Optimizing the assignment of blood in a blood banking system: Some initial results

Conference Paper · June 2012

DOI: 10.1109/CEC.2012.6256633

Aderemi Adewumi
University of KwaZulu-Natal
94 PUBLICATIONS **349** CITATIONS

WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

Optimizing the Assignment of Blood in a Blood Banking System: Some Initial Results

Aderemi Adewumi, Nigel Budlender and Micheal Olusanya
School of Computer Science
University of KwaZulu-Natal
South Africa, Durban 4001
Email: (adewumia,208504267,211560532)@ukzn.ac.za

***Abstract*—Due to the critical blood shortages in South Africa and around the world, the assignment of blood can be considered an important real world optimization problem. This paper presents a mathematical model that facilitates good management and assignment of red blood cell units in order to minimize the quantity of imported units from outside the system. The model makes use of the Multiple Knapsack Algorithm, which is implemented using several optimization techniques, in order to determine the most optimal assignments. These include a Genetic Algorithm (GA), Adaptive Genetic Algorithm (AGA), Simulated Annealing Genetic Algorithm (SAGA), Adaptive Simulated Annealing Genetic Algorithm (ASAGA) and finally a Hill Climbing (HC) Algorithm. All techniques were capable of achieving the optimal fitnesses. The AGA, SAGA and ASAGA provide some desirable results over the standard GA, whilst the HC algorithm proves to demonstrate the best results overall.**

Keywords: Blood Assignment problem, Optimization, Genetic Algorithms, Simulated Annealing, Hill Climbing, Real-World Problem.

I. Introduction

Blood transfusions save thousands of lives daily through medical treatments for patients who require blood components. There are four main components that are present in blood according to the South African National Blood Service [3], of which a patient may require any. In this research we concentrate on optimizing the assignment of the red blood cell component. According to the ABO blood group system [3], there are four different kinds of blood types: A, B, AB and O. This is due to the antigens and antibodies present in each blood type; only certain blood types are compatible with each other. Therefore it is vital that patients are assigned the correct blood type when requested, as mixing incompatible blood groups leads to blood clumping or agglutination, which is dangerous for these patients [3]. The blood type compatibilities are complicated further by the Rhesus factor (Rh) [3] and this doubles the number of blood groups. Another consideration that should be taken into account when assigning blood is that blood has a limited storage time of 30 days [1], and thus should be assigned to patients before it reaches its maximum storage time.

Charpin *et al.* [1] proposes a linear model for the management of a blood bank. This paper considers a dynamical system where everyday units of red blood enter and leave the system. This system aims at managing the blood bank efficiently by getting the proportions of each blood type at the end of the day as close as possible to the ideal proportions considered by the bank. Their model also defines the proportions that can be cross-matched between compatible blood types. This model is still a work in progress and the model has not been tested with any data.

De Angelis *et al.* [2] developed a multi-product, multi-period, multi-objective linear programming model to facilitate the good management of a blood donation-transfusion system. This model identifies that requests for units of blood are associated with an urgency, and medical and surgical interventions should be taken into consideration when assigning units of blood according to urgency and availability. This model also deals with each blood group separately and does not consider cross-matching between blood groups.

In the ideal case, each blood group should be able to supply the corresponding requests for its own blood type, however the donations and requests received by the blood bank can sometimes be unpredictable and unbalanced. This can lead to shortages in certain blood groups and this scenario was observed in the work of De Angelis *et al.* [2].

In this paper we consider a new model which dynamically determines the assignment of blood using the Multiple Knapsack Problem [4]. The idea of cross-matching between blood types is investigated and proves to give positive results. The entire model is described in the following section.

The rest of the paper is organized as follows: In Section II the model designed for the management of blood units is reviewed, as well as optimization algorithms that implement the assignment algorithm. Section III presents and discusses the experimental results obtained. Finally, in Section IV the findings of the experimentation are concluded and possible future work is proposed.

II. Methodology

The dynamic system model that is developed is based on similar approaches described in Charpin *et al.* [1] and De Angelis *et al.* [2]. However, the variation of the newly designed model breaks the problem down into separate entities, and considers a new approach for optimizing the assignment algorithm in particular. First the dynamic system model is described, followed by how the assignment algorithm generates desirable solutions and lastly the optimization techniques, which implement the assignment algorithm, are described.

