glover97]. It is interesting to note that some of these strategies are general ideas and are applicable in other algorithms.

3.4.2.1 Congenial Structure Principle

The principle of the congenial structure suggests that certain structures (solutions with certain attributes) may be more favourable towards generating good solutions. An alternative way to look at this principle is that certain properties may be common in good solutions; thus better solutions may be generated through the use of these ‘known good’ properties.

3.4.2.2 Proximate Optimality Principle

This principle suggests that good solutions at one level are likely to be found close to good solutions at an adjacent level. *Level*, in this context, refers to a specific range of the objective function value, or the presence/absence of certain attributes.

This principle may be used to initiate a search along a known good solution (initiating solution) to some other good solutions (guiding solutions). According to the principle, good solutions are likely to be found along trajectory between the initiating solution and the guiding solution.

3.4.2.3 Pyramid Principle

The Pyramid Principle suggests that there are fewer paths leading to optimality from a solution close to the optimal than from one that is far from optimal. Conceptually, the search space is a triangle with the optimal solution as the apex. Thus, the probability of reaching a global optimal solution is an inverted triangle with increasing probability of finding improving paths to global optimal solution, as the search gets closer.

By this principle, a search algorithm should explore a wider search space at the start of the search, and slowly reduce the area of search space.

3.4.2.4 Space/Time Principle

The Space/Time principle suggests that a tabu move may be dependent on not only time but also the path that the search is proceeding in.

The idea of making an attribute tabu is to prevent the attribute from being incorporated in further searches in the region that it affects. The purpose of making the attribute tabu is not achieved, if as a result of making an attribute tabu, a search then enters a region that does not interact with the tabu attribute.

3.5 Search Strategies in Meta-heuristic Algorithms

As noted in Section 3.4.2, there are similarities in the principles adopted by meta-heuristic algorithms. The following sections show how three of the principles are applied in the various meta-heuristic algorithms.

3.5.1 Congenial Structure Principle

The BBH in effect is an implementation of the Congenial Structure Principle. In the BBH, the evolution of chromosomes is done through combining fit building blocks of short order, which could be thought of as containing favourable attributes to the problem. So, the pheromone trails in the *ACO* that an ant makes can be thought of as an implementation of the Congenial Structure Principle.

3.5.2 Proximate Optimal Principle

The proximate optimal principle can be found in both *SA* and *GA*. In *SA*, this principle is observed as searches within the neighbourhood of the current best solution. In *GA*, this principle exists as part of the crossover process, in which the offspring inherits fit parent chromosomes. In *ACO*, the proximate optimal principle plays a role in the identification of the best solution through the amount of pheromone deposited.

3.5.3 Pyramid Principle

The implementation of this principle is obvious in *SA* as the reduction of annealing temperature. By cooling, the probability of solutions accepted that have fitness values less than the current best solution is reduced. Hence, the effect is that the search converges towards the region where the current best solution is found. This is in line with the principle in which the number of solutions explored is reduced as the search approaches optimality.

This principle is less obvious in *GA*, but it can also be observed in the Schema Theorem. As a *GA* progresses, the proportion of chromosomes in the population that belongs to the fitter schema increases, hence directing the search towards the region identified by the schema.

In *ACO*, the probability of an ant taking a path is dependent on the amount of pheromone attached to a trail. Initially, no pheromone is deposited and paths are equally likely to be visited. The amount of pheromone increases each time a trail is taken, thus decreasing the likelihood of exploring new paths over time. Again, it can

be observed that a large region was initially searched and as the search progressed, the region of search became progressively smaller.

3.6 JSSP Experiment

An experiment was conducted to compare the results of the *SA*, *GA* and Tabu search for the Job Shop Scheduling Problem. *ACO* was not implemented due to the lack of understanding on how it might be applied to solve the JSSP. The problem parameters used was 20 machines by 15 jobs (*ta20*) from [Taillard93].

In all three implementations, no analysis of the problem was performed other than what was necessary for implementing the algorithms. The purposes of this experiment were:

- a. To serve as a starting point for the author to understand the implementation of the various meta-heuristics algorithm.
- b. To serve as comparison of how these various algorithms perform on what was initially believed to be a problem similar to the ITAS due to the similarity having to assign tasks to machines.

3.6.1 Experiment Setup

For the *SA* implementation, the initial temperature T was chosen to be 3 standard deviations of the fitness value from a randomly generated population of solutions. This was done to avoid rapid “cooling”, i.e. to initially allow poorer solutions a high chance of being accepted by the system. The *SA* ran for 1000 cycles with a cooling

factor of 0.995. This cooling factor was chosen to allow the temperature to eventually converge to $0.00665T$ (0.995^{1000}) after 1000 cycles.

For the *GA*, a population size of 30, mutation rate of 0.1, crossover rate of 0.9, linear fitness scaling and elitism were used. The chromosome for the *GA* was an array of $M \times J$ length used to encode the JSSP parameters. Each group of J gene of the chromosome constituted the job list $\{j_1 .. j_J\}$ for a machine. That is, the genes *abbc* indicated that the first operation of machine *m* would be from the job *a*, followed by two operations from the job *b* and a fourth operation from job *c*. The relationship of the chromosome is illustrated in Figure 3.

Figure 3. Encoding Format for Job-Shop Scheduling Problems

A 1-point operator was used which copied over the genes of a parent up to the crossover point and kept track of the number of operations from each job that was copied. The remaining genes were then copied from the other parent (*p*) whenever there were leftover operations to be performed for the job depicted by the gene of *p*. The remaining genes were then filled according to the order of appearances in *p*. The effect of this crossover operator was that it preserved the job sequence of certain machines, thus ensuring that part of the parent chromosome would be propagated to the child. Finally, a mutation operator that swapped two genes within the chromosome was used.

The Tabu search implementation for the JSSP generated its neighbour set by randomly swapping 30 pairs of operations between various jobs from the current best solution. An explicit record of the last 10 solutions was used as the tabu list. No aspirant set was used.

3.6.2 Results

The population size in *GA* and the duration of the runs for both *SA* and Tabu were varied, such that the total number of objective function calls was the same for all three algorithms. The experiment was performed 10 times and the average of the best make-spans obtained from the experiments are shown in Figure 4.

Figure 4. Performance of SA, GA and Tabu search for the JSSP

From Figure 4, it was observed that the best mean make-span was obtained from SA, followed by Tabu search and GA. It was noted that these algorithms were very unlikely to be performing at their best and that there were many parameters that could be changed to improve their performance. The large fluctuations in the make-span

were probably an indication that the algorithms had yet to reach convergence. Despite this, the performance of *SA* did seem to be consistently better than Tabu search, as was the performance of the Tabu search over *GA*.

A probable reason for *GA*'s much poorer performance may be due to the crossover operator. Since the crossover operator was designed without knowing what characteristics of a solution in JSSP were desirable, it might have been unable to create better offspring from the parents and had thus hindered the search.

In comparison, *SA* did not require an understanding of how better solutions could be created and in this case, the Tabu search was not designed with domain specific knowledge and acted more like a hill climber.

3.7 Summary

Based on the study of the various meta-heuristic algorithms, it can be observed that each algorithm implements some of the various search strategies discussed in Section 3.5 in one way or another but may place more focus on certain strategies.

It was observed that all four algorithms contained some kind of selection pressure that drove the search towards optimality; however, the direction of the search (e.g. neighbour moves, crossover operator) is still up to the user.

As observed from the experiment, neither Tabu search nor *GA* were 'silver bullets' that worked miracles. Both algorithms required its user to do a proper analysis of the

problem domain so as to bring about a strategy or data structure that might be favourable for their implementation.

Genetic Algorithms is generally thought to be better at searching for regions of good fitness range, and may have difficulties in searching within the region. *SA* and Tabu search are generally considered neighbourhood search algorithms due to the nature in which the search is performed. Thus, a favourable solution may be to combine *GA* with either Tabu search or *SA* to improve the performance of *GA*.

4 Domain Knowledge

4.1 Performance Measures

The efficiency of the *ITAS* can be computed from measures, such as: (1) Tool Utilization Factor, (2) Production time.

4.1.1 Tool Utilization Factor

The Tool Utilization Factor calculates the average utilization of tools in the system based on the average time that each tool is used in the assembly process:

$$U_{tool} = \frac{1}{M} \sum_{m=1}^M \frac{T_p(m)}{T_C}$$

where M is the number of tools in the system, $T_p(m)$ is the total amount of time that the tool m was used in the duration T_C of the assembly process.

Ignoring the time required for transportation between tools, for the production of a single unit of a product, T_C would be the summation of the processing time requirement of all the steps required for the assembly of the unit.

Since the *ITAS* has a maximum capacity of 2 different products to be assembled in the system across all turrets, and that the number of times a tool is reused does not change with the increase in the number of tools, the utilization factor of the *ITAS* is likely to drop when more turrets are used in contrast with a constant tool utilization factor for more traditional pipelined assembly line approach.

4.1.2 Production time

The production time (completion time) is the time taken for the *ITAS* to produce a minimum of W units of each product type in a pair configuration assembly plans.

The *ITAS* might have started the production of a second unit of a product before the first unit of another product is completed. Thus, the production time, rather than the cycle time, is used because the pair of assembly plans may not have the same cycle time.

From U_{tool} , we note that the utilization factor is dependent on the production time as well, and that a lower T_C would result in a better utilization factor. As a result, the project uses the production time as the main measure of performance.

4.1.3 Lower Limit on Production time

The lower limit $T_{LU}(W)$ on the production time of a pair of assembly plans $\{A, B\}$ is defined as the fastest time needed to produce at least W units of each type of product in a pipelined assembly system, i.e.

$$T_{LU}(W) = \max(T_{C\min}^A(W), T_{C\min}^B(W))$$

T_C^J is the time required to produce the first unit of product J . It is computed using the sum of the processing time T_P^J and travelling time T_T^J components of the assembly plan.

$$T_C^J = T_P^J + T_T^J$$

The processing time component T_P^J is computed as the sum of the processing time, required of all the steps in the assembly plan. (s_i^J is the i^{th} step of the assembly plan of product J and $T_p(s)$ is the processing time of the step s)

$$T_P^J = \sum_{i=1}^{N^J} T_p(s_i^J)$$

The travelling time component, T_T^J , is the product of the number of steps in the assembly plan (N^J) and the time it takes to move a work-piece between two adjacent stations (T_t^J).

$$T_T^J = N^J T_t^J$$

In the pipeline, each of the next $M-1$ units will be delayed by the processing time of the slowest process in the assembly plan. Therefore, the time to complete W units is:

$$T_{C_{\min}}^J(W) = T_C^J + (M-1) \cdot \max(T_p(s_i^J))$$

Since a goal of an agile system is to reduce the time required for reconfiguring the system for different products [Rizzi97], the use of the tool utilization factor and production time is restricted to the comparison of the tool placement's efficiency.

Hence, the lower limit serves as a guide to the best possible timing that could be achieved for the assembly plans; it does not take into account the constraints imposed by the *ITAS*.

An upper limit on the production time could be defined as the slowest production time, beyond which, it may be more effective to assemble one product type at a time.

4.2 Fitness Landscape Analysis

The production time (also known as completion time) of an assembly plan is

[Chung02]: $T_C = T_P + T_T + T_W$

Production Time: (T_C) As there are two processes running at any time in the *ITAS* and the two assembly plans may be of different length, the production time of the system will be the larger of the two assembly processes.

Processing Time: (T_P)

$$T_P = \sum_{i=1}^N PT(i)$$

Time spent at a station as part of a product's assembly plan. This is simply the summation of all the processing time in an assembly plan.

Travelling Time: (T_T)

$$T_T = FT_T(S_L, S_1) + \left(\sum_{i=1}^{N-1} FT_T(S_i, S_{i+1}) \right) + FT_T(S_N, S_U) + FT_T(S_U, S_L)$$

Time spent travelling from one station to another as part of the product's assembly process, inclusive of transfer time. The travelling time is the summation of the travel time between adjacent steps in the assembly plan, plus the time

taken to move the forces to the Loading and Unloading stations and the time taken to move the forces back to the Loading station for the next work piece ($FT_T(S_U, S_L)$).

Waiting Time: Time wasted due to collision when trying to gain access to a station. This includes time spent moving from one station to another as part of the deadlock resolution process.

(T_w)

The waiting time component is obtained through simulation of the processes. No accurate mathematical estimate of the waiting time was available.

where:

N	Number of steps in assembly plan
$PT(i)$	Processing Time of the i^{th} step of assembly plan
S_L	Loading Station
S_U	Unloading Station
S_i	Station for the i^{th} step of assembly plan
$FT_T(S_x, S_y)$	Travel Time/Transfer Time between station for step x and station for step y

From the mathematical model, it can be observed that there are two ways to reduce the production time of the system: by reducing the travelling time or by reducing the waiting time.

