GA-ASAP: Radiotherapy Treatment Scheduling with Genetic Algorithm

Gong Chen

Computer Science Department

*San Diego State University*

*San Diego, CA 92182*

Abstract

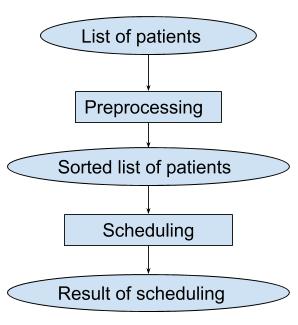
The number of cancer cases in the United States is rapidly increasing. According to a cancer statistics report from National Cancer Institute [????], the number of cancers cases will rise to 22 million within the next two decades. Among all kinds of cancer treatments, radiotherapy is considered as one of the most effective ways [????]. According to [????], the reduction of patient’s waiting time is one of the key factor for successful treatment as cancer cells are growing all the time. However, compared with the growing number of patients, the medical resources is relatively limited. Therefore, an efficient radiotherapy scheduling algorithm that aims to the delivery of right treatments at the right time becomes more important. In this project, we develop a heuristic scheduling algorithm, GA-ASAP (Genetic Algorithm [????] – As Soon As Possible[????]), to optimize scheduling regarding total tardiness for patients with radiotherapy treatments.

# 1. Introduction

Different from the traditional appointment scheduling, radiotherapy treatment is much more sensitive regarding patients’ waiting time. According to [2], the British Government claimed that it is important to reduce the patients’ waiting time in order to reach an effective treatment. The survival rate for patients with colon cancer and lung cancer in women arises, mainly, in the first six months after the diagnosis. In other words, delay of the treatment will reduce the survival rate of patients. Therefore, an effective scheduling algorithm of radiotherapy that could reduce patients’ waiting time is in need.

In 2006, [3], Petrovic et al propose another two algorithms for scheduling radiotherapy treatments, which are As Soon As Possible (ASAP) and Just In Time (JIT). In these two algorithms, patients with a more advanced level of the disease are privileged. The allocation of patients depends on the priority list. The main difference between the algorithms ASAP and JIT is that JIT assigns patients on the last available day while ASAP assigns patients from the first available day. The experiments were carried on real data corresponding to a health center in the United Kingdom. From the results obtained in this work, the algorithm ASAP, had a higher satisfaction rate for patients.

ASAP consists of two phases which are preprocessing and scheduling. Figure 1 shows the architecture of ASAP algorithm. In the first phase, patients are sorted based on their priorities, such as release day, due day, treatment day and level of emergency. In the second phase, scheduling algorithm will allocates the patients based on the sorted list which is the result of preprocessing.



**Figure 1: ASAP algorithm**

# 1.1. Motivation example

To investigate the assumption that patients order impact on the effective of solution resulted by ASAP algorithm, we consider following example which the patients set shown in TABLE I. For the sake of simplicity, we assume that the capacity of radiotherapy treatment is 2 patients per day. The scheduling patients order represented by patient ID is (5 – 2 – 1 – 4 – 3 – 6) followed by Petrovic prioritization rule [???]. The total length of waiting time is 4 days. By observing the scheduled treatment table, we find that if we swap the patients order under the same priority level, we could get less total length of waiting time. For example, if we swap patient 1 and patient 2, the scheduling order will be (5 – 1 – 2 – 4 – 3 – 6). The total length of new order will be 3 days which is less than the result of ASAP algorithm.

**Table I. List of patients associated with priority attributes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Release Day | Due Date | Treatment Days | Priority |
| 1 | 2 | 4 | 3 | E |
| 2 | 1 | 3 | 5 | E |
| 3 | 5 | 19 | 10 | R |
| 4 | 3 | 17 | 10 | R |
| 5 | 1 | 3 | 6 | E |
| 6 | 7 | 35 | 21 | R |

Therefore, we can assume that ASAP algorithm could be improved by changing the patients order. Actually, brute force algorithm can guarantee to find the best solution. However, the time cost of computing will increase exponentially over the increasing of patients’ number. Accordingly, genetic algorithm is a widely preferred algorithm to solving this kind of problem. To investigate improvement and effectiveness of genetic algorithm, we propose a new scheduling algorithm which is GA-ASAP which is combination of genetic algorithm and ASAP algorithm. The paper is organized as follows. In the following section, we briefly revise the related work of the scheduling problem in radiotherapy domain. In section 3, the details of problem will be addressed. In section 4, approach attempted to improve effective is presented. In section 5, the result of our experiments will be presented. The conclusion and a discussion of future work are given in section 6.