A. Dynamic Blood System Model

The blood banking system that was designed works as follows. The bank maintains the volumes for each blood type *O±, A±, B± and AB±*. The system runs for a specified time *t*, where each unit of *t* represents one day. Every day there are a certain number of requests for units of red blood cells, for each blood type. The bank then needs to determine the optimal amount of units for each blood type to satisfy the requests, these units are then removed from the corresponding volumes within the bank. If there is an insufficient supply of units, such that the requests cannot be met, then blood is imported from outside the system. Importation of external units is only permitted when the current supply is insufficient and we assume there is sufficient supply of the external resource. There are also a certain amount of donations every day, and these are added to the volumes at the end of each day. The objective function of this model is to minimize the total amount of red blood cell units imported into the bank:

$$minimize \sum_{t=1}^{n} I_{Total}(t) \tag{1}$$

$$where \;\; I_{Total}(t) = I_{O+}(t) + I_{O-}(t) + I_{A+}(t) + I_{A-}(t) + I_{B+}(t) + I_{B-}(t) + I_{AB+}(t) + I_{AB-}(t)$$

In Fig. 1. we observe the blood bank is the central entity that facilitates the management of the entire system. It can also be seen that the assignment algorithm is responsible for determining the optimal assignment of units of blood. The blood bank supplies the information about the number of the requests and current stock within the bank, and based on this data the the algorithm selects which units are to be supplied and if any units need to be imported.

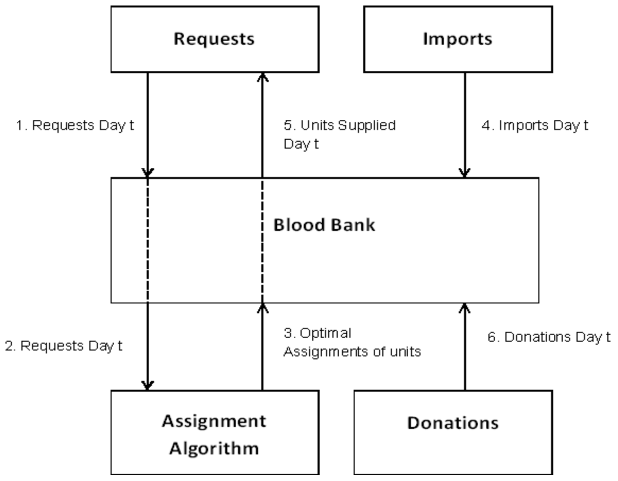


Fig. 1. Model of a Dynamic Blood Banking System

B. Simple Assignment Algorithm

This method of assignment is adapted from the work of De Angelis *et al.* [2] and is a slight variation to their approach. It differs as the urgency of each request for units of red blood cells is not considered, rather everyday the requests are scheduled immediately and the units assigned are put aside and cannot be selected again. It is designed such that each blood group supplies the corresponding request for blood, i.e. there is no cross-matching in this approach. This model is designed to give a good indication of how a system would perform with no cross-matching. It is also useful for benchmarking results to see how it performs compared to the Multiple Knapsack Assignment.

C. Multiple Knapsack Assignment Algorithm

For this approach, it is investigated whether cross-matching between compatible blood types in order to satisfy requests can be used to stabilize the proportions of blood types within the bank, as well as minimize the number of imported units. It was noted that the problem of determining the optimal assignment closely resembles a resource allocation problem, and a corresponding knapsack approach was investigated. Knapsack problems have been widely investigated and there are many approaches to these problems [4, 5]. In particular, the Multiple Knapsack Problem was used.

The Multiple Knapsack Problem can be described as follows. Choose any of *n* items (units stored in the blood bank) and pack them into *m* knapsacks (requests for particular blood types) of different capacity c_i in order to obtain the largest profit sum [4].

$$maximize \sum_{i=1}^{m}\sum_{j=1}^{n} p_j x_{ij} \tag{2}$$

$$subject\ to \sum_{j=1}^{n} w_j x_{ij} \leq c_i, \qquad i = 1, ..., m, \tag{3}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \qquad j = 1, ..., n, \tag{4}$$

$$x_{ij} \,\epsilon\, \{0, 1\}, \qquad i = 1, ..., m, j = 1, ..., n. \tag{5}$$

In (2), it can be observed that the objective is to maximize the profit by placing items into each knapsack. Constraint (3) ensures the capacity constraint of each knapsack is satisfied. Constraint (4) ensures that each item can only be used at most once. Finally, (5) indicates whether an item *j* should be placed into knapsack *i*.