The waiting time component could not be estimated accurately because the system is dynamic in nature. Collisions are dependent on the layout of the tools and the assembly plans used, and any collision that occurs will cause a shift in the timing for subsequent assembly steps. This in turn affects future collisions. In our simulation for a single turret, the waiting time differed from theoretical estimates by about 35%. [Chung02]

4.2.1 Distribution of Production Time Component

While the preliminary mathematical estimates of the waiting time component prove to be unreliable, the analysis of how the waiting time component may be estimated does provide a possible insight into how it may be reduced.

A study was conducted on the contribution of the waiting time with respect to the number of turrets through the simulation of pairs of random assembly plans for 2, 3 and 4 turrets respectively. 1000 tool placements for each pair of assembly plans were randomly generated and the production times for 100 units of each assembly item were recorded. The results are shown in Figure 5, Figure 6 and Figure 7. The assembly plans used and full size graphs of these figures are shown in Appendix B.

Figure 5. Component Contribution versus Normalized Fitness (2 Turrets)

Figure 6. Component Contribution versus Normalized Fitness (3 Turrets)

Figure 7. Component Contribution versus Normalized Fitness (4 Turrets)

From the graphs, it can be observed that the contribution of the waiting time component decreases with an increase in the number of turrets. We note that the processing time is fixed for each pair of assembly plans, thus any improvement is due to the decrease in waiting time or travelling time.

It can also be observed that if the average number of stations used in the assembly plan was large, the forces would have to move across more stations to reach its desired station. This in turn means an increased likelihood of collisions. Thus a reduction in the number of stations visited (i.e. minimizing the travelling distance) should also lead to a reduction in waiting time.

The decrease in the travelling time component can be seen from the following mathematical hypothesis:

Consider an assembly plan of L steps and a uniformly distributed collision probability at each station. Let $\bar{t}_p$ be the average processing time of each step in the assembly plan. It is assumed that:

- a) the average travel time between steps ($\bar{t}_t$) does not change with the increase in the number of turrets (T),
- b) the waiting time approximates the probability of a station being occupied, which is uniformly distributed among shared stations, i.e. $\frac{1}{4T}$, and that
- c) changes in N affect the complexity of the routes by a factor α .

The expected time components $E(\cdot)$ of the system are:

$$E(T_w) = \frac{L^\alpha}{4T} \cdot \bar{t}_p$$

$$E(T_p) = L \cdot \bar{t}_p$$

$$E(T_t) = L^\alpha \cdot \bar{t}_t$$

$$\therefore E(T_C) = L\bar{t}_p + L^\alpha \left(\frac{1}{4T} \cdot \bar{t}_p + \bar{t}_t \right)$$

Since $L^\alpha > \frac{L^\alpha}{4T}$, this implies that the contribution of the waiting time component decreases with the increase in the number of turrets, whereas the contribution of the travelling time component increases with the increase in the number of turrets.

4.3 Observation on Placement of Unique Tools

It was also observed that a change in the placement of a unique tool tends to produce a greater change in the production time compared to a change in a shared tool. This is probably because the distance between two unique tools of a process is at least 1 turret away (because each turret can only house 1 unique tool), meaning that the change in travel time from other station to a unique station will be greater than that of a shared tool. In addition, since the tools are on different turrets, a transfer time will also be incurred.

4.4 Discussions

From the graphs in Section 4.2, the contribution of the travelling time and waiting time components can be observed to span across a certain band for a single fitness value and these bands seem to decrease with the increase in the number of turrets.

The contribution of the processing time component towards completion time decreases with the increase in the number of turrets. It is believed that as the number of turrets increases, the average distances between subsequent stations in an assembly plan will need to be increased, thus leading to an increase in the travelling time component.

This confirms the hypothesis that the tool placement layout of the *ITAS* makes a difference towards the efficiency of the system and that there is a relationship between the tool placement and the production time. However, it does not give the exact relationship between tool placement and the production time.

5 Genetic Algorithm for *ITAS*

5.1 Implementation I

In the implementation of the Genetic Algorithm solution for the *ITAS*, for a T turrets problem, a simple *GA* with a population size of $N = 4T \left\lceil \frac{\log(1 - p_{occ})}{\log(4T - 1) - \log(4T)} \right\rceil$ was used with a $p_{occ} = 0.95$ (which is the probability that a shared tool will be mapped to every shared station at least once in the initial population.)

The derivation of the equation for the variable n is as follows:

$$\begin{aligned} p_{not_occ}(x, \pi) &= \frac{4T - 1}{4T} \\ p_{occ}(x, \pi, n) &= 1 - p_{not_occ}(\pi)^n \\ p_{occ}(x, \pi, n) &= 1 - \left(\frac{4T - 1}{4T} \right)^n \\ n &= \frac{\log(1 - p_{occ}(x, \pi, n))}{\log(4T - 1) - \log(4T)} \end{aligned}$$

where:

T	Number of turrets.
n	Number of different tool placements.
x	A shared tool in the system.
π	A shared station in the system.
$p_{not_occ}(x, \pi)$	The probability that a tool x is not assigned to station π in a single tool placement.
$p_{occ}(x, \pi, n)$	The probability that a tool x is assigned to station π at least once in n different tool placements.

In order to have a tool assigned to each station at least once with probability p_{occ} , we need a population size of $(4T \times N)$ because there are $4T$ shared stations.

We consider only the assignment of the shared stations because the number of permutation of the shared stations greatly exceeds that of the unique stations. Since the placement of unique tools are independent of the shared tools, if a shared tool is to appear at least once in every shared station for a population of N size, the same must be true for the unique tools.

By selecting a population of size n , each tool is expected to be assigned to a station at least once in the initial population with probability of p_{occ} . Hence, all schema of defining length 1 are also expected to be represented in the initial population with the same probability. That is, since *GA* is supposed to work through the combination of short building blocks of high fitness, it is hoped that by having every tool mapped to each station once, the chance of any such building blocks being present in the initial population would be increased.

5.1.1 Encoding Format

The chromosome-encoding format used to represent tool to station assignment in the *ITAS* is a 1-dimension array with the same number of elements as the number of stations present in the *ITAS* as shown in Figure 8. S_S represents a shared station; S_A represents a unique station for the assembly process A; and S_B represents a unique station for assembly process B. T is the number of turrets in the system, Q_i is the i^{th} tool.

Figure 8. Chromosome Encoding Format for ITAS

The index of the array represents the tool number and the value of the array, the station that the tool was assigned to. That is, if element Q_4 of the chromosome contains the value 8, this means that tool #4 is to be assigned to station #8. The stations are numbered as shown in Figure 9, with r being the turret number, which starts at 1.

Figure 9. Numbering of Station

In the implementation, the initial tool assignment maps each tool to the station number that is the same as the tool number (i.e. tool #1 maps to station #1). This means that tool $\{Q_3, Q_9, \dots, Q_{6T-3}\}$ and $\{Q_6, Q_{12}, \dots, Q_{6T}\}$ are unique tools of assembly process A and B respectively.

5.1.2 Mutation Operator

The mutation operator for the *GA* does a swap between random pairs of tool assignment. Once a chromosome is determined to undergo mutation, a random tool Q of the chromosome is picked. If the tool happens to be a unique tool, another unique tool belonging to the same process as the Q is randomly selected. Otherwise, a shared tool is selected and the station assignments of the two tools are swapped.

In this case, the mutation rate for the *GA* refers to the probability that a swap will occur within a chromosome and not the overall probability that a genes will be changed.

5.1.3 Crossover Operator

For a permutation-based *GA*, it was observed that the fitness of the chromosome is affected by two factors, namely, the absolute position that a value appears in the chromosome, the relative order of values, or a combination of the two. This is because in a permutation-based *GA*, all genes that a chromosome takes on are unique and known beforehand.

In the case of the *ITAS*, we observed that if we ignored the presence of the unique stations, and assumed that the transfer time was the same as the travelling time, then a group of tool placements would have the same travel path if the order of tools were the same except for tools that were assigned to the first or last station of the C-path.

For example, in the tool placements in Figure 10, two assembly sequences that make use of only tool 2 to 5 will have the same travel pattern because the relative distances between the tools are the same.

Figure 10. Travel Pattern of Tools

However, the presence of the unique stations, the differences between travelling time and transfer time, and the relative order of the stations affect the production time of the *ITAS*. Thus, the Partially Matched Crossover (PMX) operator was chosen as the crossover operator for the *ITAS* because this operator attempt to preserve the absolute position of the parent chromosomes.

5.2 Implementation II

5.2.1 Encoding Format

The main difference between the second implementation and the first is the choice of the encoding formation. The second implementation uses the same 1-dimension array as the first, but instead of each element of the array containing the station number (S_i), the array now contains the tool number assigned to the station. The encoding format of the Implementation II is shown in Figure 11.

The index of the array element represents a station number. Like in the first implementation, tools unique to each process are located only at the unique stations. Q_s represents a shared tool. Q_A and Q_B represent unique tools for process A and process B respectively.

S_1	S_2	S_3	S_4	S_5	S_6		S_{6T-5}	S_{6T-4}	S_{6T-3}	S_{6T-2}	S_{6T-1}	S_{6T}
Q_s	Q_s	Q_A	Q_s	Q_s	Q_B	...	Q_s	Q_s	Q_A	Q_s	Q_s	Q_B

Figure 11. Chromosome Encoding Format for ITAS

5.2.2 GA Operators

Since the location of the unique tools remain unchanged, the GA uses the same crossover and mutation from the first implementation.

The main difference is that the semantic of the crossover operator has changed somewhat. In this implementation, the operation of the GA is not to group commonly used tools together, but to group commonly visited stations together. The key different that this change brings about is that while tools may not be used in order, the movement between the stations follows a fixed C-Path.

Figure 12 and Figure 13 show the effect that swapping a pair of tools/stations has on the distances among the tools (in term of number of stations) within the shared tools/stations. The change in the distance between the tools is enclosed in brackets.

When a pair of adjacent tools is swapped (i.e. when the array index is the station number), the change in distance between the swapped tools and the rest of the tools is 1 station. However, when a pair of adjacent stations is swapped, the change will be affected by differences in their station assignments.

Tool	1	2	4	5
Station	5	2	1	4

Tool	1	2	4	5
Station	2	5	1	4

Distance between Tools				
Tool	1	2	4	5
1	0	3	4	1
2	3	0	1	2
4	4	1	0	3
5	1	2	3	0

After Swap →

Distance between Tools (Changes)				
Tool	1	2	4	5
1	0	3	1 (3)	2 (1)
2	3	0	4 (3)	1 (1)
4	1 (3)	4 (3)	0	3
5	2 (1)	1 (1)	3	0

Figure 12. Distance Matrices for Implementation I.

Station	1	2	4	5
Tool	5	2	1	4

Station	1	2	4	5
Tool	2	5	1	4

Distance between Tools				
Tool	1	2	4	5
1	0	2	1	3
2	2	0	3	1
4	1	3	0	4
5	3	1	4	0

After Swap →

Distance between Tools (Changes)				
Tool	1	2	4	5
1	0	3 (1)	1	2 (1)
2	3 (1)	0	4 (1)	1
4	1	4 (1)	0	3 (1)
5	2 (1)	1	3 (1)	0

Figure 13. Distance Matrices for Implementation II.

5.3 Production Time versus Mutation Rate

Based on the findings in [Chung02], an experiment was conducted to verify that the performance of the GA improves with increased mutation rate based on Implementation I using the crossover rate $p_c = 1 - p_m$. This means that the new population should be entirely made up of the crossover or mutation of parent's

chromosome(s) and the elitist member of the parent's population. The experiment for each p_m value was run 10 times and the average of the 10 runs was taken. The results of the experiments are shown in Figure 14 and Figure 15.

From the graphs, it can be observed that a high mutation rate gives better results compared to a high crossover rate. With no mutation, the production time obtained is longer than when no crossover operator was applied. This suggested that the effect of the crossover operator is inferior to that of the mutation operator. The test case with 4 turrets was affected more by the changes in mutation rate compared with 3 turrets.

Figure 14. Production Time versus Mutation Rate for 3 Turrets (Implementation I)

Figure 15. Production Time versus Mutation Rate for 4 Turrets (Implementation I)

The same experiment was repeated for Implementation II with the results shown in Figure 16 and Figure 17. The change in encoding format seems to have an inverse effect on the mutation rate compared to Implementation I. The GA with a lower mutation rate now performs better compared to before where mutation was the preferred operator. Overall performance of the GA has improved, with the best production time of 80356 compared to 88637 for the case of 3 turrets and 155468 compared to 158240 for the case of 4 turrets.

