Scheduling Chemotherapy Appointments: A Heuristic Approach

Calum Clark
Computer Science
2009/2010

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)__________________________

Summary

Within hospital chemotherapy outpatient clinics patients have appointment patterns that require them to attend the clinic on several days. On each of these days an appointment must be made for a patient to see a nurse to receive their pre-described treatment regime. This means that on any given day there will be a number of patients needing to have different treatment regimes provided by the nurses within that clinic. Each of these nurses will need to have a schedule produced of the patients they are to treat on a day and this schedule should be created to make the most efficient use of a nurses time, whilst also obeying all hospital employee regulations. This project's task then is examine how a heuristic approach can be used to create these schedules and the potential performance of them therein.

This is done by starting from a phase of research which selected local search methods based on a seeded start solution. These seeds are created by a constructive heuristic and overall this combined method was chosen as the best method available due to an assessment made of a variety of types of heuristic. A process of prototyping and exploratory development is then used to create a number of possibilities for both of these algorithms which are then polished and the best combination of the two is selected. This is then implemented with the appropriate user interface for input and output also being designed to allow for the creation of a user friendly piece of software.

Finally this is tested with the results showing that predominately the method does create an acceptable schedule and it is then compared to two other methods with the results showing it to be the best balance between the desirable features of quality and creation time. A conclusion is then made that there is a potential for the application of heuristic methods in solving this variety of scheduling problem.

Acknowledgements

I'd like to thank my project supervisor Natasha Shakhlevich for all the advice and guidance she has given me during the course of this project, and for not giving up with my sometimes abysmal written English especially as I'm the native English speaker.

I'd also like to that Alessandro Condotta for allowing me to use his test data and for some helpful insights into the subject area.

Contents

Chapter 1

Overview

Project Aim

The aim of this project is ultimately to produce a tool that will, when given a list of treatment pattern identification numbers, find the relative information and then produce a good schedule for one nurse to treat all their patients during a single day. The schedule produced will be aimed at avoiding any potential clashes between jobs and if this is impossible then they should be minimal.

Objectives

The objectives of this project are to:
- Research and analysis mathematical methods of optimization that are applicable.
- Prototype and refine a heuristic algorithm using one of the methods previously investigated.
- Evaluate my heuristic in comparison to other similar methods and try to define trends.

Minimum Requirements

The minimum requirements of this project are to:
- Produce designs for several heursitic algorithms for the single day problem.
- Implement at least one of the designs to produce an algorithm for the single day problem.

Extensions

Possible extensions are:
- Implement an alternative method and compare its results those from the orginial method.
- Produce a suitable input and output mechanism to allow a user to interact with the system.

Chapter 2

Problem Description

2.1 Introduction

The problem in question for the project is one which is closely related to task of the creation and optimization of schedules for a typical hospital outpatient chemotherapy clinic where nurses working a set amount of hours provide a variety of different treatment regimes for patients.

The treatment regimes for these clinics are composed of two parts: drug combinations and strict delivery patterns for these. This means that once a patient is prescribed a set regime they will then need a series of appointments with non-flexible periods of rest days between them commencing as soon as possible. This then leads to the situation where on any given day a number of patients will require an appointment, meaning that for any single day there exists a complex scheduling problem even when it is assumed that the multiday schedule has been fixed. This is especially evident when considering that the nurses treating the patient will require lunch breaks and that the same nurse should provide each stage of treatment for a patient during each appointment.

The main cause beyond these two issues for the problems complexity is that for each of the different regimes there is a strict pattern given to the appointment type needed for the administration of the prescribed treatment. These patterns can vary greatly but are always made up of two components: treatment time (where a nurse is carrying out a procedure) and a waiting period where time is needed for a drug to take affect or something similar that can only occur between two treatments. The number of either of these can vary but there must always be one initial treatment phase and then from here the two types of process should alternate with the final stage also always having be a treatment phase due a patient having to be given a check up before being allowed to leave.

The aim then will be to produce a schedule of these appointments which will make most efficient use of the facilities and personnel available whilst also avoiding or at least minimizing the number of clashes, a situation where two patients should be being treated concurrently by the same nurse. Also heavy consideration needs to be given in any potential solution to the issue of *clash density* which is defined as the maximum number of concurrent jobs clashing at any one point. This is because an increase in clash density would mean that the cost of a solution would rise greatly and it is a fair conclusion then that this should then be penalised.

The problem is then a simplified version of this task where only a single nurse is to be considered whose patients for that day have already been allocated (a successful method could realistically be extended to a group of nurses if a procedure was put in place to divide up a full days patients equally between them.)

The nurse in question would be assumed to be working a normal nine to five working day so the time frame for appointments is a maximum length of eight hours which is divided into fifteen minute unit time slots. It should also be noted that if the total amount of time treating patients is over six hours on a single day the schedule should include a fixed half hour break (two consecutive time slots) close to the middle of the working day to allow for a lunch break (it is as assumed that with less that six hours of treatment time a nurse will be able to have this break unscheduled).

The end goal of the project is to produce a good schedule that is a solution to the simplified problem from which interpretation can ascertain helpful information and observe instances of better or worse behaviour that can be useful in attempting to efficiently find a solution to the real life problem.

2.2 Example

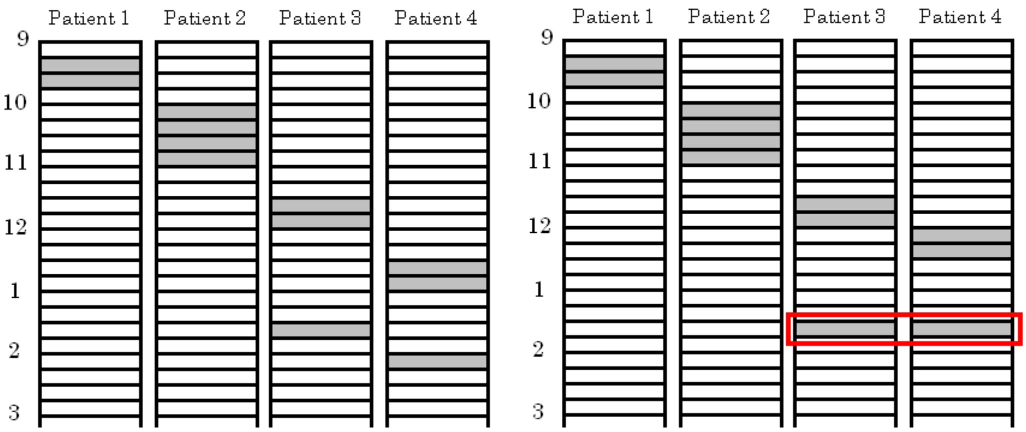
An hypothetical example of this variety of problem would be:

A single nurse working a standard 8 hour day being assigned four patients with the following information:

Patient	Regime Number	Pattern of treatment
1	1	1 2
2	28	1 2 3 4
3	29	1 2 9
4	42	1 2 7

Table 1: Patient details

From this there could be a number of schedules can be created, examples are shown in figure 1 where in (a) a clash free schedule is shown and in (b) there exists a single clash with density two.



(a) (b)

Figure 1: Two schedules: (a) Clash free, (b) With a clash between treatments for patient 3 and 4

2.3 Formal Description

A precise formal description of this problem is:

2.3.1 Input

The required inputs are:

- A set $P = \{ \ldots \}$ of n jobs; each of which corresponds to a patient
- the time horizon T consisting of m 15-minute time slots, $T = \{0,1, \ldots , m\text{-}2, m\text{-}1 \}$
- the details of the 167 regimes each in the form of a pattern of operations, each operation is a phase of treatment, 1 pattern is assigned to each patient

For each regime i, the corresponding pattern *p(i)* is given as a sequence of time slots in which the treatment procedures should happen. It is assumed for the input data set that any pattern starts in time slot 0. For example, the pattern *p(3)* of regime i=3 requires treatment procedures in time slots 0, 1 and 8. If in a schedule a patient with pattern *p(3)* has the starting time 5, then 5 is treated as an offset and the corresponding treatments should happen in time slots 5,6 and 13.

Two additional characteristics of a pattern i are $\nu(i)$, which specifies the number of operations in that job (ν:$P\rightarrow \mathbb{N}$) and $\beta(i)$, which is the total length of the job p (β:$P\rightarrow \mathbb{N}$)

The jobs shown in Table 1 are represented as shown in Table 2.

Patient i	p(i)	$\nu(i)$	$\beta(i)$
1	0, 1	2	2
2	0, 1, 2, 3	4	4
3	0, 1, 8	3	8
4	0, 1, 6	3	6

Table 2: Formal version of Table 1 with both characteristics shown.

For all the cases that will be considered in the experiments the value of m is 36.

2.3.2 Constraints

There are several constraints implied within the problem that are mainly related to the construction of a schedule. This process is formally represented through a relation s which assigns a start time from T to each of the first operations of the n jobs within the set P ($s : P \rightarrow T$) to create a schedule. Each of the assigned times must obey the constraint:

$$s(i) + \beta(i) \leq m \text{ (i} \in \text{P)}$$

This ensures that no job breaks the limit of the 36 slots time horizon, that the nurse can not have more than 36 assigned slots after clash removal and that all stages of treatment for a patient will be completed within a single day.

Another constraint is that if a nurse is scheduled to work more than 6 hours (24 time slots) then a half hour (two time slot) lunch break must be included in the schedule. This should be placed as close as possible to the middle of the day. This is assigned by way of adding an artificial job to the set P into the location *P(O)* to be scheduled along with the other jobs and it must obey the constraint:

$$\sum_{i=1}^{P} \nu(i) \leq 24$$

There is only one other constraint which is that no two operations from the same job can occur at the same time. This is trivial as each job is only assigned one start time by the function s and it can be assumed that the regimes will not specify conflicting times.

2.3.3 Objective

The objective of this problem is the production of a solution that best satisfies the aim of creating a clash minimal schedule with a correctly placed lunchslot. The assessment of how well the aim is met is done by use of a composite objective function, with each component being given different levels of importance. Three components are considered:

- C_1 - Clash Density: Which is defined as the number of operations placed into a location already assigned to another job. It is found within a solution by first using a function a which for a value within the set T finds the set of jobs from P that have an operation assigned to that slot $(a(j) = i \in P \mid j - s(i) \subseteq p(i))$. The function d is then used to find the clash density which is the size of this set found by a $(d(j) = |a(j)| - 1)$. This function is then applied to each of the slots from T with the maximum being recorded and is shown formally by:

$$max\{d(j)|j \in T\}$$

- C_2 - Number of Clashes: This is also found by using the density function d. In this instance instead of seeking the maximum value, a summation is made of the density for all the time slots within T. A maximum is taken of ether the density value with one subtracted to account for normal density or 0 to account for free slots. This is represented formally as:

$$\textstyle\sum_{j=0}^{T} max\{0, d(j) - 1\}$$

- C_3 - Lunch Time Placement: This is calculated by first checking if the job *P(0)* exists then if it does by finding the ideal time for the lunch slot which is halfway through the day (for the 36 time slot day this is position 17). This is then subtracted from the actual value and the absolute is taken which will then be the distance the lunchtime slot is from its ideal position:

$$|17 - s(0)|$$

These components are placed into the objective function with each being assigned a weight to signify its importance. These weights are represented by a α value $(\alpha_1, \alpha_2, \alpha_3)$ the actuals values of which are found by experimentation.

$$Objective\ Function = (C_1\alpha_1) + (C_2\alpha_2) + (C_3\alpha_3)$$

This results in a weighted linear construction, such as:

$$\textit{Objective Function} = (C_1 * 100) + (C_2 * 10) + (C_3 * 1)$$

2.4 Complexity Consideration

There are several pieces of literature regarding the complexity of coupled-task scheduling problems, this is not a highly investigated area but there are several on similar topics to this project that have investigated aspects of the problem complexity [16, 6, 26]. Of these all have been proved to be ether within the complexity class of NP-Hard or remain open which then gives rise to the realistic possibility that the problem in question for this project may also be of this and that further investigation would be worth while.

When considering the complexity of a scheduling problem there are several important factors that need to be evaluated as well as problem specific objectives. These are:

- Job Constraints - Are the jobs single units or complex interconnected series of units.
- Number of jobs - If the number of jobs is limited/unlimited.
- Scheduling Time Horizon - Does a boundary on the scheduling period exist or is it open.

The problem whose complexity is being considered here is the simplified option previously discussed and there is currently one other problem that is relevant and can be considered:

Wenci Yu: Scheduling exact delay coupled-task jobs on a single machine to minimise the makespan, the number of jobs is unlimited: This is proven to be strongly NP-Hard [25]

This bares strong resemblance to the problem under study with only a small number of differences:

- the problem investigated by Yu is minimising the makespan whilst the projects problem has a fixed maximum makespan in the form of the time horizon
- the jobs within the work of Yu are coupled task whilst the ones being studied may contain a varying number of tasks

In the problem under study there is a fixed time horizon on the schedule length but the number of jobs can be unlimited since clashes are allowed. This version of the problem is then also within the class NP and is likely to be NP-hard. Formulating the proof of this is beyond the scope of this project but due to this likelihood there are realistic grounds for an investigation into methods other than exact solvers. This is especially true when it considered that there is on going research into these areas and there is not yet currently a polynomial time method available.

Chapter 3

Methodology

To solve this problem a process is going to be used that is clearly divided into two very different (but not uncomplimentary) directions, that for the software engineering/project management area and that regarding the mathematical modelling aspects. Both of these directions will have there own structure along with different processes and procedures which are presented in detail below along with the reasoning for the decisions and also how they can best co-operate.

The basic structure given to the first of these, that which relates to software engineering is made up of a basic approach of *evolutionary development* with both its components *throw-away prototyping* and *exploratory development* being used. The structure will also have an additional phase of formal testing inspired by the *clean room methods*.

The structure for the second part of the project will be based around first experimenting with prototypes to create an algorithm for a constructive heuristic and then developing it. This will be followed by experimenting via prototypes with local search algorithms starting from the output of the constructive heuristic and moving on to develop one or two of these.

3.1 Software Engineering Methodology

The methodology for the software production part of the project is based around the use of *evolutionary development* [22]. This option was chosen as it allows a high degree of compatibility with other areas of the process predominantly that of the creation and modification to the mathematical model which will be discussed in more detail below.

The use that is being made of evolutionary development is two fold as the intention is to apply some degree of both main components contained therein. The first of these that of *throw away prototyping* [22] will be used when initially creating the system as it will allow for the fast creation of a working program that will then allow for testing and experimentation with the different mathematical models to ascertain their relative value with minimum delay. Once the initial model has been found the development will then proceed to use the other of the main component, *exploratory development* [22] this will be used to expand upon the basic design to refine the solutions which are generated so that they are both good and error free. This will be used then as a process of adding new features and constraints in a cyclic pattern of additions and testing to verify if any changes that have been made alter either the quality of a solution or other related characteristics (time to compute, reliability etc.)

This full on use of evolutionary development means that a method will be used which is closely related to the style *iterative development* [22]. This relationship allows the products specification to be much more dynamic as the requirements of it can be significantly more flexible as they will be able to be added (or removed) and modified right the way through the initial design process. Once the exploratory development phase begins the process will then basically follow a system of incremental delivery which will allow new parts to be added and others modified with a view to best tempering the output solutions. The choice was made to more closely follow this style of iterative delivery in the process rather than other available methods such as spiral development as they are less suited to the smaller size and lesser important of the increments that will be added to the software. This is due to the level of testing which is prescribed within them, something which would be unnecessary and time demanding in this project due to likelihood of changes being inconsiderable but in high quantity.

Due to this the development will then be leaning towards a process that will incorporate various aspects of *rapid software development.* [23] This can occur as it is not an essential part of the process that there be a high degree rigidity in the solution finding method with the high chance of potential changes/refinements in fact promoting higher flexibility. This flexibility will, as already mentioned when outlining the relationship to iterative development, be used to allow experimentation and adaption. It should be noted here that though certain aspects will be used from this area, it will not go as far as to use full *agile methods* [23] as it is felt they are not appropriate to the situation. This decision was taken when considering among others things that there is no formal client to give feedback on prototypes and that the main focus is actually on keeping the progress of both software and mathematical models combined.

By using these methods it is the hope to avoid some of the drawbacks which almost certainly would appear if a development process was used such as the waterfall method which whilst having its advantages would mean that much of the specification and design would have to be agreed upon and then become more or less static, which is felt would not be in keeping with the aim of trying to correlate the software with the mathematical model.

One of the most obvious failings of evolutionary development and the style of iterative development as a whole is that the informal cyclic adding of new features results in a situation where monitoring progress and ascertaining reliability can be hard, to combat this there will also be another element added into the hybrid process which is going to be used. This element will involve bringing a formal aspect into the latter stages of development of the software, this will be based on elements of the *clean room development techniques* that were developed by IBM [13] in the 1980s. This will ensure that the testing of the end product will be closely formalized with the view to preventing any defects from remaining undetected.

Summarizing, the software engineering methodology is going to be used is a hybrid with a generally informal approach that has a base level element taken from evolutionary development that makes use of both throwaway prototyping and exploratory development/iterative development in the creation progresses. This will then culminate with more formalized section based loosely on the clean room approach to try to minimise any potential special erroneous cases whilst also continuing to use an iterative improvement cycle based approach.

3.2 Mathematical Programming Methodology

The methodology for the mathematical modelling part will begin with the development of a simple constructive heuristic which will be an algorithm that will take all of the input treatment plans and construct a feasible solution in the form of a schedule. This will be experimented with in a number of ways with the goal being to find the procedure which best balances the dual goals of a good solution and creating it in a efficient time period. The results for this will then be used as seeds for the other phase of mathematical modelling as the evidence found suggests this is more probable to result in success than a random start.

The next stage for mathematical modelling methodology is based around the concept of experimentation with local search methods which are generally used to solve combinatorial optimization problems and as scheduling is a special type of those they are applicable here. These were chosen for use as the problem is most likely to be NP-hard so whilst the methods might not always reach an the optimal, they will find a good solution potentially near to it and without requiring a very large amount of computational resources that an exact solver would need. The methods concerned generally work around the idea of all solutions being in a global solution space and that you can navigate between them by using different methods [9]. The methods from this area that are going to be considered and analysed can be partitioned into two main groups which are each based on a different approach to the base concept.

The first of these groups is the neighbourhood search which deals with investigating models that are based around the use of different thresholds. These algorithms have a basic framework of: initialization, neighbourhood generation and acceptance/termination tests and they all begin with an initialisation that requires construction of a start solution (as created in the first stage). It is at this point they then part ways by the use of different acceptance rules to implement the threshold strategies that are used whilst moving between neighbouring solutions within the solution neighbourhood this is done via various transition structures in an attempt to find the optimal. In the project an analysis will be made of the potential of each of these options with prototypes being made for a subset of these, initially there are some rules that are believed will be most beneficial to go under investigation.

One of those rules is that of steepest descent, which is the simplest strategy to move by as it only changes to a different solution if the evaluation function marks it as a definite improvement. This approach does have the disadvantage of getting caught easily by local optimums but this can be fixed by multiple applications from various random start points (called multi-start descent approach.) It is the simplicity of both the normal and multiple versions of this which is believed would make it a potential starting point for the modelling methodology to begin on.

The next step in these rules is called threshold accepting and works around the idea that the search should be refined as progress is made to stop any large changes being made later on it the searching progress. This is done by way of the search slowly being given a smaller and smaller bound of allowed increase. This is something which hopefully would allow a greater degree of efficiency to be brought into the models and hopefully increase the solutions quality.

The next logical step after this is use of simulated annealing [1] where a solution is accepted based upon a probabilistic approach, so that bad moves can be accepted but only with a probability based on the likelihood of it being a future good move. This approach is designed to allow for potential bad moves to let the solution escape from a local optimum or a plateau which could impede the search progress and will hopefully allow the models to locate the best possible solutions meaning this could be a crucial aspect of the methodology.

The final set of rules that has possibility of becoming part of the methodology is that of the tabu search methods [11]. These work in a similar way to the others but keep a list of those moves which are not allowed (or are tabu.) This is done to try to avoid the solution cycling and repeating bad moves in a way which does not link to the search progressing in a fruitful way and could also potentially result in a sizable efficiency improvement.

The other group of local search methods which is planned to be experimented with is genetic algorithms [14, 21]. These differ from the previous grouping of threshold orientated as they are based on there being a population of solutions available and then a simulated survival of the fittest leading to a evolution procedure is applied to try to best improve the solution. This is generally done by way of a joint approach from the two methods of crossover and mutation in varying degrees to allow a population to evolve. This process is then continued in a cyclic fashion for a set period of time with a goal of producing a generally higher class of population at the end. These could be a very important part of the heuristic creation process as they may lead to a better area of the search space a characteristic which could be particularly useful if other components are combined into a genetic based hybrid model.

All of the methods from the two main groups also have a variety of styles and tricks that can be additionally applied [5, 24] in different situations to improve results. These hopefully will be made use of during the project and there potential uses should be considered when understanding the choice of methodology.

It is hoped that when experimenting with algorithms based on some of these in the throw away prototyping stage of the software development that it will be possible to find one which is best suited towards producing solutions for the problem. Once this has happened it will then fully implemented and refined in the exploratory development stage which will involve trying to apply or create additional touches whilst also trying to find optimal values for the various parameters connected with the chosen methods. All this will be done with a view to generating the best possible solution whether it is found by use of a single method or even potentially by a hybrid of several. Once this is complete it is the hope to be able test the mathematical model for robustness in different situations by comparison with other methods during a final cleanroom inspired section whilst also making any necessary further refinements.

The choice was made to use local search methods for the methodology instead of other options such as enumerative algorithms [8] or approximation scheme [3] based approaches as it was felt that for a number of reasons they would not be the most beneficial directions to take. These reasons are factors ranging from that there is simply currently a lot of research going on in those areas and there would be little progress made by further work without first observing these findings or that it was felt there is a much greater potential to find a useful method for efficiently locating solutions by investigating the local search methods.

Chapter 4

Project Plan

The components of the two parts of the methodology will combined with other project elements (such as background research) to create a plan of activities for project.

As the project progressed certain areas had to be restructured, an activity which affected the plans progresss in a variety of ways. This primarily meant that there was a small amount of change within the plan during its execution but as these changes were only to the start and end times of activities so the original plan framework was still followed. This modified plan is presented here with evidence of how the changes were made being presented in Figure 2, where both original and modified Gantt charts are presented.

The project is initially split into three broad phases of initial research, design-implementation-development and then experimentation & evaluation (and write up) with each of these areas then being further divided into sub areas which will relate directly to the individual actions that will occur within them. These activities are listed in Table 2 but it should be noted that the assigned weeks are a rounded measure and that they are not sequential activities as there will be overlap and in some cases several activities running concurrently (as can be seen in Figure 2 which is visual representation of the running time for each activity.)

Week	Activity	Milestones—Delieverables
1	Investigating the problem	
2	Background Subject Research	
3	Definition of problem	Problem Definition
4	Expanding Background Research	Background research Summary
5	Conclusion and Classification of Results	Mid Project Report
6	Design of Constructive Heuristic	Prototypes of Constructive Heuristic
7	Refinements and Implementation of Constructive Heuristic	Refined and Implemented Constructive Heuristic
8	Initial Design of Local Search Algorithm	Prototypes of L.S Algorithm
9	Design Completion and Refinements of Local Search Algorithm	
10	Local Algorithm Refinements	Refined Design for L.S Algorithm
11	Local Algorithm Implementation	Implemented L.S Algorithm
12	Testing of Solution and Comparison to ILP	Progress Meeting
13	Continuation of testing	Test Results
14	Comparison Testing	
15	Begin write up	
16	Complete write up	Final Report Produced

Table 3: Activities and Milestones

4.1 Initial Research (1)

This area will begin with a process of background research into the problem topic to produce a full problem description to allow for a good understanding of the problems attributes before any further steps are made. Once this is complete the next stage is to move onto a period of researching similar problems and how they are solved and what methods and tools are used to create those solutions. This will then be combined with any additional research into methods that are felt could be beneficial when creating methods to solve the projects problem. From this there will then follow an evaluation of all of the available possibilities before deciding upon a chosen course for the next phase.

4.2 Design-Implementation-Development (2)

This phase is primarily based around the chosen methodology, which will be applied twice to create designs for two different algorithms (the order of design being decided due to the fact of one being the start seed generator for the other.) These algorithms are:

- Constructive Heuristic
- Local Search Algorithm

Both of these will initially be designed through a process of throwaway prototyping to find the best rough models before moving onto an incremental period of exploratory development where they will be analysed, improved and refined. After this the chosen algorithms will be implemented through the programming language, Python.

This particular language has been chosen as it will allow the quick development and implementation of prototypes, characteristics which whilst being very attractive for use with this projects methodology do lead to certain developmental risks. These risks though have been proven to be avoidable in the correct circumstances, an example of this happening can be see in a case study carried out by Eric Newton [15] where he develops reliable applications in an area of very little allowance for failure and reports generally favourable conclusion.

It should also be noted that the easy to use nature of this particular language comes with the price of in some areas less than efficient implementation of data structures and other constructs. This risk has been seen in other areas such as scientific computation where a number of papers [7] have been written with content aimed at evaluating this risk with the overall conclusion being that it will only become a problem of note if not prepared for and action taken to avoid it by sensible and efficient use of structures.

This all then means that the software produced for the project will whilst being viable and accurate for the purposes of the initial approach would be best re-implemented in another language (such as C++) before being put into extended use. Though by common knowledge this can be seen improvement overall it would not be such a large improvement that it is pointless to implement via python as there has been at least one study [18] reporting similar speeds for python and other languages whilst they are all preforming the same tasks.

Once this has been done for both algorithms there will then be produced a final combination of the two that will solve the problem in the chosen fashion and that has been refined to be efficient as possible.

4.3 Evaluation & Experimentation (3)

The evaluation of the solution will have a number of different stages to try to assess the potential benefits of the methods, these will work on the algorithms on there own and when placed in comparison to others.

The two attributes which will predominately be accessing through all of the testing stages of the evaluation process will be:

- Quality of solution (assessed using the objective function based on number of clashes, clash density and correct placement of lunch break)
- Time to create solution (assessed on a quantity based on ether the actual time taken (in minutes/seconds) or possibly based on the number of algorithm iterations taken)

It is hoped that a compromise will be found between these two which will allow for the selection of the method that has the overall best characteristics of all of those produced.

