Particle Swarm Optimization Algorithm for Optimizing Assignment of Blood in Blood Banking System

Article *in* Computational and Mathematical Methods in Medicine · March 2015

DOI: 10.1155/2015/713898

Martins A. Arasomwan
University of KwaZulu-Natal
11 PUBLICATIONS **78** CITATIONS
SEE PROFILE

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

Hindawi Publishing Corporation
Computational and Mathematical Methods in Medicine
Volume 2015, Article ID 713898, 12 pages
http://dx.doi.org/10.1155/2015/713898

Research Article

Particle Swarm Optimization Algorithm for Optimizing Assignment of Blood in Blood Banking System

Micheal O. Olusanya, Martins A. Arasomwan, and Aderemi O. Adewumi

School of Mathematics, Statistics, and Computer Science, University of Kwazulu-Natal, Private Bag Box X54001, Durban 4000, South Africa

Correspondence should be addressed to Aderemi O. Adewumi; laremtj@gmail.com

Received 9 June 2014; Accepted 20 October 2014

Academic Editor: David A. Elizondo

This paper reports the performance of particle swarm optimization (PSO) for the assignment of blood to meet patients' blood transfusion requests for blood transfusion. While the drive for blood donation lingers, there is need for effective and efficient management of available blood in blood banking systems. Moreover, inherent danger of transfusing wrong blood types to patients, unnecessary importation of blood units from external sources, and wastage of blood products due to nonusage necessitate the development of mathematical models and techniques for effective handling of blood distribution among available blood types in order to minimize wastages and importation from external sources. This gives rise to the blood assignment problem (BAP) introduced recently in literature. We propose a queue and multiple knapsack models with PSO-based solution to address this challenge. Simulation is based on sets of randomly generated data that mimic real-world population distribution of blood types. Results obtained show the efficiency of the proposed algorithm for BAP with no blood units wasted and very low importation, where necessary, from outside the blood bank. The result therefore can serve as a benchmark and basis for decision support tools for real-life deployment.

1. Introduction

Blood is a living tissue of unique medical value to the human body [1]. It is responsible for carrying the substances needed for healthy living to various parts of the body and carries away those not needed as wastes. A normal blood is made up of four different components, namely: red cells, white cells, platelets, and plasma [1], which serve separate functions in human organism and different use in the medical treatment of patients in the hospitals or health clinics. The need for blood every day in the hospitals for various reasons and uses in the treatment of patients has made it become of high demand. As a result, there exits collection sites where blood is collected from donors, processed, and shipped to a Hospital Blood Bank to be stored and be available to meet the demands for transfusions to patients. Handling blood transfusion in hospitals involves some complexities due to blood compatibility issues and stochastic nature of the daily demands for blood. Therefore, assignment of blood can be seen as NP-Hard problem. As a limited resource with limited shelf life, there is need for its efficient management during the process of assigning it to patients from hospitals blood banks. The component of blood considered in this paper is the red cells and it is made up of four main types, namely: A, B, AB, and O. For convenience, blood type AB will be denoted by C in this work. With the Rhesus factor [2], the number of blood types is doubled, resulting in A^+, A^-, B^+, B^-, C^+, C^-, O^+, and O^-. Optimizing the assignment of these components is the focus of this paper.

The assignment problem is one of the basic combinatorial optimization problems in the fields of discrete optimization. Many optimization techniques have been used to solve various assignment problems including vehicle assignment problem, transportation problem, task allocation problem (TAP), and blood assignment problem (BAP) [3–5]. For instance, Jean et al. [3] used PSO to solve the problem of allocating a set of cabs to some customers with the goal of minimizing the distance traveled by the fleet with result that showed that PSO is capable of achieving optimal results. Similarly, a discrete PSO (DPSO) and genetic algorithm (GA) were compared for the problems of finding optimal solution to the allocation of the expected number of people in flooded areas to various types of available vehicle in an evacuation process [6] with DPSO having better performance than GA. DPSO has also shown great performance with different degrees of difficult knapsack problems [7, 8]. In addition, experimental results in [9] showed that DPSO algorithm is highly efficient for solving the multiple knapsack problems (MKP) even with large problem instances. This motivates the investigation of DPSO in solving the BAP defined in this paper.