Figure 16. Production Time versus Mutation Rate for 3 Turrets (Implementation II)

Figure 17. Production Time versus Mutation Rate for 4 Turrets (Implementation II)

5.4 Discussion

5.4.1 Implementation I

For Implementation I, it is noted that, if the effect of collisions is ignored, each gene i in the chromosome contributes (based on the QAP equation in Section 2.6.3):

$$f_{gene}(i) = \sum_{j=1}^N a_{ij} b_{\pi(i)\pi(j)}$$

Thus, when a single gene of a chromosome is copied over, the full information of the gene is not copied because $b_{\pi(i)\pi(j)}$ depends on the assignment of the other tools.

Let l denote the number of genes transmuted from the parent chromosome, and L denote the total length of the chromosome. A total of l^2 summation can be performed with l genes, so the ratio of information transmuted from the original parent to the offspring is $\frac{l^2}{L^2}$. Thus, the amount of information from the parent increases as a square of the number of genes copied from the parent.

Therefore, while the PMX copies genes from its parent and by doing so, preserves part of the travelling time incurred by the parent tool assignment, some of the information is lost because of the property of the QAP equation. For example, if $L=12$, and $l=6$, only 25% of information from the parent chromosome are preserved even though half the genes were copied.

Hence, the poor performance of the crossover operator in implementation I appeared to be due to the problem with the chromosome representation and not the crossover operator used.

5.4.2 Implementation II

To understand the cause for the change in behaviour brought about by the new encoding format, an experiment was conducted to observe the fitness landscape in the neighbourhood of a chromosome. The experiment was repeated with a few other randomly generated chromosomes to verify that the characteristic displayed is not an isolated case. The results of the repeated experiments showed similar graphs as those in Figure 18 and Figure 19 for every chromosome.

The graphs show the production time, sorted according to the swap distance, of the set of the neighbourhood tool placements of a randomly chosen tool placement Z (for a case with 4 turrets) using Implementation I and II respectively. The swap distance is the number of stations between the pair of tool assignments that were swapped.

Figure 18. Neighbour of Tool Placement Z (Implementation I)

Figure 19. Neighbour of Tool Placement Z (Implementation II)

The distribution of the production time seems to be uniform across the entire neighbourhood in Implementation I and does not seem to be affected by the swap distance. For Implementation II, the distribution of production can be observed to vary according to the swap distance. A larger general change in production time can be expected for pairs of swaps which are located farther away from each other.

The minimum, average and standard deviation (in production time) of the neighbourhood are 162029, 190981 and 4475 for the implementation I and 158785, 185791 and 5051 for implementation II. This suggests that the difference in encoding format changes the distribution of values in the landscape as well. Thus, it is believed that the ability of the crossover operator in *GA* to perform well is partially dependent on the encoding format used.

In the case of the *ITAS*, we see that a chromosome receives a block of 4 genes during a crossover. The change in the production time is likely to be smaller for Implementation II. This is because the change in the 4 genes can be considered a 2 pairs of swaps with the distance of 3 and 1 (1st with 4th, 2nd with 3rd), 2 and 2 (1st with 3rd, 2nd with 4th), or 1 and 1 (1st with 2nd, 3rd with 4th). Figure 19 suggests that the likelihood of a large change in production for short swap distance is smaller than that of Implementation I. Thus, this allows higher order building blocks of the chromosome to survive the crossover operation.

5.5 Conclusion

For *GA*, the intensification process occurs in the crossover operation. When two parent chromosomes perform a crossover operation, part of the genes of each parent is retained in the child. The resulting child chromosome can be thought of as containing some higher order schemas than the parent. For example, if a parent chromosome with a chromosome string of 11000100 produces a child chromosome with the chromosome string of 11001011, the child chromosome would bring about a new solution for the schema 1100****. In this way, more points from the schema 1100**** is being explored and an intensification of the schema occurs. The mutation operator also helps in the intensification process by introducing small changes to the chromosome, resulting in more solutions from a high order schema to be explored.

Interestingly, the same crossover process of the *GA* is responsible for the diversification process. Using the same example in the previous paragraph, the schema 11001011 can also be thought of as a diversification, as the new chromosome is not a member of the 11000*** contained in the parent chromosome.

Therefore, depending on the order of the building blocks presented, the effect of diversification may overwhelm that of intensification as the diversification process may cause high order building blocks to be broken up. This may be why the difference in encoding format changes the effect of the crossover operation. The proper choice of a crossover operator and crossover probability is thus important for *GA*.

6 Simulated Annealing for *ITAS*

6.1 Implementation

The neighbour moves of the Simulated Annealer were generated by randomly swapping a pair of tool to station assignments.

To determine k in the Boltzmann's equation $P_{acc}(\Delta E) = \begin{cases} \exp\left(\frac{E_i - E^*}{kT}\right) & E_i < E^* \\ 1 & E_i \geq E^* \end{cases}$ in

Section 3.2.1, the standard deviation (in production time) of an initial population consisting of randomly generated solutions was computed at the start of the algorithm.

The size of the population is the same as that used in the *GA* implementation.

This use of 1 standard deviation as the scaling factor was based on the recommendation in [CSEP91].

The standard deviation is then used as the scaling factor. This was done because the number of turrets and assembly steps in the *ITAS* depends on the actual assembly plans used, thus a constant k is unlikely to keep up with the changing parameters.

k is also used as the number of iterations the Simulated Annealer must complete before changing its temperature. This is to allow the Simulated Annealer to visit all its neighbour state with a probability of 0.95.

6.2 Production Time versus Cooling Factor

6.2.1 Experiment

The main variable that affects the *SA* is the choice of the cooling schedule which is determined by the scaling factor k and cooling factor α .

To determine suitable values for k and α , the same pair of random assembly plans of 2, 3 and 4 turrets used in the experiment from Section 4.2 were simulated with different values of α and k . Each value of α and k was ran for 10 times to obtain the average production time needed to complete the assembly of 100 units of each product. The average production time of the 10 runs is recorded in Figure 20.

6.2.2 Results

The results of the production time versus cooling factor are as shown in Figure 20. To plot the graph, each value of x in the set of results X is normalized using the formula

$$\frac{x - \min(X)}{\max(X) - \min(X)}.$$

The *SA* was able to converge close to the production time of 47971 for all values of α in the case of 2 turrets. For an *ITAS* with 3 and 4 turrets, the production time improved with the increase in α , and seemed to peak around 0.8 to 0.9. This is consistent with the finding in [Skiena97].

In addition, it was found that for all three cases, when the value of k is changed to 2,3,4 times of the standard deviation computed from an initial randomly generated

population. There were no significant improvement over the production time as compared to just using the standard deviation.

$\alpha \backslash T$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
2	48278	47971	47971	47971	48265	47971	47971	47971	47971	47971
3	91628	90192	89388	89181	88989	88907	88910	88779	88594	88594
4	162012	1588775	157679	156945	156310	155632	155262	154372	154124	153751

α : Cooling Factor, T: Number of Turrets

Figure 20. Production Time of Simulated Annealing versus Cooling Factor

6.3 Production Time versus Number of Fitness Evaluation

An experiment was conducted to review the performance of the Simulated Annealer with respect to the number of fitness evaluations performed.

The results of the experiment are shown in Figure 21. As in Section 6.2.2, the graph is

plotted using normalized production time with the formula $\frac{x - \min(X)}{\max(X) - \min(X)}$.

$\begin{matrix} m \\ T \end{matrix}$	10	20	30	40	50	60	70	80	90	100
2	48240	48004	48012	47979	47973	47973	48003	47990	47987	47981
3	91526	90829	90075	89415	88836	88787	88787	88596	88583	88736
4	164903	161477	158936	156906	15554	154364	154535	153993	154183	154149

m : Number of Fitness Evaluations (In multiples of population size)

T: Number of Turrets

Figure 21. Production Time of Simulated Annealing versus Number of Fitness Evaluations

From Figure 21, the Simulated Annealer converges after (50 x population size)^[1] fitness evaluation for the two test cases.

Thus, the Simulated Annealer appears to work for 3 and 4 turrets with about the same normalized performance when the number of fitness evaluation is scaled by the population size.

6.4 Effect of Change in Encoding Format on Simulated Annealing

To investigate the effect of changing the encoding format observed in Section 5.4, the experiment in Section 6.3 is repeated (for 3 and 4 turrets) with the encoding format described in Section 5.2. The result of the experiment is shown in Figure 22.

No major change in the performance of the Simulated Annealing was observed with the change in encoding format. A possible reason why the Simulated Annealer is unaffected by the change in encoding format is that Simulated Annealing selects the neighbour solution randomly. The condition for a new solution to be accepted is dependent only on the Boltzmann's equation (see Section 3.2.)

^[1] The population size for 2, 3, 4 turrets is 184, 420 and 736 respectively.

$\begin{matrix} m \\ T \end{matrix}$	10	20	30	40	50	60	70	80	90	100
3	91522	90816	89988	89459	88896	88922	88756	88584	88487	88388
4	165002	161644	158847	156982	155604	154314	154502	154186	154293	154033

m : Number of Fitness Evaluations (In multiples of population size)

T: Number of Turrets

Figure 22. Production Time of Simulated Annealing versus Number of Fitness Evaluations

(Implementation II)

6.5 Conclusion

For SA, the intensification process occurs due to the acceptance policy. Since the acceptance of new solutions is dependent on the difference between the current and neighbour fitness for poorer performing solutions, a region which has smaller change in fitness is more likely to keep the SA within that region because the moves have a higher probability of being accepted.

Like *GA*, the diversification process of the *SA* is also controlled by the same process which controls the intensification process. However, as *SA* accepts poorer solution with a certain probability based on the solution's fitness and the current annealing temperature, the chance for the *SA* to diversify to other regions is independent of the encoding format used. The consistent results that were obtained from the experiments suggest that *SA* is a robust meta-heuristic algorithm that can be easily implemented for most problems.

7 Tabu Search for *ITAS*

7.1 Implementation

The neighbour set of a solution s in the *ITAS* is the set of solution that differs from it by one pair of tool assignment. Thus, each solution in the neighbour set can be reached by swapping a pair of tool assignments within s .

Since there are $4T$ ($T = \text{number of turrets}$) shared stations, and each of these stations may swap with all other shared stations other than itself and that swaps are symmetric, there are $4T(4T-1)/2$ neighbour solutions within 1 swap (of shared stations) from the current solution. Inclusive of the unique stations of each process, there is a total of $4T(4T-1)/2 + 2(T)(T-1)/2$ solutions in the neighbour set.

For the *ITAS*, the aspirant set is chosen as the set of all solution that has a production time that is better than the current solution by k standard deviation calculated from a randomly generated initial population. The effect of k is discussed in Section 7.5.2, with $k=0$ being the case where no aspirant set is used. The size of the population used is the same as that of the *GA* implementation.

7.2 Tabu List and Tenure

The use of the Tabu list is that it prevents cycles shorter than the size of the Tabu list from occurring by keeping track of recently visited solutions. If a move will result in a solution that matches one of the items on the tabu list, the move is forbidden unless there is a strong advantage in doing so.

In the Tabu implementation for the *ITAS*, moves in the tabu list are not given explicit Tabu tenure; instead, a move is tabu so long as it is in the Tabu list. The oldest tabu move is removed from the Tabu list whenever the list is full and a new move needs to be added to it, or when no non-tabu moves are available.

The maximum size of the tabu list is chosen to be the same as the population size in *GA*.

7.3 Diversification Strategies

7.3.1 Non-Improving Move

When no improving or aspirant moves are available, a non-improving, non-tabu move is selected from the current neighbour set. Two possible schemes for this selection are: best non-improving and worst non-improving.

In the best non-improving move, the next move selected is the best move in the current neighbour set that is not marked tabu. In the worst non-improving move, the worst solution that is not marked tabu in the current neighbour set is selected.

In both cases, the current pair of tool assignments to be changed for the move is then marked tabu to prevent the Tabu search from moving back to the current solution from the new solution.

7.3.2 Random Restart

Besides the non-improving move strategy, a random restart strategy is applied that will randomly choose a starting point if no improving moves were found for the past R moves. The purpose of this is to allow the Tabu search to explore more regions rather than remain in a region that shows no sign of having a better solution.

A large value of R is expected to allow a more thorough check of the region, while a smaller R allows more explorations. It is believed that R is likely to be affected by the number of turrets involved since the more turrets there are, the larger the neighbour set. Hence, more explorations may be needed for the neighbour set.

7.4 Tabu Attribute

Two possible choices are available for use as the tabu attribute: the information of a move (i.e. the pair of tools and their current station assignment), or all the tool assignment of a solution.

The choice of the Tabu attribute may affect the Tabu list size. If the full tool assignment is used as the Tabu attribute, only that solution will be made tabu. When a pair of tool assignments is made tabu, all solutions with the matching pair of tool assignments will be made tabu and is therefore more restrictive.

7.5 Experiments

The following experiments were conducted with a maximum tabu list size equal to the initial population size. In each test case, the number of fitness evaluations is set to (100 x population size). Each experiment is repeated 10 times and the average of the 10 experiments is used.