The actual experimentation will vary depending on the nature of the algorithm in question, this is due to the fact that some of them will be strictly deterministic whilst others may contain different levels of random input meaning that there is no guarantee that any two applications of the method will produce the same result. It is with this possibility in mind that a clear division will be made between the testing of these two groups, for those that are based on a deterministic approach they will only be applied once to each of the different problem instances. Whilst those of a less predictable nature will be applied several times and will then consider both the worst and the best outputs and also the average output when considering how good the quality of the solutions is and how reliable the methods solution production abilities are.

The test instances that will be used are taken directly from the results of a data collection process which is part of a similar project but experimenting with Integer Linear Programming instead of heuristics. This data collection was concerned with gathering and anonymising actual patient data which then means that the instances that will be used to test the system will be made up entirely of real life information taken from a chemotherapy clinic.

Each of the methods will then be applied to the test instances the appropriate number of times and then a series of comparisons based on the generated information will be made to try to access to potential characteristics of each of the algorithms. There will then be a process of trying to fit patterns or trends to the information with a view of drawing the conclusions which are a big part of the aim of the investigation. Any further details of the actual solution evaluation cannot be planned at this point due to a level of dependency on the actual algorithms produced.

This final phase will then conclude with a period of writing up the project and considering the results to draw as many useful conclusions as possible. There will also be a final consideration period given to review the decisions and choices made within each of the phases of the project.

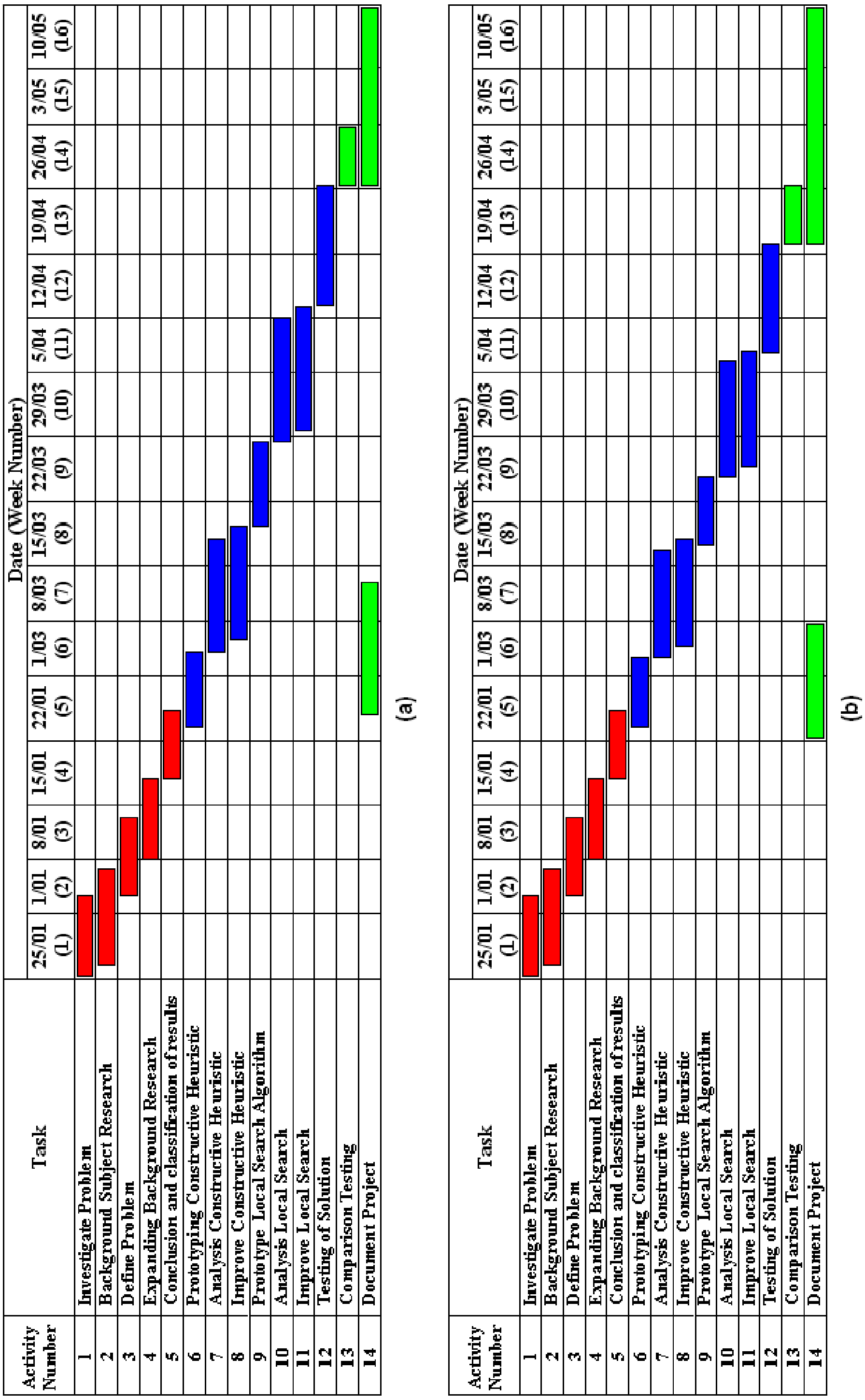


Figure 2: Gantt Charts showing activities in all three phases with start and end times, with (a) original and (b) modified.

Chapter 5

Literature Review

There has been and still is a wide range of research going on into the project area and others that are very closely related to it, as a starting point for the research an investigation was made into a range of those that were considered to be most relevant. This was done with a view to both being able to find an applicable and up to date direction for the projects investigation whilst also trying to analysis and evaluate the application of methods that have worked in other similar areas when considering the projects topic. In all of the material which was initially found there was a clear division between that which was relevant and that which was discounted for a number of reasons, though in some cases useful tricks or extra ideas were learnt from them.

The papers which were read and whose content is not going to be used further, concern a number of different areas. These are:

- Identical Jobs [12]
- Unit Execution Time [6]
- Approximation Algorithms [3, 2]
- Flowshop Scheduling [20, 19]

These papers were discounted for a number of reasons, in several cases it is simply that the methods used within them whilst being very helpful to the problem situation are not useful for this project as there is a clear difference between the two problems which cannot be easily be over come. This can be seen in situations such as where the jobs are identical, as in the work that done on this using pattern identification within finite windows by Lehoux-Lebacque [12] and also in the work done by Blazewicz [6] where a polynomial algorithm is developed to schedule sub-parts of jobs with unit execution time. These two situations were then discounted due to these differences but gained from them was the idea of representing with unit execution operations (larger blocks being decomposed into single ones with zero lags).

Another reason for some of the research being discounted was that in some cases the methods themselves whilst being applied to seemingly similar situations would in actual fact be very difficult to adapt to suit the projects problem, a clear example of this is the approximation algorithms that where developed by Ageev [3, 2] which whilst recording very good results for his scenario would be very complex to alter.

A final reason for some of the papers being classified as irrelevant was down to the fact that though they were seemingly in a situation that was closely linked to the project it would not in actual fact be possible to bring the two together whilst still applying the same methods, this was due to problems that they were being applied to falling into categories such as flow shop [20, 19] which requires multiple machines where as the problem in question only has one (a single nurse.)

Whilst these papers themselves were not overly informative it was a useful to carry out the broadening of the search away from a small confined area and as it allowed for a high degree of surity that the method which would end up being selected to use would be the best one available.

The papers which were read whose content is going to be using as a starting basis and inspiration for the project work are manly based around two topics:

- Coupled Operation Scheduling [16, 17]
- Local Search Methods [10, 9]

These topics were found due to coupled operation scheduling being a very similar problem to the projects which whilst only involving two tasks instead of an arbitrary number, the results can still realistically be extended in most cases to deal with the greater number of operations. The main body of work which was found on this topic was the research done into the problem complexity and creation of heuristic solvers carried out by Potts with a variety of others [16, 17]. This body of research introduced the idea of this problem being NP-Hard in all but special cases which was proved by way of selection of proofs dealing with a variety of problems with different constraint values. From this it can then be inferred that speedy progress could be made using solution finding methods such as local search (tabu and descent are the main ones discussed) something which the authors conclude has lesser than expected benefits when compared to the constructive heuristic methods they also applied. This is then explained with good reasoning based on the inefficiency of the neighbourhood search methods when combined with the feasibility constraints placed upon any given solution. A important observation made here is that of the local search methods the one which is most successful is that which is based on a seed created by the constructive heuristic rather than a random start selection. This then gives potentially more of a justification for the inclusions of a investigation of constructive heuristics in this project.

Further analysis and investigation of these was carried out once the more general search had completed. This was done as it was felt use of a similar approach would give the most promising results for the particular problem based on evidence gained so far. It was then from here then that the decision was made to use local search methods as they would appear to be an approach with a lot of potential. After this decision was made the process of intensifying the research into this area began, regarding both its applications into areas surrounding the problem and also into the methods themselves (this part of the research is covered in detail previously in Section 2.2 so wont needlessly be repeated here.)

The research that was found when intensifying investigation around the scheduling application of these local search methods was split between two groups, first those which like the previously mentioned work by Potts [17] and also work by Gupta [10] that use a variety of tactics including neighbourhood search and genetic algorithms. This is done to create a composite approach to find a method which will hopefully result in the best possible solution by drawing from a number of different places. From studying these papers it was felt this group and their varying approaches can offer a lot of interesting opportunities and they also appear to be fairly malleable in nature which could allow them to be used in different situations so could have a lot of potential for the project.

These are then followed by the group of applications which solely concentrate on developing genetic algorithms possibly as a hybrid with other factors, examples of this being the research carried out by both Tseng [24] and Amirthagadeswaran [4]. Both of which consider a variety of types of potential genetic algorithm to use and then progress on to develop a final improved algorithm. In the case of the work by Tseng this resulting algorithm is a hybrid that combines the standard genetic approach with other varieties of local search. This group seemed to also pose a selection of good opportunities and it was believed to be in best interest at this stage not to rule out each of the two directions of local search development, it is then an aim to analyse and possibly experiment with elements from both during the prototyping stage of local search algorithm development.

Chapter 6

Constructive Heuristic

The next stage in the project was to move onto the start of the second phase of the plan that deals with 'Design-Implementation-Development' of a solution and is where the design of the actual algorithms began. The first of these algorithms which was designed was the simple constructive heuristic which is primarily to be used as a start solution generation mechanism to provide the seeds for the local search methods.

6.1 Initial development

To begin creating this a process of idea creation and exploration was used. This was done by way of mind mapping with an aim of generating as many different approaches to the creation of a constructive heuristic as possible. This was done by working from initial ideas, then trying to either: consider new directions that could be taken or find possible links and benefits between each of the different groups which were gradually beginning to appear within the spread of ideas.

6.2 Throwaway Prototyping

The next stage in the development of the constructive heuristic was to take as many of the ideas generated previously and try to turn them into simple prototypes. This was achieved primarily based on a process of grouping together a number of related approaches and combining them into a single prototype. From this process five prototypes were created all of which were considered to have the potential of producing a good feasible solution. These prototypes are given below and were all based from a starting situation of having a list of jobs each of which is defined as the treatment pattern required for a patient to be seen by the nurse on that day.

1. *Basic Greedy Constructive Heuristic* uses a process of first ordering the job list and then adding each job in turn into the schedule as early as possible and if a job cannot be added into the nearest free slot then each following slot is attempted until an acceptable one can be found.

2. *Simple Sequential Improvement Heuristic* is a method based on starting from a schedule where each job is placed in a sequential order without any interleaving followed by a process of shifting each job to an earlier time to try to introduce as much interleaving as possible, in a process starting from the final job.

3. *Composite Pairing Greedy Heuristic* is a process based on a similar method to that used by Potts and Whitehead, where jobs are combined into composite jobs and then further combined until only one remains.

4. *Expansion Based Interleaving Heuristic* works on a method based on selecting a first job to place in the middle of the schedule and then trying to place all the other jobs incrementally around the edges of the current schedule based on trying to maximise the global benefits of the resulting solution through considered positioning.

5. *Global Pattern Based Heuristic* is a system based on having a range of pre computed information about how different jobs can fit together best and then using this to create a good feasible schedule.

Once this stage was reached there then was a more serious consideration of both the actual implementation of these prototypes and how the algorithms themselves would work (considering at this stage issues such as time complexity and feasibility of produced solutions.) After this was done the decision was made that several of the prototypes did not merit any further consideration these were:

- *Simple Sequential Improvement* due to the fact that imposing either time horizon constraints or including lunchtime slots was a going to be overly complicated procedure

- *Expansion Based Interleaving Heuristic* due to complex nature of the considerations needing to be examined for placement of the surrounding jobs this would not be efficient

- *Global Pattern Based Heuristic* due to the large amount of pre-processing needing to be carried out to find all of the combinations this did not seem to be a realistic possibility with the time period of the project but could still be a good possibility for an efficient system.

Once those which were considered to be too impractical had been discounted there then began a period of extension and consideration towards the implementation of those prototypes still remaining taking into account features such as the complexity of the prototypes and the actual mechanics of implementing them.

6.2.1 Basic Greedy Constructive Heuristic

The first of the algorithms which had been selected to continue developing was the most basic one which is a greedy process of job ordering followed by trying to add each of the jobs as early as possible to the schedule with an end goal of producing a feasible solution in a relativity small period of time.

Development of this was continued by working with several examples of first small data sets then gradually becoming larger until the experimentation was with a real selection of jobs for a nurse on a single day. This was done to try to ascertain how well the algorithm worked on different problem instances and also how much, if at all the algorithm would need to be modified to deal with a real data set as until this point the prototypes had only been experimented with on a small subset of jobs. The results of this process were that it was discovered that overall no large changes were required to be made, with only a couple of small changes being needed after which the algorithm functioned correctly.

A basic description of the final prototype of the algorithm is:

1. Take list of jobs to be scheduled and order them by a preselected method.
2. Before starting the iterative adding process, define a value 'start' this will be the starting location to add each job into the schedule initialised as '0'. Also have a list 'busy' which holds all unit time slots that have been allocated and a list that contains the 'schedule' both are initialised empty.
3. Iterative process for each job to be included in schedule:
   (a) Working from 'start' value check if each operation in job can be added into the location (found by operation+start) without clashes by a comparison process to those time slots contained in 'busy'
   (b) If possible add to job to 'schedule' at this location, append 'start' value to job information and update 'busy' with operation+start values
   Else increment 'start' and try again (return to Step 2)
   (c) Once a job has been added reset 'start' to be the earliest free time slot in (as in one not in 'busy') then move onto next job to be put into schedule and returning to start (Step 1)

The time complexity for this algorithm was found by a process of assigning each of these steps a complexity for a single job then finding the overall complexity for each of those steps to be carried out on a full problem instance of n jobs. A number of values were needed within the time complexity calculations these were: δ = the maximum number of operations in any of the n jobs, l_j = length of job j, L = summation of the total length of all n jobs which is also the maximum makespan for the schedule . ($L = \sum_{i=1}^{n} l_j$)

1. The ordering operation has a time complexity of $O(nlogn)$
2. The variable definition has complexity of $O(1)$
3. The iterative process runs at most L applications of the substeps which are $O(\delta L)$ so overall is $O(\delta L^2)$
   (a) The operation addition process tries to add each of the δ operations from the job at most into each of the L possible locations within in schedule so is $O(\delta L)$
   (b) The comparison process checks each operation in the job so has a complexity of $O(\delta)$
   (c) The reseting of start to the first empty slot checks each part of schedule so is $O(L)$

This then means that the time complexity for a single step is $O(\delta L^2)$ with the total time complexity for the entire process when altered for all n jobs being $O(n\delta L^2)$
An example of this algorithm working is presented in Figure 3.

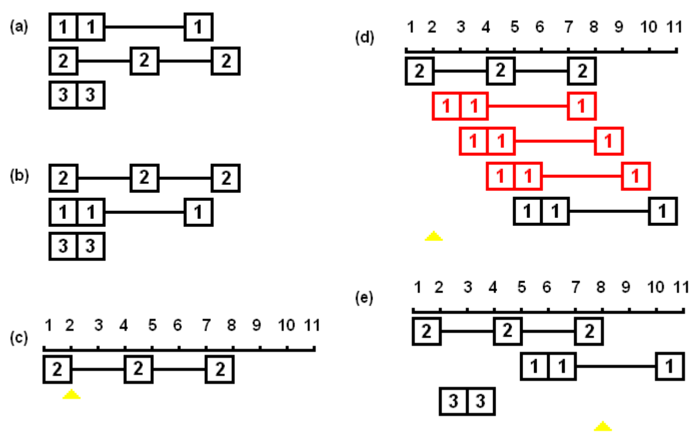


Figure 3: (a) List of jobs to be scheduled, (b) Jobs ordered based on total length (optional), (c) First job added to schedule at position 1 and set 'start' to be first empty position (2), (d) Attempt to add second job starting form start value, keep increasing 'start' until can add all operations and set 'start' again to be first free place (2), (e) Add in third job to schedule and set 'start' to be first free position (8)

6.2.2 Composite Pairing Greedy Heuristic (2)

The second of the algorithms that was developing further was the one which was based around a approach inspired by work carried out by Potts and Whitehead [9] which is based around an approach of first ordering the jobs by one of the various possible methods (to be further discussed at later stage) then starting an iterative process of picking two of them to combine into a composite job with a goal of ending up with one final composite job that contains all of the initial jobs which can then be decomposed into a feasible schedule hopefully all within a reasonable time period for an algorithm to be executed in.

The development was continued on this algorithm in a very similar way to that of first, in that there were experiments with a variety of datasets of increasing size to try to fully consider the requirements imposed by having a real life dataset. The only real factor which came to light in doing this was that it was discovered that the method had some problems dealing with jobs with identical patterns which meant that it was not able to differentiate between them. This was fixed by introducing a numbering system for the initial jobs which then as far as could be ascertained with analysis of the datasets resulted in the algorithm functioning correctly for even the largest real life instances.

A simple description of the resulting algorithm is:

1. Before starting the iterative process several lists are defined: a starting list of the initial jobs, a list to contain all 'active' jobs (both composite and those unused original jobs) and a list 'composite jobs' to contain all of the composites along with information involved with them which will be used in the decomposition process. All but the initial jobs list are initialised empty.

2. Initial jobs are ordered in a predefined manner (to be selected later) then placed in the active list

3. Iterative process to create best composites to be included in the schedule:

   (a) Iterative Process to find all possible pairs of jobs (job1 and job2) from 'active' list, define a new empty list of 'possible composites' to hold all composites created at this stage:

      i. Check job1 and job2 are not the same, define a new empty temporary structure.

      ii. Add operations from job1 to temporary composite structure

      iii. Start process of attempting to add operations from job2 to the temporary composite, If can't add immediately due to 'clashes' introduce a 'bias' value to job2 where the position of each operation will be increased by that the 'bias.'

Keep increasing 'bias' until all operations can be placed into the temporary structure.

iv. Temporary structure then holds a potential composite, add to 'possible composite' list and move onto next pair (return to step 3.a.i)

(b) Select best possible composite available using an ordering method based on a number of different features (to be explored in detail later)

(c) Add this to the 'active' list, remove component jobs. Also add to 'jobs' then continue to find next pair until all complete (Return to Step 3)

4. Once all jobs are combined a process of decomposition back to a permutation of the original jobs with start times for each job being assigned by accumulating 'bias' values.

The second algorithms time complexity was found in the same manner as the first by calculating complexity per step but in this case the complexity was for the entire schedule not a single job. Again a number of values were needed within the time complexity calculations these were: δ = the maximum number of operations in any of the n jobs, l_j = length of job j and K = is the maximum length of any one job within the problem instance ($K = max\ (l_j)$)

1. The variable definition includes has a time complexity of $O(n\delta)$

2. The ordering of the jobs has a time complexity of $O(nlogn)$

3. The total number of possible composites is n^2 so time complexity is $O(\delta K n^4)$

   (a) The number of possible pairs to be made at any stage is n^2 so complexity is $O(\delta K n^2)$

      i. Comparing the job IDs has a complexity of $O(1)$

      ii. Adding the δ operations to the temporary structure has a complexity of $O(\delta)$

      iii. Finding a possible composite has a complexity of $O(\delta K)$

      iv. Adding the composite to the possible list has a complexity of $O(\delta)$

   (b) Selecting the best possible composite has complexity of $O(n\delta)$

   (c) Adding and removing jobs from the active has complexity of $O(\delta)$

4. The decomposition process has a complexity of $O(\delta n^2)$

This means the overall time complexity for this method is O (Kδn^4) which would appear significantly worse than that of the first algorithm. This is primarily due to the n^4 that comes about due to the generation of so many different composites at each iteration with the algorithm.

An example of this method in use is presented in Figure 4.

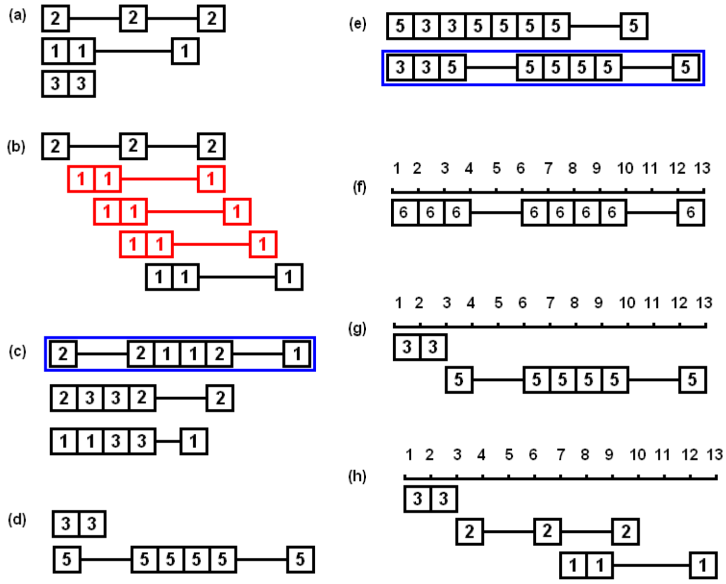


Figure 4: (a) Placing jobs into active list in order of longest total time first, (b) Example of selecting a first job then a second and combining them into a composite structure using a bias to shift the second job until it can be placed in a clash free position, (c) Selecting a composite from those generated (only a sample of these are shown), (d) Adding composite to active list and removing components, (e) Composites made and selection occurs again, (f)(g)(h) Only one composite left so decompose back to permutation of original jobs by using information held in 'jobs' list which contains all composites chosen during previous stages.

6.3 Exploratory Development

The final stage in this algorithms development was a exploratory process of first implementing the prototypes then of analysing the resulting programs and there output and attempting to find any features which they would benefit from having added whilst also attempting to find areas in which they weak and required improvement in. Also part of the analysis process was a clean room inspired testing period to achieve the two goals of accessing the reliability of the programs and of finding the best values for various different parameters within the algorithms (such as the different potential ordering strategies or the criteria used to select the 'best' composite.)

6.3.1 Additional Features

There were two additional features which were found that the programs required to have added, these were firstly the ability to produce schedules which would contain clashes if there was not a feasible way to arrange the jobs within the time horizon boundary (which is defined as 36 unit times slots.) Then secondly the ability to judge when a 'lunch break' was required by the nurse and if it is, then it should then be included within the algorithm. These were both then implemented within the programs in the following ways:

Time Horizon

This feature was added to both of the programs by adding a callable function which when run at the end of the schedule production phase introduces another component to both of the programs which checks if any part of any of the jobs runs over the time limit of 36. If this is so, the component then moves the offending job or jobs to places within the schedule that have minimum overlap and records this move along with keeping an overlap record (which it also considers when finding a new position for a job to try to minimise overlap density).
The process of finding the location within the time horizon where a job can be placed with minimum overlap/resulting clashes is relativity complex, a rough outline of it is done follows:

- For the job to be moved the information is recorded and it is removed from the schedule, a 'best-so-far' tuple is then introduced containing a location and a clash number value.
- Starting from 0 an assessment is made for each location by considering both the busy and overlap lists then a record is made of how that location performs when considering the number of clashes that would be created by adding the job to that location.
- This record is then compared to the tuple and if the number of clashes is less than that one currently being held by the value, the tuples information is swapped for that in the record otherwise the next location is moved onto.
- Once all locations have been assessed then the information contained in the tuple is where the job should be placed.

Lunch Break

This was added to the first program, the basic greedy constructive by simply pre calculating if the number of operations from all jobs was sufficiently large and if so then added, first a special job (denoted by an additional field containing a 'L') to the schedule and then added additional times to the busy list. All of this was done before any of the scheduling process took place so that the lunchtime slot could be guaranteed to be assigned its optimal position of halfway through the day.

This feature was implemented for the second program by an opposite procedure to that of the first where the lunch time slot is added at the very end of the scheduling process. This involved checking for all free time slots and then trying to find two adjoining as close to the middle as possible and placing the lunch break there. This may not seem to be a very effective method but it will allow for penalisation of similar schedules where only some of them have bad lunch break placement whilst also allowing for bad lunch break placement to be weighed against other good and more important features for any potential schedule. Also this reflects a common feature of some areas within hospitals were lunch breaks can occasionally have to be taken at non-optimal times to account for a days workload but this will hopefully not be an issue with.

6.3.2 Parameter Selection

Next there began a process of formal cleanroom inspired experimentation to find the best parameter values. This was done by using three randomly selected problem instances to evaluate the different solution lengths generated based on changes to the different parameters with the each test only needing to be repeated once due to the algorithms deterministic nature.

Firstly there were experiments with changing the start ordering of jobs in both of the algorithms to try to create a more efficient end solution (Note: this experimentation was done on each of the algorithms without applying the time horizon constraint or any ordering to second algorithms composite selection so the schedule length might be over 36. This was done to try to improve the quality of the solutions before applying the time horizon constraint to minimise the amount of work needed to comply to it by endeavouring to minimise idle time and therefore shorten the solution.) In these two tests the ratio of total schedule length to amount of working time was calculated:

Problem Instance	Ordering Strategy: Total Length Longest First	Total Length Shortest First	Length Operations Longest First	Length Operations Shortest First
1	1.343	1.257	1.2	1.2857
2	1.2727	1.848	1.3030	1.7878
3	1.366	1.166	1.1	1.233

(a)

Problem Instance	Ordering Strategy: Total Length Longest First	Total Length Shortest First	Length Operations Longest First	Length Operations Shortest First
1	1.2	1.2	1.228	1.342
2	1.5757	1.7878	1.636	1.727
3	1.2	1.333	1.2	1.3

(b)

Table 4: Shows the ratios of total schedule length to amount of working time as results from start order tests with (a) showing basic greedy constructive and (b) showing composite pairing greedy heuristic

From these results the chosen orders were to use Length Operations Longest First for the first algorithm and Total Longest Length First for the second due to these two having the smallest ratio between total schedule length and the amount of working time for the tested problem instances. This suggested that they could be the most efficient. At this point more test instances could have be studied to try to gain a fuller evaluation but it was considered that the constructive heuristics fine tuning were not of sufficient importance to spend further time on beyond these brief tests.