Equations (2) - (5) outline the standard structure of the Multiple Knapsack Problem, however some additional constraints need to be considered for it to work in the dynamic blood banking system. Firstly, there are 8 known blood types when including the Rhesus factor. Therefore each day there are 8 sets of requests for the various blood types, and these correspond to 8 knapsacks. To satisfy these requests, units of red blood cells (items), from either within the bank or imported from outside the system, must be placed into each knapsack. Each knapsack can only be satisfied with a compatible unit of blood and this is an important constraint that cannot be violated.

In order to satisfy the objective function of the system in (1), the profit of the Multiple Knapsack Problem must be maximized and therefore suitable values need to be determined for the profit of each item. The profit is determined by the following important factors:

- The value of each blood type. This is determined by how many different blood types can be placed in each knapsack and the sum for all 8 types amounts to 27. The value of each unit is then defined as this initial value over the total sum resulting in an even distribution.

$$\begin{aligned} O^+ &= \frac{2}{27} = 0.07 & O^- &= \frac{1}{27} = 0.04 \\ A^+ &= \frac{4}{27} = 0.15 & A^- &= \frac{2}{27} = 0.07 \\ B^+ &= \frac{4}{27} = 0.15 & B^- &= \frac{2}{27} = 0.07 \\ AB^+ &= \frac{8}{27} = 0.3 & AB^- &= \frac{4}{27} = 0.15 \end{aligned} \tag{6}$$

  The more knapsacks a unit can be placed in, the more valuable it is. Valuable blood types should therefore have a lower profit as they are important in the sense that they can supply other blood groups when they face shortages.
- The actual volume levels (*V*) of each blood type compared to the desired volume levels (*P*). If a certain blood group has a volume greater than double the desired level, its profit is increased in order to encourage greater usage of this blood type. The reason for the volume having to be greater than double is because some of the smaller blood groups occupy very small percentages of the population and if they are only slightly bigger than the desired level they might try to supply too many requests for larger blood groups. Blood type *AB* is denoted as *C* for brevity.

$$if \ \frac{V_{O^+}}{V_{Tot}} > (P_{O^+} \times 2) \ \& \ \frac{V_{O^+} - S_{O^+}}{V_{Tot} - R_{Tot}} > P_{O^+} \quad then \quad O^+ = 0.15 \tag{7}$$

$$if \ \frac{V_{O^-}}{V_{Tot}} > (P_{O^-} \times 2) \ \& \ \frac{V_{O^-} - S_{O^-}}{V_{Tot} - R_{Tot}} > P_{O^-} \quad then \quad O^- = 0.07 \tag{8}$$

$$if \ \frac{V_{A^+}}{V_{Tot}} > (P_{A^+} \times 2) \ \& \ \frac{V_{A^+} - S_{A^+}}{V_{Tot} - R_{Tot}} > P_{A^+} \quad then \quad A^+ = 0.3 \tag{9}$$

$$if \ \frac{V_{A^-}}{V_{Tot}} > (P_{A^-} \times 2) \ \& \ \frac{V_{A^-} - S_{A^-}}{V_{Tot} - R_{Tot}} > P_{A^-} \quad then \quad A^- = 0.15 \tag{10}$$

$$if \ \frac{V_{B^+}}{V_{Tot}} > (P_{B^+} \times 2) \ \& \ \frac{V_{B^+} - S_{B^+}}{V_{Tot} - R_{Tot}} > P_{B^+} \quad then \quad B^+ = 0.3 \tag{11}$$

$$if \ \frac{V_{B^-}}{V_{Tot}} > (P_{B^-} \times 2) \ \& \ \frac{V_{B^-} - S_{B^-}}{V_{Tot} - R_{Tot}} > P_{B^-} \quad then \quad B^- = 0.15 \tag{12}$$

$$if \ \frac{V_{C^+}}{V_{Tot}} > (P_{C^+} \times 2) \ \& \ \frac{V_{C^+} - S_{C^+}}{V_{Tot} - R_{Tot}} > P_{C^+} \quad then \quad C^+ = 0.6 \tag{13}$$

$$if \ \frac{V_{C^-}}{V_{Tot}} > (P_{C^-} \times 2) \ \& \ \frac{V_{C^-} - S_{C^-}}{V_{Tot} - R_{Tot}} > P_{C^-} \quad then \quad C^- = 0.3 \tag{14}$$