# 2. Related Work

The problem of scheduling radiotherapy treatment has been studied for years. One of the very first approaches can be found in [1], conducted by the Department of Oncological Radiation in Manitoba, Canada. The main objectives of the proposed system were (1) to generate a radiotherapy scheduling, in a reasonable time; (2) cheap to implement; (3) easy to use; and (4) capable to manage around 100-150 patients each month. This application was developed using Excel macros. Authors in [????] propose a Greedy Randomize Adaptive Search Procedure (GRASP) for solving the problem without priorities.

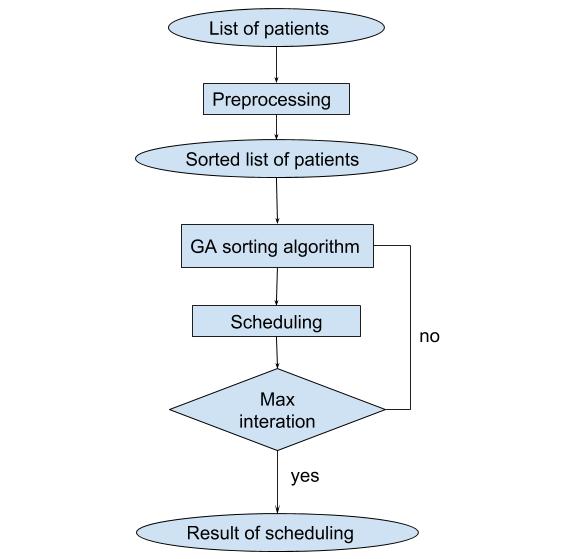
# 3. Problem formulation

The problem to be addressed is how to improve ASAP algorithm by applying genetic algorithm such that the total tardiness will reduce. It is formally defined as follows: Given a set of patients P = {}, each patient is defined by a tuple as where

* r(i): release data for patient to be scheduled in the system
* d(i): due date by the patients p(i) starts the treatment
* t(i): number of determined treatment session for patient p(i)
* priority(i): level of patient p(i), which are emergency or regular

The total tardiness is (1) where s(i) is the actually starting date for patient p(i). The quality of scheduled result is measure by the total tardiness of patients.

tardiness = ) (1)



**Figure 2. GA-ASAP algorithm**

# 4. GA-ASAP algorithm

**4.1. Architecture**

In this section, we propose a heuristic algorithm, GA-ASAP. Figure 2 is the outline of architecture. Same as ASAP, we first sort patients list based on their priority. Then we assign the sorted list to GA algorithm along with other lists of patients with random sequence. Those list together compose the 1st generation. We apply ASAP to the 1st generation and calculate the tardiness of each. Then we select the lists with less tardiness and generate our 2nd generation based on them by crossover and mutation, two basic random operator in GA. Then we repeat the process on 2nd, 3rd until max generation and select the one with least tardiness as our best solution.

|  |
| --- |
| GA-ASAP Algorithm  Input: a list of patients  Output: bestSolution  1: initialize the bestOrder by prioritization rule  2: **while** not reach max iteration **do**  3: random list = randomly generate patients list  4: nth generation = random list and bestOrder  5: offspring = crossover (nth generation) and  mutation(generation)  6: **for** each offspring **do**  7: tardiness(offspring) = ASAP(offspring)  8: **if** tardiness(offspring) < bestTardiness **do**  8: bestOrder = offspring  9: **end while** |