This was then followed by experimentation with the comparison rules used in the composite pairing greedy heuristic at the stage of accessing the different possible composites and selecting one to use (Note: this was done without the use of a start job ordering strategy.) In this test the ratio of total schedule length to amount of working time was again calculated:

Problem Instance	Comparison Rules					
	Total Length Longest First	Total Length Shortest First	Length Operations Longest First	Length Operations Shortest First	Max Idle Time	Min Idle Time
1	1.4	1.285	1.371	1.342	1.371	1.111
2	1.636	1.787	1.636	1.454	1.636	1.394
3	1.233	1.166	1.2330	1.333	1.3	1.066

Table 5: Is in the same form as Table 4 and shows results from the second heuristic.

From this Minimum amount of idle time was chosen as it produced schedules which have the smallest ratio implying they are both the shortest and have the least amount of total idle time.

When all of the parameter values had been set it then was possible to conclude on the other function of the testing process, that of giving a measure of the programs reliability. As all tests concluded correctly and without errors, the decision was made that the program was considered to be at a satisfactory level of reliability.

6.3.3 Weakness Removal

Only one weakness was identified within either of the algorithms that was considered to be something that required action to avoid it potentially causing some large future difficulties. This weakness was in the second algorithm there was a chance of confusion when differentiating between composite jobs if they had the same structure which only became a noticeable problem during the decomposition stage of the algorithm as it could lead to jobs being incorrectly placed with the end resulting schedule. This issue was fixed by a process of extending the earlier added feature of identification numbers from just the initial jobs to include each of the composites created to allow them to be referenced and then the original start jobs could be efficiently traced through the entire decomposition process.

6.4 Summary

Finally an evaluation of the two different algorithms was made to try to choose one which would be used as a starting point for other methods to try to improve upon. To start this process another series of tests was run with 10 problem instances being used (a number which should have been larger again but due to this project times constraints was kept lower for this part due to its lesser importance.) The results of this are then shown in table 6.

Problem Instance	Basic Greedy Constructive	Composite Pairing Constructive
1	1.25	1.111
2	1.3030	1.8
3	1.1	1.166
4	1.058	1.2
5	1.425	1.5757
6	1.1	1.2
7	1.22	1.1
8	1.11	1.11
9	1.333	1.394
10	1.2	1.333

Table 6: Results of comparisons between two methods

After analysing these results the basic greedy constructive appears to produce overall better solutions and when giving consideration to the time complexities of the two where it appears to be significantly faster. For these reasons and the fact that the lunchtime slot was guaranteed its ideal position it was decided that the first algorithm (Basic Greedy Constructive) would be used with the other algorithm being a potential variation to be considered in the future when trying to find the overall best solution generation method.

6.4.1 Random Element

Also considered was the idea of using random start job ordering with a set amount of iterations being preformed with the view of selecting the best generated results. The result of these tests were that a boundary amount of iterations needed to find a lowest achievable schedule was found, this was approximately 200 iterations. This type of method was not considered to be necessary for the basic constructive algorithm due to the increased amount of time for the algorithm to run and the subsequent drop in efficiency but could possibly be used as a method for later comparison and evaluation.

Chapter 7

Local Search Algorithm

The next part of the development phase of the project was to design the algorithms which would work from an initial solution to alter it and find improvements meaning the schedule produced would have a better objective function values. The development of these algorithms followed a number of standard steps, each of which was the design of a different component. The required components were:

- A solution representation
- A structure to modify a solution
- The method of evaluating a solution
- A method of generating an initial solution
- A search process
- An acceptance function
- The termination test

The first three of these steps need to be carried out before the process of creating any prototypes for algorithms can begin as their results would need to be included within any prototype. These are predominantly the functions which are used to carry out the basic operations of changing solutions and evaluating them so it is vital they be created first. The components from the other steps will then be entirely contained within the prototypes and will vary depending on the algorithm design and type.

7.1 Solution Representation, Modification and Evaluation

After the research stage of the project it was decided that the development of a local search algorithm was going to investigate both genetic algorithms and those working on a traditional local search basis. This then means that there will need to be designed both a neighbourhood structure to allow the neighbours to be found for a solution and a genetic operation to allow reproduction to get a new population of solutions. Once this is done it will then be possible for solution transition to occur in both of the potential algorithm directions.

The first step to designing either of these functions was to select a solution representation which would allow for easy alterations to the schedule. This was done by starting from a period of prototyping which produced a range of ideas, this was then followed by an analysis, experimentation and consideration phase that resulted in the selection of what was preserved to be the best method. The selected method was, that a solution was to be represented by a collection of 36 lists with each time slot having an initially empty list assigned to it and each of the jobs being represented by a marker appearing in the list which represented the time slot where the first operation of that job was scheduled to start. An example of this being used is shown in Figure 5.

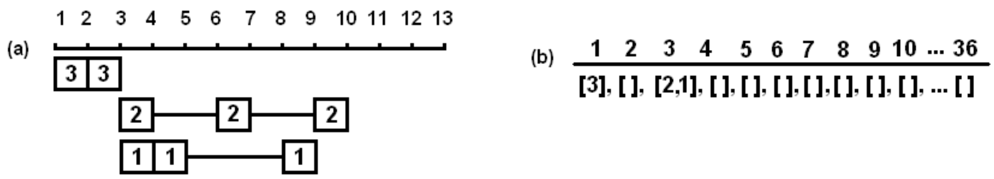


Figure 5: (a) An example of a schedule, (b) Schedule representation in a list form

The main reasons behind the selection of this particular representation being chosen are that it is simple and easy to manipulate which hopefully counteracts any disadvantages due to the larger amount of memory required.

7.1.1 Neighbourhood Function Creation

As soon as the representation had been chosen the process could then move onto the creation of the neighbourhood function and the genetic operation. Both of these were going to be created by a process of prototyping to create a range of different possibilities that can be tested later to select the best for the chosen algorithms.

The first of these two to be created was the neighbourhood function which began with a process of generating ideas about the different ways a solution could be altered in a standardised pattern. From this generation process a number of concept ideas were then firmed up to create the range of different neighbourhood functions. These functions will each alter the solution in different manner, they are:

- *Two Swap* two different elements of the solution are selected at random and then values contained within the parts are swapped.
- *Insert* an element of the solution is selected at random and then placed into a different random location within the solution, causing some of the other values within the solutions to shift in one direction or another.
- *Multiple Swap* this is very similar to Two Swap but varies as to the number of different components being swapped.
- *Shuffle* part of or all of the solution is shuffled using one of several random distribution strategies (Examples being Weibull or Gaussian distribution).

After a period of evaluation and considering the practical implications of the their use it was decided that the final two functions would not be considered further due to in the case of *Multiple Swap* there being a obvious possibility of multiple applications of *Two Swap* and in *Shuffle* there being far too large a neighbourhood for any solution to allow it work as effectively as the others. This then meant that two neighbourhood structures which were to be experimented with later in the local search algorithm design process were going *Two Swap* and *Insert*. Examples of these are shown in Figure 6.

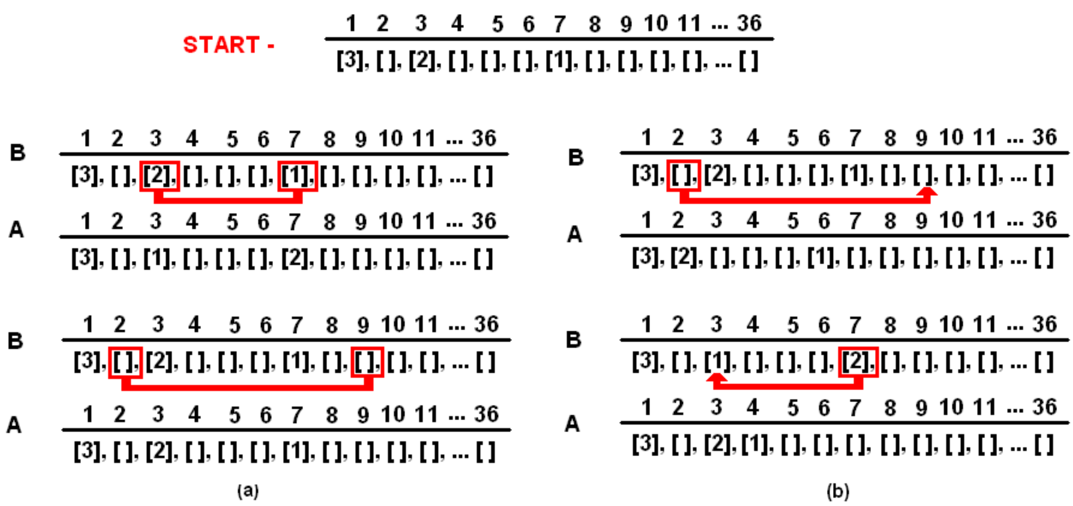


Figure 6: Schedule from figure 5 modified via (a) Two Swap neighbourhood, (b) Insert neighbourhood with Before [B]/After [A] being shown for each.

7.1.2 Genetic Operation Creation

The next of these solution modification functions to be created was the genetic operation to allow for a new generation to be created. This also began by a process identical to that of the neighbourhood function where ideas were generated and then firm concepts were attempted to be constructed from them.

It was at the point where the ideas were based around the two standard areas of crossover (where two solutions are split at a random locations then recombined) and mutation (where a random change it made to elements from the solution), that it became clear that the strict rules on the solution contents would prohibit the normal use of ether of these methods. This is due to constraints meaning that a solution must contain a marker for each of the jobs to be scheduled once and only once. This is something which cannot be guaranteed when ether of the two methods mentioned previously are applied to a solution.

With this issue in mind the process of creating prototypes then began to also consider hybrid genetic operations (meaning that they would now become genetically inspired rather that true genetic operations as they would not be using the standard implementations of crossover or mutation that were described previously. This means that for all that from here on when a genetic operation is referred to, what is actually being discussed is one of these genetically inspired approaches.) These would combine elements from outside of genetic search domain to repair solutions that which would allow for the output to consist only of those that were valid. The outcome of this was two different functions which were:

- *Random crossover with Repair* Two solutions are taken and crossover procedure is used (a random point is chosen within the solution range and at that point two solutions are split then recombined with a section of the other solution to create two new offspring solutions.) These two new solutions then have a repair process applied to the problematic areas within them. This repair alters repeated values to be ether blank or missing values and inserts remaining missing values into random positions until the solution is valid.

- *Extended Random Mutation* A solution is taken and at a random point is mutated where the value is altered to be a random value from the range of all those possible, this random selection and mutation process is then repeated until the solution is again classified as valid.

Both of these prototypes were then evaluated and it was decided that only the first of the two *Random Crossover with Repair* would be the chosen to be used within any genetic algorithms. This decision is due to the way that, this function would preserve as much as possible of the parents in the offspring. This is preferable the other option, a process of purely random mutation from which there is no guarantee of a result that is an improved combination of the parents and therefore the evolution process would not working as desired.

An example of this chosen function being applied to a smaller schedule is shown in Figure 7.

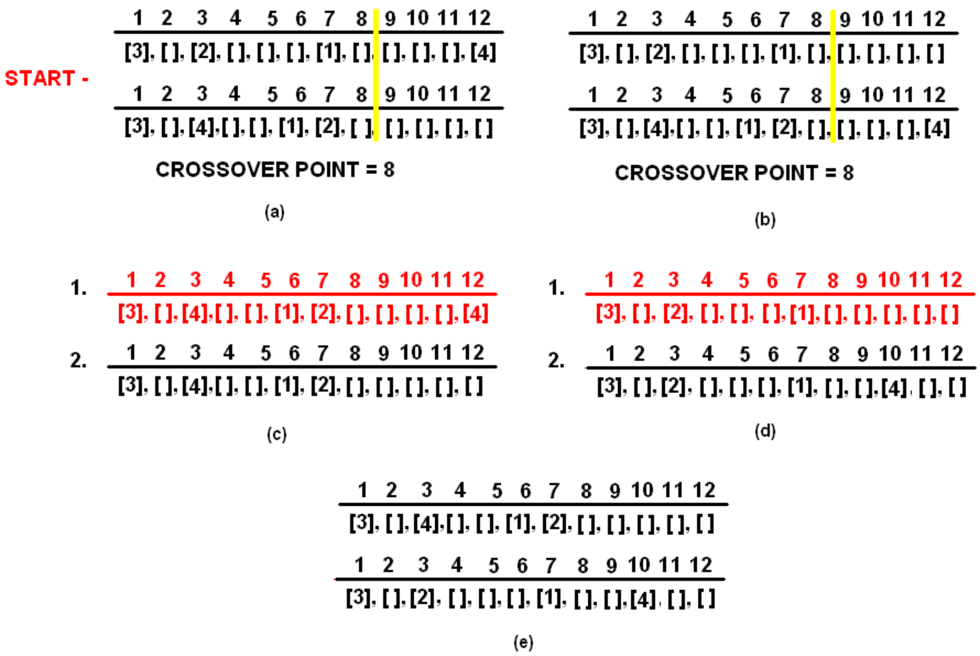


Figure 7: Input of two start solutions, (a) Two Parents with crossover point marked, (b) Two offspring after crossover, (c) First repair operation to remove repeat instance, (d) Second repair operation to add missing job, (e) Result two valid offspring.

7.1.3 Solution Evaluation

The next key component to be designed prior to the prototyping was the objective function that when given a schedule would calculate three different measures and return a numerical value to show the relative goodness of the input and allow comparisons to be carried out.

The calculation of three measures is outlined below:

1. To calculate the number of clashes, a list was made of all busy time slots and then all overlapping slots (as used in the constructive heuristic) the number of clashes was then simply the length of this overlap list.

2. To calculate the clash density, the list created in the previous stage was used with a counter being kept of how many times each value appeared; the highest counter therefore equals the clash density.

3. To calculate the distance from the perfect lunch location, the assigned location of lunch was subtracted/added (depending on if the location was greater/smaller that desired) from the perfect value to get the deviant from the ideal placement.

These three values were then combined into a weighted linear equation as outlined previously with the weights of 100, 10 and 1 being selected after brief series of tests.

7.2 Throwaway Prototyping

Once the basic components have been created, the design process then moved on to the creation of a number of different prototypes for a local search algorithm whose content would cover to remaining standardised design steps. This prototyping was done in the same manner as the creation of the constructive heuristic where an initial idea gathering process occurred followed by a consolidation procedure to bring together ideas into a set of prototypes. All of the prototypes use a seeded started point due to evidence found during the earlier research phase as it has been widely suggesting this was more efficient starting choice. The resulting prototypes from local search and the genetic approach are shown below.

Standard Local Search Prototypes
This entire group of prototypes uses the same basic method; starting from an initial seeded solution the algorithms then move between neighbours comparing each ones objective function value and recording the best found so far. This process continues until one of the following terminations conditions is met: a total iteration limit is met (100,000), a single solution iteration limit is met (1000) or a trivial lower bound is reached (Objective Function returns 0.) The prototypes differ in how their acceptance of a better solution is decided, the different acceptance rules of each of these are:

1. *Iterative Improvement* uses a process where each new neighbour has there objective function value found and then compared to the current best. If the neighbour is better of these it is then taken and recorded.

2. *Threshold Accepting* is a method that differs in the constraints used to denote an improvement as here a worse solution can be accepted if it falls within a threshold value. This threshold value will be slowly decreased throughout the search process allowing for smaller worse moves to be accepted.

3. *Tabu Search* - is the same as Iterative Improvement but includes a list of solutions that are tabu and cannot be revisited to avoid solutions getting caught in cyclic movements whilst trying to improve the solution.

Genetic Inspired Approach Prototypes
For each of these prototypes it was decided to extend the idea of a seeded start solution. This

meant that the initial population would be found by using the constructive heuristic with the diversity being created by random job input ordering.

1. *Basic Population Evolution* uses a method very close to the natural development of animals, where all members of the population are evaluated and the better half of the population is taken and then used to as parents to create offspring that will form with their parents a new generation. The process will then occur repeatedly until the termination condition of a predefined number of generations is met, with the population constantly improving.

2. *Variable Size Population Evolution* uses a similar basic structure to the first prototype but instead of a static sized population the size of the populace can vary as the better part is found not by a numerical cut off but by a percentage variance from the best value found for any individual member. This is done with an aim of overall population improvement so attempts not to loose good members.

Once the stage of having a wide selection of prototypes was reached, a more in depth period of consideration and evaluation was then carried out giving thought to the potential effectiveness of the methods and also the practical considerations of how they would be implemented. After this had been completed it was decided that only two of the prototypes would be further investigated. Those chosen two where the first genetic approach one, *Basic Population Evolution* and the last standard variety local search *Tabu Search* though it was at this point decided to modify this as explained below. The reasons for discounting each of the others were:

- *Iterative Improvement* this was discounted due to its strong nature of getting easily caught in local minimas with no realistic possibility of escape other than by a large number of repeat applications.

- *Threshold Accepting* this was not chosen as it was felt that it would be a better option to take the good features from it and combine them with another to create a hybrid that would potentially function better than ether of the two components.

- *Variable Size Population* due to the complex nature of having the population size vary it was foreseen that there could be the potential for difficulties if the percentage threshold should not be able to be met or if returned number were not of an even size which could lead to offspring with good potential being missed.

As soon as those which were considered not to be the better prospects for an algorithm were discounted, there was an extension period for each of the chosen prototypes. This was done as until this point they had only been experimented with on small datasets and now needed to progress to the realistic sized instances for the problems in question.

7.2.1 Modified Tabu Search (1)

This first selected prototype that was chosen to pass through to this stage of the process with the condition that it be modified to include aspects from one of the other potential prototypes, *Threshold Accepting*. This modification consisted of incorporating the threshold allowance into the algorithms acceptance rule. It would then mean that the potential solutions being found and possibly accepted would need to be checked against both a list of those solutions that were tabu and also checked to see if there values fell within the acceptable threshold. This would have the result that during a search process for an improved schedule a worse solution could be chosen at one stage in the hope of allowing a better end result whilst also being protected from any unproductive cycling of solutions that may occur from this more lenient approach to acceptance.

The period of expansion to allow for the transition to realistic job instances was very brief for the prototype as only one alteration was required. This alteration was to add a way of automatically choosing good threshold values and appropriate decreasing strategies for the objective function values of the solutions in the neighbourhood surrounding the current solution. This was done by having the value being assigned for the initial threshold based on a predefined percentage of the starting solution and having the decrease amount being set to a predefined percentage of the current threshold. Both of these predefined values are to be chosen via testing at a later stage. After this problem was resolved there were no further issues in use of real problem instances.

A more detailed description of how this more refined algorithm would work is given below:

- A variable Best So Far holds the best solution found up to that point as well as its objective function value; a value is defined as the Threshold which is altered after each change in solution and a list is defined with a predefined length to store recent solutions called the Tabu List. Two counters of Total Iterations and Single Solution iterations are defined and initialised as 0.
- A start solution is generated using the constructive heuristic, then it is evaluated and from the returned value both the initial threshold and first amount to decrease are calculated, this solution is initially set to be best so far
- Iterative Loop is started:
    1. A check is made to see that none of the termination conditions are being met: (1)Trivial Lower Bound Reached, (2)Max Total Iterations Boundary Reached , (3) Max Iterations on a Single Solution Boundary Reached.

2. From the current Best so far solution a random neighbour is found using the chosen neighbourhood structure and this is then evaluated by the objective function and both Total and Single Solution Counters are incremented.
3. A check is made to see if the generated neighbour is in the Tabu list. If it is then no further action is taken and the loop returns to step 1.
4. Else a comparison is made between the value for this neighbour and the value recorded for the Best so far, this comparison is of the form:

$$\textit{Best So Far} + \textit{threshold} \geq \textit{Neighbour}$$

5. If this true then the neighbour is set to equal Best so far, the old best so far is added to the Tabu List, the threshold is decreased by the pre defined amount and single solution counter is reset then the loop returns to step 1.
6. Else no action is taken and the loop returns to step 1

An example of this algorithm at work is given in Figure 8 (with the solutions being shown in a shortened representation, listing just Job Numbers and Start times.)

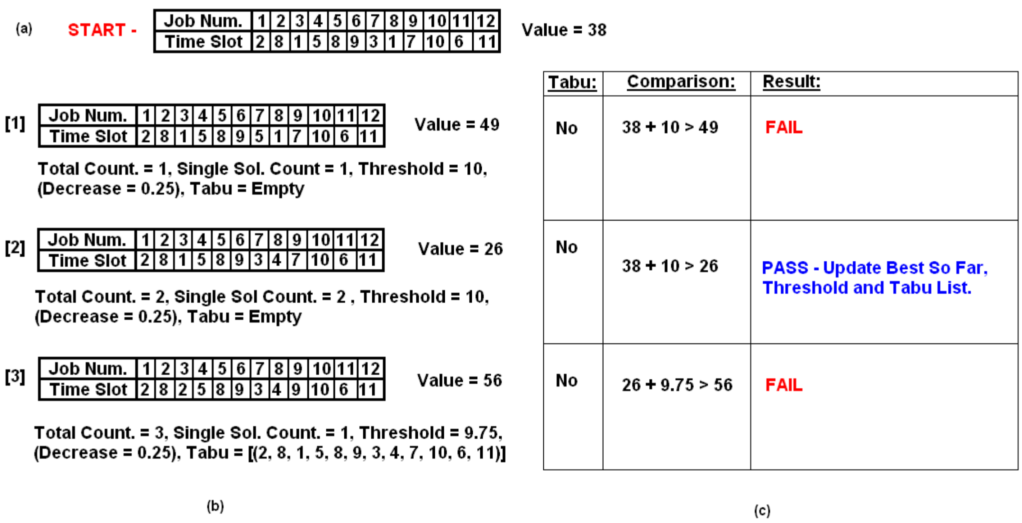


Figure 8: (a) Initial start solution generated by the constructive heuristic with its objective function value, (b) Three neighbours are generated from the current Best so far and then also evaluated through objective function, (c) Comparisons being made: [1] - Is not accepted due to being outside of the threshold, [2] Is accepted and is made into the new best so far so the next neighbour is generated from this not the start solution, [3] Is not accepted due to being outside of threshold.

7.2.2 Basic Population Evolution (2)

This second selected prototype for the local search algorithm was chosen as it offered a very different approach to a method of improving an original solution and had potential to get better results than the more ordinary version of local search due to taking routes through the search space the other may not. It works by using a process that starts from an initial population of feasible solutions all of which will be evaluated and had a value assigned to them. From this the better half will then be selected, paired and then passed one at a time to the genetic operation which then produce offspring that can then be combined with there parents to produce a new generations population. This will then be evaluated and so on until a certain number of generations have passed and the process will terminate with the overall best population at that point.

The process of expanding this particular prototype to work with the realistic sized instances was carried out without any issues arising that required action to be taken. It was noted that as the size increased there was a uniform increase in the time taken to produce a result when the termination conditions remained the same something which may become an issue at a later stage of evaluation.

A fully explanation of how this process would work is given below:

- Start by defining value number of generations and current generation as a counter both starting from 0.
- A start population is generated using the constructive heuristic with a random order applied to the jobs being given to it to generate a diverse start population. This then has the evaluation function applied to each of its members and the values are recorded.
- Iterative Loop started:
  1. A check is made to see if generation limit has been reached
  2. Population is sorted based on evaluation vales
  3. Population is split in half and each element of the better half is randomly paired with another.
  4. Each pair is given to the genetic operator which creates two offspring per pair, these then have the evaluation function used on them and the value is recorded.
  5. Offspring and parents are returned and then all combined into the next generation of the population, the generation counter is incremented

An example of this in practice is shown in Figure 9.

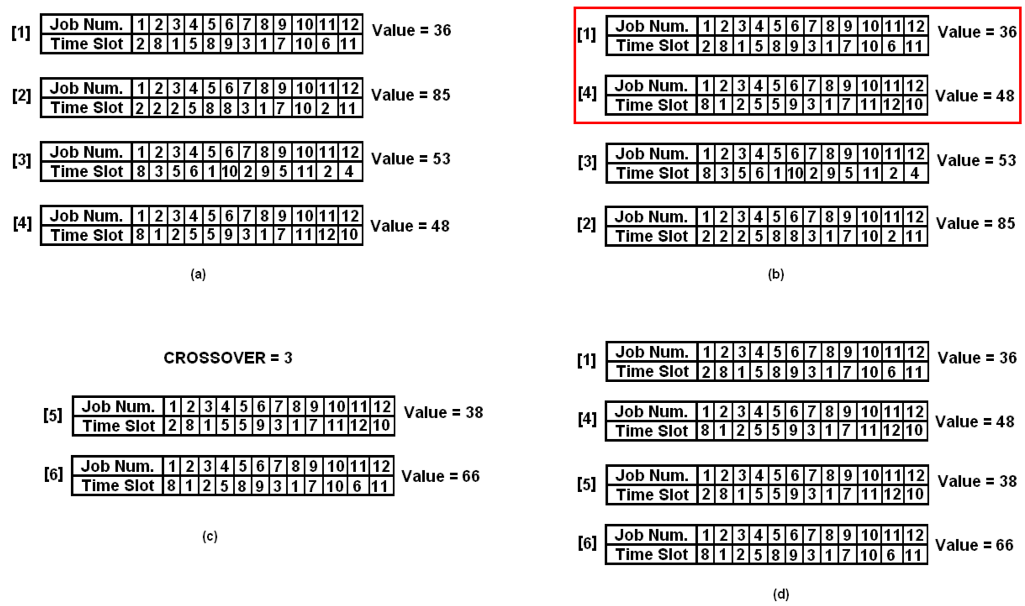


Figure 9: (a) Start population created by random ordering of input to constructive heuristic, (b) Population is ordered and then better half paired at random, (c) Off spring created by Genetic Operator, offspring have evaluation function applied to them, (d) Parents and offspring are combined into new population

7.3 Exploratory Development

Once the prototyping stage of the process had been completed with all the designs not considered to be better options being discarded, the algorithm creation process then moved onto the final stage. This stage was a period of exploratory development starting from implementation of the remaining prototypes. Once this was complete it was followed by a phase of consideration of what features the resulting programs would benefit from or needed to fix discrepancies in performance. This was then completed by a phase of experimentation to fix values for various different parameters within each of the algorithms to allow them to be fine tuned and to also ascertain that the programs were reliable over a number of instances.