  The second part to each equation ensures that after assignment the volume is still greater than desired value, otherwise too many units have been supplied (*S*) to other blood groups. The desired volume levels for each type depend on the distribution of blood in the region of the blood bank. For this case we are considering the region of South Africa and the distributions are $P_{O^+} = 39\%$, $P_{O^-} = 7\%$, $P_{A^+} = 32\%$, $P_{A^-} = 5\%$, $P_{B^+} = 12\%$, $P_{B^-} = 2\%$, $P_{C^+} = 3\%$ and $P_{C^-} = 1\%$ [3]. If both conditions hold in any of the equations the profit for that type is doubled. This means that these units will be encouraged to supply less valuable blood groups and essentially stabilize the proportions in the bank.
- Imported units placed in knapsacks do not increase profit. When imported units are added to the knapsack, these transactions must be reversed. As this directly relates to the objective function (1), imported units can decrease the profit such that entire knapsack profits can become negative. This factor encourages cross-matching between blood groups as it is more profitable to add another blood type from the bank rather than an imported one.

The bank also needs to ensure that units of blood are not wasted by passing their expiration date. The bank can efficiently handle this problem by storing the units in the bank in a first-in first-out queue. This means than when units are supplied to satisfy requests, the units are taken from the front of the queue. Similarly, when units are donated into the bank, they are added to the back of the queue.

As one request is satisfied by one unit, the weight w_j of every unit is equal to 1. As the weights of every unit are equal, we can focus on selecting the most profitable units in order to satisfy each request in order to maximize the profit. Various optimizations can then be employed to find the optimal distributions in order to satisfy the requests.

D. Optimization Techniques

Finding the optimal units to place into each knapsack is a difficult task. This is because the profit for each unit changes depending on the number of units selected and the amount of available units in the bank. Therefore optimization algorithms can be applied to implement the Multiple Knapsack Assignment and determine the optimal assignment by finding the largest profit in (2). The simple assignment algorithm does not require such techniques and can be implemented by a basic heuristic. The following optimization techniques implement the Multiple Knapsack Assignment.

1) Genetic Algorithm: Genetic Algorithms (GA) [6] consist of populations of individuals who use some selection, recombination and mutation operators to evolve from one generation to the next. Each individual or chromosome in the population represents a candidate solution to the optimization problem and these evolve toward the optimal solution. The operators used for this algorithm were one-point recombination for crossover, random-point mutation and tournament selection.

2) Adaptive Genetic Algorithm: In order to improve the search ability of the GA an Adaptive Genetic Algorithm (AGA) is defined. This algorithm has the same structure as the previous GA, however the probabilities for crossover and mutation change from generation to generation. Zhang *et al.* [7] proposes an AGA that has adaptive parameters for crossover rates and identifies that using constant parameters settings have the ability to reduce the GAs efficiency and can result in immature convergence. Using adaptive parameter settings according to the individual fitnesses can improve the search ability of the algorithm. The adaptive equations for the implemented AGA are adapted from their work [7].

$$p_c = \begin{cases} p_{c1} - \frac{(p_{c1}-p_{c2})(f'-f_{avg})}{f_{max}-f_{avg}} & \text{if } f' \geq f_{avg}, \\ p_{c1} & \text{if } f' < f_{avg}. \end{cases} \tag{15}$$

$$p_m = \begin{cases} p_{m1} - \frac{(p_{m1}-p_{m2})(f'-f_{avg})}{f_{max}-f_{avg}} & \text{if } f' \geq f_{avg}, \\ p_{m1} & \text{if } f' < f_{avg}. \end{cases} \tag{16}$$

In (15) and (16) the probabilities of crossover and mutation are adjusted dynamically with the values p_{c1} and p_{m1} set higer than the values p_{c2} and p_{m2}.