7.5.1 Best Non-Improving versus Worst Non-Improving Move

To determine the difference in performance of the various tabu attributes and strategies, experiments were conducted for the different tabu attributes and strategies without the use of an aspirant set.

The average production times and standard deviations of the experiment are as follows:

Tabu Attribute, Strategy	Average Production Times			
	3 Turrets		4 Turrets	
	Average	Standard Deviation	Average	Stand Deviation
Full, BNI	89590	784	157355	1782
Full, WNI	89025	1080	156716	2359
Pair, BNI	89322	707	156994	1531
Pair, WNI	89074	1005	156817	2237

WNI: Worst Non-Improving, BNI: Best Non-Improving

Table 1. Effect of Tabu Attribute and Diversification Strategy

In all the test cases, it was observed that the best non-improving scheme was inferior to the worst non-improving scheme in generating better solutions. A lower standard deviation was also observed for test cases using the best non-improving scheme. This behaviour was expected because the worst non-improving strategy attempts to diversify the search further away from the current solution, thus allowing more regions to be explored.

For the worst non-improving move strategy, better results are produced when using the full tool assignment as the tabu attribute compared to using the information of tool assignment pairs. The reverse is true when the best non-improving move strategy was used.

A probable reason for this could be due to that the tool assignment pairs make more regions tabu, forcing the tabu search to explore a larger region. However, it is a paradox that the *pair BNI* has a smaller standard deviation than the *full BNI*.

7.5.2 Effect of k

To determine the effect of k (see Section 7.1), an experiment was conducted with k having the value of 0, 1 and 2 for the different tabu attributes and strategies. Each experiment is repeated 10 times and the average is used. The results of the experiment are shown in Table 2.

When $k=0$, all moves that are better than the current solution is accepted as an aspirant move.

From Table 2, it can be seen that for most cases, a larger k results in a shorter production time. The production times obtained were also longer than when no aspirant set ($k = 0$) was used.

A trace through the execution of the search shows that the aspirant moves often causes the Tabu Search to revisit past solutions. This is especially so when they occur just after a diversification move.

Tabu Attribute, Strategy	Average Production Times					
	3 Turret			4 Turret		
	$k=0$	$k=1$	$k=2$	$k=0$	$k=1$	$k=2$
Full, BNI	90587	89733	89898	160450	158967	157621
Full, WNI	90380	89145	89021	160327	156852	156238
Pair, BNI	90597	89425	89925	160486	158744	157377
Pair, WNI	90446	89277	89137	160334	157516	156625

Table 2. Effect of k

7.5.3 Effect of R

To determine the effect of R , an experiment was conducted with the value of b varied from 0 to 1, where $R = (b \times \text{neighbourhood size})$. This corresponds to an R value of 0 to the size of the neighbourhood ($4T(4T-1)/2 + 2(T)(T-1)/2$) (see Section 7.3.2). For each value of b , 10 experiments were conducted and their average was used. The results of the experiments are shown in Figure 23 and Figure 24.

Figure 23. Effect of R (3 Turrets)

Figure 24. Effect of R (4 Turrets)

With $b=0$, random restarts are taken whenever no non-improving moves can be found. This probably prevents the Tabu Search from performing a proper intensification in the search region it is currently in. This explains why there is a mark improvement when $b>0$ is used.

Tabu Search implemented using the *full* attribute seems to be affected more by the change in b . This might be because the probability that a move is forbidden due to being on the tabu list is lower compared to *pair* attribute. Thus, the *full* attribute tabu search is likely to perform more detailed search of the neighbour. Due to this, a smaller b allows the *full* attribute Tabu Search to escape from a potentially poor region earlier.

7.5.4 Effect of Encoding Format on Tabu Search

To investigate if the Tabu Search is affected by the change in the encoding format described in GA's Implementation II, the experiments in 7.5.1 were repeated. The results of the experiments are as follows:

Tabu Attribute, Strategy	Average Production Times			
	3 Turrets		4 Turrets	
	Average	Standard Deviation	Average	Stand Deviation
Full, BNI	89684	826	156377	1848
Full, WNI	88917	1133	155168	2528
Pair, BNI	89471	867	155653	1870
Pair, WNI	89107	1149	155408	3193

Table 3. Effect of Tabu Attribute and Diversification Strategy

Comparing Table 2 and Table 3, there do not appear to be much changes in the Tabu Search performance for 3 turrets. However, an improvement was seen for the case of 4 turrets. Considering that the search strategies implemented are generalized exploration techniques which should work with any landscape, the improvement seen for the case of 4 turrets was unexpected.

7.6 Conclusion

The local optimal is often quickly reached due to Tabu Search's aggressive intensification process which seeks out the best possible solution in each neighbourhood. Thus, to allow Tabu search to sufficiently explore the search space, the choice of a diversification strategy is important. This can be seen in the *ITAS* by the better performance of strategies which favour diversification such as the Worst Non-Improving Move and the Random Restart with a properly chosen value.

8 Ant Colony Optimisation

8.1 Pheromone

Two different types of pheromone were used in the implementation. The first pheromone ρ_s^Q stores the expected production time based on the absolute positional information of tools. That is, the expected production time of tool placement which has tool Q at the station s . Pheromone values are kept in a 2-dimensional array which has rows representing the station number and columns representing the tool number. The Q^{th} column of s^{th} row contains the pheromone value for which tool Q appeared at station s .

The update rules for this type of pheromone are:

$$\begin{aligned} \rho_s^Q &= \rho_s^{Q'} + \frac{1}{m} \sum_i^m (T_{C_i} - \rho_s^{Q'}) & | \quad m > 0 \\ \rho_s^Q &= \rho_s^{Q'} & | \quad m = 0 \end{aligned}$$

where $\rho_s^{Q'}$ is the current value of the pheromone, m is the number of solutions in the ant population that has tool Q at station s . T_{C_i} is the production time of the i^{th} solution.

The second type of pheromone σ_v^Q stores the expected production based on the relative order of tools instead. That is, it stores the expected production time of tool placement which has tool Q located at the next station when the tool located at the current station is v .

The update rules for this type of pheromone are the same as the first type of pheromone mentioned earlier. Like the case of the first implementation, the storage of the pheromone is a 2-dimensional array. However, the rows now represent the current tool and the columns represent the next tool. An additional row is added to store the pheromone value of the initial edge (i.e. before the tool for the first station was selected). The only change to the update rule is that it is now based on the current tool instead of the current station. i.e.

$$\begin{aligned} \sigma_v^Q &= \sigma_v^{Q'} + \frac{1}{m} \sum_i^m (T_{C_i} - \sigma_v^{Q'}) & | \quad m > 0 \\ \sigma_v^Q &= \sigma_v^{Q'} & | \quad m = 0 \end{aligned}$$

The initial value of all the pheromones is set to the average production time computed from a randomly generated set of tool placements. Thus, the effect of the rules is to adjust the pheromones towards the average production time observed for the edges in the current tool placements. For edges not in the current tool placements, no update is done.

8.2 Diversification using Blind Ants

Evaporation was not used in this implementation because the pheromone used is in term of the differences between the current pheromone value and the average production time for the edge, which should automatically adjust itself. So to achieve the effect of diversification brought about by the evaporation process, “blind” ants, which are randomly generated tool placements, were used to augment the exploration process instead.

8.3 Construction of Trails

The trails used in the *ACO* implementation of the *ITAS* are 1-dimensional arrays consisting of the station-to-tool mapping as shown in *GA*'s Implementation II.

To construct the trail, a selection process starts off with an empty list and selects a tool from list L which is one of L_S, L_A, L_B . The lists L_S, L_A, L_B initially contain the full set of shared tools, unique tools for process A and unique tools for B respectively.

Once a tool is selected for the construction of the trail, it is removed from L . The probability $P(Q, s, v)$ of assigning a tool Q to the station s , if the current tool is v , is the ratio of pheromones ρ_s^Q, σ_v^Q to the sum of pheromones ρ_s^k, σ_v^k of the remaining tools in L . The sum of the pheromones ρ_s^k, σ_v^k of the remaining tools in list L shall be denoted as ω_v^s .

Since pheromone values are in terms of production time whereby a smaller value is better, they are converted by subtracting from the maximum and minimum value for ω_v^s (e.g. $\rho_s^{\max} + \rho_s^{\min} - \rho_s^Q$).

Therefore, the equation for $P(Q, s, v)$ is:

$$P(Q, s, v) = \frac{\alpha(\rho_s^{\max} + \rho_s^{\min} - \rho_s^Q) + (1-\alpha)(\sigma_v^{\max} + \sigma_v^{\min} - \sigma_v^Q)}{\sum_{k \in L} [\alpha(\rho_s^{\max} + \rho_s^{\min} - \rho_s^k) + (1-\alpha)(\sigma_v^{\max} + \sigma_v^{\min} - \sigma_v^k)]}$$

The distribution of the contribution of each pheromone type in the selection process is controlled by the value of α . When $\alpha = 0$, the selection process will be purely based

on the relative positional information; when $\alpha = 1$, the selection process will be purely based on the absolute positional information.

Some ants are constructed as blind ants which are randomly generated tool placements.

After all the ants are constructed, the production time of each solution is evaluated. The result of the evaluation is used to update the pheromone using the update rule in Section 8.1. The total number of ants in the system is chosen to be the same as the population size used in *GA*.

8.4 Experiments

The following experiments were conducted with 100 sets of ants for exploration. Each experiment is repeated 10 times and the average of the 10 experiments is used.

8.4.1 Absolute versus Relative Positional Pheromone

To determine whether the absolute or relative positional information of tool placement is more important to the *ACO*, the value of α (alpha) is varied from 0 to 1. The results of the experiments are shown in Figure 25.

From the graphs, a general decrease in production time can be observed when the value of α tends towards 1. This corresponds to the increased use of the pheromone with absolute positional information. This supports the observation that the absolute

positions of tool placements are important, perhaps because of the presence of unique tools.

Another observation is that the combined effect of both types of pheromone has an adverse effect on the performance of *ACO*. This can be seen by the pure absolute positional pheromone has a better performance compared to either the use of purely relative positional pheromone or a mixture of both.

Figure 25. Effect of α (3 Turrets).

8.4.2 Ratio of Blind Ants

One of the concerns in the development of the *ACO* implementation is the effect of the blind ants.

Figure 26. Change in Production Time w.r.t Ratio of Blind Ants (3 Turrets)

Figure 27. Change in Production Time w.r.t Ratio of Blind Ants (4 Turrets)

From Figure 26 and Figure 27, the production time of the *ACO* is seen to fluctuate with the change in the ratio of blind ants. The amplitude of the oscillation in production time for the case of 3 turrets does not seem to be affected by the ratio of blind ants.

In terms of performance, the case of 3 turrets (Figure 26) seems on par with Tabu Search (Table 1 to Table 3) and Simulated Annealing (Figure 21 and Figure 22), while the case of 4 turrets (Figure 27) is worse than both Tabu Search and Simulated Annealing.

There are two possible explanations for the oscillations. The first explanation is that the combination of good individual edges does not necessarily lead to the formation of a good solution and the *ACO* is not converging. However, the hypothesis contradicts with the finding that the choice of α has an effect on *ACO*'s performance.

.

The second explanation is that the *ACO* is capable of training itself using both bad and good solutions. Thus, the *ACO* fluctuates, or constantly adjusts itself to compensate for the error in its estimate. This is likely, as Figure 27 shows that when the ratio of the blind ants increases, the oscillations die down. This is because when the ratio of blind ants is high, the pheromones of the edges are effectively not being used.

8.5 Hybrid *ACO* + *GA*

8.5.1 Implementation I

8.5.1.1 Experiment

The hybridization of *ACO*+*GA* was first proposed by [Acan2002]. In this implementation, the blind ants in the *ACO* were replaced by a *GA*. The results of the crossover operation are fed into the *ACO* to adjust the weights of the pheromone and solutions from the *ACO* participate in crossover process as well. This was done because the *ACO* appears to be relatively unaffected by the change in the ratio of blind ants and that the *GA* Implementation II has the best performance among all the algorithms.

To observe the effect of the use of *GA*, the experiments in Section 8.4.2 were repeated. The results of the experiments are shown in Figure 28 and Figure 29.

8.5.1.2 Results

From the graphs, it can be observed that increasing the ratio of the *GA* to ants, i.e. the number of solutions generated via *GA* crossovers, above 0.6 leads to a sharp increase in the performance of the algorithm. In addition, the performance of *ACO*+*GA* at 0.9 is about the same as the best performance of the pure *GA* implementation II (Figure 16 and Figure 17). This suggests that the use of the *ACO* did not improve the performance of the *GA*. Any change in performance is likely to be due to the *GA* crossover process.