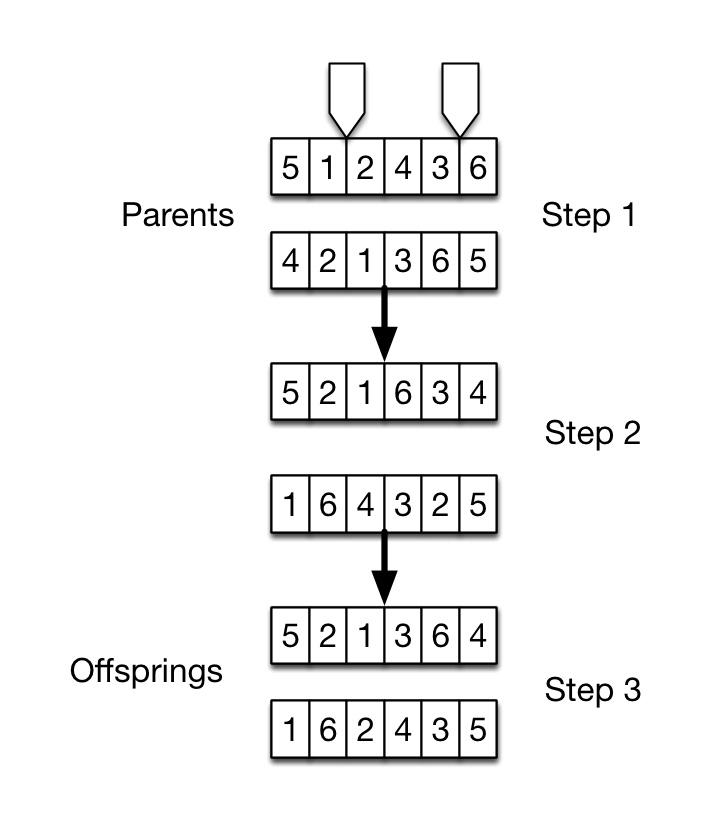
**Figure 3. GA-ASAP pseudocode**

**4.2 Crossover**

The crossover operator introduced in this section were among the first to be designed for the algorithm. This operator first randomly selects two cut points on both parents. Then, the inverse replacement is applied outside of the cut points, in order to eliminate duplicates and recover all patients. In order to create an offspring, the substring between the two cut points in the first parent replaces the corresponding substring in the second parent.

Figure 4 is one of the example shown how the crossover works on generate offspring from parent. As it shown, [5 – 1 – 2 – 4 – 3 – 6] and [4 – 2 – 1 – 3 – 6 – 5] are two parents. The number represents the patient ID and the sequence indicates the order of those patients in terms of priority. If we set the two cuts point at middle of 2nd and 3rd patient and middle of 5th and 6th patient. Then the substring of crossover will be 2 – 4 – 3 in parent 1 and 1 – 3 – 6 in parent 2 (step 1). We can’t simply swap the two substring. Because the new offspring will contain duplicated patient. For example, one of the new offspring will be [5 – 1 – 1 – 3 – 6 – 6]. So we need to another preprocessing by swap the patients inside and outside substring (step 2). Then swap the substring to get new offspring (step 3).

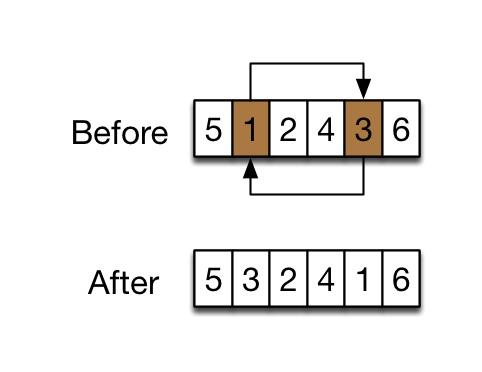
Clearly, this operator tries to preserve the absolute position of the patients when they are copied from the parents to the offspring. In fact, the number of patients that do not inherit their positions from one of the two parents is at most equal to the length of the string between the two cut points.



**Figure 4. Generate offspring by crossover**

**4.3 Mutation**

Mutation operators for the algorithm are aimed at randomly generating new permutations of the patients. As opposed to the classical mutation operator, which introduces small perturbations into the chromosome, the permutation operators for the algorithm often greatly modifies the original order. These operators are summarized below. Two patients are randomly selected and swapped. Figure 5 is an example of mutation. Before mutation, the list is [5 – 1 – 2 – 4 – 3 – 6]. Then the mutation happens randomly on two of patients’ order, which is patient 1 and patient 3. We swap their order to have a new list of [5 – 3 – 2 – 4 – 1 – 6]. This mutation operator is the closest in philosophy to the original mutation operator, because it only slightly modifies the original order.



**Figure 5. Mutation**

**4.4 Select and keep the best offspring**

For each GA iteration, offspring will be generated by crossover and mutation. Offspring with better performance should ‘survival’ and be chosen as the parent for the next iteration. Therefore, for each iteration, we initialize the parents with the random list and best offspring from pervious iteration. Then we produce new offspring by crossover and mutation. At the end of current iteration, we scheduling each offspring based on ASAP algorithm, calculate and choose the best one as one of the parents of next iteration based on their total tardiness.

# 5. Preliminary results analysis

**5.1 Simulation setup**

The goal of this experiment is to compare the proposed GA-ASAP against the conventional ASAP algorithm.