7.3.1 Additional Features

There was only one feature that was added to both of the algorithms in exploratory stage. This was a way to impose the time horizon upon the outputs of ether the neighbour generation or the genetic operation. This was implemented as detailed below:

Time Horizon

This feature was added to both of the algorithms in the same manner as it was added to the constructive heuristic. This simply meant that the same callable function to move violating jobs to the best alternative location was used.

7.3.2 Parameter Selection

The next part in the exploratory process was another stage of formal Cleanroom inspired experimentation with various different values and options within the two algorithms being fine tuned in order to achieve the algorithms maximising their potential. For all of these tests all other values within the algorithm were set as default to be: Threshold = 100%, Decrease = 5%, Tabu = 10, Population = 32 and Generations = 100. All tests results presented are averages of the results of 10 applications per instance with 10 instances being used (only averages from all instances are shown with more detailed results available in Appendix B.)

In the results shown below the average resulting objective function value is given for each of the potential parameter options. In addition to this for those tests regarding the *modified tabu search* the average number iterations to reach a result is also given in brackets. (This is only shown for one algorithm as the other has a set number of iterations.)

Neighbourhood Creation Method
The first value to be experimented was the choice of which of the neighbourhood search methods should be used in the standard local search algorithm. The results were:

	Neighbour Creation Method	
	2 Swap	Insert
Average	3.67 (3370)	3.99 (5011)

From these results it was chosen that the best method is that of the 2 Swap method which appeared in most cases to produce good results in a smaller number of iterations.

Tabu List Length
The next value to be experimented was the length of the list that would hold all of the tabu values in the *Modified Tabu List algorithm*. The results were:

	Tabu List Length				
	5	10	15	20	30
Average	3.72 (4624)	3.68 (2847)	3.68 (2530)	3.88 (1940)	4.56 (2371)

From the results it can be concluded that the best length is for the tabu list to take is 20 this is due to the general trend of lower values that are found quickly when using this.

Threshold Value

The next value to be experimented with was the threshold value which was set to be a percentage of the first initial solution to be created by the constructive heuristic and this will then alter how large the accepted bad moves can be. The results were:

	Percentage of Initial Solution				
	25%	50%	75%	100%	200%
Average	3.74 (1510)	3.6 (2712)	4 (2339)	3.6 (2450)	3.8 (2570)

From these results it can be seen that there is no clear cut best option but that 50% and 100% seem to be a slight improvement, 100% is selected due to the opinion that a larger threshold will result in a more diverse search process.

Decrease Size

Once the threshold value had been chosen it was then possible to choose the speed at which it would be decreased, which is a percentage of the current threshold. The results were:

	Percentage of Initial Threshold				
	1%	5%	10%	20%	50%
Average	11.1 (2152)	11 (1271)	11.06 (1968)	11.03 (1189)	12.12 (619)

From this it was decided that the best option would be to use a value of 20% based on a compromise between the varying results with good values coming from both 10, 20 and 50 percent.

Population Size

The experimentation then moved to the genetic method based algorithm where the first test was to ascertain the size of the population that should be used. The results were:

	Population Size					
	8	16	32	64	128	256
Average	11.98	11.89	11.8	11.8	11	11

These results appear to be limited due to many of the instances reaching 0 with no extra information available but it was noted that the larger values took predictably longer to return a result so when this was combined the available results the decision was taken to use 128.

Number of Generations

This was then followed by an experiment to try to gauge how the population affected the outcome and what size was required for the algorithm to best perform. The results were:

	Number of Generations				
	25	50	100	200	500
Average	11.26	11.18	12.1	11.9	11.08

These results had a similar issue as those previously mentioned but it was decided that in this case the longer time period was worth the improvement seen between the highest two values so a population size of 500 was selected for use.

The tests that were carried were also done to ascertain a reliability level for the methods over an extended period of use on varying input. All of the tests were successful so there were no immediate flaws within the method and a good level of reliability can be assumed.

7.4 Summary

Finally there was one last comparison to be made between the two algorithms production capabilities after they had both been fully polished. This was done by comparing the two algorithms over the ten problem instances used previously. The results to be considered in this test were as before the average final objective function value and now a timer was introduced so the average time for the algorithm to run could also be considered for both. For this test averages were drawn from the results of 25 applications per instance.

Problem	Algorithm (Average over 25 applications)	
Instance	Modified Tabu Search	Basic Population Evolution
1	111.2 (1.391)	110 (14.845)
2	0.5 (1.58)	0 (17.233)
3	1 (0.965)	0 (18.483)
4	0 (0.201)	0 (15.620)
5	0 (0.388)	0 (15.483)
6	1 (0.599)	0 (16.022)
7	0 (0.415)	0 (15.235)
8	110 (2.100)	110 (18.956)
9	0 (0.281)	0 (13.997)
10	0 (688)	0 (15.421)

Table 7: Results of comparisons between two methods

The result of these tests was that the *Modified Tabu Search* algorithm was chosen as the better of the two and subsequently was selected to be the primary method to be used within the project. This decision was made due to the fact that the though the results are occasionally worse for it than those produced by the *Basic Population Evolution* method, it is not significantly so. (In general the difference is only an extra clash with the same clash density or a slight deviant in the desired location for the lunchtime slot.) This when considered alongside the vast difference in time to produce a schedule (which is nearly always a ratio of over 17:1) makes the first method a clear choice.

Chapter 8

GUI and Graphic Output

Once the main part of the projects development had been completed and the algorithms had been designed and implemented it was then decided that the project required an appropriate method to allow input of the various jobs and output of the completed schedule.

8.1 Output

The first of these issues to be addressed was the one regarding the output and how it should best be presented for the user to view. The initial solution to this task was a simple text based representation that would show: the actual schedule, the busy time slots and the overlapping time slots as well as listing each jobs start time and give a simple representation of the schedule. This was then further extended to allow for this representation to be saved as a text file. A sample of this kind of data is show in Figure 10.

```
Job List:

Job 1 starts at 0
Job 2 starts at 3
Job 3 starts at 5
Job 4 starts at 8
Job 5 starts at 10

Lunch Break: No

Graphic Display:
 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2

 1 1 - - - - - - - - - - - - - - -
- - 2 - - - - - - - - - - - - - -
```

Figure 10: Sample of textual output

It was then decided that a more visual output would be attractive option to allow for more easy user interpretation of the produced schedule. This was achieved by creating a template for a web page that can automatically display any generated schedule in a visual fashion along with all the other data in the text version. In this template each of the jobs were shown along with information about each of the time slots as well as clearly showing the lunch break slot.

A sample of this web page's table is shown in Figure 11 with a full example available in Appendix C.

Chemotherapy Nurse Day Schedule

Date: 01.02.1993

Patient ID:	1	2	3	4	5
Regime:	1	3	6	23	5
9:00					
9:15					
9:30					
9:45					
10:00					
10:15					

Figure 11: Sample of web page output

The use of such a template was inspired by othes work [8] but the implementation here was done independantly.

8.2 Input

The other of these issues to be addressed was the input of the jobs that were to be scheduled to appear on a single day. For this to occur easily it was decided that a simple graphical user interface would be the best choice. On this interface there would need to be an place for the job information to be entered and a button for the schedule creation action to be commenced. In addition to this it was decided that for ease of user only the treatment ID number would be required to be entered into the user interface, with a error message being displayed if this was not done correctly.

This simple design was then extended to allow for an option of how the output should be produced which the user could then select, to make the choice between previous described the two output methods.

An outline of this graphical user interface is shown in Figure 12.

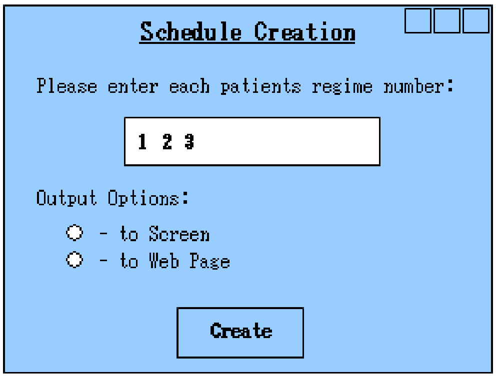


Figure 12: Outline of design for user interface

After some thought into the evaluation of the different local search algorithms it was decided that the user should be given the option of which to use due to differing benefits of the two methods. To this end it was decided that a selection box should be added to the user interface from which the algorithm could be selected. This complete user interface is shown in Figure 13.

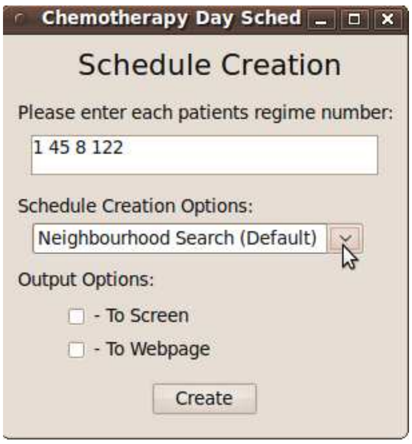


Figure 13: systems Graphical User Interface

Chapter 9

Experimentation

9.1 Plan

Once the algorithms to create schedules had been implemented and then polished the next phase of the project could begin. This phase was the one which dealt with the testing of the final product to access how well it met its target of creating good solutions (good being defined as one which best met the composite objective function criteria) within a small time period. To this end there was wide range of different complexity problem instances used to test the system with the goal of getting the best assessment its of the capabilities and robustness.

The test instances used for this process were taken from schedules which had been assigned to the nurses working in a real outpatient chemotherapy clinic and each comprised of up to 12 different patients to be scheduled for treatment on a single day with the regimes for the patients varying from 1 unit treatments to others comprising of 5 operations with multiple delays between them and with total lengths of over half the predefined scheduling period.

Within these problem instances an assessment was carried out to gauge the relative 'hardness' of the different problems to allow for some tests to be done only upon a percentage of those which were considered the most complex. To do this end an investigation was carried out into what would cause some problems to be harder to create schedules for than others. The result of this investigation was that predominately the problems that were harder were those which which contained the highest total number of operations to be carried out within the jobs. This then was a separate value to that of the number of jobs as it meant that it was more important to consider the length of the regime than how many of them there were to be scheduled. Also during this investigation it became clear that those regimes which had delays within them were harder to place, so schedules containing high numbers of these were normally among the most complex (it should also be noted that there is a probable link between these two conclusions as the regimes with the higher numbers of operations are generally those which also contain delays within them.)

Within the tests that were to be carried out it was decided to monitor two attributes within the resulting solutions, the first of these was the value assigned to the final solution by the objective function with the second being the time taken from starting the algorithm until the final value was returned. It was also decided that due to the none deterministic nature of the local search algorithms that each of the instances should be tested more than once to allow for a true measure of the ability of the system to be gained along with a view of the range of its performance. With this in mind the values of both time and objective function result were then recorded for each application of the instance with the best and worst results being kept along with also working out the average results for the number of applications.

To allow for wider consideration of the final products abilities, all of the tests which were carried out upon it were also repeated for two different methods which had been developed during the project. These two methods were firstly the alternative local search algorithm *Basic Population Evolution* and secondly a method of repeat application of the constructive heuristic using random ordering for job input which was briefly discussed at the end of the Constructive Heuristic Section. This repetition of tests would then hopefully allow for a comparisons to be drawn and the relative success of the final product to be accessed.

The tests that were carried out were:

- 400 Mixed Difficulty Instances each being applied 100 times
- 50 Hardest Instances each being applied 200 times

All the system tests were run on a computer with a Dual Core Intel Pentium D Processor running at 3.4Ghz with the times being based on that computers internal system clock.

9.2 Results

A summation of the results of the experimentation upon the final product is shown below with detailed results available within Appendix B.

9.2.1 Test 1 - 400 Mixed Difficulty Instances each being applied 100 times

This test was a general one over a mixed selection of problem instances and was primarily to assess the general behaviour of the algorithm over a range of input and over a suitably large number of applications.

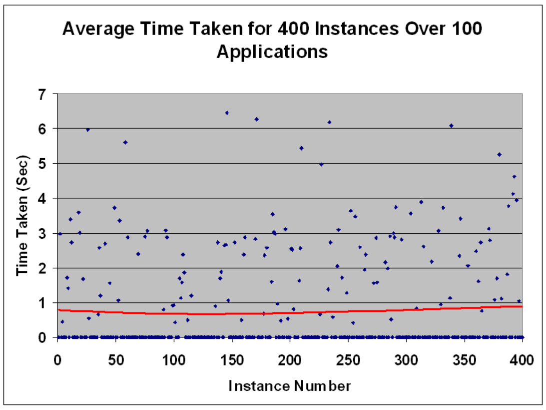
The important statistics that were gained from these results were firstly for the accuracy of the results: the best returned objective function value was 0, with the worst being returned being 133 and the overall average being 5 meaning that the range was 133. These numbers can then be translated into facts about the schedules produced with numbers over 100 representing a schedule containing at least one clash and numbers under 20 representing clash free schedules with only a slight variation in the placement of the lunchtime.

Next were the time taken to return the schedules, the best being 0.0002 seconds with the worst value of 28.8 seconds and with the overall average being 0.746 seconds. This then was taken to imply that the larger percentage of solutions must be closer to the lower end of this scale.

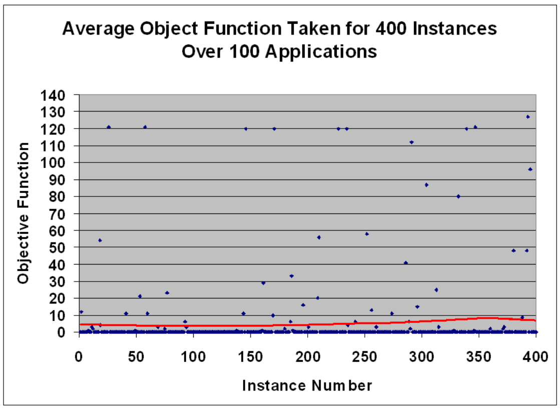
As well as this information there were a number of interesting observations regarding the distribution of results in the different sets of values:

- Firstly of the best values 385 (96.25%) returned the best result 0 meaning that after repeat applications nearly all of the instances will return a fault free schedule.
- Secondly of the worst results 281 (70.25%) are 0 so it can assumed that 281 always return 0.
- Thirdly of the best values 384 (96.0%) returned an answer in under 1 sec meaning that even those who returned 0 only 1 (0.25%) took longer than a second.
- Finally of the worst times, 300 (75.0%) are under 5 sec meaning that even for those instances not quickly finding a solution the end result is still found relatively fast.

To further access the general behaviour it was then decided to create scatter graphs on which to plot the average value retrieved from each instance to gain a rough measure of the overall behaviour within both the time and the objective function results. From this it was possible to see that general trend is towards a time of roughly 1 second with the objective function result being returned being between 5 and 10. The graphs used to assess this are shown in Figure 14.



(a)



(b)

Figure 14: Scatter graphs showing 400 instances with trend lines for: (a) Average times, (b) Average objective function results

The information of both the results and the trend line information on the scatter graphs then can be interpreted as that on average a solution was being returned without any clashes (due to all indications being of a normal value below 100) and in under a second, which can be seen hopefully as satisfying the projects aim of high accuracy schedules created in a minimal amount of time.

9.2.2 Test 2 - 50 Hardest Instances each being applied 200 times

This test was designed to further push the method to test how well it would perform when its input was made up of those instances judged to be the 'hardest' and most complex and also further ascertain its robustness when dealing with the harder problems over an extended number of applications.

The important statistics that were gained from these results were firstly for the accuracy of the results: the best objective function value was again 0, with the worst increasing slightly to 138 and the overall average becoming 26. This information was again translated with the average result this time suggesting that there was now a ratio of roughly (1:4) between those instances that were returned clash free and those containing a single clash. This whilst being an increase was not beyond that which should be expected when only considering those cases that were most complex.

Next was the times taken to return the schedule with the best time being 0.00047 seconds and the worst being 27.5 seconds this then leading to an overall average of 3.21 seconds which whilst being an extension on the mixed test results was reasonably fast when it is considered that all of the easiest and fastest timed instance have been removed.

As well as this information there were again a number of interesting observations that could be drawn from the sets of data:

- Firstly of the 50 best results 44 (88.0%) are 0 meaning that even over the hardest instances it was still possible in over three quarters of the cases to return a fault free schedule.
- Next of the 50 best times 43 (86.0%) are under 1 sec meaning that of the best results it was still possible to return values quickly.
- Finally of the 50 worst times 47 (94%) are between 5 and 20 seconds meaning that of the times when the instances complexity caused the method to take much longer to run it was still not a overly extended period of time in all but a few cases.

This information was then interpreted as meaning that when only the hardest instances were considered there was still relatively good results obtained but that there was the expected rise in the averages for both motioned characters due to the complexity of the problems.

9.2.3 Comparison to Alternative Methods

Once the tests had been carried out on the final product and suitable conclusions had been drawn, they were repeated upon the two alternative methods. The results of testing these two methods are shown alongside the same results for the main algorithm in Tables 8 and 9 below with more detailed results for both being available in the appendix B.

Attribute	Method 1 Modified Tabu Search	Method 2 Basic Population Evolution	Method 3 Multiple Random Iterations of Constructive Heuristic
Best Result (% At this)	0 (96.25)	0 (96.25)	0 (68.25)
Worst Result (% At this)	133 (0.25)	130 (0.75)	170 (0.25)
Average Result	5	5	37
Best Time	0.0002	6.47	0.014
Worst Time	28.8	19.9	0.367
Average Time	0.746	12.9	0.066

Table 8: Results for the first test (400 instances with 100 applications) being run on all 3 methods.

Attribute	Method 1 Modified Tabu Search	Method 2 Basic Population Evolution	Method 3 Multiple Random Iterations of Constructive Heuristic
Best Result (% At this)	0 (88)	0 (84)	0 (2)
Worst Result (% At this)	138 (2)	130(2)	170(2)
Average Result	26	27	122
Best Time	0.00047	8.66	0.058
Worst Time	27.5	21.36	0.267
Average Time	3.21	13.15	0.14

Table 9: Results for the second test (50 instances with 200 applications) being run on all 3 methods.

From these two tables a number of different conclusions can be drawn with the most important being centred on how the two extra methods results compare to the main set. Also important is how the two features of time taken and objective function value are comparatively balanced within each of the methods results.

These conclusions are:

- The objective function results for methods 1 and 2 are very similar in both tests with the only difference being slightly better values for worst results for method 2.
- The time values have a clear division between the three methods with a order always being apparent of method 3 coming first followed by method 1 and then method 2.
- The objective function results of Method 3 appear to not be close to the quality of the others with only 2% of the instances having a best result of 0 in the second test.

To allow for further comparison between the methods further scatter graphs where created showing the results for all three methods for each of the categories ’best’, ’worst’ and ’average.’ The most conclusive of these were the results for the 50 hardest instances; this is due to the number of points being small enough for realistic human interpretation whilst also being large enough for groupings to be identified. These scatter graphs are presented in figures 15, 16 and 17. It should be noted that within these figures it is clear to see that there is division between those that are clash free (the group at roughly 0 on on the objective function axis) and those which contain clashes (the group at the level of 100 and higher on the objective function axis.)

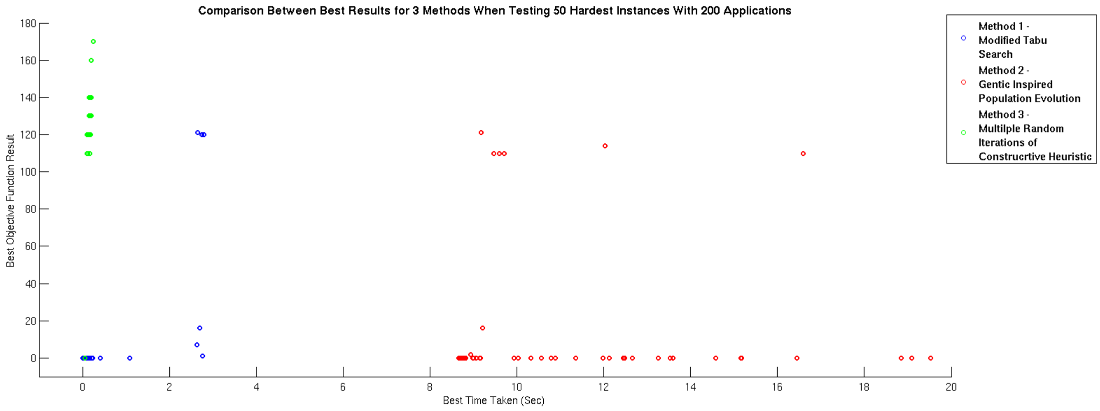


Figure 15: Best results for each of the instances with clear groupings being seen for each of the methods.

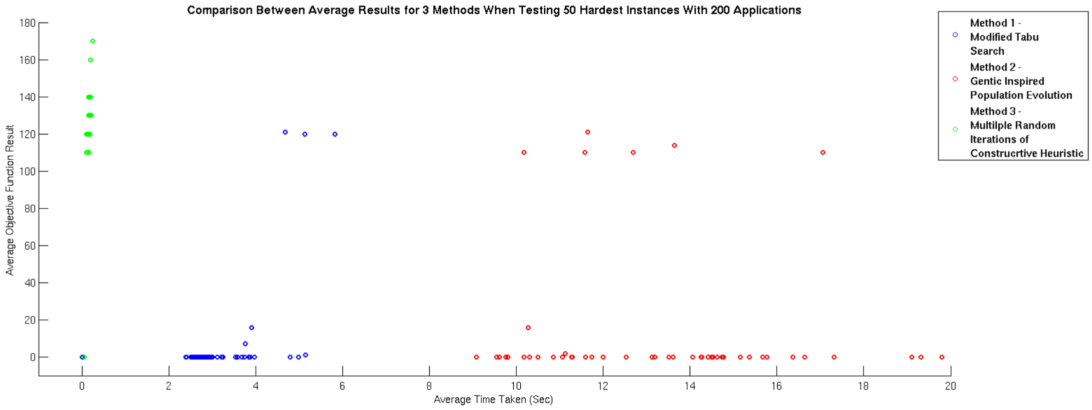


Figure 16: Average results for each instance again showing clearly the groupings thought less closely grouped than Figure 15.

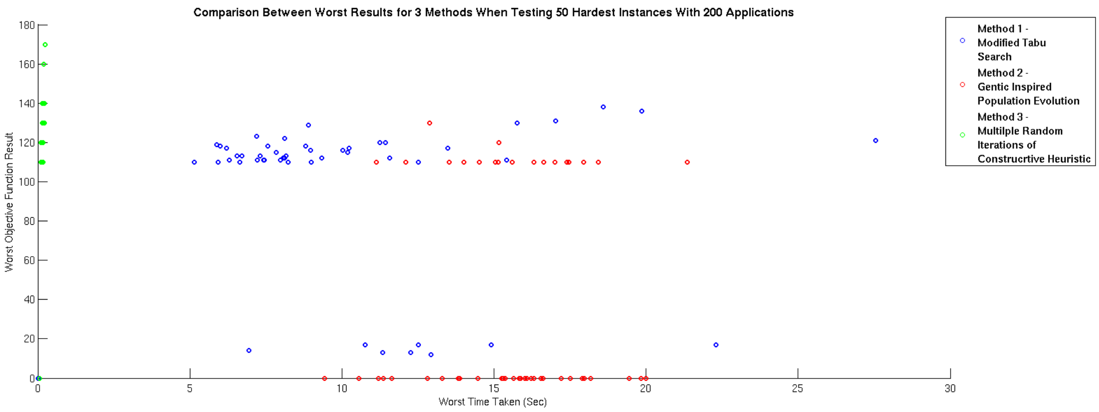


Figure 17: Worst results for each instance with a clear group of the results from method 3 but a more equal group from methods 1 and 2 with the only difference being that the greater percentage of method 1 is produced at a faster time than those of method 2.

From the three graphs it can be seen that in most cases each of the methods results group together into a clearly defined areas. These areas in general can be related in an approximate manner to the behaviour of the representative methods and when combined with the results shown in the tables above can be used to make summary statements about each of the three methods and how they compare to one another.

Method 1 Appears to be a good trade off between the positive characteristics of the other two methods. This is because it has better objective function results than method 3 and roughly similar ones to those found method 2 but these results are achieved in general faster than those in method 2 meaning that it is the clear mid point between the two and a good overall balance of the two examined features.

Method 2 Is comparable with the objective functions result for method 1 and in the more complex cases can occasionally produce better results but in general runs significantly slower than method 1 or method 3 signifiying again the main reason for it not being selected as the primary method for the program.

Method 3 Is uniformly faster that the other 2 methods but also uniformly limited in the quality of its results meaning that they rarely compete with those of other methods and so cannot then be seen to be true competitor.

9.3 Conclusion

Experimentation and analysis of the results has shown that the project's end product successfully finds a method of scheduling a nurse's patients with minimal clash density and schedule creation time. This is shown clearly in the results of test one and again in test two under less favourable circumstances, the latter also establishes the system's robustness.

It is also clear that the methods employed by the software successfully conclude with the best option being returned as the primary choice. This is shown by evidence gathered from comparisons undertaken with *modified tabu search* appearing as quantifiably the best mediation between the desirable features.

Also in overall conclusion, the end product has proven itself to be very reliable since for a total of 50,000 applications over all instances correct and error-free results were produced. Therefore, provided a correct input is provided it can be reasonably assumed that the system will always provide a schedule.

Chapter 10

Evaluation

Once the development phase had ended and the experimentation had been completed and reviewed there was then a period of evaluation of how the different stages of the project had progressed and the positives and negatives of the decisions made within them.

10.1 Evaluation Satisfaction of Minimal Requirements

The minimum requirements that were stated at the start of this project were based around the creation of simple heuristic method of creating schedules and then possibly extending them to develop other methods for comparison or improvement. At this point it is believed to be safe to firmly conclude both of these have been met with the problem being solved as results that were reported in the previous chapter confirm. In that chapter it was found that on the whole of the wide range of instances tested all would return a schedule and that 70% of them would always return a 'perfect solution' (which is clash free and with the lunchtime slot placed in the ideal position) whilst doing this 75% would also return an answer in less than 5 seconds. These two facts then appear to give the impression the problem has gone beyond just being solved and appears to surpass the original objectives. This then has provided the evidence to make it possible to conclude that there is a good potential for the application of heuristics in this area and that if they are designed properly could both compete with the accuracy levels of the exact solvers and also potentially do so faster.