Interestingly, since the selection and crossover process of *GA* is based on the entire population, which includes solutions generated by both the *ACO* and *GA*, the *GA* should have been able to produce fitter chromosomes from two fit parents that were generated by the *ACO* as well.

Another observation is that some kind of oscillations was present when *ACO* was not combined with *GA*, but the amount of oscillations becomes smaller when *GA* becomes the dominant component in the *GA+ACO* implementation. This is logical as the performance of *ACO+GA* tends to follow the dominant (*ACO* or *GA*) algorithm.

Figure 28. Production Time versus Ratio of *GA* to Ants (3 Turrets)

Figure 29. Production Time versus Ratio of GA to Ants (4 Turrets)

8.5.1.3 Discussion

While it was expected that the combination of GA and ACO would bring about better results, it did not. In this hybrid ACO+GA implementation, the ACO acts as a second cross-over operator of the GA, whereas GA acts like a second trail construction mechanism in the ACO. [Stutzle99] reported that ACO was most successfully used when it was combined with a local search when applied to Quadratic Assignment Problems because “constructive algorithms often results in a poor solution quality compared to local search algorithms”. [Stutzle99] also proposed the use of pheromone for modification of edges rather than the construction of solutions to overcome this problem.

For the ITAS, observation from Sections 5.2 and 5.4.2 also shows that there could be significant change in Production Time when pairs of tool-assignments are changed. Thus, just because tool 1 was assigned to station 1 resulted in a good solution does not mean that good solutions will be obtained from combining other tool assignment with tool 1 initially assigned to station 1. There has to be a larger inter-dependency between the tools. This inter-dependency, which is missing in ACO, is present in the GA as the set of station-tool assignment from each parent used during crossover.

8.5.2 Implementation II

8.5.2.1 Pheromone-based Cross-Over Operator

To increase the inter-dependency in Implementation I, we hybridized the algorithms by changing the GA cross-over operator to make use of information generated by the ACO pheromones.

To generate a new child chromosome, the cross-over operator picks the station-tool assignment with the highest pheromone value from one parent and fills the remaining tool-station assignments using chromosome from the second parent in the same manner defined by the Partially Mapped Crossover operator.

8.5.2.2 Experiments

To confirm that the performance of Implementation II is better, the same experiments in Section 8.5.1.1 were performed. However, instead of the ratio between GA and ACO generated solutions, the ratio is now the number of solutions generated by the ACO pheromone-based crossover (APX) and that of the PMX. The results of the experiments are shown in Figure 30 and Figure 31.

Figure 30. Production Time versus Ratio of APX in GA population (3 Turrets)

Figure 31. Production Time versus Ratio of APX in GA population (4 Turrets)

8.5.2.3 Discussion

It is interesting to observe that there are now two stages of selection; one at the chromosomes level and another at the genes level.

The new APX operator let genes identified by the ACO to be interesting to have a higher chance of being part of the new population. In this way, the APX forces the GA to perform more intensification based on its pheromone values.

8.6 Summary

In the implementation of the *ACO* for the *ITAS*, a self-adjusting update rule for pheromone was used. The performance of this update rule appears to be relatively unaffected by the amount of random exploration that the ants perform. The effect of α suggests that the importance of absolute tool placement information is more important compared to the relative tool placement information.

In the Hybrid *ACO+GA* implementation, the performance of the algorithm did not appear to have improved from *GA* implementation II. Therefore, to use *ACO* for the *ITAS* problem, an *ACO* using only the absolute positional information for pheromone with no blind ants is recommended.

9 Conclusion and Future Work

9.1 Comparison of Algorithms

The best production time of the various algorithms, based on the experiments conducted so far, are shown in Table 4.

	Algorithm	<i>GA I</i>	<i>GA II</i>	<u><i>SA</i></u>	<i>TS</i>	<i>ACO I</i>	<u><i>ACO II</i></u>
3 turrets	Time	88637	80356	88388	88917	83490	79654
	Rank	5	2	4	6	3	1
4 turrets	Time	158240	155468	153751	155168	155712	154608
	Rank	6	4	<u>1</u>	3	5	<u>2</u>
Average Time		123438.5	117912	121069.5	122042.5	119601	117131
Rank		6	2	4	5	3	1

GA I Genetic Algorithm Implementation I

GA II Genetic Algorithm Implementation II

SA Simulated Annealing

TS Tabu Search

ACO I Ant Colony Optimisation Implementation I

ACO II Ant Colony Optimisation Implementation II

Table 4. Best Production Time Comparison of Meta-heuristic Algorithms for *ITAS*.

From the results, it can be seen that *GA I* performed worse than most other algorithms in both cases of 3 and 4 turrets. This is attributed to the chromosome encoding format which was discussed in Section 5.4.

Algorithms (such as *SA* and *TS*) that favour local search seemed to give better results for cases with 4 turrets. A probable explanation is that as the solution length (e.g Chromosome/Trails) for 4 turrets is longer, the operator they used (namely swapping of pairs) is typically less destructive compared to the operators used by other algorithms such as *ACO I* and *GA II*. The increase in the solution length also means that *SA* and *TS* have larger neighbourhoods to search in, therefore reducing the chance of being stuck in regions with local minima.

Conversely, for cases of 3 turrets, where the solution length is shorter, the effects of the crossover or construction operators in *ACO I* and *GA II* are less destructive and help to diversify the search into more regions.

9.2 ACO Pheromone-based Crossover

In contradiction to the Basic Building Blocks hypothesis, the strength of GA probably lies not only in its ability to combine basic building blocks, but also in its ability to cluster these building blocks into a higher order schema. This is something which is lacking in Ant Colony Optimisation.

The Partially Mapped Crossover operator (PMX) uses a gene selection process. The ACO Pheromone-based Crossover operator (APX) replaces that with a selection based on the pheromone values of the station-tool assignments. As a result, the GA is encouraged to intensify its search into station-tool assignments identified by the ACO as likely to be better.

The APX, unlike the ACO pheromone trail construction process, does not destroy the order and placements of the parent genes. Therefore, APX preserves any interdependency relations between station-tool assignments. These relations may be useful as higher order building blocks for the GA and probably explains how *ACO II* was able to improve upon *ACO I* and *GA II* to obtain better production time.

9.3 Future Work

The self-adjusting pheromone used in the *ACO I* and *II* is a very simple algorithm. It could potentially be replaced by more complex forms of learning networks, e.g. neural network. The use of such learning networks could lead to the realisation of the approach outlined in Section 2.2.

Through the experiments and research done in this project, the key point noted in the use of meta-heuristic algorithms is that there should be a balance between diversification and intensification strategies. The choice of the various parameters in a meta-heuristic algorithm often affects this balance, which in turn affects their performance. Thus, one possible area towards improving the performance would be to design a self-calibrating GA. A possible implementation of such a GA would

increase or decrease its population size [Eiben04], crossover and mutation rate according to changes in the population statistics.

Other possible areas of interest could be the combination of the various meta-heuristics, for example, the inclusion of the Tabu Search or *SA* as part of a local search process for the *GA* or *ACO*.

References

- [Acan02] A. Acan. GAACO: A *GA* + *ACO* Hybrid for Faster and Better Search Capability. Lecture Notes in Computer Science 2463. Springer 2002
- [Bäck00] T. Bäck (ed), D.B. Fogel and Z. Michalewicz. Evolutionary Computation 1, Institute of Physics Publishing, 2000
- [Blum01] C. Blum and A. Roli. ‘Metaheuristics in combinatorial optimisation: Overview and conceptual comparison’, Tech. Rep. TR/IRIDIA/2001-13, Universite Libre de Bruxelles, IRIDIA, 2001.
- [Beasley93] D. Beasley, D. Bull and R. Martin. ‘An Overview of Genetic Algorithms: Part 1, Fundamentals’, vol. 15, no. 2, pages 58—69, 1993.
<http://citeseer.nj.nec.com/16527.html>
- [Chung02] M.H. Chung, K.K. Lee, L.P Tay and H.W. Tan. ‘Genetic Algorithm Solution for the Intelligent Turret Assembly Systems’, 7th ICARCV, 2002.
http://www.ntu.edu.sg/home5/ps7633120j/ITAS/GASaMAC_paper.pdf
- [CSEP91] Computational Science Education Project. Mathematical Optimisation, Computational Science Education Project, 1991
<http://csep1.phy.ornl.gov/mo/mo.html>
- [David87] L. David. Genetic Algorithms and Simulated Annealing, Pitman, London, 1987.

- [Dorigo00] M. Dorigo and T. Stutzle. 'The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances', Metaheuristic Handbook, Kluwer, 2001.
- [Eiben04] A.E. Eiben. 'Evolutionary Algorithm with on-the-fly Population Size Adjustment', X.Yao et al. editors, Parallel Problem Solving from Nature, PPSN VIII, volume 3242 of LNC, Springer, pp. 41-50, 2004.
- [Fang94] H.L. Fang. Genetic Algorithms in Timetabling and Scheduling, PhD Thesis, Department of Artificial Intelligences, University of Edinburgh, 1994
<ftp://ftp.dai.ed.ac.uk/pub/user/chrisg/Fang.PhD.Thesis.ps.gz>
- [Galinier99] P. Galinier and J.K. Hao. 'Hybrid Evolutionary Algorithms for Graph Coloring', *Journal of Combinatorial Optimization*, 1999
<http://citeseer.nj.nec.com/galinier99hybrid.html>
- [Glover93] F. Glover and M. Laguna. Modern Heuristic Techniques for Combinatorial Problems, John Wiley & Sons, Inc, 1993
- [Glover97] F. Glover, M. Laguna, and F. Laguna, Tabu Search, Kluwer Academic Publishers, 1997
- [Goldberg89] D.E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning, 1989, Addison-Wesley
- [Grefenstette92] J.J. Grefenstette. "Deception Considered Harmful", *Foundations of Genetic Algorithms 2*, pages 75—91, Morgan Kaufmann, 1992.
- [Hertz92] A. Hertz, E. Taillard, and D. Werra. "A Tutorial on Tabu Search", *Proceeding of Giornate di Lavoro AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes)*, 1992
<http://citeseer.nj.nec.com/73006.html>

- [Ingber93] L. Ingber. “Simulated Annealing: Practise versus Theory”, *Mathematical and Computational Modelling*, 18(11):29-57, 1993.
http://www.ingber.com/asa93_sapvt.pdf
- [Jong89] K.A.D. Jong and W.M. Spears, ‘Using Genetic Algorithms to Solve NP-Complete Problems’, Proceedings of the Third International Conference on Genetic Algorithms, pages 124—132. Morgan Kaufmann, 1989
<http://www.aic.nrl.navy.mil/~spears/papers/icga89.pdf>
- [Karaboga00] D. Karaboga and T.P. Duc. Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks, Springer Verlag, 2000
- [Koza99] J.R. Koza. Genetic Programming III: Darwinian Invention and Problem Solving, Morgan Kaufmann Publishers, 1999
- [LagunaW1] M. Laguna, and F. Glover. “What is Tabu Search” (Web page)
http://www.tabusearch.net/Tabu_Search/What_is_Tabu_search.ASP
- [Rizzi97] A. A. Rizzi and R. L. Hollis. “Opportunities for Increased Intelligence and Autonomy in Robotics Systems for Manufacturing”, in Proc. 8th International Symposium of Robotic Research, 1997
- [Skiena97] S.S. Skiena. The Algorithm Design Manual, Springer- Verlag, 1997
- [Stutzle99] T. Stutzle and M. Dorigo. New Ideas in Optimization, McGraw-Hill, 1999.
- [Taillard93] Source code: <http://www.ms.ic.ac.uk/jeb/pub/jobshop2.txt>
- [Tate92] D.M. Tate and A.E. Smith. “A Genetic Approach to the Quadratic Assignment Problem”, *Computer & Operations Research*, 1992
<http://www.pitt.edu/~aesmith/postscript/genpaper.pdf>

- [Wolpert96] D. H. Wolpert and W. G. Macready. “No Free Lunch Theorems for Optimization”, in *IEEE Transactions on Evolutionary Computation*, 1996

<http://citeseer.nj.nec.com/wolpert96no.html>

Appendices

A. Production Time Component Distribution

Test Case for 2 Turrets:

Assembly Process A

Travel Time = 7

Transfer Time = 225

Loading Time = 825

Unloading Time = 199

Number of Steps = 42

Assembly Steps (Tool, Duration):

8 578 9 812 10 158 7 402 5 744 11 958 3 167 2 259 5 744 4 437 5 744 8 578 7 402 11 958 7 402 10
158 1 743 2 259 11 958 7 402 4 437 11 958 2 259 9 812 10 158 3 167 2 259 10 158 8 578 4 437 11 958
3 167 4 437 8 578 9 812 1 743 3 167 10 158 9 812 3 167 8 578 2 259