We have designed a generator of random instances which allow to evaluate different configurations with a different number of patients. The instances used in the experiments are similar to our real-world data. We design instances with following Table II.

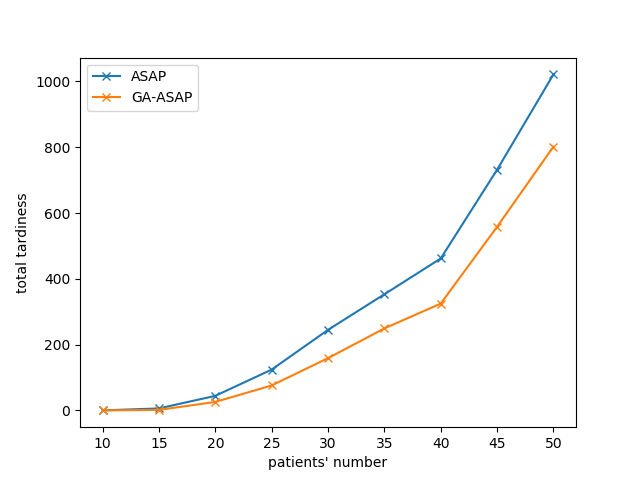
**Table II: Experiment parameters**

|  |  |
| --- | --- |
| release date | Due data: patients’ due date of each patient is randomly from 1-14 |
| treatment day | Total treatment day of each patient is randomly from 15-30 |
| number of patients | Number of total patients is from [10,15,20,25,30,35,40,45,50] |
| priority | The priority of each patient is randomly assigned as Emergency or Regular. The ratio of them is 1:5 |
| capability | The capability of radiotherapy facility is [12,18,24] patients per day |
| max iteration | The max iteration of GA-ASAP is 10 |
| population | The population of each generation is 4 |

We conduct 3 experiments which are total tardiness, total time cost and improvement (2) of GA-ASAP against ASAP. The sub-sections follows show the result of our experiments. The result is the mean average of 100 experiments.

**5.1 Total tardiness**

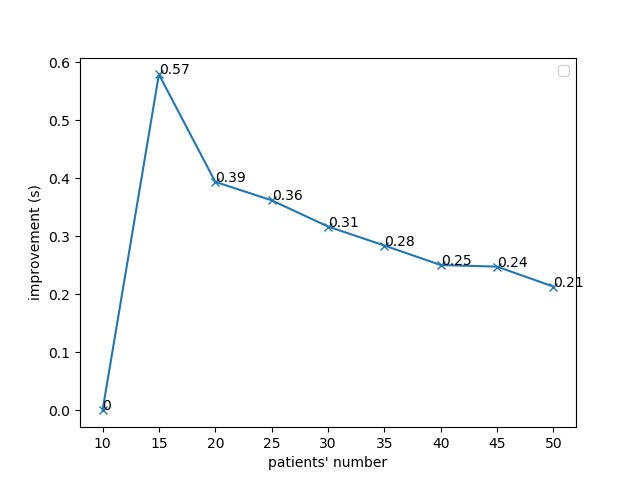
Figure 6 shows the total tardiness of GA-ASAP and ASAP over the increasing of patients’ number. Along with the increasing of patients’ number, the total tardiness of each two algorithm increases. However, the tardiness of GA-ASPA are always less than those of ASAP. From figure 6 and figure 7-1 we can figure that the GA-ASAP has an advantage against ASAP regarding the reduction of total tardiness.



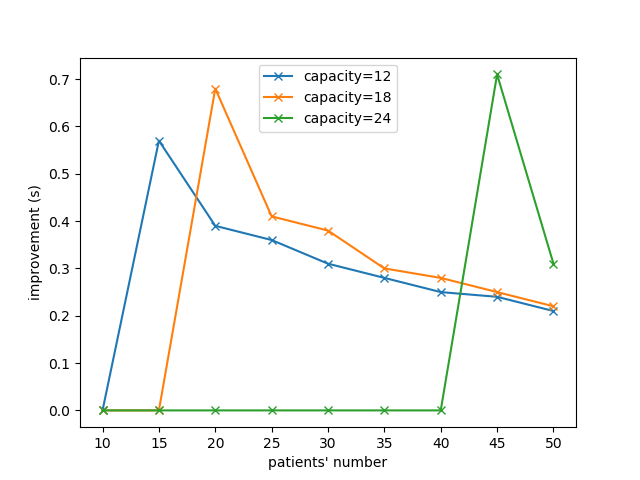
**Figure 6. Total tardiness of GA-ASAP and ASAP**