10.2 Methodology Evaluation

The methodology selected for this project was a hybrid one based around a core of the two main parts evolutionary development, throwaway prototyping and exploratory development that was then combined with a phase of clean room inspired testing.

The throwaway prototyping worked well and allowed for a wide range of approaches to be tested and then compared to others. This hopefully allowed for the best methods to be developed something which could possibly have been assured by an extension to introduce a quantifiable marking scheme for assessment within the prototypes but in this instances worked well without.

The exploratory development also appeared to function well but the necessity of designing working prototypes did mean that a lot of the development that would normally have been associated with exploratory section was instead present within the throwaway prototyping component. The result of which was that this became less useful than it had potential to be and perhaps giving an indication of something that could be reviewed if this hybrid methodology was to be used again.

The clean room based testing was used to ascertain the methods reliability and also to quickly test parameter values when fine tuning the algorithms and potentially resulted in them reaching the best possible solutions and becoming polished to a high degree.

10.3 Planning Evaluation

The planning within this project was one of the components that was identified at an early stage to be a clear point of importance with obviously so much of the projects progress being linked to it.

The final evaluation of this element is that it appears to have worked well with only a few slight oversights regarding the time taken to produce documents (which is more down to the process of improving drafts than the production of the original version). When not considering this issue it was clear that the overall plan did work as can be seen by the comparison between the two Gantt chart included previously in the planning section. This comparison shows that there are only small changes which are due to the delays that were forced to occur due to the previously mentioned oversight.

10.4 Algorithm Design Evaluation

Two categories of algorithms were designed within this project; these were the constructive heuristic and the local search algorithm.

The constructive heuristic development and the resulting final product was a successful process as the heuristic did exactly what it was intended to create a valid start solution. In retrospect though the design process of this part would have been improved if more thought had been given to a higher degree of efficiency at an earlier stage. This became clear during the time complexity analysis within the latter stages of the development as it was discovered that neither of the two final candidate algorithms were at there most efficient. Due to this there are then possible improvements which could be made to the constructive heuristic but it was not deemed important enough due to this method only providing an initial seed for the local search algorithm to spend further time on here.

The local search algorithm development was also a very successful process with the result of two methods which produced comparably good results with final selection being made on which was the faster option. The production and design of these algorithms was a successful procedure with the only issue arising being as observed previously that the constraints upon the solutions stopped the standard methods for genetic algorithms being implemented directly. This though did not overly distract from the success of the method once a suitable repair mechanism had been designed. The main area which was not explored to its desired extent was that of further development of the genetic design to allow for inclusion of aspects of other parts of local search a composite which had promise to allow for even further improvements in the solution. The only other way this algorithm could have been improved would have been to test it over the hardest instances then activity attempt to take steps to improve the results specifically for those which returned the worse results, though this is something which equally have degraded the performance when dealing with other areas so was not counted as an essential task.

10.5 Implementation Evaluation

The implementation of the solution was done in the programming language python and overall the process was problem free. The only issue that did arise was with certain rules regarding the manipulation of the list data structure causing issues until a method of avoiding this could be discovered. Beyond this issue the implementation was a smooth process with all design features being transferred easily and the language of choice proving to be both versatile and dynamic. Two positive qualities which were greatly emphasized by the later stage decision to extend the project to include friendly input and output which possible with the minimum amount of additional work.

10.6 Experimentation and Testing Evaluation

The testing procedure was based around recording the average, worst and best values retrieved for the results of a number of applications over a variety of instances. A 'hardness' measure was also taken and this allowed for the most complex instances to be selected for particular testing to ascertain the robustness of the solution. The tests were then carried out on both the final solution and two other methods to allow for comparison.

This comparison then resulted in clear evidence that the final solution method was a balanced point between the extremes of the other two and appeared that to suggest the final product met the criteria of relatively good solution schedules being produced quickly. The main desirable feature that was not present for the testing if there had been a way in which to contrast the results of the final product against results found for the same problem by an exact solver to ascertain the differences between the two methods and give a much more realistic measure of the end products potential usefulness.

10.7 Further Work

Within the scope of this project there were a great many interesting directions only a small percentage were possible to be explored. This has resulted in there still be a sizeable number of possible directions for research to be carried out into related subjects, an example of this being the application of simulated annealing within the problem.

Some interesting future directions have been mentioned previously and one of these which is assigned particular importance is the investigation of hybrid local search with elements being taken from both genetic algorithms and the more traditional areas that were explored. This would have the potential to possibly produce the higher accuracy results generated from the *basic population evolution* whilst also achieving comparable speeds to the *modified tabu search*. This would be the next logical step in the development of the particular direction of work followed here.

A final interesting future move would be the exploration of a full NP-hardness proof for the problem which is strongly believed to be possible but was forced to be outside of the scope of this project due to the higher importance assigned to good algorithm production and the overall time constraints.

Bibliography

[1] E. H. L. Aarts, J. H. M. Korst, and P. J. M. van Laarhoven. *Local Search in Combinatorial Optimization*, chapter 4, pages 91–120. Addison Wesley, 1997.

[2] A. A. Ageev and A. E. Baburin. Approximation algorithms for uet scheduling problems with exact delays. *Operations Research Letters*, 35:533–540, 2007.

[3] A. A. Ageev and A. V. Kononov. Approximation algorithms for scheduling problems with exact delays. *Lecture Notes in Computer Science*, 4368:1–14, 2006.

[4] K. S. Amirthagadeswaran and V. P. Arunachalam. Improved solutions for job shop scheduling problems through genetic algorithm with a different method of schedule deduction. *International Journal of Advanced Manufacturing Technology*, 28:532–540, 2006.

[5] R. Battiti and G. Tecciolli. The reactive tabu search. *O.R.S.A - Journal on Computing*, 6:126–140, 1994.

[6] J. Blazewicz, K. Ecker, T. Kis, C. N. Potts, M. Tanas, and J. D. Whitehead. Scheduling of coupled tasks with unit processing times. Submitted to Journal of Schduling.

[7] X. Cair, H. Langtangan, and H. Moe. On the performance of python programming language for serial and parallel scientific programming. *Scientific Programming*, 13:31 – 56, 2005.

[8] A. Condotta. Personal Communication.

[9] C. A. Glass and C. N. Potts. A comparison of local search methods for flow shop scheduling. *Annals of Operations Research*, 63:489–509, 1996.

[10] J. N. D. Gupta. Comparative evaluation of heuristic algorithms for the single machine scheduling problem with two operations per job and time lags. *Journal of Global Optimization*, 9:239–253, 1996.

[11] A. Hertz, E. Taillard, and D. de Werra. *Local Search in Combinatorial Optimization*, chapter 5, pages 121–137. Wiley-Interscience, 1997.

[12] V. Lehoux-Lebacque, N. Brauner, and G. Finke. Identical coupled task scheduling: polynomial complexity of cyclic case. *Centre National de la Recherche Scientifique*, 2009.

[13] H. D. Mills and M. Dyer. Cleanroom software engineering. *IEEE Software*, 4:19–24, September 1987.

[14] H. Muhlenbein. *Local Search in Combinatorial Optimization*, chapter 6, pages 137–170. Wiley-Interscience, 1997.

[15] E. Newton. Python for critical applications a case study. In *10th International Python Conference*, 2002.

[16] A. J. Orman and C. N. Potts. On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72:141–154, 1997.

[17] C. N. Potts and J. D. Whitehead. Heuristics for a coupled-operation scheduling problem. *Journal of Operation Research Society*, 58:1375–1388, 2006.

[18] L. Prechelt. An empirical comparison of c, c++, java, perl, python, rexx and tol for a search/string processing programs. Technical report, International Report Faculty of Computer Science, Univerisity Of Karlsruke,, 2005.

[19] J. Riezebos and G. J. C. Gaalman. Time lag size in multiple operations flow shop scheduling heuristics. *European Journal of Operation Research*, 105:72–90, 1998.

[20] J. Riezebos, G. J. C. Gaalman, and J. N. D. Gupta. Flow shop scheduling with multiple operations and time lags. *Journal of Intelligent Management*, 6:105–115, 1995.

[21] K. Sastry, D. Goldberg, and G. Kendall. *Search Methodologies: Introductary Tutorials in Optimization and Decision Support Techniques*, chapter 4, pages 97–125. Springer, 2 edition, 2006.

[22] I. Sommerville. *Software Engineering*, chapter 4, pages 65–81. Addison Wesley, 8th edition, 2007.

[23] I. Sommerville. *Software Engineering*, chapter 17, pages 396–407. Addison Wesley, 8th edition, 2007.

[24] L. Tseng and Y. Lin. A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 198:84–92, 2007.

[25] W. Yu. *The two-machine flow shop problem with delays and the one-machine total tardiness problem*, chapter 3, pages 30–31. 1996.

[26] W. Yu, H. Hoogeveen, and J. Lenstra. Minimizing makespan in a two-machine flowshop problem with delays and unit-time operations is np-hard. *European Journal of Operation Research*, 7:333 – 348, 2004.

Appendix A

Personal Reflection

I found the project to an interesting insight into how skills included within a computer science degree can be applied. During the course of the project I learnt some valuable lessons and also gained a much greater level of insight into some of the techniques and methods I was applying. Those I consider to be the most important of these I have listed below:

The first of these was that I learnt a lot about the amount of consideration that is required when first analysing a problem such as this, there are generally many different levels of detail which have to be considered then ether assigned an importance level or discarded. This whilst sounding trivial can actually dramatically affect both the complexity of the problem and the usefulness of the answers produced. I have discovered that this is something that requires a significant amount of thought prior to any attempt being made to produce a solution and I would advise anyone starting a project in a similar area to allocate sufficient time for this as I felt that the success of my project was partially down to the time spent considering the problem.

Another of these regards the solution finding styles that were used, this is primarily centered around advice for anyone considering scheduling problems to try to include experimentation with global solution methods. This is advised due to my observation that whilst the local greedy methods are easy to design and implement there is an substantial amount of promise in global methods which would possibly result in efficient algorithms being created if the correct amount of time and effort was put into there production. This is something which was breifly touched upon in the constructive heuristic prototyping and that I would have persuaded further in my project had the constructive heuristic been more important.

Also one of these concerns specifically the application of genetic local search methods, these are as I discovered a very useful tool and definitely something that I would advise be considered for any problem that concerns scheduling. The lesson that was learnt from this though was that when these are being considered for a problem the way the information is represented and the constraints upon it need to be investigated fully, otherwise there is the potential to start experimenting with prototypes before it is acknowledged that there is the potential for large problems should ether of the standard operations (crossover/mutation) be applied. This is down to the way that if there are strict constraints upon the representation then large proportion of the offspring generated by the method may be be invalid so a repair procedure maybe required (as in my project) for the effective use of the these methods. This I felt could have been avoided if the representation had been design with this risk in mind.

A final observation regards the time planning aspect of the project, this is not the standard comment relating to needing to better manage my time, as from the start of this project I made it a personal objective not to fall behind. Rather it is more based around advising that a flexible approach be taken especially when considering the different deliverables that have to be produced. This is due to there being a range of unforeseen circumstances that will most likely require time to be spent reviewing and altering documents and presentations that can cause delays if not prepared for so a buffer would be best built in around these.

Appendix B

Experimentation

B.1 Evolutionary Development Testing

Results of tests carried out during the local search exploratory development to ascertain values and fine tune the end product.

B.1.1 Neighbourhood Creation Method

Problem Instance	Neighbour Creation Method	
	2 Swap	Insert
1	110.1 (3482)	112.0 (7199)
2	0 (2111)	0 (4523)
3	0 (6673)	0 (7660)
4	0 (2333)	1 (4161)
5	0 (2255)	0 (1510)
6	0.25 (2901)	0 (5023)
7	0 (3542)	0 (4288)
8	111.4 (4009)	111.9 (6188)
9	0 (2605)	1.4 (6552)
10	0 (3338)	0 (3956)

B.1.2 Tabu List Length

Problem Instance	Tabu List Length				
	5	10	15	20	30
1	111.6 (6882)	111.4 (3369)	111.4 (4703)	111.4 (6001)	121.8 (4285)
2	0 (4981)	0 (1227)	0 (2632)	0 (447)	0 (2495)
3	0 (6673)	0 (7660)	0 (3435)	0 (1231)	0 (2454)
4	0 (2333)	0 (469)	0 (435)	0 (564)	0 (369)
5	0 (2255)	0 (1510)	1 (1446)	0 (1257)	1 (2255)
6	0 (4882)	0 (2874)	0 (2501)	0 (1780)	0 (2587)
7	0 (3200)	0 (3302)	0 (2228)	0 (2140)	0 (2109)
8	111.5 (3698)	111.9 (4102)	111.5 (4454)	111.2 (2509)	120.2 (5082)
9	0 (5268)	0 (2001)	0 (2780)	0 (948)	0 (2505)
10	0 (5899)	0 (4156)	0 (1999)	0 (3420)	0 (2370)

B.1.3 Threshold Value

Problem Instance	Percentage of Initial Solution				
	25%	50%	75%	100%	200%
1	111.4 (2615)	111 (3215)	112 (4555)	111 (2108)	111.2 (2358)
2	0 (1227)	0 (5166)	0 (2131)	0 (160)	0 (862)
3	0 (1991)	0 (3528)	0 (3870)	0 (6281)	0 (1113)
4	0 (742)	0 (582)	0 (268)	0 (856)	0 (2204)
5	1 (975)	0 (1068)	0 (870)	0 (4714)	0 (4311)
6	0 (1509)	0 (2674)	1.8 (2455)	0 (1901)	0 (2458)
7	0 (1340)	0 (2890)	0 (2107)	0 (2785)	0 (2690)
8	111.2 (2501)	111 (3301)	112.2 (3699)	111 (2895)	111.6 (3698)
9	0 (605)	0 (1900)	0 (899)	0 (369)	0 (2023)
10	0 (1805)	0 (2523)	0 (2488)	0 (2400)	0 (2566)

B.1.4 Decrease Size

Problem Instance	Percentage of Threshold				
	1%	5%	10%	20%	50%
1	110 (2155)	111 (1235)	110.6 (1562)	110.7 (1802)	121 (1108)
2	0.5 (1909)	0 (950)	0 (1021)	0 (1230)	1 (380)
3	0 (554)	0 (740)	0 (1973)	0 (453)	0 (338)
4	0 (1103)	0 (1097)	0 (201)	0 (230)	0 (193)
5	2 (5041)	1 (1335)	0 (5087)	0 (2232)	1 (1080)
6	0 (2058)	0 (1071)	0 (1895)	0 (1125)	0 (788)
7	0 (2265)	0 (1028)	0 (2041)	0 (981)	0 (1102)
8	110 (2150)	111.2 (1119)	110.9 (2158)	111 (2156)	121.6 (1023)
9	0 (1900)	0 (1425)	0 (1650)	0 (1180)	0 (453)
10	0 (2177)	0 (1325)	0 (1899)	0 (1259)	0 (1522)

B.1.5 Population Size

Problem Instance	Population Size					
	8	16	32	64	128	256
1	122.6	125	124.6	112.2	110	110.1
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	123	124.4	119.6	112.4	110	108.9
9	0	0	0	0	0	0
10	0	0	0	0	0	0

B.1.6 Number of Generations

Problem Instance	Number of Generations				
	25	50	100	200	500
1	121.6	121.3	121.5	123.7	112.7
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	1	0	0	0
6	1	0	0	0	0
7	0	0	0	0	0
8	120.6	120.9	121.4	123.1	112.9
9	0	0	0	0	0
10	0	0	0	0	0

B.2 Evaluation Testing

B.2.1 Method 1: Modified Tabu Search

Full results of the large amount of testing carried out on the final system with 400 instances being used with 100 applications of each for the modified tabu method.

Inst. Num.	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
1	0	0	0	0.000583887022	0.00233697835	0.00063584
2	0	111	12	0.119632959366	6.58366722	2.97349398
3	0	0	0	0.0004160400532	0.000616687	0.000432493
4	0	0	0	0.0033540725708	1.87308001518	0.4504167366
5	0	0	0	0.000378842192	0.000557475098	0.00039255
6	0	0	0	0.000406984526	0.000626188721	0.00043099

Inst. Num.	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
7	0	0	0	0.000365018844604	0.00053596496	0.00037928
8	0	13	1	0.0622971057892	4.53768396	1.70457417727
9	0	5	0	0.0521790981293	6.27193999	1.40764209986
10	0	0	0	0.000352144241333	0.000545071826	0.000370
11	0	17	3	0.0961410999298	7.896933077	3.3846368574
12	0	17	2	0.0751891136169	7.431444109	2.722356928
13	0	0	0	0.000436067581177	0.00064382153	0.0004541
14	0	0	0	0.000381946563721	0.000587064	0.0004066
15	0	0	0	0.000324010848999	0.00049321582	0.0003437
16	0	0	0	0.000288009643555	0.000466144	0.0003021
17	0	0	0	0.000409841537476	0.0006519526	0.0004302
18	0	115	54	0.541888952255	7.852305387	3.59682622
19	0	110	4	0.149537086487	5.82425308	3.001589686
20	0	0	0	0.000419855117798	0.0007779292	0.0004489
21	0	0	0	0.00041389465332	0.0006699275	0.00043698
22	0	13	0	0.0265700817108	5.40851902	1.679467730
23	0	0	0	0.000252962112427	0.0004375493	0.0002637
24	0	0	0	0.000329971313477	0.0005105015	0.00034255
25	0	0	0	0.000297784805298	0.000488737	0.00030645
26	120	130	121	2.77711200714	11.85239	5.96552836
27	0	0	0	0.0162670612335	2.434061050	0.5441145137
28	0	0	0	0.00043797492981	0.0006518394	0.000461301
29	0	0	0	0.000391960144043	0.000602011	0.00040830
30	0	0	0	0.000296115875244	0.0006530459	0.00032079
31	0	0	0	0.000293016433716	0.000514053	0.00031174
32	0	0	0	0.000322103500366	0.0005240065	0.00033683
33	0	0	0	0.000463962554932	0.0008220242	0.00049203
34	0	0	0	0.00040602684021	0.000675911	0.000428304
35	0	2	0	0.00735092163086	4.707567932	0.6594123649
36	0	12	0	0.0301649570465	11.42247602	2.579914562
37	0	1	0	0.0306611061096	5.31450200081	1.1948862603
38	0	0	0	0.000391960144043	0.0007531291	0.000414699
39	0	0	0	0.000301837921143	0.0004928203	0.000316352
40	0	0	0	0.000279903411865	0.000441071	0.00028599
41	0	114	11	0.0472569465637	4.88873791	2.6946564575
42	0	0	0	0.000383138656616	0.00259187	0.000437433
43	0	0	0	0.000278949737549	0.00180793	0.000326345
44	0	0	0	0.000375986099243	0.00179905	0.000411452
45	0	4	0	0.00173377990723	6.950800184	1.557773439
46	0	0	0	0.00036883354187	0.000577925	0.0003939712
47	0	0	0	0.000303983688354	0.00047490	0.0003183749
48	0	0	0	0.000383138656616	0.00064807	0.000399539
49	0	14	1	0.165038108826	13.28139805	3.73252603

Inst. Num.	Best Res.	Worst Result	Aver. Result	Best Time	Worst Time	Average Time
50	0	0	0	0.000395059585571	0.000602002231	0.000427937
51	0	0	0	0.000370979309082	0.000648027998	0.000386484
52	0	2	0	0.0163221359253	5.633172032	1.066789237
53	0	114	21	0.0419099330902	13.30081937	3.360940324
54	0	0	0	0.000396966934204	0.00059454858	0.000413217
55	0	0	0	0.0004429817171	0.00064587306	0.000460292
56	0	0	0	0.000252008431	0.0004458492	0.00026237211
57	0	0	0	0.000345945358276	0.000547885894775	0.000359109
58	120	128	121	2.76895594597	19.1318039894	5.602410469
59	0	0	0	0.000413894632	0.0027688146	0.0004601631
60	0	114	11	0.0884749889374	5.90038895607	2.86994430
61	0	0	0	0.000298976898193	0.000478251099	0.000317742
62	0	0	0	0.000411987304688	0.00059157593	0.000428451
63	0	0	0	0.000302076339722	0.000502783325	0.000323637
64	0	0	0	0.000288963317871	0.000477576782	0.000302635
65	0	0	0	0.000414133071899	0.00076656897	0.0004397436
66	0	0	0	0.000414848327637	0.00060535376	0.0004297683
67	0	0	0	0.000405073165894	0.000579984375	0.000422983
68	0	0	0	0.000396013259888	0.000593424805	0.0004159747
69	0	110	3	0.0631411075592	5.39887853	2.39407066
70	0	0	0	0.000391006469727	0.000635094727	0.00040458911
71	0	0	0	0.000419855117798	0.000692135132	0.0004552030
72	0	0	0	0.000360012054443	0.000580402954	0.0003805041
73	0	0	0	0.000425815582275	0.000828399536	0.0004495692
74	0	0	0	0.000326871871948	0.000500434692	0.0003387546
75	0	15	2	0.123622894287	6.320387847	2.9013403726
76	0	0	0	0.000287055969238	0.000494295898	0.0003032207
77	0	116	23	0.268184185028	6.02354099	3.0609108403
78	0	0	0	0.0003900527541	0.0003980542	0.0004012717
79	0	0	0	0.0003159046731	0.0004858472	0.000326397
80	0	0	0	0.000298976898193	0.000478251099	0.00377610
81	0	0	0	0.000425100326538	0.000680461914	0.0004391
82	0	0	0	0.000411033630371	0.000581751587	0.0004608
83	0	0	0	0.000411033630371	0.00068936792	0.00042568
84	0	0	0	0.000288009643555	0.00049908606	0.00030958
85	0	0	0	0.000288009643555	0.000487157104	0.0003359
86	0	0	0	0.000301122665405	0.000487157104	0.00038348
87	0	0	0	0.000366926193237	0.000632653198	0.00035327
88	0	0	0	0.000220060348511	0.000411630371	0.00027768
89	0	0	0	0.000396013259888	0.000669532959	0.00046887
90	0	0	0	0.000372886657715	0.000587216064	0.0003973
91	0	11	0	0.0361490249634	3.64324497	0.79208057642

Inst. Num.	Best Res.	Worst Result	Aver. Result	Best Time	Worst Time	Average Time
92	0	0	0	0.000411033630371	0.00062084197998	0.000428837730
93	0	111	6	0.0653789043427	13.424959898	3.074589994
94	0	110	3	0.0875599384308	6.10278201103	2.87200146
95	0	0	0	0.000430107116699	0.000607967376709	0.0004471659
96	0	0	0	0.000420093536377	0.000607013702393	0.0004489374
97	0	0	0	0.00037693977356	0.00055193901062	0.000394377708
98	0	0	0	0.000428915023804	0.000617980957031	0.0004459023
99	0	1	0	0.000619888305664	4.99218082428	0.9113142204
100	0	1	0	0.0271589756012	5.4008641243	0.9374041005
101	0	0	0	0.010290145874	1.57975482941	0.4296668362
102	0	0	0	0.000348806381226	0.000539064407349	0.00036238673
103	0	0	0	0.000277996063232	0.000456094741821	0.00029293775
104	0	0	0	0.000311851501465	0.000514030456543	0.0003832550
105	0	4	0	0.0263729095459	5.8445289135	1.7014604044
106	0	9	0	0.0327529907227	5.52825379372	1.131644942
107	0	9	0	0.0269558429718	6.35019803047	1.579400990
108	0	11	0	0.0398740768433	8.15576601028	2.376331724
109	0	6	0	0.0365970134735	6.54781985283	1.865455777
110	0	0	0	0.000280857086182	0.000487089157104	0.0002959513
111	0	0	0	0.000446081161499	0.000797986984253	0.0004697394
112	0	1	0	0.0199368000031	4.27877306938	0.50319822549
113	0	0	0	0.00047492980957	0.000686168670654	0.00049540511
114	0	0	0	0.000427007675171	0.000724077224731	0.0004447675
115	0	5	0	0.047257900238	5.94633698463	1.19382843256
116	0	0	0	0.000304937362671	0.000510215759277	0.00031849158
117	0	0	0	0.000408172607422	0.000733137130737	0.00043489277
118	0	0	0	0.000378131866455	0.000728130340576	0.00040254344
119	0	0	0	0.000286817550659	0.00047492980957	0.000297257902
120	0	0	0	0.000387907028198	0.00058388710022	0.000410344606
121	0	0	0	0.000367879867554	0.000798940658569	0.00039406764
122	0	0	0	0.00041389465332	0.000618934631348	0.000435953102
123	0	0	0	0.000399827957153	0.000593900680542	0.00041405962
124	0	0	0	0.000290870666504	0.000476121902466	0.00030751283
125	0	0	0	0.000408172607422	0.000787019729614	0.00042729661
126	0	0	0	0.000305891036987	0.000582218170166	0.00032810964
127	0	0	0	0.00031304359436	0.000488996505737	0.000328898498
128	0	0	0	0.000418901443481	0.000801801681519	0.00044714692
129	0	0	0	0.000416040420532	0.000627040863037	0.00044924760
130	0	0	0	0.000462055206299	0.000674962997437	0.00047726143
131	0	0	0	0.00041389465332	0.000617027282715	0.000433938532
132	0	0	0	0.000483989715576	0.00070595741272	0.000514450032
133	0	0	0	0.000356912612915	0.000563144683838	0.00038066371