Assembly Process B

Travel Time = 9

Transfer Time = 697

Loading Time = 609

Unloading Time = 415

Number of Steps = 42

Assembly Steps (Tool, Duration):

11 958 12 929 5 744 6 765 2 259 10 158 2 259 5 744 1 743 12 929 1 743 7 402 1 743 2 259 7 402 11
958 1 743 10 158 4 437 12 929 2 259 7 402 6 765 8 578 4 437 12 929 4 437 7 402 5 744 2 259 7 402 8
578 4 437 1 743 4 437 12 929 6 765 12 929 6 765 8 578 5 744 12 929

Test Case for 3 Turrets:

Assembly Process A

Travel Time = 8

Transfer Time = 600

Loading Time = 393

Unloading Time = 277

Number of Steps = 70

Assembly Steps (Tool, Duration):

15 660 1 810 4 794 16 322 1 810 8 658 10 484 4 794 2 208 4 794 8 658 3 740 9 515 4 794 3 740 17
805 5 385 8 658 2 208 17 805 4 794 2 208 1 810 16 322 10 484 3 740 2 208 17 805 8 658 7 780 14 407
13 944 9 515 1 810 10 484 7 780 17 805 14 407 13 944 3 740 1 810 14 407 17 805 8 658 4 794 3 740
17 805 15 660 11 547 10 484 3 740 10 484 7 780 4 794 11 547 13 944 17 805 14 407 15 660 16 322 9
515 5 385 8 658 11 547 14 407 15 660 13 944 15 660 13 944 15 660

Assembly Process B

Travel Time = 10

Transfer Time = 217

Loading Time = 933

Unloading Time = 943

Number of Steps = 94

Assembly Steps (Tool, Duration):

11 547 1 810 13 944 10 484 6 624 7 780 16 322 10 484 8 658 2 208 4 794 10 484 18 323 12 304 16
322 10 484 14 407 2 208 16 322 8 658 10 484 7 780 16 322 5 385 2 208 17 805 6 624 11 547 6 624 8
658 12 304 16 322 13 944 6 624 11 547 12 304 16 322 5 385 17 805 7 780 2 208 16 322 6 624 8 658 7
780 13 944 2 208 4 794 18 323 13 944 18 323 11 547 18 323 7 780 11 547 7 780 4 794 17 805 13 944
17 805 18 323 2 208 8 658 17 805 2 208 16 322 17 805 6 624 17 805 10 484 12 304 6 624 11 547 10
484 7 780 1 810 5 385 12 304 17 805 8 658 17 805 5 385 14 407 18 323 7 780 16 322 8 658 18 323 14
407 17 805 5 385 4 794 7 780 10 484

**Component Contribution vs Normalized Fitness
(3 Turrets)**

Test Case for 4 Turrets:

Assembly Process A

Travel Time = 6

Transfer Time = 895

Loading Time = 610

Unloading Time = 400

Number of Steps = 92

Assembly Steps (Tool, Duration):

22 657 19 938 5 496 21 379 14 483 16 907 17 991 9 883 19 938 9 883 8 931 21 379 20 254 11 292 8
931 14 483 11 292 7 458 15 532 20 254 9 883 13 350 3 742 4 999 15 532 2 237 13 350 1 754 22 657
19 938 11 292 5 496 9 883 22 657 7 458 10 923 11 292 10 923 2 237 23 182 20 254 17 991 14 483 22
657 19 938 22 657 5 496 4 999 23 182 2 237 23 182 19 938 20 254 23 182 3 742 10 923 20 254 14 483
23 182 8 931 14 483 4 999 3 742 16 907 13 350 20 254 8 931 17 991 5 496 20 254 1 754 4 999 16 907
7 458 19 938 8 931 17 991 8 931 15 532 3 742 20 254 13 350 14 483 16 907 3 742 10 923 8 931 15
532 13 350 9 883 13 350 22 657

Assembly Process B

Travel Time = 8

Transfer Time = 194

Loading Time = 702

Unloading Time = 951

Number of Steps = 138

Assembly Steps (Tool, Duration):

4 999 5 496 10 923 19 938 13 350 20 254 5 496 14 483 1 754 6 718 5 496 4 999 24 356 19 938 8 931
18 743 19 938 7 458 13 350 10 923 6 718 24 356 4 999 13 350 5 496 18 743 11 292 8 931 20 254 24
356 20 254 13 350 16 907 7 458 4 999 19 938 1 754 8 931 10 923 14 483 18 743 14 483 24 356 2 237
17 991 8 931 24 356 8 931 11 292 16 907 8 931 7 458 24 356 10 923 12 433 20 254 11 292 8 931 19
938 6 718 24 356 14 483 4 999 13 350 18 743 22 657 17 991 8 931 13 350 17 991 2 237 16 907 12 433
19 938 17 991 22 657 6 718 19 938 4 999 5 496 18 743 7 458 14 483 13 350 10 923 22 657 5 496 10
923 1 754 20 254 18 743 12 433 18 743 11 292 18 743 11 292 22 657 14 483 7 458 1 754 12 433 17
991 19 938 10 923 19 938 20 254 6 718 2 237 17 991 2 237 1 754 24 356 20 254 23 182 4 999 19 938
20 254 4 999 6 718 18 743 12 433 14 483 13 350 12 433 10 923 11 292 20 254 5 496 2 237 16 907 24
356 19 938 5 496 12 433 8 931 19 938 7 458 17 991

B. Experiment Results for Effect of R for Tabu Search

3 Turrets				
B	Full, BNI	Full, WNI	Pair, BNI	Pair, WNI
0.1	103848	102432	103366	102416
0.2	97770	92764	97161	94495
0.3	93268	90675	93017	90904
0.4	91031	90115	91858	90425
0.5	90458	88456	90233	89030
0.6	89921	88238	88958	88537
0.7	89878	87656	89626	88374
0.8	89780	87584	89701	87674
0.9	89913	88118	89885	88391
1.0	89960	88516	90118	88775

4 Turrets				
B	Full, BNI	Full, WNI	Pair, BNI	Pair, WNI
0.1	162226	161330	162460	161924
0.2	159775	158475	160522	159312
0.3	158597	156560	159674	157162
0.4	158022	155352	158493	155709
0.5	157408	155159	157197	155276
0.6	156901	155462	156764	155778
0.7	157029	155975	156832	155991
0.8	157381	156201	157149	156317
0.9	157705	156100	156952	156325
1.0	157713	156064	156969	156276

C. Publications

Chung Meng Hee, Lee Keok Kee, Tay Leng Phuan and Tan Hock Woo, 'Genetic Algorithm for Intelligent Turret Assembly System', Conference Proceeding of the International Control, Automation, Robotics and Vision Conference (ICARCV), Dec 2002.

Tan Hock Woo, Lee Keok Kee and Tay Leng Phuan, 'Algorithm for Intelligent Turret Assembly System', Conference Proceeding of the International Conference on Information Technology and Applications (ICITA), July 2005.

Genetic Algorithm Solution for The Intelligent Turret Assembly System

Chung Meng Hee [#], Lee Keok Kee ^{*}, Tay Leng Phuan ^{*}, Tan Hock Woo ^{*}

[#] Singapore Institute of Manufacturing Technology
71 Nanyang Drive, Singapore 638075

^{*} School of Computer Engineering,
Nanyang Technological University
Nanyang Avenue, Singapore 639798

Abstract

The Intelligent Turret Assembly System (*itas*) was designed to provide a reconfigurable assembly system with a small footprint using modular units that can be fitted with a variety of tools. Due to the many possible tool placements, Genetic Algorithm was used to generate solutions that would help maximise the utilization of the *itas*.

The implemented Genetic Algorithm solution uses an array-based chromosome that corresponds to the tool-placement. To evolve the solution, a combination of Partially Mapped Crossover operator and a mutation operator that acts by swapping of genes was used.

Due to the difficulty in predicting collision in the system, the objective function was implemented through the simulation of the behaviour of the *itas*. This is believed to be due to the orthogonality that exists between different assembly processes as well as the time varying nature of the system.

1 Introduction

The Intelligent Turret Assembly System (*itas*) was designed with the aim of overcoming the deficiency of conventional assembly systems by allowing easy configuration of the system for the assembling of a variety of products, yet avoiding the large footprint requirements of typical Flexible Assembly System. To achieve this, the *itas* uses a modular design, consisting of turrets with tool-heads that can be fitted with a variety of tools.

The modularity of the *itas* allows for different structural configuration and interchangeability of tools allows for many possible tool placements, thus making it difficult for the user to determine a tool-placement order that would allow for optimal employment of the system. As such, a software system was implemented to help find tool-placements that will offer better utilization of the system.

2 Intelligent Turret Assembly System

The *itas* is made up of one or more turrets (or Unit System) and supporting devices or mechanisms such as transfer-arms, loaders and un-loaders, etc.

Each turret (Figure 1) is a physical unit that houses 6 tool-heads, 2 *Forces*, and other mechanical and electrical components.

Figure 1. A Turret (Unit System)

Tool-heads are arranged in a 3x2 configuration in the turret. Each tool-head of the turret may be fitted with a tool necessary for the assembly. As assembly operations are carried out only at these locations, the term *station* shall be used to denote locations within the *itas* where a tool-head is present.

Forces are platforms used for transporting work-piece to locations within the turret. *Forces* are driven by electromagnetic force with air-cushion (Air-Bearings) and may travel to any location within the turret. In order to maximise the performance of the conveying mechanism, there shall only be two *Forces* in a turret at any one time.

Transfer-arms are bidirectional devices used to carry work-pieces, which fasten on an upper-fixtured to or from a turret. These machineries are located between any physically adjacent corners of two turrets

Loaders and un-loaders are respectively responsible for placing the work-piece onto the *itas* and removing the assembled product. Loaders and un-loaders may be

located at any station that is accessible by the respective *Forces*.

3 Assembly Processes

With two *Forces*, the *itas* can support two assembly lines concurrently. The two assemblies may be of the same product, or of a different product each. Each assembly process shall consist of several sequential steps, each of which requires the use of a tool for a certain amount of time. To simplify the problem, concurrent assemblies of more than one unit of a product in a single process is not considered.

The order in which the tools are to be used and the amount of time needed are input into the system by the user in a text file together with timing parameters that determine the time required to transport a work-piece between adjacent stations (travel-time), the time required to transfer a work-piece across two turrets (transfer-time) and the time required to load or unload a work-piece (loading-time/unloading-time).

4 Operational Characteristic of the *itas*

4.1 Movement Path

Due to the wiring attached to each *Forces*, the movement of the *Forces* follows a 'C'-shaped path shown in Figure 2. to avoid the tangling of the wires. That is, *Forces A* travels in the path $\{0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4\}$, while *Forces B* travels in the path $\{3 \rightarrow 4 \rightarrow 5 \rightarrow 0 \rightarrow 1\}$.

Due to the C-path, there will be one station within each turret that is accessible to only one of the *Forces*, e.g. station 5 for *Forces A* and station 2 for *Forces B*.

Figure 2. Movement Path within Turret

The movement of the *Forces* are independent of each other, i.e., both *Forces* may move concurrently. The *Forces* follow the same movement path across turrets.

4.2 Collisions

A collision occurs when a *Forces* move towards a station occupied by another process, or when both

processes are moving towards the same station. Such occurrences are undesirable and should not be allowed.

Since a collision is basically a race condition between two processes trying to reach the same station, it can be resolved by resource locking. Once a station is occupied (or if a process is already moving towards it), the other process must wait until the station becomes free again.

4.3 Deadlocks

A deadlock is a special case of collision that occurs when the next station that both processes are moving to (i.e. target station) are currently occupied by each other. Hence a deadlock can be detected by checking that the target station of one process (e.g. *A*) is not the current station of the other process (e.g. *B*) and vice-versa.

Once a deadlock is detected, it can be resolved by getting one of the processes to move back to its previous station.

5 Problem Parameters

In determining the relationship of the various problem parameters, a simulator for a single turret was coded.

Due to the lack of real assembly processes, 1000 pairs of randomly generated assembly plans comprising between 18 and 24 steps, with each pair using six shared tools and one unique tool and have a processing time between 100 and 1000 time-units, were used in an attempt to simulate an 'average' assembly process.

5.1 Comparison Criteria

The criteria for determining the efficiency of a tool-placement is based on the time required to assemble *X* unit of a product (i.e. production time) using the respective tool-placement.

The production time is used because it is believed that the speed of the assembly process remains an important factor in the manufacturing industry.

5.2 Components of Production Time

Based on the observation of how the processes flow and the behaviour of the *itas*, it is observed that the production time can be broken up into three components, namely:

- Waiting Time: Time wasted due to collision when trying to gain access to a station.
- Travelling Time: Time spent moving from one station to another.
- Processing Time: Time spent at a station as part of a product's assembly process.