BAP recently stated attracting interests among meta-heuristics and optimization researchers [2, 10–14] as the sourcing for blood for transfusion purpose is currently posing global challenge. An early work reported in [10] modelled the problem as a multiobjective linear programming model to help determine the best assignment of blood from donors to requests. The model was based on variables that represent the units of blood coming from both within the blood bank and outside the system and assigned available blood on daily basis depending on demand. A case study of Italian Red Cross System in Rome was considered. The model is however incapable of handling global request of different blood types at the same time. In [2] a dynamical system model was developed to handle the management of blood in a blood bank but without considering Rhesus factors. The model in [2] was improved upon in [11] where the authors modelled BAP as a MKP and compared the performances of five different (meta)heuristics in solving the problem. The goal was to minimize the amount of blood unit imported from outside the blood banking system. To further improve on the work in [11], a new cross and mutation techniques were introduced into the algorithm in [12] with the result that GA was reported as more efficient than other compared techniques. In [13], two local search optimization techniques, namely, dynamic programming and greedy randomized adaptive search procedure (GRASP) were tested for the BAP with GRASP performing better than dynamic programming. Tabu search and simulated annealing with their hybrid were also tested on BAP [14] where the hybrid showed better results than the individual techniques.

The problem of assigning blood in blood bank system principally involves attending to requests for blood and accepting donated blood from donors with the lifespan of each blood type of critical considerations. These activities are on daily basis and stochastic in nature. In situations where there is not enough volume of compatible blood type(s) to meet demands, such blood types are imported from other sources outside the blood bank system and this could be very expensive. All these put together show that the objective of assignment of blood in blood banks is to meet the daily demand for blood and to minimize the volume as well as the number of blood types imported from external source(s). Also, the volume of expired blood should be minimized. There is no report of the application of any swarm intelligence algorithms to this new domain of BAP. This paper is therefore set to investigate the application of a well-known swarm intelligence algorithm, particle swarm optimization (PSO), to the BAP.

Table 1: Blood type compatibility with Rhesus factor.

Receiver	Donor							
	A^+	A^-	B^+	B^-	C^+	C^-	O^+	O^-
A^+	YES	NO	NO	NO	NO	NO	YES	YES
A^-	NO	YES	NO	NO	NO	NO	NO	YES
B^+	NO	NO	YES	YES	NO	NO	YES	YES
B^-	NO	NO	NO	YES	NO	NO	NO	YES
C^+	YES	YES	YES	YES	YES	YES	YES	YES
C^-	NO	YES	NO	YES	NO	YES	NO	YES
O^+	NO	NO	NO	NO	NO	NO	YES	YES
O^-	NO	NO	NO	NO	NO	NO	NO	YES

PSO was introduced in [15, 16]. It is a fast and efficient technique used in solving complex and simple optimization problems. This technique is nature-inspired and draws inspiration from social behavior of animals, like birds and fishes. Thus, it belongs to the swarm intelligence group of evolutionary computation techniques. From the period of its inception, it has been used by researchers to solve a wide range of simple and complex optimization problems [17–24].

In the sections that follow, Section 2 gives the description of BAP considered in this paper. Section 3 describes the method that was applied in solving the problem. Section 4 focuses on the numerical simulations conducted which comprises the analysis and discussion of results. Finally, Section 5 concludes the paper and identifies some possible areas of future research.

2. Problem Description for Blood Assignment

The blood bank stores and issues the appropriate blood units to satisfy transfusion requests. On each day the bank receives a random number of transfusion requests for each blood type and each request for a random number of units. Once a request is received, the appropriate number of units of that type is removed from the bank upon successful cross-matching. We will define demand to be the number of units requested and usage to be the number of units transfused. Any units which are not used within their shelf life are considered expired and are discarded from the bank. Normally, a patient should be transfused his or her own blood type when there is a need for it. However, there could be situations when the blood type in the bank may not be enough to meet the blood units of the type requested. In order to address such challenges compatible blood types are supplied instead, where possible. The compatibility between blood types is presented in Table 1 (in this paper, blood types "C^+" and "C^-" refer to AB^+ and AB^- throughout and are thus chosen for typo simplicity.). In the table, "YES" means that there is compatibility between the donor and receiver, while "NO" means there is no compatibility. Also, in the table, it is evident that O^- is the universal donor, while C^+ is the universal receiver. In Republic of South Africa, 86% of the population has positive Rhesus factor, while 14% have negative Rhesus factor leading to the repartition presented

TABLE 2: Proportion of blood types in South Africa[2].