3) Simulated Annealing Genetic Algorithm: Simulated Annealing (SA) is based on the idea of dynamically altering the probability of accepting inferior solutions [8]. Initially, when the temperature is high, the algorithm accepts worse solutions and has a high exploration rate. As the temperature cools the probability of accepting worse solutions decreases, and this improves the fine-tuning of the algorithm. The algorithm was considered to be merged with the GA because it has the ability to give better solutions for a given number of fitness evaluations and it gives more consistency over many runs [8]. The approach used by Adler [8], was considered for implementation. This defines new operators for the GA, SA-Recombination(SAR) and SA-Mutation(SAM). These operators work the same as in the GA, however they decide whether or not to accept the new solution after the operators are applied depending on the Metropolis Criterion. This is defined as follows:

$$p_{accept} = e^{-c/t} > r \tag{17}$$

c represents the change in the fitness function, whilst t represents the current temperature and r is random number between 0 and 1. The cooling schedule, which is updated every generation, was defined as to allow the temperature to become low enough by the latter stages:

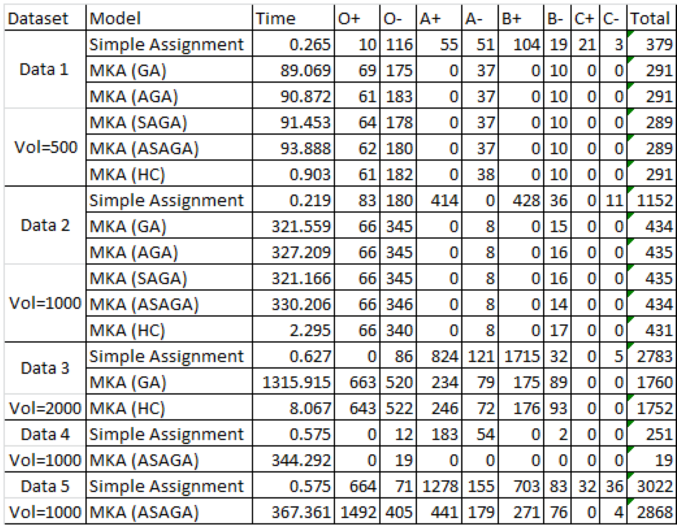
$$t' = t \times 0.95 \tag{18}$$

4) Adaptive Simulated Annealing Genetic Algorithm: The Adaptive Simulated Annealing Genetic Algorithm creates a hybrid algorithm with the desirable properties of both SA and the AGA mentioned above. This algorithm is a straight forward merger of those algorithms.

5) Hill Climbing Algorithm: The Hill-Climbing (HC) algorithm works on a current solution to find the next better solution [6]. In the HC algorithm we use a single chromosome solution. For each new step, the chromosome is mutated and if the fitness of the mutated chromosome is better than the previous chromosome then the mutated chromosome replaces the current chromosome. No crossover or selection operators are involved in HC, making it a simpler version of the GA. This algorithm is also adapted so that the initial solution uses the Simple Assignment Algorithm. This domain knowledge is inserted into the HC algorithm, as the optimal assignment of units is usually only a slight variation of this initial solution.

III. Results and Discussion

The blood banking model was tested using the Simple Assignment as well as the Multiple Knapsack Assignment (MKA) algorithms. These were tested using 5 randomly generated data sets each over period of 90 days. Datasets 1 - 3 are created with initial volumes of size 500, 1000 and 2000, and the amount of requests and donations for each day are both equal to randomly generated amounts between 25-75% of the initial volumes of each blood type. Datasets 4 and 5 have initial volumes of 1000 units, however in dataset 4 the number of donations is between 30-75% and the number of requests is between 25-75%, and conversely dataset 5 has donations between 25-75% and requests between 30-75% each day. The datasets are used to measure how the models perform with different volumes of units as well as how they perform when the ratio of of donations to requests is not even. The parameters used for the population-based meta-heuristics were as follows: population size of 500, number of generations 25% of initial population size, probability of crossover 0.7 and probability of mutation 0.005 $\times$ initial volume. The number of steps for the HC algorithm was set at 10 $\times$ initial volume of the system.