5.3 Estimation of Processing Time Component

It is observed that the processing time component is fixed by the assembly plans. Thus, the processing time component can be obtained by summing up the processing time of the individual steps in the assembly plan, i.e.,

$$T_{PC} = \sum_{i=1}^N PT(i) \quad (1)$$

Where:

N	Number of steps in assembly plan
T_{PC}	Processing Time Component
$PT(i)$	Processing Time of the i^{th} step of assembly plan

With the processing time fixed, the effectiveness of a tool-placement would mainly be dependent on the travelling time and the waiting time.

5.4 Estimation of Travelling Time Component

Like the processing time, it can be observed that the travelling time component is relatively fixed. This is because the only additional movements besides those defined in the assembly plans are movements needed for deadlock resolution.

While additional movements may be needed for deadlock resolution, such occurrences are small compared to the number of movements required by the assembly sequences. It was found that although deadlocks may sometimes occur once for every piece of product produced, deadlocks are typically responsible for less than 1% of all the movements.

Thus, the travelling time component (for 1 run) can be estimated using:

$$T_{TC} = FT_T(S_L, S_1) + \left(\sum_{i=1}^{N-1} FT_T(S_i, S_{i+1}) \right) + FT_T(S_N, S_U) + FT_T(S_U, S_L) \quad (2)$$

Where:

N	Number of steps in assembly plan
T_{TC}	Travelling Time Component
S_L	Loading Station
S_U	Unloading Station
S_i	Station for the i^{th} step of assembly plan
$FT_T(S_x, S_y)$	Travel Time/Transfer Time between station for step x and station for step y

The simulation of 1000 assembly plans for 1000 runs each shows a maximum deviation from between the actual and estimated travelling time component of less than 2%.

5.5 Estimation of Waiting Time Component

Unlike the processing time component or the travelling time component, the waiting time component cannot be

directly estimated using the assembly plans. This is because the waiting time component is dependent on the amount and time of collision that occurs in the system.

Figure 3. Percentage Contribution of Waiting Time Component (T_{WC}) towards Production Time (T_{PT})

In general, it was found that the contribution of the waiting time component decreases with the increase in the travel time (Figure 3). Since real assembly plans would typically have stations' processing time that are much larger than the travel time, the contribution of the waiting time component would be important towards reducing the overall production time.

However, it is noted that besides the waiting time component, a large percentage of the production time is made up of the processing time and travelling time. Thus, another possible way to reduce the production time would be through arranging the tools such that they would require the shortest average travelling time between adjacent steps in the assembly plan.

5.6 Orthogonality of Processes

One possibility that the project looked at was the prediction of the waiting time that will be incurred at a station through the use of statistical methods. Ideally, the waiting time that is incurred at each station would be a function of the station's processing time over the entire production time, i.e.

$$P_s(P, S_i) = \frac{T_p(P, S_i)}{T_{TC}(P) + T_{PC}(P)} \quad (3a)$$

$$T_{WC}(P, S_i) = (1 - P_s(P, S_i)) \times T_p(P, S_i) \quad (3b)$$

Where:

P	Process P
$P_s(P, S_i)$	Probability of finding the Forces of P at station S_i
$T_p(P, S_i)$	Total Processing Time for Station S_i for P

$T_{WC}(P, S_i)$	Estimated Waiting Time component for station S_i for 1 run.
$T_{TC}(P)$	Travelling Time component for P
$T_{PC}(P)$	Processing Time component for P

By treating the time spent at a station S_i by a process P_x (without considering collision) as a periodic signal $f_{usage}(P, S_i)$ consisting of a series of unit steps starting from the time which the station is required by the process till the time the process leaves the station. Two processes P_A, P_B are orthogonal iff $\forall S_i, \int_0^T f_{usage}(P_A, S_i) f_{usage}(P_B, S_i) dt = 0$, where T is the larger of the production time needed for processes P_A or P_B to complete 1 unit of its product in the absence of collisions.

Using T_{WC} , an error of up to 30% between the estimated and the actual waiting time was observed. This error was believed to be in part due to the orthogonality that exists between the two assembly processes. This is because statistically, the probability of collision does not take into consideration the path relationship between two processes.

5.7 Time Varying Nature of *itas*

To minimise waiting, the *itas* does not synchronise the start of two assembly processes, thus each assembly process will start production of its next unit immediately on completion of the prior unit. The effect is a shift in the timing between adjacent assembly cycles that may lead to changes in the collision pattern. These changes also make it difficult to determine an orthogonality factor between two assembly plans.

Thus, unlike the Travelling Salesman Problem (TSP) or the Open Job-Shop Problem (OJSP) where the problems are mostly end-to-end, e.g. finding the shortest cost path from a starting town to all other towns and back in TSP, or finding the best schedule to fit all jobs under certain criteria, the *itas* is concerned with finding tool placement with a pair of assembly plans that will be used repeatedly for multiple assembly cycles.

6 Problem Complexity

Since each turret is capable of holding four tools that are accessible by both processes (i.e. shared tools), and another two tools, each of which is accessible by one of the processes (i.e. unique tools), letting n be the number of turrets, the number of tool placement possibilities is

$$O(n) = (4n)! + 2(n!) \quad (4)$$

The term $(4n)!$ of $O(n)$ is derived from the number of permutations available to the $4n$ stations which houses the shared tools. The term $2(n)!$ is derived from the number of permutations available from the n stations

available for housing unique tools for each of the processes.

Thus, since $O(n)$ is not of polynomial time, the problem of finding a optimal tool-placement would require the use of algorithms used for NP problems such as Genetic Algorithms [5].

7 Genetic Algorithms

The basic principles of Genetic Algorithms (GA) are similar to that of the Darwin's evolution theory. They are [6]:

- Better individuals are more likely to be selected than inferior individuals;
- Reselection is allowed. That is, better individuals can be selected for breeding more than once; and
- Selection is probabilistic.

Fundamental to GA are the Schema Theorem (ST) and the Basic Building Block Hypothesis (BBH). The ST basically states that the number of individuals in a population is a function of schema fitness, i.e. a fitter schema would have more chromosomes in the population [2], while the BBH proposed that the evolution of GA is through the recombination of short blocks of high fitness [4].

7.1 Objective Function

Due to the difficulty in obtaining an accurate estimate of the waiting time using statistical methods, a simulator that mimics the *itas* was used as the objective function. The output of the objective function is the production time required to complete a user specified number of units of the pair of assembly plans.

Since the simulator requires all parameters (i.e., timing and tool-placement) to be known before it can proceed, the GA for the *itas* must ensure that the result of any crossover or mutation operators always result in a chromosome that maps into a set of valid tool-placement.

The need for all parameters to be present also suggested that there is tight coupling between the various parameters as the change of any parameters is likely to affect the entire system.

7.2 Chromosome Format

The format of the chromosome used for the *itas* encodes each tool-placement parameter into an integer array. The index of the array is used to map the element to the station and the content of the array is used to map to the tool that is to be located at the station.

Figure 4. shows the format for the first turret. The N^{th} Turret follows the same format but starts at an array index of $6(N-1)$.

Figure 4. Semantic of Array Based Chromosome

Where:

- S ID of shared tool to assign to Station i .
 U_A ID of unique tool for process A to assign to Station i .
 U_B ID of unique tool for process B to assign to Station i .

As the simulator requires chromosomes to be mapped into valid tool-placements, operators that act on the chromosome needs to be able to operate on ‘station’ boundary. Thus, it would be more convenient for each station to be housed in an array element rather than in a bit string.

7.3 Genome Landscape of a Pair of Assembly Plans

To gain a better understanding of the genome landscape of the *itas*, the pair of assembly plans in Table 1. was simulated for 250 units for each possible tool-placement configuration.

Table 1. A Simple Pair of Assembly Plans

P_A	Processing Time	P_B	Processing Time
Tool A,	100	Tool D	100
Tool B	200	Tool C	150
Tool C	150	Tool B	200
Tool D	100	Tool A	100

The production time (T_C) for each of the 24 possible tool-placements were recorded. The Schema Mean Fitness, $ST_{C(mean)}$ and Schema’s Standard Deviation, S_{SD} is computed by taking the average and standard deviation of production time of all members within the schema respectively. The mean production time, $PT_{C(mean)}$, is computed by averaging all 24 tool-placements’ production time.

Within the schema’s fitness, a standard deviation approximately 20% of the difference between the maximum and minimum production time was observed. Thus, for this pair of assembly plans, the genome landscape appears to be spiky.

Based on the findings, it is felt that although some forms of building blocks do exist, the building blocks

may not be well defined due to the large variance between members of the same schema [3].

Table 2. Schema Mean Fitness and Standard Deviation for Assembly Plans in Table 1.

Schema	Schema’s Mean Fitness $ST_{C(mean)}$	Schema’s Relative Static Fitness = $ST_{C(mean)} - PT_{C(mean)}$	Schema’s Standard Deviation S_{SD}
A****	234647	-2868.75	17547.39
*A**	233803	-3712.08	18129.30
**A*	236218	-1297.08	13186.00
***A	245393	7877.92	11875.40
B****	234970	-2545.42	19923.87
*B**	237056	-458.75	16387.04
**B*	239133	1617.92	13370.15
***B	238901	1386.25	14023.22
C****	235811	-1703.75	12213.52
*C**	239551	2036.25	13987.52
**C*	240146	2631.25	17515.22
***C	234551	-2963.75	19377.07
D****	244633	7117.92	10881.58
*D**	239650	2134.58	15306.65
**D*	234563	-2952.08	19262.44
***D	231215	-6300.42	14362.47

7.4 Partially Mapped Crossover

The Partially Mapped Crossover (PMX) is used to perform the crossover operations for the *itas*.

In the PMX operator, the crossover point occurs at the boundary of genes, i.e. it will never break up a gene like in the case of the standard crossover operator on a binary bit-string encoded chromosome. Genes within the crossover region from the first parent are copied to the child’s chromosome with the remaining genes obtained from the genes of the second parent based on the order of their appearances (in the second parent’s chromosome). The following is the pseudo-code of a 1-point PMX operator:

New chromosome (N) = copy genes from parent 1 up to crossover point (X)

For each genes (G) in N starting from X+1

G = first gene in parent 2 not already in N

End For

It can be observed that the PMX operator will always produce a valid chromosome that preserves part of the genes configuration (i.e. tool-placement) of the first parent. Thus, any building block (i.e. groups of tool-placement responsible for the fitness) that exists within the crossover region should be transmuted to the child as required by the Basic Building Block Hypothesis.

An experiment using the PMX with a two-point crossover with random crossover points based on the recommendation in Bierwirth's paper [1] were performed. The observation from the experiment was that while the PMX operator appears to evolve the population initially, the evolution tends to stop after a few generations, i.e. newly generated populations do not contain better individuals compared to its parent population.

The large variance between members of the same schema as observed in Section 7.3 could be a possible reason why the PMX operator was unable to converge to the optimum solution.

7.5 Gene Swapping Mutation

The mutation operator for the *itas* operates by randomly swapping two genes within the same section of the chromosome (i.e. shared or unique tools).

Thus, unlike the PMX, it can be observed that the mutation operator operates on building blocks of higher orders as only a pair of genes was swapped.

It was found that it is possible to evolve the population by coupling the mutation operator with elitism. Elitism is necessary in a mutation-based GA to ensure that good chromosomes in the population do not accidentally become lost.

Since mutation operates on higher order building blocks, the rate of evolution was found to be typically slower than that of using the PMX operator. However, unlike the PMX operator, the GA using Gene Swapping Mutation (GSM) does not seem to get stuck.

7.6 PMX + Gene Swapping Mutation

An experiment using both PMX and GSM was performed and it was found that this GA was capable of evolving faster than one based purely on GSM and does not get stuck as that based purely on PMX.

This is believed to be due to the ability of the PMX operator to bring together low order building blocks, which complements the GSM operator that is believed to work on building blocks of higher orders.

The rationale behind this hypothesis is based on the observation that the GSM affects only a pair of genes and thus, retains the majority of the genes in their original positions. Hence, the disruption to the schemata is low and the building block of higher orders is likely to survive the mutation.

On the other hand, the PMX is observed to be more disruptive in that it may retain less than 50% of the genes from either parent in their original positions.

Thus, building blocks which are likely to survive this operation are those of lower orders.

8 Conclusion

The *itas* is unique from other NP search problems such as the TSP and OJSP in that it takes into consideration the effect of multiple runs of the assembly processes.

On choosing a suitable representation of the problem parameters and GA operators, it was found that the use of Gene Swapping Mutation together with the use of the Partially Mapped Crossover appear to produce the best results.

Like the TSP, GA is a viable solution for the *itas*. However, the large variance between members in the same schema suggests GA may not be an efficient solution to the problem.