Inst. Num.	Best Res.	Worst Result	Aver. Result	Best Time	Worst Time	Average Time
134	0	0	0	0.000352144241333	0.000566005706787	0.000368311
135	0	0	0	0.000304937362671	0.000505924224854	0.000320904
136	0	3	0	0.000653028488159	4.65219902992	0.893235135
137	0	0	0	0.000405788421631	0.000686168670654	0.000422921
138	0	11	1	0.0276799201965	10.8460850716	2.72170696
139	0	0	0	0.00039005279541	0.00059700012207	0.000415863998
140	0	1	0	0.042717218399	9.51933407784	1.69706550
141	0	9	0	0.0763649940491	5.70989990234	1.88254074
142	0	0	0	0.00030517578125	0.000494956970215	0.00031996488
143	0	0	0	0.000371932983398	0.000576972961426	0.000312702
144	0	115	11	0.118249893188	4.64356422424	2.638173342
145	0	5	0	0.0523569583893	8.99896001816	2.65445747
146	120	128	120	2.85134291649	11.5997879505	6.45924942
147	0	1	0	0.033951997757	6.2104511261	1.065534296
148	0	0	0	0.00036883354187	0.000584840774536	0.0003839015
149	0	0	0	0.000366926193237	0.000573873519897	0.000388522
150	0	0	0	0.000293970108032	0.000486135482788	0.000312051
151	0	0	0	0.00043797492981	0.000705003738403	0.0004585599
152	0	8	0	0.0558750629425	10.9200088978	2.723233904
153	0	0	0	0.000444889068604	0.000808954238892	0.0004687929
154	0	0	0	0.000293016433716	0.000496864318848	0.0003110599
155	0	0	0	0.000396966934204	0.000619173049927	0.0004290390
156	0	0	0	0.00040602684021	0.000740051269531	0.00043126344
157	0	0	0	0.00037693977356	0.000565052032471	0.00039795637
158	0	0	0	0.0204658508301	2.26740002632	0.4940696859
159	0	0	0	0.000258922576904	0.000457048416138	0.00027875434
160	0	14	1	0.0298180580139	8.13037014008	2.370470511
161	0	117	29	0.0974628925323	6.13223195076	2.8717919281
162	0	0	0	0.000408887863159	0.000618934631348	0.000431972
163	0	0	0	0.000361919403076	0.00057315826416	0.0003811930
164	0	0	0	0.000409841537476	0.00061297416687	0.0004283074
165	0	0	0	0.00040602684021	0.000769138336182	0.0004277499
166	0	0	0	0.000277996063232	0.000617980957031	0.000293963
167	0	0	0	0.000288963317871	0.000488996505737	0.000299514
168	0	0	0	0.000449895858765	0.000664949417114	0.000473964
169	0	0	0	0.000468015670776	0.000716924667358	0.000499528
170	0	111	10	0.0942540168762	4.86743092537	2.822302958
171	120	122	120	2.56967282295	11.7211389542	6.271421485
172	0	0	0	0.000406980514526	0.000615119934082	0.000424069
173	0	0	0	0.000365972518921	0.000555038452148	0.000383520
174	0	0	0	0.000304937362671	0.000488042831421	0.000319892
175	0	0	0	0.000419855117798	0.000626802444458	0.000434998

Inst. Num.	Best Res.	Worst Result	Aver. Result	Best Time	Worst Time	Average Time
176	0	0	0	0.000395059585571	0.000662088394165	0.000416426863
177	0	1	0	0.00116205215454	4.99644207954	0.678282507
178	0	6	0	0.0323660373688	9.70146489143	2.361216668
179	0	0	0	0.000447034835815	0.000634908676147	0.000468432329
180	0	13	2	0.0738000869751	9.10759186745	2.582974622
181	0	0	0	0.000392913818359	0.000762939453125	0.000422484875
182	0	0	0	0.000375032424927	0.000559091567993	0.000387916561
183	0	0	0	0.000285863876343	0.000455141067505	0.000298783144
184	0	7	0	0.0270431041718	6.96048402786	1.591696122
185	1	111	6	2.52334499359	6.87815308571	3.538330098
186	0	119	33	0.190171957016	6.798609972	3.026743962
187	0	11	1	0.0707900524139	20.0039110184	2.975819444
188	0	0	0	0.000391960144043	0.000663042068481	0.000414271675
189	0	6	0	0.0435411930084	4.40428805351	0.965352685
190	0	0	0	0.000257015228271	0.000417947769165	0.000264909267
191	0	0	0	0.000290155410767	0.000476121902466	0.000303783416
192	0	1	0	0.000594139099121	3.94620203972	0.477638804
193	0	0	0	0.00043797492981	0.000811100006104	0.00046197175
194	0	0	0	0.000399112701416	0.000672101974487	0.000420863628
195	0	0	0	0.000378131866455	0.000577926635742	0.000395338535
196	0	117	16	0.262812137604	6.05369710922	3.11045557
197	0	0	0	0.000380992889404	0.000615119934082	0.000395920276
198	0	1	0	0.00059700012207	4.45708990097	0.523984100
199	0	0	0	0.000293970108032	0.000468969345093	0.000307152271
200	0	0	0	0.000290870666504	0.000459909439087	0.000302546024
201	0	110	3	0.0264949798584	6.93407487869	2.54805401
202	0	3	0	0.088268995285	7.40833187103	2.521121256
203	0	3	0	0.0280380249023	4.99908709526	0.813179805
204	0	0	0	0.00047492980957	0.000695943832397	0.0004933786391
205	0	0	0	0.000257968902588	0.000427007675171	0.000268790723
206	0	0	0	0.000328063964844	0.000530004501343	0.000347378257
207	0	0	0	0.000418901443481	0.000642061233521	0.000437068939
208	0	13	0	0.0256628990173	6.82102704048	1.62513528585
209	0	114	20	0.188678026199	5.87189102173	2.553343894
210	1	114	56	2.61755895615	10.0749871731	5.43715947866
211	0	0	0	0.000448942184448	0.000638008117676	0.000466861754
212	0	0	0	0.000384092330933	0.000562906265259	0.000394515911
213	0	0	0	0.000432968139648	0.000732183456421	0.000459456487
214	0	0	0	0.000296115875244	0.000640153884888	0.003137469169
215	0	0	0	0.000288009643555	0.000486850738525	0.000303782562
216	0	0	0	0.000452041625977	0.000653982162476	0.000477779033
217	0	0	0	0.000366926193237	0.000583171844482	0.000383932585

Inst. Num.	Best Res.	Worst Result	Aver. Result	Best Time	Worst Time	Average Time
218	0	0	0	0.00037693977356	0.000611066818237	0.00088324260
219	0	0	0	0.00042200088501	0.000779867172241	0.00042335605
220	0	0	0	0.000442028045654	0.00066089630127	0.000465190410
221	0	0	0	0.00046706199646	0.000656843185425	0.000483129024
222	0	0	0	0.000416994094849	0.000630140304565	0.000443911554
223	0	0	0	0.000279903411865	0.000483989715576	0.00029084444
224	0	0	0	0.00039005279541	0.000660181045532	0.00041161060
225	0	0	0	0.000409841537476	0.00062894821167	0.0004293632532
226	0	0	0	0.0124840736389	3.20656490326	0.6694785308
227	120	123	120	4.04434680939	6.8920340538	4.972030289
228	0	0	0	0.000430107116699	0.000701904296875	0.000455148262
229	0	0	0	0.000352144241333	0.000653982162476	0.000371928227
230	0	0	0	0.000380039215088	0.000588893890381	0.000401029592
231	0	0	0	0.000331163406372	0.00053596496582	0.00035168170
232	0	0	0	0.000426054000854	0.000616073608398	0.000449459552
233	0	4	0	0.00119400024414	10.1328239441	1.38417696
234	120	127	120	3.15065908432	10.929017067	6.1799449
235	0	110	4	0.0853729248047	6.09051799774	2.72923197
236	0	0	0	0.00035285949707	0.000560998916626	0.000369839664
237	0	1	0	0.0176749229431	3.2122631073	0.587268846
238	0	0	0	0.000391006469727	0.000580072402954	0.000419476257
239	0	0	0	0.000272035598755	0.000478029251099	0.000291702345
240	0	0	0	0.000401973724365	0.000715970993042	0.000430526398
241	0	9	0	0.0392501354218	6.71106386185	2.037558246
242	0	111	6	0.11052107811	9.63923311234	3.100390227
243	0	0	0	0.000296831130981	0.000507116317749	0.00030683517
244	0	0	0	0.000401973724365	0.000581026077271	0.000424809452
245	0	5	0	0.0223660469055	8.08109688759	1.71881270
246	0	0	0	0.000293970108032	0.000638008117676	0.000938841248
247	0	0	0	0.000383853912354	0.0028281211853	0.0004327911835
248	0	0	0	0.000401020050049	0.000669002532959	0.000433235157
249	0	2	0	0.0344820022583	7.01573610306	1.281712017
250	0	0	0	0.000433921813965	0.000646114349365	0.00046627521
251	0	0	0	0.000445127487183	0.000817060470581	0.000470371246
252	1	115	58	2.62476110458	8.18746614456	3.634564857
253	0	0	0	0.000281095504761	0.00042986869812	0.00028838631
254	0	1	0	0.00257897377014	3.69050002098	0.423732144
255	0	0	0	0.000372886657715	0.000576972961426	0.000401537415
256	1	111	13	2.46162605286	6.08390307426	3.48289408
257	0	0	0	0.000396966934204	0.000607013702393	0.000418581965
258	0	0	0	0.000380039215088	0.000565052032471	0.000395328996
259	0	0	0	0.000351190567017	0.000556945800781	0.000370454788

Inst. Num.	Best Res.	Worst Result	Aver. Result	Best Time	Worst Time	Average Time
260	0	110	3	0.0699918270111	6.06930398941	2.58782799
261	0	0	0	0.000410079956055	0.000656127929688	0.000428078171
262	0	0	0	0.000301837921143	0.000480890274048	0.000317771434
263	0	0	0	0.000408887863159	0.000614881515503	0.000431628227
264	0	8	0	0.0180420875549	6.51721906662	1.94609815
265	0	4	0	0.0659079551697	9.14377593994	2.376274111
266	0	0	0	0.000363826751709	0.000571966171265	0.00038369178
267	0	0	0	0.000395059585571	0.000610113143921	0.000419373512
268	0	0	0	0.00042200088501	0.000746965408325	0.00044448137
269	0	0	0	0.000363826751709	0.000566959381104	0.000387051105
270	0	0	0	0.000324964523315	0.000523090362549	0.000339181423
271	0	0	0	0.000357866287231	0.000494003295898	0.000364112854
272	0	3	0	0.0563368797302	6.7817800045	1.565188319
273	0	0	0	0.000396013259888	0.000597953796387	0.000416519641
274	0	113	11	0.252691030502	8.17179512978	2.861914
275	0	2	0	0.0393319129944	6.95368599892	1.58158450
276	0	0	0	0.000494956970215	0.000739812850952	0.000515167713
277	0	0	0	0.000301837921143	0.000475883483887	0.000310332775
278	0	0	0	0.000297069549561	0.000485181808472	0.000312857627
279	0	0	0	0.000324010848999	0.000516176223755	0.000344576835
280	0	0	0	0.000374794006348	0.000582933425903	0.000396881103
281	0	0	0	0.000365972518921	0.000547170639038	0.000380022525
282	0	7	0	0.0562169551849	17.1685659885	2.169108755
283	0	0	0	0.000438928604126	0.00070595741272	0.000463678836
284	0	11	0	0.0336389541626	6.53076696396	1.98119160
285	0	0	0	0.000297069549561	0.000497102737427	0.000314035415
286	0	118	41	0.0665571689606	5.67559504509	2.91159420
287	0	1	0	0.0166010856628	4.38088488579	0.522362599
288	0	0	0	0.000389099121094	0.000701189041138	0.000408809185
289	0	115	6	0.0697720050812	15.9351890087	2.99796425
290	0	13	2	0.00180697441101	9.28744387627	2.88387381
291	16	124	112	2.63871502876	5.87258815765	3.745874097
292	0	0	0	0.000367879867554	0.00057315826416	0.000386664867
293	0	0	0	0.000309944152832	0.000483989715576	0.000321216583
294	0	0	0	0.000308036804199	0.0020010471344	0.000342552661
295	0	0	0	0.000271081924438	0.000470876693726	0.00028683423
296	0	114	15	0.110318899155	5.73281598091	2.817840456
297	0	0	0	0.00049901008606	0.000731945037842	0.000527341365
298	0	0	0	0.00040602684021	0.000592947006226	0.000423188209
299	0	0	0	0.000353097915649	0.000649929046631	0.000375847816
300	0	0	0	0.00043511390686	0.00066614151001	0.0004560232162
301	0	0	0	0.000362873077393	0.000555038452148	0.000377180576

Inst. Num.	Best Res.	Worst Result	Aver. Result	Best Time	Worst Time	Average Time
302	0	0	0	0.000393867492676	0.000602960586548	0.000412046909
303	0	0	0	0.000297784805298	0.000518083572388	0.000314092636
304	0	120	87	0.18839097023	8.51165819168	3.55806590
305	0	0	0	0.000415086746216	0.000694036483765	0.000442736148
306	0	0	0	0.000380039215088	0.000568866729736	0.000392904281
307	0	0	0	0.000442981719971	0.000658988952637	0.00045615196
308	0	0	0	0.000423908233643	0.000647068023682	0.000448231697
309	0	1	0	0.0290620326996	5.92952299118	0.829476675
310	0	0	0	0.00032901763916	0.000538110733032	0.000342116355
311	0	0	0	0.00025200843811	0.000417947769165	0.000259807109
312	0	0	0	0.000375986099243	0.000586986541748	0.000394420623
313	0	112	25	0.123091936111	9.32272696495	3.886912352
314	0	0	0	0.000365018844604	0.000576972961426	0.000382741547
315	0	17	3	0.168774843216	7.2411301136	2.603286635
316	0	0	0	0.000308036804199	0.00052285194397	0.00032910585
317	0	0	0	0.000351905822754	0.000716924667358	0.000366342067
318	0	0	0	0.000283002853394	0.000458002090454	0.000296316146
319	0	0	0	0.000391006469727	0.000604867935181	0.000415062904
320	0	0	0	0.000418901443481	0.000633001327515	0.000443031787
321	0	0	0	0.000397920608521	0.000613927841187	0.000414893627
322	0	5	0	0.0396928787231	9.89019584656	2.178125188
323	0	0	0	0.000315189361572	0.000527143478394	0.000324940681
324	0	0	0	0.00041389465332	0.000625848770142	0.000432124137
325	0	0	0	0.000427961349487	0.000637054443359	0.000452361106
326	0	0	0	0.000367879867554	0.000590085983276	0.000397963523
327	0	0	0	0.000304937362671	0.000503063201904	0.000327618122
328	0	17	1	0.0470139980316	9.44801282883	3.056050889
329	0	0	0	0.000293970108032	0.000497817993164	0.000319292545
330	0	3	0	0.00414204597473	4.04531598091	0.947127742
331	0	0	0	0.000401973724365	0.00061297416687	0.000419898033
332	7	124	80	2.56359410286	6.44858598709	3.71826069
333	0	0	0	0.00036883354187	0.000732898712158	0.000387420654
334	0	0	0	0.000238180160522	0.000402927398682	0.000248231887
335	0	0	0	0.000397920608521	0.000631093978882	0.000428411960
336	0	0	0	0.000358104705811	0.00050687789917	0.00036657333
337	0	0	0	0.000411033630371	0.000596046447754	0.000438144207
338	0	5	0	0.0370578765869	6.46820187569	1.13281336
339	120	125	120	3.04546403885	9.0378639698	6.08122876
340	0	0	0	0.000431060791016	0.000704050064087	0.000451033115
341	0	0	0	0.00038480758667	0.000602960586548	0.0004026222
342	0	0	0	0.000405788421631	0.000766038894653	0.000428848266
343	0	0	0	0.000287055969238	0.000593900680542	0.000308933258

Inst. Num.	Best Res.	Worst Result	Aver. Result	Best Time	Worst Time	Average Time
344	0	0	0	0.000481843948364	0.000843048095703	0.00050811290
345	0	0	0	0.000380039215088	0.000567197799683	0.000400383472
346	0	17	1	0.0263879299164	8.8095369339	2.35147966146
347	120	131	121	2.43421912193	6.10506105423	3.41659301
348	0	0	0	0.000409841537476	0.000728130340576	0.000435597896
349	0	0	0	0.000378847122192	0.000599145889282	0.000397624969
350	0	0	0	0.000283002853394	0.000576972961426	0.000295920372
351	0	0	0	0.00029993057251	0.00049901008606	0.000316090583
352	0	0	0	0.000396013259888	0.000607013702393	0.000412585735
353	0	4	0	0.0387759208679	11.0017981529	2.064029009
354	0	0	0	0.000424861907959	0.000802993774414	0.000441505908
355	0	0	0	0.0003981590271	0.000664949417114	0.000414795875
356	0	0	0	0.000402927398682	0.000728845596313	0.000420215129
357	0	0	0	0.000424861907959	0.00075888633728	0.000454678535
358	0	0	0	0.000296115875244	0.000499963760376	0.000316250324
359	0	0	0	0.000366926193237	0.00277709960938	0.000406422615
360	0	17	2	0.077467918396	5.8242418766	2.48066154
361	0	0	0	0.000372886657715	0.00056004524231	0.000391728878
362	0	16	0	0.0230600833893	6.27112698555	1.618376467
363	0	0	0	0.000416040420532	0.000638961791992	0.000437037944
364	0	12	0	0.0338928699493	9.40454602242	2.728610422
365	0	1	0	0.0314700603485	5.60331702232	0.7609343004
366	0	0	0	0.000338077545166	0.000561952590942	0.000350811481
367	0	0	0	0.000294923782349	0.000496864318848	0.000308389663
368	0	0	0	0.00040078163147	0.000607967376709	0.000420076847
369	0	0	0	0.000353097915649	0.000565052032471	0.000369112491
370	0	0	0	0.000372886657715	0.000640869140625	0.000388748645
371	0	17	2	0.133727073669	16.209487915	3.13109951
372	0	115	3	0.0827069282532	5.7597720623	2.7938662
373	0	0	0	0.000357151031494	0.000566959381104	0.0003760985
374	0	0	0	0.000308036804199	0.00049090385437	0.000316517353
375	0	0	0	0.000264883041382	0.000519037246704	0.000280046413
376	0	1	0	0.0496470928192	6.2929558754	1.075599896
377	0	0	0	0.000389099121094	0.000606060028076	0.000411188608
378	0	13	0	0.000875949859619	4.16318583488	1.69013925552
379	0	0	0	0.000463962554932	0.000696897506714	0.000488600896
380	0	118	48	2.93175601959	28.8192369938	5.253036284
381	0	0	0	0.000298023223877	0.000572919845581	0.000320315323
382	0	2	0	0.0447850227356	6.96436095238	1.11762479
383	0	0	0	0.000290870666504	0.000499963760376	0.000301990509
384	0	0	0	0.000398874282837	0.000760078430176	0.000421857833
385	0	0	0	0.00043511390686	0.000645160675049	0.000450749397

Inst. Num.	Best Res.	Worst Result	Aver. Result	Best Time	Worst Time	Average Time
386	0	0	0	0.000406980514526	0.000689029693604	0.000430171489
387	0	2	0	0.000741004943848	8.53111314774	1.8070745
388	0	110	9	0.225622177124	9.87241220474	3.77052291
389	0	0	0	0.000358819961548	0.000568151473999	0.00037376858
390	0	0	0	0.00034499168396	0.000547170639038	0.00036301614
391	0	0	0	0.000420093536377	0.000725984573364	0.000452449747
392	0	120	48	0.267019987106	9.203592062	4.1313092
393	121	133	127	2.77009105682	14.2058699131	4.62148711
394	0	0	0	0.000447034835815	0.000698089599609	0.00044984741
395	0	120	96	0.899717092514	8.45113682747	3.93631808
396	0	0	0	0.000426054000854	0.000743865966797	0.0004468321
397	0	1	0	0.0309200286865	5.11185193062	1.05038484
398	0	0	0	0.000403881072998	0.000613212585449	0.00041850324
399	0	0	0	0.000314950942993	0.000494003295898	0.00032679799
400	0	0	0	0.000320911407471	0.00169110298157	0.00036090135

B.2.2 Method 2: Basic Population Evolution

Full results of the large amount of testing carried out on the final system with 400 instances being used with 100 applications of each for the basic population evolution method.

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
1	0	0	0	14.030629158	14.4216599464	14.2124638081
2	0	0	0	8.68752193451	13.543804884	10.9318365812
3	0	0	0	13.9767830372	14.6799938679	14.4569061756
4	0	0	0	11.5416448116	11.8307499886	11.6783052444
5	0	0	0	12.1240420341	12.5278699398	12.370746398
6	0	0	0	11.4897589684	12.0748329163	11.8032512903
7	0	0	0	12.302562952	12.9080178738	12.5870318651
8	110	110	110	9.44590592384	10.705960989	10.0261810303
9	0	0	0	16.8279058933	17.5780258179	17.0710722446
10	0	0	0	12.2731359005	13.1228151321	12.657212472
11	0	0	0	9.15906405449	15.0141499043	12.9384136915
12	0	0	0	8.31127500534	10.556071043	8.63818340302
13	0	0	0	16.2517409325	16.676887989	16.482494998
14	0	0	0	12.986207962	13.3269860744	13.1386026144
15	0	0	0	9.48957896233	9.68177080154	9.60897705555
16	0	0	0	8.34067416191	8.55930995941	8.42019279003
17	0	0	0	14.1865608692	14.6796469688	14.3876071453

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
18	110	110	110	10.3150849342	15.439677	12.3554690838
19	0	0	0	8.79004406929	10.4193940163	9.47929711342
20	0	0	0	14.2564451694	14.8281559944	14.5787715197
21	0	0	0	15.6898920536	16.1250879765	15.8175359011
22	0	0	0	10.7497119904	12.0567619801	11.3322118044
23	0	0	0	8.07642006874	8.25925397873	8.18675351143
24	0	0	0	11.2433440685	11.6181311607	11.3864315271
25	0	0	0	9.66508388519	9.97924685478	9.82639427185
26	0	0	0	18.8572978973	19.6412258148	19.2320447445
27	0	0	0	12.3815360069	12.7666511536	12.5201146364
28	0	0	0	15.466837883	16.0912489891	15.7585091114
29	0	0	0	12.2571718693	12.9005408287	12.4955423117
30	0	0	0	9.20995903015	9.48963403702	9.32360818386
31	0	0	0	9.02666711807	9.29762101173	9.17569093704
32	0	0	0	10.890709877	11.2329549789	10.996707058
33	0	0	0	17.7244451046	18.5013229847	18.048867321
34	0	0	0	15.5238380432	16.3036510944	15.7468993902
35	0	0	0	10.0125348568	11.375524044	10.7223068237
36	0	0	0	10.5616538525	13.615046978	12.0598263741
37	0	0	0	18.2661068439	19.3150048256	18.8077957869
38	0	0	0	13.7565889359	14.6145939827	14.1390902042
39	0	0	0	9.62530589104	10.2516231537	9.89182271957
40	0	0	0	8.11382317543	8.46134090424	8.24401283264
41	0	0	0	11.6206338406	16.8852910995	13.7184675217
42	0	0	0	14.3854908943	15.0266718864	14.6883607626
43	0	0	0	7.72148394585	8.03304791451	7.88327586651
44	0	0	0	12.9791038036	13.8558869362	13.2757093668
45	0	0	0	16.280217886	16.8201489449	16.5780512571
46	0	0	0	12.316423893	13.0684130192	12.5597120047
47	0	0	0	9.34334683418	9.7070608139	9.48906054497
48	0	0	0	13.8234272003	14.4196429253	14.1654099941
49	0	0	0	13.8224580288	16.895512104	15.0745512247
50	0	0	0	12.5386190414	13.3007199764	12.9233713865
51	0	0	0	11.7557799816	12.0748779774	11.9330757856
52	0	0	0	15.7624738216	16.3629209995	16.0489760876
53	110	110	110	11.4779219627	13.2056968212	12.6504248619
54	0	0	0	13.531867981	14.0235390663	13.7463816643
55	0	0	0	14.0229539871	14.5837731361	14.3223958731
56	0	0	0	7.41489291191	7.52694106102	7.462525177
57	0	0	0	12.3271751404	12.7235910892	12.4727300644
58	0	0	0	19.4238638878	19.8979380131	19.6830458641
59	0	0	0	13.8079519272	14.3241209984	14.0841724157

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
60	0	110	44	8.74314713478	15.7637209892	12.0323325872
61	0	0	0	8.68993210793	8.97144198418	8.78943793774
62	0	0	0	15.0213410854	15.4107398987	15.2381629944
63	0	0	0	9.50699281693	9.69016695023	9.59112696648
64	0	0	0	9.15720391273	9.47913098335	9.28346238136
65	0	0	0	12.2004611492	12.5622019768	12.37810781
66	0	0	0	13.1235408783	13.7079792023	13.5019618511
67	0	0	0	15.0897238255	15.6828980446	15.3684955835
68	0	0	0	14.4708199501	14.6453142166	14.5576378107
69	0	0	0	12.0211799145	17.2955698967	15.939307642
70	0	0	0	13.2443399429	13.5478971004	13.3300920963
71	0	0	0	15.6614370346	16.1605279446	15.9609905243
72	0	0	0	11.423707962	11.9031610489	11.6523471117
73	0	0	0	14.0868220329	14.4055309296	14.2607854366
74	0	0	0	10.8276538849	11.4855029583	11.086542201
75	0	0	0	13.226339817	15.1461379528	14.2216372013
76	0	0	0	8.93243098259	9.45199203491	9.09347422123
77	0	110	11	8.92879009247	14.5535919666	10.3434803963
78	0	0	0	13.1121430397	13.594602108	13.3590569496
79	0	0	0	9.64503216743	10.0919418335	9.97765479088
80	0	0	0	9.57664585114	9.76268076897	9.64703435898
81	0	0	0	15.0113170147	15.6291561127	15.3338881731
82	0	0	0	14.0715990067	14.7026641369	14.379659605
83	0	0	0	13.9656841755	14.391505003	14.2528054953
84	0	0	0	8.69665288925	8.95509719849	8.85753741264
85	0	0	0	8.58389687538	8.92794179916	8.72072784901
86	0	0	0	9.06351494789	9.30803108215	9.17912709713
87	0	0	0	13.877104044	14.2257020473	14.1390922785
88	0	0	0	6.47310900688	6.584004879	6.52858181
89	0	0	0	13.3705611229	13.7412409782	13.5645014763
90	0	0	0	11.3550000191	11.816806078	11.5196293116
91	110	110	110	7.64423394203	7.65270900726	7.64763536453
92	0	0	0	15.9717290401	16.6079821587	16.3013782501
93	0	0	0	8.95594906807	13.2801151276	10.076102829
94	0	0	0	8.95206594467	16.1095991135	13.8595967054
95	0	0	0	14.5210499763	15.2461280823	15.0209049702
96	0	0	0	11.9374759197	12.4564127922	12.0880395412
97	0	0	0	13.9288089275	14.2335679531	14.0718086481
98	0	0	0	15.0269520283	15.4230079651	15.2639744282
99	0	0	0	12.891272068	13.7299048901	13.3623599768
100	0	0	0	16.4689028263	17.2881000042	16.9510671616
101	0	0	0	11.4784109592	11.9020271301	11.6724989891