Dataset	Model	Time	O+	O-	A+	A-	B+	B-	C+	C-	Total
Data 1	Simple Assignment	0.265	10	116	55	51	104	19	21	3	379
	MKA (GA)	89.069	69	175	0	37	0	10	0	0	291
	MKA (AGA)	90.872	61	183	0	37	0	10	0	0	291
Vol=500	MKA (SAGA)	91.453	64	178	0	37	0	10	0	0	289
	MKA (ASAGA)	93.888	62	180	0	37	0	10	0	0	289
	MKA (HC)	0.903	61	182	0	38	0	10	0	0	291
Data 2	Simple Assignment	0.219	83	180	414	0	428	36	0	11	1152
	MKA (GA)	321.559	66	345	0	8	0	15	0	0	434
	MKA (AGA)	327.209	66	345	0	8	0	16	0	0	435
Vol=1000	MKA (SAGA)	321.166	66	345	0	8	0	16	0	0	435
	MKA (ASAGA)	330.206	66	346	0	8	0	14	0	0	434
	MKA (HC)	2.295	66	340	0	8	0	17	0	0	431
Data 3	Simple Assignment	0.627	0	86	824	121	1715	32	0	5	2783
	MKA (GA)	1315.915	663	520	234	79	175	89	0	0	1760
Vol=2000	MKA (HC)	8.067	643	522	246	72	176	93	0	0	1752
Data 4	Simple Assignment	0.575	0	12	183	54	0	2	0	0	251
Vol=1000	MKA (ASAGA)	344.292	0	19	0	0	0	0	0	0	19
Data 5	Simple Assignment	0.575	664	71	1278	155	703	83	32	36	3022
Vol=1000	MKA (ASAGA)	367.361	1492	405	441	179	271	76	0	4	2868

Fig. 2. Table of Results for Total Imports and Time

From the entire set of results in Fig. 2. we can observe that the MKA algorithm always results in less units having to be imported from outside the system than the Simple Assignment algorithm. This positive result is clearly illustrated in Fig. 3. We can also observe that the MKA algorithm still performs well regarding datasets 4 and 5 when the ratio between donations and requests is not equal.

In Fig. 4. the number of imports for each blood group can be observed. It is notable how the MKA algorithm greatly reduces the need to import most of the other blood types that were required in the Simple Assignment Algorithm. Although it does reduce the total number of imports, it also results in

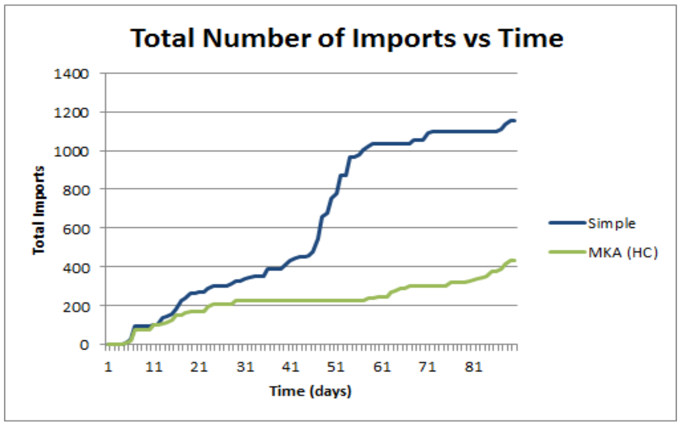


Fig. 3. Total Imports for Both Models on Dataset 2

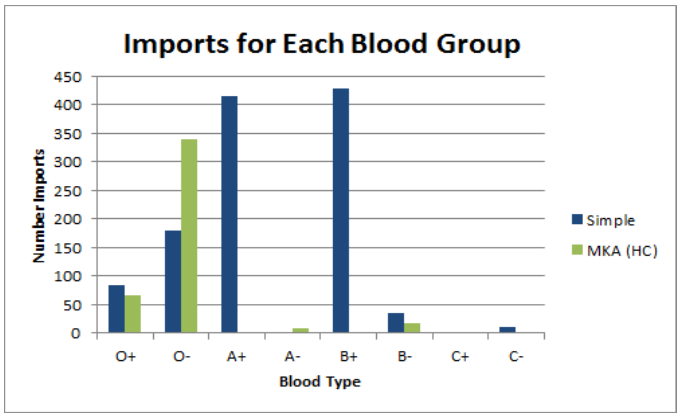


Fig. 4. Imports for Each Blood Group on Dataset 2

the system needing to import more of the blood type O^-. This is also a general trend in all the results in Fig. 2. and it occurs because this blood type is often used for cross-matching and emergency scenarios and this causes it to often run low itself. This is a negative side-effect of this system.