However, from figure 7-1 which shows the improvement (1) of tardiness of GA-ASAP against ASAP algorithm. We figure out that within a relatively small amount of patient, for example, number from 10 to 15, the improvement of GA-ASAP is over 57%. However, along with the increasing of patients’ number, the improvement reduces to a number around 21%. We assumes that the reason which affect the improvement is the capacity. Therefore, we conduct another experiment figure out the relationship between capacity and improvement. Figure 7-2 show the improvement of GA-ASAP among different of capacity (12, 18, 24 per day). We assume that the utilization rate which is related to the capability and patients’ number affects the improvement. Along with the increasing of patients’ number, the capability remains the same, which lead to the increase of utilization rate. With relatively low utilization rate, both GA-ASAP’s and ASAP’s total tardiness is 0. For those tardiness is not equal to 0, the one with higher capacity (which leads to low utilization rate) has higher improvement rate.

(2)

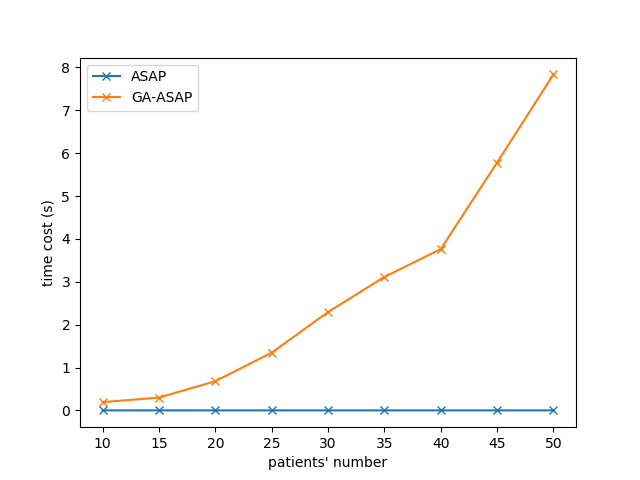


**Figure 7-1. Percentage of improve of GA-ASAP against ASAP in terms of total tardiness on capacity of 12**



**Figure 7-2. Percentage of improve of GA-ASAP against ASAP in terms of total tardiness on capacity of 12, 18 and 24**

Figure 8 is the time cost of GA-ASAP and ASAP algorithm. Although GA-ASAP shows significant advantage against ASAP regarding to the reduction of total tardiness, it consumes more computing resources than ASAP. Along with the increasing of patients’ number, the time cast of GA-ASAP increase obviously. However, the cost of ASAP is remained at a relatively low level. For example, when dealing with 50 patients, the time cost of GA-ASAP is around 8 second, however, the cost of ASAP is within the magnitude of millisecond.

****

**Figure 8. Time cost of GA-ASAP against ASAP in terms of total tardiness**

We observe from result above that GA-ASAP has an advantage ranged from 21 % to 68% against ASAP over 10 - 50 patients.

# 6. Conclusions and future work

**6.1 Conclusion**

To get radiotherapy treatment in time will significant increase the survival rate of patients with cancer. Therefore, an efficient radiotherapy treatment scheduling algorithm is in demand. ASAP algorithm is good solution. However, it still can be optimized. We proposed a new radiotherapy scheduling algorithm GA-ASAP that is able to improve the reduction of total tardiness against ASAP.

Experimental results showed that GA-ASAP can reduce the total tardiness of ASAP up to 68% percentage. However, the improvement is affect by the utilization rate of the overall system which is related to patients’ number and system’s capacity.

**6.1 Future work**

From the previous experimental result, we can figure out GA-ASAP has a significant advantage against ASAP. However, the improvement is affect by multiple parameters. For example, capacity, patients’ number and the details of patients, such as treatment day. For the future work, we are going to investigate more details between these facts and the improvement of GA-ASAP.

**References**

[1] S. L sson, "Radiotherapy patient scheduling using a desktop personal computer," *Clinical Oncology*, vol. 5, no. 2, pp. 98-101, 1993.

[2] P. Spurgeon, F. B well, and D. Kerr, "Waiting times for cancer patients in england a er general practitioners' referrals: retrospective national survey," *8MJ*, vol. 320, no. 7238, pp. 838-839, 3 2000.

[3] S. Petrovic, Leung, X. Song, and S. Sund , "Algorithms for radiotherapy treatment booking," in *Proceedings of the 25th Workshop of the UK Planning and Scheduling Special Interest G up (PlanSIG), Nottingham*, UK, 14-15th December 2006,