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
102	0	0	0	13.1685729027	13.5717959404	13.3836406946
103	0	0	0	8.23661804199	8.3770878315	8.3226149559
104	0	0	0	10.017813921	10.2382228374	10.1456351519
105	0	0	0	14.6662108898	15.5886580944	15.0749562979
106	0	0	0	14.4373371601	14.8918128014	14.5993003607
107	0	0	0	14.9315719604	15.699434042	15.3682990313
108	0	0	0	13.0422480106	14.893652916	14.0208758354
109	0	0	0	12.9664111137	14.2132999897	13.6209135294
110	0	0	0	8.03184199333	8.24424290657	8.1374982357
111	0	0	0	14.0401070118	14.6912071705	14.3862167358
112	0	0	0	12.3640451431	12.6587150097	12.522907114
113	0	0	0	17.3774278164	17.8266429901	17.6248727798
114	0	0	0	14.9184250832	15.4131300449	15.2491390467
115	0	0	0	17.8773329258	18.391479969	18.1059161186
116	0	0	0	9.35130405426	9.63835597038	9.45768101215
117	0	0	0	11.9366829395	12.5727441311	12.1820529222
118	0	0	0	12.3651099205	12.6988739967	12.5714313984
119	0	0	0	8.68393397331	8.80823802948	8.73069477081
120	0	0	0	12.288310051	12.5663568974	12.3985011816
121	0	0	0	9.50616192818	9.77123785019	9.66072583199
122	0	0	0	12.0661509037	12.5044260025	12.2326000929
123	0	0	0	15.832201004	16.3959479332	16.1603251934
124	0	0	0	14.295083046	14.6794581413	14.5354435444
125	0	0	0	9.05554485321	9.27233886719	9.17938432693
126	0	0	0	12.2759521008	12.6748130322	12.4566010475
127	0	0	0	9.18979001045	9.47880005836	9.33783869743
128	0	0	0	10.3934288025	10.5775120258	10.4822574854
129	0	0	0	15.6730480194	16.0869121552	15.8683982372
130	0	0	0	14.1570470333	14.5705761909	14.3538256168
131	0	0	0	17.3679029942	18.0660159588	17.7732966661
132	0	0	0	13.8701570034	14.5763399601	14.0848338127
133	0	0	0	17.8404171467	18.3281629086	18.1409883022
134	0	0	0	12.2883579731	12.7778658867	12.5972929955
135	0	0	0	12.4251589775	12.9165680408	12.7725175142
136	0	0	0	9.05688405037	9.29127597809	9.22782049179
137	0	0	0	15.0674898624	15.5776269436	15.350166297
138	0	0	0	13.9317009449	14.1867220402	14.1084138393
139	0	0	0	11.7414329052	13.9174878597	12.537435627
140	0	0	0	14.1202611923	14.613920927	14.2875530481
141	0	0	0	15.4303190708	15.8631289005	15.6137943983
142	0	0	0	11.0452771187	11.7813198566	11.3512776613
143	0	0	0	9.7198741436	10.0611829758	9.93059973717

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
144	0	0	0	13.4870779514	13.7692329884	13.5980178833
145	0	110	22	8.65139722824	14.007586956	10.6773985863
146	0	0	0	15.6057128906	17.4554760456	16.3214024067
147	0	0	0	15.3173310757	15.8750510216	15.6569056988
148	0	0	0	15.4796700478	15.9977889061	15.6986217976
149	0	0	0	12.1442968845	12.7510719299	12.5119447231
150	0	0	0	12.6471960545	12.9550409317	12.7915615559
151	0	0	0	9.33339810371	9.57042217255	9.45166714191
152	0	0	0	15.9646449089	16.1577279568	16.0599740982
153	0	0	0	14.6792840958	16.1903171539	15.3225470304
154	0	0	0	15.3800477982	15.876363039	15.6332152367
155	0	0	0	8.98683404922	9.23487091064	9.13531413078
156	0	0	0	12.6916251183	13.1502511501	12.8953145742
157	0	0	0	14.3938698769	14.8839039803	14.6642579794
158	0	0	0	12.0486228466	12.3970620632	12.2401875496
159	0	0	0	12.4157369137	13.0435550213	12.6827623367
160	0	0	0	7.14994311333	7.39993214607	7.30877475739
161	110	110	110	11.3335258961	11.7233700752	11.5405329943
162	0	110	33	9.00719690323	15.0723979473	11.4994078159
163	0	0	0	14.2776119709	15.1904020309	14.7066645384
164	0	0	0	12.1117341518	12.4980881214	12.2595234394
165	0	0	0	14.9110569954	15.2113659382	15.085430336
166	0	0	0	15.465749979	15.8777940273	15.6930107117
167	0	0	0	8.2794251442	8.34423112869	8.31027417183
168	0	0	0	9.37956690788	9.57397603989	9.48650281429
169	0	0	0	15.1994121075	15.7106199265	15.3838535547
170	0	0	0	15.4413330555	16.7615590096	16.2540694237
171	0	0	0	8.68803620338	9.46192097664	9.05055458546
172	0	0	0	14.8952560425	15.3301341534	15.0037643909
173	0	0	0	14.0557329655	14.7217059135	14.3382217646
174	0	0	0	11.4091320038	11.7065820694	11.5521026611
175	0	0	0	9.25760602951	9.48103809357	9.36101276875
176	0	0	0	14.571395874	15.1341280937	14.859291482
177	0	0	0	12.6847989559	13.0875189304	12.8294245958
178	0	0	0	10.9997389317	11.6116089821	11.3587395906
179	0	0	0	14.8170881271	16.4804880619	15.8682341099
180	0	0	0	16.4976260662	17.5100250244	16.9804667711
181	0	0	0	14.3701930046	16.2901289463	15.3514087677
182	0	0	0	12.4598441124	12.8079738617	12.6425858974
183	0	0	0	10.864084959	11.233880043	11.0341265917
184	0	0	0	8.86241388321	9.00895690918	8.9539637804
185	0	0	0	13.137914896	14.0487911701	13.5815652132

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
186	130	130	130	8.26712179184	8.2731590271	8.27006947994
187	0	110	11	8.75600099564	16.6203770638	10.140224123
188	0	0	0	13.3775670528	15.6211719513	14.3173770189
189	0	0	0	13.674380064	14.075097084	13.9040951967
190	0	0	0	15.6918890476	16.1269431114	15.8998486757
191	0	0	0	7.91910791397	8.09880208969	7.98883476257
192	0	0	0	9.08547210693	9.30744695663	9.19946990013
193	0	0	0	14.3863759041	14.7562367916	14.5818876743
194	0	0	0	15.5461421013	16.3976910114	15.8606206656
195	0	0	0	12.9539730549	13.4159669876	13.2572651863
196	0	0	0	12.5053889751	13.1997420788	12.9610645533
197	0	0	0	8.78119897842	14.7353508472	11.5382413149
198	0	0	0	13.3450829983	13.6460959911	13.4923054218
199	0	0	0	14.4649960995	15.3143661022	14.8802974224
200	0	0	0	8.78276491165	9.05588793755	8.94970786572
201	0	0	0	8.4234521389	8.56811404228	8.49658772945
202	0	0	0	13.4900968075	16.4504039288	15.5389746428
203	0	0	0	15.642318964	16.9512631893	16.4241503239
204	0	0	0	15.3770840168	16.1975271702	15.8696782351
205	0	0	0	18.6350190639	19.0452029705	18.8596811056
206	0	0	0	7.91729593277	8.0385890007	7.97923998833
207	0	0	0	10.3493609428	10.7682340145	10.6134018898
208	0	0	0	13.7718789577	14.084485054	13.8887638569
209	0	0	0	11.0641000271	11.6640770435	11.4381683588
210	0	110	11	8.68187403679	16.2571558952	10.0357796907
211	13	110	100	9.50534200668	12.1282749176	10.3876820326
212	0	0	0	15.0432360172	15.459679842	15.2024589777
213	0	0	0	13.5598130226	14.194051981	13.9483033895
214	0	0	0	15.475538969	15.9694240093	15.7396854639
215	0	0	0	9.13074994087	9.35250115395	9.2176784277
216	0	0	0	9.34645986557	9.61791014671	9.47536978722
217	0	0	0	16.8930358887	17.3204028606	17.101235342
218	0	0	0	11.778523922	12.0268111229	11.9043124676
219	0	0	0	11.0129370689	11.4968378544	11.27920928
220	0	0	0	15.3462030888	15.7074718475	15.5274889231
221	0	0	0	16.1672079563	16.8533260822	16.4847648144
222	0	0	0	16.4064440727	16.9287910461	16.5630333424
223	0	0	0	14.7826941013	15.0891442299	14.9935480118
224	0	0	0	8.37931585312	8.63808512688	8.4803170681
225	0	0	0	12.6118330956	13.0794789791	12.9010899067
226	0	0	0	12.9359228611	13.771643877	13.3174055815
227	0	0	0	11.0728030205	11.7378001213	11.3774867773

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
228	0	0	0	17.8813779354	18.4382469654	18.1693605185
229	0	0	0	14.2964560986	14.998816967	14.6257017136
230	0	0	0	11.5181000233	11.8236510754	11.6568199635
231	0	0	0	12.6715219021	13.035476923	12.8978234529
232	0	0	0	10.518832922	10.7719831467	10.6057362318
233	0	0	0	14.2726700306	14.8235981464	14.5515042305
234	0	0	0	15.8977768421	17.2211921215	16.6494238615
235	0	0	0	18.036465168	18.7631280422	18.3293639421
236	0	0	0	8.67455816269	11.8851840496	9.71349451542
237	0	0	0	11.6852030754	11.8588371277	11.7646560192
238	0	0	0	11.8547489643	12.2388761044	12.0608263254
239	0	0	0	11.9927239418	12.4143190384	12.2248736858
240	0	0	0	8.16807198524	8.37058496475	8.2664331913
241	0	0	0	13.9349999428	14.5532839298	14.2875968933
242	0	0	0	15.1970970631	15.9250249863	15.6124324083
243	0	0	0	8.7857708931	11.8948569298	9.83528959751
244	0	0	0	9.60019087791	9.74726986885	9.66489236355
245	0	0	0	13.0367529392	13.5454359055	13.2502048731
246	0	0	0	13.1428639889	14.1037092209	13.7181153774
247	0	0	0	9.30610609055	9.59748005867	9.39191334248
248	0	0	0	14.6583490372	15.0026187897	14.8575559139
249	0	0	0	13.4339520931	13.6245639324	13.5442928314
250	0	0	0	17.1678140163	17.6849381924	17.469247508
251	0	0	0	14.5539360046	15.3099758625	14.9756747723
252	0	0	0	14.6935579777	15.408285141	15.0410300493
253	110	110	110	8.99503707886	13.3546090126	10.6382198811
254	0	0	0	9.01508903503	9.30891609192	9.16990549564
255	0	0	0	14.3490030766	14.739948988	14.5219923973
256	0	0	0	12.2258071899	12.6227359772	12.467617631
257	130	130	130	8.2682158947	8.27613377571	8.27290878296
258	0	0	0	14.6858489513	15.1466920376	14.9331856489
259	0	0	0	13.1651899815	13.41700387	13.2929197788
260	0	0	0	11.4362089634	11.8591990471	11.6181064129
261	0	0	0	9.15155315399	14.2813410759	12.3371331215
262	0	0	0	12.8376309872	13.3490550518	13.1652738094
263	0	0	0	9.61113405228	9.94661998749	9.79722168446
264	0	0	0	13.2630870342	13.7718069553	13.4938366175
265	0	0	0	9.40993118286	11.75750494	10.8239661217
266	0	0	0	18.792899847	19.8762040138	19.3394528389
267	0	0	0	13.1181190014	13.5469810963	13.2448047161
268	0	0	0	13.5813391209	13.9218649864	13.7889617205
269	0	0	0	14.7723200321	15.5514659882	15.0982162952

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
270	0	0	0	13.5941829681	13.9518749714	13.7607182026
271	0	0	0	10.3838078976	10.7247700691	10.5505062819
272	0	0	0	11.1016981602	11.4371099472	11.2576108217
273	0	0	0	15.2421250343	15.5845668316	15.4476844788
274	0	0	0	14.0220899582	14.868639946	14.4789270878
275	0	0	0	8.72564506531	14.6551599503	11.6414064646
276	0	0	0	15.6551311016	16.6553490162	16.2377561569
277	0	0	0	17.1492421627	17.9506480694	17.3887416363
278	0	0	0	9.98052597046	10.1039330959	10.0316255331
279	0	0	0	9.47405004501	9.70446014404	9.55890603065
280	0	0	0	10.708796978	11.0776929855	10.966245079
281	0	0	0	13.647698164	14.0196809769	13.7777718067
282	0	0	0	12.4955010414	12.8126490116	12.6466923952
283	0	0	0	15.1427419186	16.4095230103	16.0694008589
284	0	0	0	15.5559298992	15.9915480614	15.795372963
285	0	0	0	9.99984812737	11.8185458183	10.8025110483
286	0	0	0	9.09460020065	9.36821198463	9.22269461155
287	0	110	33	8.8480451107	16.9226009846	12.4892864943
288	0	0	0	11.8053789139	12.5272469521	12.2667206287
289	0	0	0	13.808216095	14.2076981068	14.0335824251
290	0	0	0	8.84698104858	17.5976760387	13.8566097975
291	0	0	0	13.1241109371	15.3800749779	14.1099755287
292	120	120	120	11.838449955	15.1674451828	13.5253822088
293	0	0	0	12.3014581203	12.7880408764	12.549630785
294	0	0	0	9.69093513489	9.93288397789	9.84293141365
295	0	0	0	9.6078338623	9.7872428894	9.70609550476
296	0	0	0	8.60967803001	8.99782705307	8.69978542328
297	0	0	0	8.9459810257	16.8809919357	11.1649678469
298	0	0	0	16.4575619698	17.0469448566	16.8125922203
299	0	0	0	14.1587028503	14.4069349766	14.2858355045
300	0	0	0	12.1780660152	13.0054788589	12.5358244658
301	0	0	0	13.3126440048	14.0219681263	13.6538824797
302	0	0	0	12.0085828304	12.3974850178	12.2356664419
303	0	0	0	12.7105250359	13.1257669926	12.9538663149
304	0	0	0	9.05174803734	9.3364648819	9.23786816597
305	0	110	88	9.61018204689	17.5586800575	14.734325099
306	0	0	0	15.2065169811	15.6955761909	15.4019831419
307	0	0	0	12.3290610313	12.845389843	12.6070286989
308	0	0	0	15.6062381268	16.199878931	15.8445985556
309	0	0	0	13.6789691448	14.401350975	14.0224227905
310	0	0	0	14.5322070122	14.9409451485	14.7168886185
311	0	0	0	10.8277769089	11.0858061314	10.9497610807

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
312	0	0	0	7.56781196594	8.53463697433	7.85420424938
313	0	0	0	12.9511759281	13.3916380405	13.1518942118
314	0	110	88	9.39128494263	17.9356491566	14.8446029902
315	0	0	0	11.4909911156	11.8011660576	11.6149060488
316	0	0	0	13.1897888184	17.747492075	15.8188625574
317	0	0	0	9.53400111198	10.0462729931	9.83519742489
318	0	0	0	11.2255101204	11.5925087929	11.4235198498
319	0	0	0	8.83485007286	9.07296991348	8.99425415993
320	0	0	0	12.035574913	12.5739760399	12.291433692
321	0	0	0	14.8344957829	15.5686650276	15.0441023588
322	0	0	0	14.1100821495	14.6232140064	14.3649730206
323	0	0	0	14.1386499405	15.2740471363	14.8575890064
324	0	0	0	8.53199386597	8.83726882935	8.71508367062
325	0	0	0	10.2964010239	11.1563827991	10.6592033386
326	0	0	0	13.4661591053	14.7116868496	13.8762130499
327	0	0	0	13.5521600246	14.0024869442	13.8212526321
328	0	0	0	9.09271216393	9.59346318245	9.22396438122
329	0	0	0	13.9410099983	17.6298120022	16.4676264048
330	0	0	0	8.1243288517	8.35877895355	8.24485290051
331	0	0	0	14.2770922184	14.9316751957	14.6381337643
332	0	0	0	13.3593530655	13.9388170242	13.5772735834
333	121	130	127	9.7801630497	12.7400510311	11.4218926907
334	0	0	0	11.4737741947	11.6858069897	11.589875412
335	0	0	0	7.2463350296	7.41209197044	7.34197444916
336	0	0	0	12.5081298351	12.8004179001	12.6654434681
337	0	0	0	11.4224758148	11.6459958553	11.5427343369
338	0	0	0	12.9449641705	13.9097089767	13.4568031311
339	0	0	0	16.1303591728	16.5616879463	16.3173149347
340	0	0	0	17.4974889755	18.9554688931	18.2241002083
341	0	0	0	16.9852941036	17.3622908592	17.1781461716
342	0	0	0	13.1050460339	13.5684459209	13.3002448797
343	0	0	0	13.5976319313	14.023460865	13.7999388695
344	0	0	0	9.2438659668	9.49990296364	9.37382237911
345	0	0	0	17.7313940525	18.4720461369	18.0370947599
346	0	0	0	11.5814261436	11.892829895	11.7290416718
347	0	0	0	12.0000391006	13.5691621304	12.8709734201
348	110	110	110	14.3199419975	15.0585310459	14.6271434307
349	0	0	0	14.0072610378	14.575054884	14.3497641563
350	0	0	0	12.782282114	13.1111931801	12.9821629763
351	0	0	0	8.75582814217	9.03773498535	8.84385020733
352	0	0	0	9.7400598526	10.2040250301	9.87472791672
353	0	0	0	13.6950871944	14.5147399902	14.0636461496

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
354	0	0	0	16.0068540573	17.4125440121	16.5040928602
355	0	0	0	13.5911591053	13.9583480358	13.7237333298
356	0	0	0	12.3823709488	12.7544679642	12.5262400866
357	0	0	0	15.14894104	15.5199940205	15.3709661245
358	0	0	0	12.868224144	13.6652948856	13.3528644562
359	0	0	0	9.38935613632	9.59006500244	9.48430452347
360	0	0	0	13.3094601631	13.7376439571	13.472266078
361	0	0	0	14.7448730469	16.5213708878	16.0113194942
362	0	0	0	11.4027280807	11.972725153	11.7934348583
363	0	0	0	11.3199250698	11.9620330334	11.5506464005
364	0	0	0	14.0624740124	14.4979159832	14.3163800478
365	0	0	0	16.3694629669	17.7287011147	17.0169185638
366	0	0	0	10.8945670128	11.7691719532	11.4476727009
367	0	0	0	10.7885479927	11.0100841522	10.8897591352
368	0	0	0	9.19436788559	9.45507502556	9.34086339474
369	0	0	0	14.2346701622	14.7564620972	14.5457406282
370	0	0	0	11.0074617863	11.4480631351	11.235624671
371	0	0	0	13.1270508766	13.4749000072	13.2737922907
372	0	0	0	10.5425441265	13.0344660282	11.7570909739
373	0	0	0	13.5513370037	15.8075351715	14.3124559402
374	0	0	0	12.9590220451	13.4640378952	13.2272160292
375	0	0	0	9.11114501953	9.26656293869	9.15693457127
376	0	0	0	8.02431511879	8.19123983383	8.1140460968
377	0	0	0	14.2707560062	14.8555598259	14.5422162533
378	0	0	0	12.7101840973	13.062485218	12.8699214458
379	0	0	0	14.2331690788	15.0499110222	14.6101565123
380	0	0	0	14.7685091496	15.3396570683	15.0905883074
381	110	110	110	9.40435504913	13.4279260635	11.2182431459
382	0	0	0	9.69179320335	10.1052899361	9.87237770557
383	0	0	0	16.4330539703	17.4343450069	16.742745018
384	0	0	0	9.5805721283	9.76120996475	9.67345452309
385	0	0	0	13.4055290222	13.7030460835	13.6025837898
386	0	0	0	14.6904430389	15.1721258163	14.9024167538
387	0	0	0	13.7311480045	14.4722321033	14.0564888716
388	0	0	0	15.2033851147	15.6109650135	15.4007085323
389	0	110	22	9.24633693695	17.0375709534	12.1178014517
390	0	0	0	12.377822876	13.0316359997	12.7450106859
391	0	0	0	11.2127718925	11.5895659924	11.3826948166
392	0	0	0	13.9863858223	15.550773859	14.9842743635
393	0	110	55	10.0811669827	19.1269278526	14.6994389772
394	110	110	110	16.8221189976	17.3301348686	17.0558335304
395	0	0	0	16.4989349842	17.5783760548	16.9841219187

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
396	110	110	110	9.56777787209	12.4536540508	10.3442390919
397	0	0	0	15.4444589615	16.2464659214	15.7794953585
398	0	0	0	17.8356099129	19.1981370449	18.5004611731
399	0	0	0	15.6472361088	16.1023919582	15.7639323235
400	0	0	0	10.5648808479	10.8997321129	10.7339096069

B.2.3 Method 3: Random Iteration of Constructive Heuristic

Full results of the large amount of testing carried out on the final system with 400 instances being used with 100 applications of each for the random iteration based method.

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
1	0	0	0	0.0496559143066	0.094113111496	0.052326669693
2	120	120	120	0.115566015244	0.124088048935	0.118163433075
3	0	0	0	0.0492570400238	0.0627291202545	0.0510562825203
4	110	110	110	0.085214138031	0.0946271419525	0.0872134160995
5	0	0	0	0.0370359420776	0.0634360313416	0.0389927124977
6	0	0	0	0.0471179485321	0.0509521961212	0.0478384447098
7	0	0	0	0.0391190052032	0.0436718463898	0.0395075964928
8	120	120	120	0.100091218948	0.106219053268	0.10135355711
9	0	0	0	0.0501132011414	0.0539329051971	0.0511242246628
10	0	0	0	0.0355100631714	0.0382270812988	0.0361488747597
11	130	130	130	0.154448032379	0.159623146057	0.15621491909
12	110	110	110	0.100378036499	0.104303121567	0.101402788162
13	110	110	110	0.106340885162	0.11518406868	0.108238780499
14	0	0	0	0.0415859222412	0.0457229614258	0.0422873282433
15	0	0	0	0.0304579734802	0.0328159332275	0.0309222197533
16	0	0	0	0.0261390209198	0.0273580551147	0.0264051318169
17	0	0	0	0.0483119487762	0.0518848896027	0.0488832497597
18	140	140	140	0.15608215332	0.17289686203	0.158521175385
19	110	110	110	0.106431007385	0.110671043396	0.107635097504
20	0	0	0	0.0545339584351	0.0588538646698	0.0551670265198
21	0	0	0	0.0466160774231	0.087660074234	0.0491211867332
22	120	120	120	0.103049993515	0.168305873871	0.106769268513
23	0	0	0	0.0191640853882	0.025426864624	0.019719388485
24	0	0	0	0.0340430736542	0.0385401248932	0.0347082495689
25	0	0	0	0.0260829925537	0.0318789482117	0.0267738366127
26	120	120	120	0.1087911129	0.137446880341	0.110744390488

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
27	110	110	110	0.0854699611664	0.0900928974152	0.0864362335205
28	0	0	0	0.0565021038055	0.0606851577759	0.0572878575325
29	0	0	0	0.0481400489807	0.0521838665009	0.048583714962
30	0	0	0	0.0239679813385	0.0269739627838	0.0243918275833
31	0	0	0	0.0270109176636	0.0295851230621	0.0273263692856
32	0	0	0	0.026908159256	0.0298399925232	0.0272332954407
33	0	0	0	0.0617761611938	0.0664501190186	0.0624977660179
34	0	0	0	0.0507299900055	0.0547330379486	0.0511209821701
35	0	0	0	0.0419290065765	0.0456428527832	0.0423254728317
36	140	140	140	0.144775867462	0.149916887283	0.145946135521
37	120	120	120	0.0932388305664	0.0977041721344	0.0939924883842
38	0	0	0	0.0426659584045	0.054267168045	0.0436580467224
39	0	0	0	0.0291299819946	0.0368142127991	0.0296415138245
40	0	0	0	0.0236051082611	0.0282080173492	0.0241089701653
41	120	120	120	0.101900100708	0.10804605484	0.102634489536
42	0	0	0	0.044830083847	0.0460870265961	0.0454448151588
43	0	0	0	0.0233361721039	0.0242819786072	0.0235535240173
44	0	0	0	0.0438809394836	0.0453810691833	0.0442208528519
45	120	120	120	0.15395116806	0.158123970032	0.15566539526
46	0	0	0	0.0408639907837	0.0531740188599	0.0413023495674
47	0	0	0	0.0283200740814	0.0294940471649	0.0286332941055
48	0	0	0	0.0396230220795	0.0412700176239	0.039947283268
49	120	120	120	0.106703996658	0.107582092285	0.107078938484
50	0	0	0	0.0376760959625	0.0381979942322	0.0378700137138
51	0	0	0	0.0329570770264	0.0338070392609	0.0332756710052
52	120	120	120	0.0973110198975	0.0986480712891	0.0977945637703
53	130	130	130	0.142982006073	0.144871234894	0.143854813576
54	0	0	0	0.0428409576416	0.0437841415405	0.0430500674248
55	0	0	0	0.0452370643616	0.0460958480835	0.0454759860039
56	0	0	0	0.0191400051117	0.0195310115814	0.0193167853355
57	0	0	0	0.0333580970764	0.0340430736542	0.0336854720116
58	120	120	120	0.10723400116	0.108610153198	0.107789361477
59	0	0	0	0.0494701862335	0.0504930019379	0.0497664308548
60	130	130	130	0.169901847839	0.171468019485	0.170678527355
61	0	0	0	0.0276379585266	0.0281829833984	0.0279912400246
62	0	0	0	0.0508639812469	0.0517809391022	0.0511528730392
63	0	0	0	0.0286478996277	0.0290861129761	0.0288627362251
64	0	0	0	0.0258460044861	0.0264251232147	0.0260882258415
65	0	0	0	0.0446481704712	0.0451350212097	0.0448766231537
66	0	0	0	0.0388371944427	0.0394721031189	0.039068608284
67	0	0	0	0.0470900535583	0.0483410358429	0.0474383163452
68	0	0	0	0.0430281162262	0.0436899662018	0.0432226514816