References

- [1] Bierwirth, C, "On Permutation Representation for Scheduling", *Parallel Problem Solving From Nature - PPSN IV, volume 1141 of Lecture Note in Computer Science*, Springer, pages 960—970, 1996
- [2] Fang, H L, *Genetic Algorithms in Timetabling and Scheduling*, Department of Artificial Intelligences, University of Edinburgh, 1994
- [3] Forrest, S, Mitchell, M, "Relative Building-Block and the Building Block Hypothesis", *Foundations of Genetic Algorithms 2*, Morgan Kaufmann, pages 109—126, 1993
- [4] Grefenstette, J J, "Deception Considered Harmful", *Foundations of Genetic Algorithms 2*, Whitley, pages 75—91, 1992
- [5] Jong, K A D, Spears, W M, "Using Genetic Algorithms to Solve NP-Complete Problems", *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, pages 124—132, 1989
- [6] Koza, J R, *Genetic Programming III; darwinian invention and problem solving*, Morgan Kaufmann Publishers, 1999

Algorithms for Intelligent Turret Assembly System

Tan Hock-Woo, Lee Keok-Kee, Tay Leng-Phuan

Nanyang Technological University
School of Computer Engineering
Nanyang Avenue, Singapore 639798

Abstract

The Intelligent Turret Assembly System (ITAS) is an agile assembly system that allows for a variety of configurations through modular units. This configurability makes it difficult to determine the optimum tool placement configuration. Experiment suggested that the distribution of production time in the ITAS occurs in steps and that the increase in the number of turrets leads to a reduction in the importance of waiting time. Simulated Annealing was found to perform better than Genetic Algorithm and Tabu Search. This is possibly due to the lack of domain specific knowledge on how Tabu Searches may be directed and problems with the chromosome representation of the tool-station mapping in Genetic Algorithm.

1. Introduction

The Intelligent Turret Assembly System (ITAS) is an agile assembly system that adopts a modular design consisting of cages, known as turrets. Unlike the agent based system interacting on a common space with no rigid barriers proposed by the Agile Assembly Architecture (AAA) [5], the ITAS imposes limitations on the travel path.

In the ITAS, each turret houses 6 tool-heads arranged in a 3x2 configuration. Each tool-head can be fitted with one of a variety of tools independent of the tool heads location. Two mobile platforms (*forces*) transport one workpiece each to the various tool heads for processing, and transfer arms are used to pick and place the workpiece from one turret to another [1]. Due to the possibility of cable entanglement, *forces* within each turret are restricted to move in a C-shaped path. Hence, there are 4 stations accessible by both *forces* (shared stations), and 1 station uniquely accessible by each force (unique stations). Currently only two workpieces may be simultaneously assembled in the ITAS.

The location of tool heads, i.e. stations, within a system of N turrets is labelled from 1 to $6N$. The C

paths within each turret, along with the numbering of the stations in the n^{th} turret ($n = 1, 2, \dots, N$) are shown in Fig. 1. Turrets are assumed to be cascaded to form a linear assembly line, i.e. for *forces* A to move from station 1 to station 11, the path taken will be $\{1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11\}$.

Figure 1. C-Paths and Station Numbering

As the two *forces* can share common tools within each turret, the system has to prevent collisions and deadlocks by prioritizing the work process and access to a tool. Such actions result in a loss of efficiency for the ITAS. Thus, the objective of the project is to determine the optimum tool placement that will result in the best possible routing for its assembly processes, given these constraints.

2. Distribution of Production Time Component

The production time (completion time) T_C is the time taken for the ITAS to produce a minimum of W units of each product type in a pair of assembly plans [1]:

$$T_C = T_P + T_T + T_W \quad (1)$$

where T_P , T_T , T_W are the processing time, travelling time and waiting time components respectively.

A study was conducted on the contribution of the waiting time (with respect to the number of turrets) through simulations using pairs of randomly generated assembly plans. The result for the case of 2 turrets is shown in Fig. 2.

Figure 2. Component contribution versus Normalised Fitness (2 Turrets)

It is observed that the contribution of the T_T and T_W span across a certain band for a single fitness value and these bands seem to decrease with the increase in the number of turrets. The contribution of the T_P towards T_C also decreases with the increase in the number of turrets.

It is believed that the increase in the number of turrets causes the frequency of occurrences of consecutive usage of any two arbitrary tool to decrease and the travel distance to increase. As the T_P is fixed for each pair of assembly plans, any improvement is due to the decrease in T_W or T_T . This is believed to be due to the decrease in likelihood of collisions.

3. Neighbourhood Analysis

Fig. 3 shows the production time, sorted according to the swap distance, of the set of the neighbourhood tool placements of a randomly chosen tool placement Z for a case with 4 turrets. The swap distance is the number of stations between the pair of tool assignments that were swapped. Interestingly, the distribution of the production time does not seem to be affected by the swap distance. Even though all the solutions differ from Z by just one pair of tool assignments, the production time has a standard deviation of about 18.5% across the range of the production times. This suggests that the production time is sensitive to small changes in the tool assignments.

Figure 3. Change in Production time versus Swap Distance.

4. Genetic Algorithms

4.1. Chromosome Representation and Operators

The chromosome representation of the Genetic Algorithm (GA) is a 1-dimensional array with $N^* = 6N$ elements. Fig. 4 shows the format of the chromosome, with the content of the array element shown inside the boxes and the station number shown on the top of each cell. A, B represents unique tools that are available only to forces A, B respectively. S represents shared tools.

Figure 4. ITAS's chromosome representation.

The Partially Matched Crossover (PMX) operator was used to favour the preservation of the absolute positions of genes [3] because unique tools can only be fitted at certain stations. For mutation, a random pair of tool assignments within each chromosome is swapped, with the mutation rate being the number of chromosomes per population subjected to this swapping.

4.2. Population Size

The population size (*popsiz*) is made a function of the number of turrets using equation (2) to allow all schema of order 1 to be present in the initial population with probability of P_{occ} . This is done because [3] suggested that the

conditional sampling distribution of schemas is conditionally independent and is identical to the parent population.

$$popsize = 4N \left\lceil \frac{1 - P_{occ}}{\log(4N - 1) - \log(4N)} \right\rceil \quad (2)$$

4.3. Results

Fig. 5 show the production time versus mutation rate (p_m) of the GA for 4 turrets. To compare the effect of mutation versus crossover, the experiments use the crossover rate, $p_c = 1 - p_m$.

Figure 5. Production time versus Mutation Rate for 4 turrets

From the graphs, it can be observed that a high mutation rate gives better results (shorter production times) compared to a high crossover rate. This suggested that the effect of the crossover operator is inferior to that of the mutation operator.

We note that, ignoring the effect of collisions, the *ITAS* can be modelled as a Quadratic Assignment Problem (QAP) with a mathematical model [7] of:

$$f(\pi) = \sum_{i=1}^n \sum_{j=1}^n a_{ij} b_{\pi(i)\pi(j)} \quad (3)$$

where a_{ij} is the traffic intensity between object i and j , $b_{\pi(i)\pi(j)}$ is the distance between the sites containing object i and object j . Thus, each gene i in the chromosome contributes $\sum_{j=1}^N a_{ij} b_{\pi(i)\pi(j)}$. Therefore, when a single gene of a chromosome is copied over, only part of the information associated with the gene is copied because $b_{\pi(i)\pi(j)}$ depends on the placement of the other tools.

Let l denote the number of genes transmuted from the parent chromosome, and L denote the total length of the chromosome. The expected ratio of information transmuted from the original parent is $\frac{l^2}{L^2}$ because l^2 summations can be

performed with l genes. While the actual amount of information that is transmuted would depend on the amount of gene interaction (epistasis) of the problem [4], the estimates of $\frac{l^2}{L^2}$ suggest that the amount of information from the parent increases exponentially with the number of genes copied from the parent.

Hence, the poor performance of the crossover operator appears to be due to the problem with the chromosome representation.

5. Simulated Annealing

5.1. Parameters and Setup

The solution for the SA uses the same format as the chromosome in the GA. The neighbourhood move of the SA was also implemented by swapping a pair of tool assignments. New solution T_{ci} replaces the current solution T_{c*} with probability of $P_{acc}(\Delta T_c)$ based on the Boltzmanns equation:

$$P_{acc}(\Delta T_c) = \begin{cases} \exp\left(\frac{T_{c*} - T_{ci}}{kT_i}\right) & T_{ci} \geq T_{c*}; \\ 1 & T_{ci} < T_{c*}; \end{cases} \quad (4)$$

k is the scaling factor and has the value equal to one standard deviation in production time of a randomly generated set of tool placements. The step size uses the same *popsize* (2) as that of the GA implementation. The annealing temperature T_i is reduced by the cooling factor α to αT_i every *popsize* number of evaluations. The value of α was found through experiments to be between 0.8 to 0.9, which is consistent with the finding in [6].

5.2. Results

Experiments on the performance of the SA were conducted for assembly plans requiring 3 and 4 turrets. From the results (Fig. 6), it can be observed that the SA converges after approximately 60 steps.

Interestingly, the performance of the SA is similar for both 3 and 4 turrets. This could be due to the use of the standard deviation as the scaling factor k which should cause most $\Delta \frac{T_c}{k}$ to fall within 0 to -1. This agrees with the recommendation in [2]

6. Tabu Search

6.1. Parameters, Setup and Attributes

The neighbour set for the Tabu Search (TS) implementation for the *ITAS* used the same format for its solutions as that in the implementations of the GA and SA.

Figure 6. Performance of SA

The neighbour set of a solution Q is the set of solutions that can be obtained by swapping exactly 1 pair of tool placement within Q , which has a total of $2N(4N - 1) + N(N - 1)$ tool assignments. The aspirant set is chosen as solutions that are better than the current solution by at least 1 standard deviation calculated from an initial random sample of $popsiz$ tool placements.

Two tabu attributes considered in the implementation are: the information of a neighbour move, i.e. the position and current tool placements of the pair being swapped (Pair); and the complete tool placement information, i.e. the solution string (Full).

For diversification strategy, the TS uses a non-improving move strategy. That is, when no better solutions are available from the neighbour and aspirant set, a move based on one of the poorer solutions will be taken.

The two schemes used are: best-non-improving move scheme (BNI), in which the tool placement with the best production time is used as the new point to conduct the search; and the worst-non-improving move scheme (WNI), in which the tool placement with the worst production time is chosen instead.

6.2. Results

Experiments were conducted to evaluate the performance of the Tabu Search based on the different tabu attributes and diversification schemes. Each experiment has a termination criteria based on the number of fitness evaluation of $100 \times popsiz$, which equals that of the GA and SA. The results of the experiments are shown in Table 1.

In all the test cases, it was observed that the BNI scheme was inferior to the WNI scheme in generating better solutions. A smaller standard deviation was also observed for test cases using the BNI.

This behaviour was expected because the WNI strategy

Table 1. Results of Tabu Search

Tabu Attribute, Strategy	3 Turrets		4 Turrets	
	Production Time	Stand. Deviation	Production Time	Stand. Deviation
Full, BNI	89684	826	156377	1848
Full, WNI	88917	1133	155168	2528
Pair, BNI	89471	867	155653	1870
Pair, WNI	89107	1149	155408	3193

attempts to diversify the search further away from the current solution, thus allowing more regions to be explored. For the WNI strategy, better results are produced when using the full tool assignment as the tabu attribute compared to using the information of tool assignment pairs. The reverse is true when the BNI move strategy was used. A probable reason for this could be due to that the tool assignment pairs make more regions tabu, forcing the tabu search to explore a larger region. However, it is a paradox that the full BNI has a smaller standard deviation than the pair BNI.

7. Conclusion

From the experiments, SA was found to outperforms GA and TS, with TS being second best for the case of 4 turrets. The lack of domain specific strategies for Tabu Search and the problem with the suitability of the chromosome representation in GA, as seen by the distribution of production time and large standard deviation in the neighbour landscape, may be why SA was able to outperform GA and TS.

References

- [1] M. H. Chung, K. K. Lee, L. P. Tay, and H. W. Tan. Genetic algorithms for intelligent turret assembly system. In *7th International Conference on Control, Automation, Robotics and Vision*, pages 354 – 359, 2002.
- [2] CSEP. *Computational Science Education Project*. <http://csep1.phy.ornl.gov/mo/mo.html>, 1991.
- [3] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publisher Co., Inc., 1989.
- [4] P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Trans. Evolutionary Computation*, 4(4):337–352, 2000.
- [5] P. Muir, A. Rizzi, and J. Gowdy. Minifactory: A precision assembly system that adapts to the product life cycle. In *SPIE Symposium on Intelligent Systems and Advanced Manufacturing*, 1997.
- [6] S. S. Skiena. *The algorithm design manual*. Springer-Verlag New York, Inc., 1998.
- [7] D. Tate and A. Smith. A genetic approach to the quadratic assignment problem. *Computers & Operations Research*, 22(1):73 – 83, 1995.