In terms of the algorithms used to implement the MKA, all of these algorithms were capable of converging to the optimal solution and achieving similar results in almost all cases. This can be observed in the results in Fig. 2. for datasets 1 - 2. However, the AGA, SAGA and ASAGA were capable of converging to the optimal solution slightly faster than the GA in most cases and this can be observed in Fig. 5. They also avoid immature convergence and are considered to be more consistent than the standard GA approach in general.

The HC algorithm proved to be the most efficient of all the algorithms as it converges to the optimal solution the fastest and in the quickest time. This is because the HC algorithm deals with only one candidate solution, and for each step it only requires one fitness function evaluation.The HC algorithm is well suited to problems when the optimal solution is near the initial solution and only requires a few adjustments, which is one of the reasons it performs so well. The execution time of the other population-based meta-heuristics also becomes extremely slow as the size of the blood bank increases. This is because of the increase in total fitness function evaluations as well as the genetic operators having to manipulate large

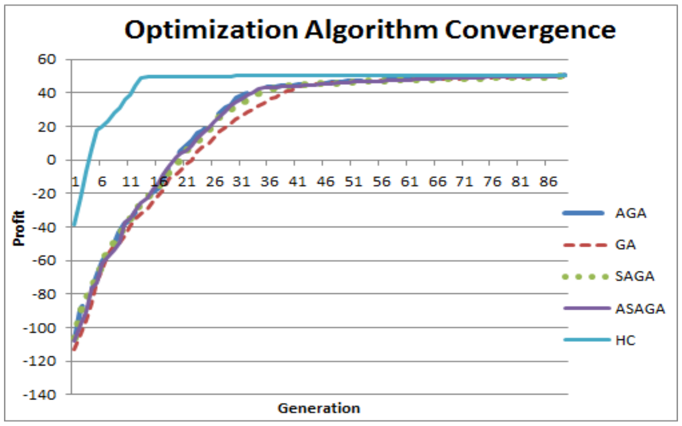


Fig. 5. Convergence Rates for Random Day t on Dataset 2

chromosome strings for crossover and mutation. Therefore, these algorithms are not recommend for large blood bank sizes using this model.

IV. Conclusion and Future Work

We present an efficient model for the management of a blood banking system. An approach for determining the assignments of units is explored using the Multiple Knapsack Problem and this yields positive results. The success of this approach shows encouragement that cross-matching between blood types can be used to minimize the amount of units imported from outside the system when a particular type is scarce. The assignment algorithm can be be implemented by a number of optimization algorithms when the size of the blood bank is small, however the HC algorithm is the only suitable candidate for use with a larger blood bank and it proves to be the best suited algorithm to the model overall.

An area for future improvement will be to implement the Multiple Knapsack Model with considering urgency requests for each blood type. The idea of urgency requests has already proved to improve the management of a blood bank system [2] and this could be used to improve the proposed system further. Another area for future work is to test the results of this system using real data, which needs to be performed before the system can be considered for use in a real world system.

References

[1] J.P.F Charpin, A.O Adewumi (2011), *Blood Assignment*, Technical Report, Mathematics in Industry Study Group (MISG 2011), Jan 9 to 14.

[2] V. De Angelis, N. Ricciardi, and G. Storchi, *Optimizing blood assignment in a donation - transfusion system*, International Transactions in Operational Research, August 2001.

[3] South African National Blood Service, *What's your type*, http://www.sanbs.org.za, [Accessed 13 October 2011].

[4] D. Pisinger, *Algorithms for Knapsack Problems*, Department of Computer Science, University of Copenhagen, Denmark, February 1995.

[5] S. Martello and P. Toth, *Knapsack Problems*, Algorithms and Computer Implementations, University of Bologna, Italy, 1990.

[6] T. Weise, *Global Optimization Algorithms - Theory and Application*, University of Kassel, Distributed Systems Group, Germany, pages 141-149 and 253-258, June 2009.

[7] W. Zhang, W. Chen and Y. Wang, *The Adaptive Genetic Algorithms for Portfolio Selection Problem*, International Journal of Computer Science and Network Security, VOL. 6 No.1, January 2006.

[8] D. Adler, *Genetic Algorithms and Simulated Annealing: A Marriage Proposal*, IEEE International Conference on Neural Networks, pp. 1104-1109, 1993.