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
69	110	110	110	0.116548061371	0.117974996567	0.117038543224
70	0	0	0	0.0450940132141	0.0459110736847	0.0453917622566
71	0	0	0	0.0497140884399	0.0508878231049	0.0502061843872
72	0	0	0	0.0373630523682	0.0378820896149	0.0375590610504
73	0	0	0	0.0525300502777	0.0532600879669	0.0528325605392
74	0	0	0	0.0301768779755	0.0306968688965	0.030406639576
75	140	140	140	0.146105051041	0.261359930038	0.151420311928
76	0	0	0	0.0253548622131	0.0273001194	0.0257448124886
77	130	130	130	0.152754068375	0.16094493866	0.154390714169
78	0	0	0	0.0440471172333	0.048840045929	0.044670381546
79	0	0	0	0.0307590961456	0.0328230857849	0.0312331795692
80	0	0	0	0.0279150009155	0.0321300029755	0.0284163427353
81	0	0	0	0.0516409873962	0.0546889305115	0.0521724510193
82	0	0	0	0.0499830245972	0.0536251068115	0.0505566692352
83	0	0	0	0.0442368984222	0.0473620891571	0.0448612260818
84	0	0	0	0.0247468948364	0.0277121067047	0.025024907589
85	0	0	0	0.0257019996643	0.0287370681763	0.0261027050018
86	0	0	0	0.0292959213257	0.0384750366211	0.0301448369026
87	0	0	0	0.0390250682831	0.0702650547028	0.0399679660797
88	0	0	0	0.0137820243835	0.0169289112091	0.0139652109146
89	0	0	0	0.0400290489197	0.0439040660858	0.0404689884186
90	0	0	0	0.0414199829102	0.0456340312958	0.0418427395821
91	110	110	110	0.0885050296783	0.103165149689	0.0899881458282
92	120	120	120	0.089015007019	0.115070819855	0.0897792267799
93	120	120	120	0.105010986328	0.110002994537	0.105935454369
94	130	130	130	0.151407003403	0.241286993027	0.155793862343
95	0	0	0	0.0531389713287	0.0549020767212	0.0534213805199
96	110	110	110	0.095104932785	0.167237043381	0.0968285250664
97	0	0	0	0.0395720005035	0.0570678710938	0.0408009505272
98	0	0	0	0.0475568771362	0.0523529052734	0.048278324604
99	110	110	110	0.0967559814453	0.106359004974	0.0977344965935
100	0	0	0	0.05619597435	0.0568201541901	0.0564796328545
101	110	110	110	0.0857999324799	0.115807056427	0.0872610783577
102	0	0	0	0.0346891880035	0.0656020641327	0.0366877865791
103	0	0	0	0.0221960544586	0.0281147956848	0.0229219198227
104	0	0	0	0.0296149253845	0.0337100028992	0.0300550866127
105	120	120	120	0.15439414978	0.159656047821	0.155627810955
106	110	110	110	0.0965120792389	0.10321187973	0.0973339962959
107	110	110	110	0.103091955185	0.107581853867	0.103896286488
108	110	110	110	0.11647605896	0.121008872986	0.117341096401
109	120	120	120	0.111763000488	0.116070985794	0.11260073185
110	0	0	0	0.0240540504456	0.0264489650726	0.0243892240524

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
111	0	0	0	0.0576617717743	0.0616519451141	0.0581876707077
112	120	120	120	0.088418006897	0.0928549766541	0.0890985989571
113	0	0	0	0.0621471405029	0.0658440589905	0.0627804398537
114	0	0	0	0.0516970157623	0.0542709827423	0.0521567392349
115	120	120	120	0.0932419300079	0.0976521968842	0.0939276456833
116	0	0	0	0.0259649753571	0.0283360481262	0.026344537735
117	110	110	110	0.0895700454712	0.0940001010895	0.0902992129326
118	0	0	0	0.0409889221191	0.0449869632721	0.0413822960854
119	0	0	0	0.0252249240875	0.0284039974213	0.0255985927582
120	0	0	0	0.0453398227692	0.0493619441986	0.0457785511017
121	0	0	0	0.025787115097	0.0294880867004	0.0263616919518
122	0	0	0	0.0403971672058	0.0440399646759	0.0407548761368
123	0	0	0	0.0469660758972	0.0509641170502	0.0474152874947
124	0	0	0	0.0418298244476	0.0440428256989	0.0421893811226
125	0	0	0	0.0258619785309	0.0296368598938	0.0261776351929
126	0	0	0	0.0492839813232	0.0529761314392	0.0497709560394
127	0	0	0	0.0284879207611	0.0323979854584	0.0288065195084
128	0	0	0	0.0257911682129	0.0284168720245	0.0262671041489
129	110	110	110	0.101558923721	0.120894908905	0.102776377201
130	0	0	0	0.0497479438782	0.0507898330688	0.0502099967003
131	0	0	0	0.0554869174957	0.0565991401672	0.0559123301506
132	0	0	0	0.0422899723053	0.0431289672852	0.0425380802155
133	0	0	0	0.0594570636749	0.0604069232941	0.059770822525
134	0	0	0	0.0373661518097	0.0407979488373	0.0376842045784
135	0	0	0	0.0336489677429	0.0346541404724	0.0339985489845
136	0	0	0	0.0296831130981	0.0483140945435	0.0306499099731
137	110	110	110	0.104043960571	0.132189035416	0.105327422619
138	0	0	0	0.0421049594879	0.0439820289612	0.0426387405396
139	140	140	140	0.14422416687	0.182739019394	0.146451518536
140	0	0	0	0.0412409305573	0.043309211731	0.0416885828972
141	110	110	110	0.0985610485077	0.104468107224	0.099453458786
142	110	110	110	0.0909280776978	0.0920920372009	0.0914871025085
143	0	0	0	0.0294818878174	0.0479779243469	0.0316082072258
144	110	110	110	0.0846290588379	0.091224193573	0.086375412941
145	120	120	120	0.101259946823	0.111741781235	0.102714371681
146	130	130	130	0.148128032684	0.226861953735	0.151783440113
147	0	0	0	0.0541801452637	0.0606639385223	0.0551668787003
148	110	110	110	0.101431131363	0.108732938766	0.102985920906
149	0	0	0	0.0404009819031	0.0444929599762	0.0408610892296
150	0	0	0	0.0384900569916	0.0427761077881	0.0389377975464
151	0	0	0	0.0269510746002	0.0309500694275	0.027436234951
152	0	0	0	0.0518679618835	0.0557589530945	0.0524069666862

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
153	140	140	140	0.141403198242	0.147403001785	0.142687880993
154	0	0	0	0.0533740520477	0.057736158371	0.0539935183525
155	0	0	0	0.0278460979462	0.0317490100861	0.028225979805
156	0	0	0	0.0425379276276	0.0455429553986	0.043069460392
157	0	0	0	0.0427579879761	0.0461089611053	0.0432976126671
158	0	0	0	0.0421531200409	0.046245098114	0.0426575684547
159	110	110	110	0.100584030151	0.109452009201	0.10146351099
160	0	0	0	0.0205600261688	0.0231750011444	0.020863571167
161	110	110	110	0.0962069034576	0.100555896759	0.0970277929306
162	120	120	120	0.152873039246	0.158874034882	0.154567885399
163	0	0	0	0.046993970871	0.076621055603	0.0480807042122
164	0	0	0	0.0390679836273	0.0429410934448	0.0394655394554
165	0	0	0	0.0456819534302	0.0488529205322	0.0461804270744
166	0	0	0	0.0464680194855	0.050901889801	0.0471006441116
167	0	0	0	0.0245521068573	0.0268440246582	0.024879386425
168	0	0	0	0.0279750823975	0.0310649871826	0.0283529186249
169	0	0	0	0.0534970760345	0.0575098991394	0.0539257049561
170	0	0	0	0.0535669326782	0.0579879283905	0.0540858578682
171	120	120	120	0.105209112167	0.109487771988	0.105840220451
172	110	110	110	0.100945949554	0.105854034424	0.102066857815
173	0	0	0	0.0446419715881	0.0485599040985	0.04514523983
174	0	0	0	0.0341289043427	0.0371849536896	0.0343847513199
175	0	0	0	0.030711889267	0.0346579551697	0.0310666131973
176	0	0	0	0.0509378910065	0.0631239414215	0.0519495677948
177	0	0	0	0.0471389293671	0.0527000427246	0.0476813292503
178	110	110	110	0.0889930725098	0.0907118320465	0.0896314907074
179	120	120	120	0.0943241119385	0.0952179431915	0.0946757149696
180	110	110	110	0.10436797142	0.112588167191	0.105275850296
181	120	120	120	0.180444955826	0.185348987579	0.181194648743
182	0	0	0	0.0449190139771	0.046128988266	0.045616080761
183	0	0	0	0.0360169410706	0.0366578102112	0.0362880468369
184	0	0	0	0.0231640338898	0.0236790180206	0.0234036564827
185	110	110	110	0.0968990325928	0.0980410575867	0.097365694046
186	130	130	130	0.142926931381	0.144441843033	0.143687095642
187	120	120	120	0.172919034958	0.179719924927	0.173922998905
188	110	110	110	0.10377407074	0.160430908203	0.10489300251
189	0	0	0	0.0423908233643	0.0591349601746	0.0429549217224
190	120	120	120	0.0892879962921	0.0902519226074	0.0897048401833
191	0	0	0	0.020311832428	0.0207738876343	0.0205529260635
192	0	0	0	0.0228750705719	0.0233120918274	0.0230605220795
193	0	0	0	0.0424280166626	0.0475611686707	0.0427536320686
194	0	0	0	0.0622959136963	0.0633268356323	0.0627817988396

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
195	110	110	110	0.0913529396057	0.0922961235046	0.0917904090881
196	0	0	0	0.0422070026398	0.0430607795715	0.0424107599258
197	120	120	120	0.161842107773	0.167789936066	0.162604076862
198	0	0	0	0.0391550064087	0.0396950244904	0.0393599462509
199	110	110	110	0.0920441150665	0.0933558940887	0.0926117873192
200	0	0	0	0.0270109176636	0.03209400177	0.027444870472
201	0	0	0	0.0263178348541	0.0311439037323	0.0266432976723
202	110	110	110	0.107219219208	0.122668981552	0.108066422939
203	0	0	0	0.0515351295471	0.0534520149231	0.0520129823685
204	0	0	0	0.0509009361267	0.123127937317	0.0531327795982
205	0	0	0	0.0580198764801	0.103552103043	0.0610372400284
206	0	0	0	0.0199880599976	0.0231828689575	0.0202863335609
207	0	0	0	0.0335988998413	0.0370600223541	0.0340301799774
208	0	0	0	0.0496740341187	0.0533390045166	0.0502678656578
209	120	120	120	0.0938239097595	0.098335981369	0.0945540547371
210	160	160	160	0.197010040283	0.202018022537	0.198366184235
211	130	130	130	0.160308122635	0.165540933609	0.161578412056
212	0	0	0	0.0575249195099	0.0618109703064	0.0581239509583
213	0	0	0	0.0459821224213	0.0497989654541	0.0463192653656
214	0	0	0	0.0534060001373	0.097972869873	0.0550887298584
215	0	0	0	0.0250010490417	0.0325601100922	0.0258263039589
216	0	0	0	0.0253050327301	0.0258588790894	0.0256198048592
217	0	0	0	0.0478851795197	0.0531339645386	0.0484745168686
218	0	0	0	0.0355229377747	0.0361759662628	0.035716342926
219	0	0	0	0.0427129268646	0.0434560775757	0.0430948925018
220	110	110	110	0.0965328216553	0.0979990959167	0.0971311330795
221	0	0	0	0.050684928894	0.0514950752258	0.0510220122337
222	0	0	0	0.050106048584	0.0509219169617	0.0504538798332
223	0	0	0	0.0494148731232	0.0501589775085	0.0496656394005
224	0	0	0	0.0239148139954	0.0245501995087	0.0242541694641
225	110	110	110	0.085736989975	0.0868721008301	0.0861972761154
226	0	0	0	0.0479559898376	0.0488219261169	0.0482026648521
227	110	110	110	0.0874180793762	0.174237012863	0.0914021444321
228	0	0	0	0.0573890209198	0.0962710380554	0.0593956208229
229	0	0	0	0.0446178913116	0.0486881732941	0.0451079750061
230	0	0	0	0.0341200828552	0.067948102951	0.035733935833
231	0	0	0	0.043319940567	0.050637960434	0.0437674593925
232	0	0	0	0.0307879447937	0.0368678569794	0.0315376067162
233	0	0	0	0.0443429946899	0.0459520816803	0.044845559597
234	130	130	130	0.146174907684	0.147849798203	0.146853275299
235	0	0	0	0.0576961040497	0.0585241317749	0.0580104422569
236	110	110	110	0.16411614418	0.165604114532	0.164820408821

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
237	0	0	0	0.0362591743469	0.0368571281433	0.0364805388451
238	110	110	110	0.0894169807434	0.0903768539429	0.0898636603355
239	0	0	0	0.046147108078	0.0467121601105	0.046341676712
240	0	0	0	0.0225009918213	0.0231759548187	0.0229357457161
241	0	0	0	0.0468049049377	0.0472660064697	0.0470622110367
242	130	130	130	0.145211935043	0.146790027618	0.145993614197
243	110	110	110	0.117101192474	0.12982583046	0.118103153706
244	0	0	0	0.0270130634308	0.0337948799133	0.0277351021767
245	0	0	0	0.0490081310272	0.054309129715	0.0494151687622
246	110	110	110	0.0917620658875	0.0994729995728	0.0926747059822
247	0	0	0	0.023973941803	0.0246431827545	0.0242858099937
248	0	0	0	0.036789894104	0.0374631881714	0.0370454835892
249	0	0	0	0.0399169921875	0.0403320789337	0.04006305933
250	0	0	0	0.0577938556671	0.0588719844818	0.0582545852661
251	0	0	0	0.0530450344086	0.0538070201874	0.0533518671989
252	0	0	0	0.0555369853973	0.0568490028381	0.0560135388374
253	130	130	130	0.168934106827	0.170382022858	0.169541909695
254	0	0	0	0.0267050266266	0.0273790359497	0.026908864975
255	110	110	110	0.088583946228	0.0915338993073	0.0891682696342
256	0	0	0	0.0411939620972	0.0417859554291	0.0414839076996
257	130	130	130	0.142972946167	0.189895868301	0.144621019363
258	0	0	0	0.041403055191	0.042308807373	0.0417053818703
259	0	0	0	0.0400388240814	0.0408489704132	0.0402913331985
260	0	0	0	0.0362298488617	0.0367739200592	0.036462392807
261	120	120	120	0.167933940887	0.169866085052	0.168626616001
262	0	0	0	0.0425510406494	0.0433270931244	0.0428165721893
263	0	0	0	0.0278120040894	0.028450012207	0.0281262469292
264	0	0	0	0.0401248931885	0.0407569408417	0.0403466105461
265	110	110	110	0.107434034348	0.114368915558	0.108100581169
266	120	120	120	0.0989689826965	0.166193962097	0.101685068607
267	0	0	0	0.0389859676361	0.0424120426178	0.039330227375
268	110	110	110	0.0891149044037	0.137600898743	0.0916637682915
269	110	110	110	0.0971651077271	0.162809848785	0.0988910913467
270	0	0	0	0.0386710166931	0.0436880588531	0.0389632821083
271	0	0	0	0.0297789573669	0.0336768627167	0.0301615190506
272	0	0	0	0.0377531051636	0.0390338897705	0.037916200161
273	0	0	0	0.0499029159546	0.12539100647	0.0561211109161
274	0	0	0	0.0434060096741	0.0491349697113	0.0438756990433
275	140	140	140	0.16029715538	0.367347002029	0.167220611572
276	0	0	0	0.0536270141602	0.0548820495605	0.0540663719177
277	0	0	0	0.0534188747406	0.0659101009369	0.054429936409
278	0	0	0	0.0272059440613	0.0287780761719	0.0276324343681

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
279	0	0	0	0.0279829502106	0.0286009311676	0.028221514225
280	0	0	0	0.0324862003326	0.0331370830536	0.0328169822693
281	0	0	0	0.0478069782257	0.0884649753571	0.0502428150177
282	0	0	0	0.0383269786835	0.0387401580811	0.0385629940033
283	120	120	120	0.0988450050354	0.100520133972	0.0992358541489
284	0	0	0	0.0528719425201	0.0540900230408	0.0531343913078
285	110	110	110	0.107731819153	0.164661169052	0.110755913258
286	0	0	0	0.0282120704651	0.0335841178894	0.0287639093399
287	130	130	130	0.205417871475	0.25937795639	0.209085288048
288	110	110	110	0.0877239704132	0.144962787628	0.0888978862762
289	0	0	0	0.045658826828	0.059553861618	0.0460863351822
290	120	120	120	0.107913017273	0.109524011612	0.108340425491
291	110	110	110	0.11502790451	0.120102882385	0.115667581558
292	130	130	130	0.15317606926	0.154617071152	0.153809401989
293	0	0	0	0.0399479866028	0.04514503479	0.0403043484688
294	0	0	0	0.0275659561157	0.0284278392792	0.0281073236465
295	0	0	0	0.0295078754425	0.0349080562592	0.0299942541122
296	0	0	0	0.022402048111	0.0230689048767	0.0226625037193
297	140	140	140	0.204725980759	0.210968971252	0.205613541603
298	0	0	0	0.0525929927826	0.0532228946686	0.0529281497002
299	0	0	0	0.0394420623779	0.0400900840759	0.039623041153
300	0	0	0	0.0361731052399	0.0367679595947	0.0363198184967
301	0	0	0	0.0537478923798	0.0543661117554	0.0540637326241
302	0	0	0	0.0375311374664	0.0384418964386	0.0378574323654
303	0	0	0	0.0462260246277	0.0470309257507	0.0465603947639
304	0	0	0	0.023754119873	0.0244228839874	0.0239508271217
305	120	120	120	0.166773080826	0.168387889862	0.167414841652
306	0	0	0	0.0456261634827	0.0465829372406	0.0459975481033
307	0	0	0	0.0388648509979	0.0838050842285	0.0400058960915
308	0	0	0	0.0562980175018	0.0639710426331	0.0566978764534
309	0	0	0	0.0512580871582	0.0564568042755	0.0515799856186
310	110	110	110	0.102589130402	0.11070394516	0.103373064995
311	0	0	0	0.0307168960571	0.0317311286926	0.031077940464
312	0	0	0	0.0197570323944	0.0351510047913	0.0214538645744
313	0	0	0	0.0415380001068	0.0425109863281	0.0418085622787
314	120	120	120	0.106746912003	0.109905004501	0.10749774456
315	0	0	0	0.0381591320038	0.0388150215149	0.0383869934082
316	140	140	140	0.156733989716	0.167254924774	0.157789099216
317	0	0	0	0.0261878967285	0.0269181728363	0.0266342186928
318	0	0	0	0.0373990535736	0.0382490158081	0.0377261853218
319	0	0	0	0.0223891735077	0.0229671001434	0.0226274204254
320	0	0	0	0.0453858375549	0.0462009906769	0.0457191634178

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
321	0	0	0	0.0444808006287	0.0495638847351	0.0449372243881
322	0	0	0	0.0433218479156	0.0486650466919	0.0438178920746
323	130	130	130	0.14691901207	0.152347803116	0.147756543159
324	0	0	0	0.0250308513641	0.0303730964661	0.025495994091
325	0	0	0	0.0471489429474	0.0549778938293	0.0474142003059
326	0	0	0	0.0518050193787	0.0523829460144	0.0520178794861
327	0	0	0	0.0398190021515	0.0449380874634	0.0402103543282
328	0	0	0	0.0291540622711	0.0299217700958	0.0295031762123
329	120	120	120	0.113414049149	0.128154993057	0.114491815567
330	0	0	0	0.0261180400848	0.0265920162201	0.0263215470314
331	120	120	120	0.0900619029999	0.0971648693085	0.0906645202637
332	0	0	0	0.0445921421051	0.0775470733643	0.0467071127892
333	170	170	170	0.245864868164	0.261065006256	0.248330309391
334	0	0	0	0.0406899452209	0.0941998958588	0.0430207228661
335	0	0	0	0.0163218975067	0.0230021476746	0.01674523592
336	110	110	110	0.088259935379	0.0926179885864	0.0891556310654
337	0	0	0	0.0319151878357	0.035572052002	0.03222635746
338	0	0	0	0.0472280979156	0.0572938919067	0.0479260253906
339	120	120	120	0.155865907669	0.17658996582	0.157384440899
340	130	130	130	0.149626016617	0.154409885406	0.150567986965
341	0	0	0	0.0494799613953	0.0510940551758	0.0498322629929
342	0	0	0	0.0401167869568	0.0406670570374	0.0403039336205
343	0	0	0	0.0399730205536	0.0409038066864	0.0403252434731
344	0	0	0	0.0247139930725	0.0300397872925	0.0250907897949
345	0	0	0	0.0523600578308	0.0534009933472	0.0528555774689
346	0	0	0	0.0429220199585	0.048171043396	0.0432945084572
347	110	110	110	0.102617025375	0.129700899124	0.103640789986
348	130	130	130	0.191762924194	0.198933124542	0.192847926617
349	0	0	0	0.0439138412476	0.0492880344391	0.0445358610153
350	0	0	0	0.0355200767517	0.0361630916595	0.0357541704178
351	0	0	0	0.023451089859	0.0248990058899	0.0236693120003
352	0	0	0	0.0285592079163	0.0356798171997	0.0292410755157
353	110	110	110	0.087070941925	0.0928931236267	0.0876731204987
354	120	120	120	0.139060974121	0.145006895065	0.139743721485
355	0	0	0	0.0484218597412	0.0492298603058	0.0488010025024
356	0	0	0	0.0396800041199	0.0402290821075	0.0398619675636
357	0	0	0	0.0458791255951	0.0466639995575	0.046100564003
358	0	0	0	0.0507371425629	0.0823359489441	0.051460776329
359	0	0	0	0.0273590087891	0.0300590991974	0.0276557588577
360	0	0	0	0.0391211509705	0.0403470993042	0.0393459868431
361	140	140	140	0.15007686615	0.156263113022	0.150766561031
362	0	0	0	0.0408489704132	0.0414600372314	0.0410478544235

Instance Number	Best Result	Worst Result	Average Result	Best Time	Worst Time	Average Time
363	110	110	110	0.0975470542908	0.101248979568	0.0980833768845
364	0	0	0	0.0502059459686	0.0511591434479	0.0506198716164
365	120	120	120	0.0977909564972	0.104001998901	0.0983244848251
366	110	110	110	0.0864429473877	0.0921912193298	0.0870484161377
367	0	0	0	0.0356161594391	0.0363719463348	0.0359356403351
368	0	0	0	0.0276579856873	0.0283639431	0.0279165840149
369	0	0	0	0.0439009666443	0.0445759296417	0.0441519784927
370	0	0	0	0.0370051860809	0.0373339653015	0.0371716141701
371	0	0	0	0.0462200641632	0.0489549636841	0.0465096569061
372	110	110	110	0.120441913605	0.130411863327	0.121280210018
373	110	110	110	0.110706090927	0.292850971222	0.113921084404
374	0	0	0	0.0362119674683	0.0720970630646	0.0382264876366
375	0	0	0	0.027480840683	0.0284209251404	0.027860519886
376	0	0	0	0.019907951355	0.0202569961548	0.0201093482971
377	110	110	110	0.100075006485	0.105656862259	0.10061016798
378	0	0	0	0.0402820110321	0.0415549278259	0.0405454778671
379	110	110	110	0.106014966965	0.11611199379	0.10645578146
380	0	0	0	0.0569319725037	0.0577480792999	0.0573105382919
381	130	130	130	0.1636531353	0.169852972031	0.16449190855
382	0	0	0	0.0249030590057	0.0257639884949	0.0251887512207
383	110	110	110	0.105440139771	0.113557100296	0.106216440201
384	0	0	0	0.0257070064545	0.0263381004333	0.0260341858864
385	0	0	0	0.0399839878082	0.0407609939575	0.0402080702782
386	0	0	0	0.0501918792725	0.0559070110321	0.0506551074982
387	0	0	0	0.0475649833679	0.0743980407715	0.0485756230354
388	0	0	0	0.0494349002838	0.0606949329376	0.0501068162918
389	130	130	130	0.151811122894	0.178202867508	0.153364162445
390	0	0	0	0.0367050170898	0.0381710529327	0.0369458842278
391	0	0	0	0.0331959724426	0.0387070178986	0.0338664007187
392	0	0	0	0.0494470596313	0.0502009391785	0.0497253298759
393	120	120	120	0.168467998505	0.205666065216	0.170596206188
394	140	140	140	0.204308986664	0.247154951096	0.206739850044
395	110	110	110	0.104656934738	0.112908840179	0.105696568489
396	130	130	130	0.180577993393	0.188539028168	0.1818486166
397	0	0	0	0.0551319122314	0.0608129501343	0.0558828902245
398	120	120	120	0.103153944016	0.12224984169	0.10395544529
399	120	120	120	0.0848140716553	0.0934991836548	0.085708322525
400	0	0	0	0.0308129787445	0.0316970348358	0.031131067276

Appendix C

Webpage Output

Example of a full web page produced when requested, displaying schedule and additional information.

Chemotherapy Nurse Day Schedule

Date: 01.02.1993

Patient ID:	1	2	3	4	5	6	7	8	9	LUNCH
Regime:	1	1	27	34	48	84	157	167	123	LUNCH
9:00										
9:15										
9:30										
9:45										
10:00										
10:15										
10:30										
10:45										
11:00										
11:15										
11:30										
11:45										
12:00										
12:15										
12:30										
12:45										
13:00										
13:15										
13:30										
13:45										
14:00										
14:15										
14:30										
14:45										
15:00										
15:15										
15:30										
15:45										
16:00										
16:15										
16:30										
16:45										
17:00										
17:15										
17:30										
17:45										

Patient Information

Patient ID: 1, **Regime Number**: 1, **Start Time:** 0
Patient ID: 2, **Regime Number**: 1, **Start Time:** 3
Patient ID: 3, **Regime Number**: 27, **Start Time:** 26
Patient ID: 4, **Regime Number**: 34, **Start Time:** 5
Patient ID: 5, **Regime Number**: 48, **Start Time:** 8
Patient ID: 6, **Regime Number**: 84, **Start Time:** 11
Patient ID: 7, **Regime Number**: 157, **Start Time:** 20
Patient ID: 8, **Regime Number**: 167, **Start Time:** 14
Patient ID: 9, **Regime Number**: 123, **Start Time:** 24

Statistics:

Number Clashes: 0
Number Free Slots: 8

Calum Clark, University Of Leeds