Blood type	A^+	A^-	B^+	B^-	C^+	C^-	O^+	O^-
Proportion (%)	32	5	12	2	3	1	39	7

[2]Source: South African National Blood Service—http://www.sanbs.org.za/index.php/donors/what-s-your-type.

in Table 2 [2, 11–14]. Further description of the BAP can be found in [2, 10, 12].

From the preceding description, it is evident that the assignment of blood in blood banks is a complex problem. Furthermore, it becomes more complex because blood products have limited shelf life. As a result, there is therefore the need for algorithms that could efficiently assign available blood resources in the blood bank to receivers, in order to minimize the blood units imported from external source(s) into the blood bank and at the same time minimize wastage of blood unit because of their limited shelf life.

2.1. Objective Function. The objective function of the BAP is given in (4). The aim is to minimize the volume of red blood cells imported into the bank over a period of n days:

$$\min \sum_{t=1}^{n} I_{\text{Total}}(t), \tag{1}$$

where $I_{\text{Total}}(t) = I_{O^+}(t) + I_{O^-}(t) + I_{A^+}(t) + I_{A^-}(t) + I_{B^+}(t) + I_{B^-}(t) + I_{C^+}(t) + I_{C^-}(t)$ and $I_{O^+}, I_{O^-}, I_{A^+}, I_{A^-}, I_{B^+}, I_{C^+}$, and I_{C^+} are volumes of the respective blood types imported in day t.

2.2. Constraints. Description of different constraints that are to be satisfied in implementing the proposed algorithm to solve the BAP is as follows.

(i) The total volume of blood requested must be satisfied for each day, either by using the ones in the bank or by importing blood from external sources.

(ii) The volume of blood assigned each day must not exceed the total volume available in the bank.

(iii) The minimum volume of each blood type must not be less than zero.

(iv) Each unit of blood types must not exceed 30 days of shelf life; otherwise, it should be incinerated.

2.3. Variables. Explained below are the descriptions of possible variables that can be used in the algorithm and could be represented using any meaningful variable names:

(i) volume of blood types in storage at day t;

(ii) proportion of blood types in the population of the region considered;

(iii) donations of blood into the bank at day t;

(iv) requests for unit blood at time t;

(v) volume of incinerated expired blood at day t;

(vi) volume of blood supplied at day t;

(vii) volume of blood imported from external source(s) into the bank at day t.

2.4. Updating Blood Volume. On daily basis, the volumes of the blood types are updated relative to the volumes supplied, donated, and expired as stated in (15).

2.5. Assumptions. The model was defined with the following assumptions.

(i) There is sufficient supply from the external source(s) of blood, when units are required to be imported.

(ii) The desired level of the volumes of each blood type is relative to the proportional of blood distribution in the region under consideration.

(iii) If emergency arises, no optimization may be performed, but patients straightaway receive O^- blood.

(iv) The validity date of the blood products is 30 days.

3. Methodology

In the daily management of the blood bank, the demand and supply of blood units vary; thus, there is a need for an effective method to determine the assignment of blood units for the purpose of good management. The assignment of blood units would be considered optimal if there is no importation of units from external sources; however, this may not be a possibility. Therefore, an algorithm that could be able to assign blood units relative to the available volume of blood units and volume requested with no or minimum units imported from outside the blood bank will suffice.

Three techniques are combined to provide solution to the BAP considered and thus to determine (near) optimal assignment of blood units relative to demands. They are particle swarm optimization which is the optimization technique used, multiple knapsack assignment technique which is used to handle the cross-matching of blood types in order to satisfy requests and stabilize the proportions of blood types stored in the bank, and queuing technique which is used to monitor the expiration date of the units of each blood type. Also described in this section is a bottom-up technique that was used to assign the blood units before importing any blood types from external source(s).

3.1. Particle Swarm Optimization Technique. The PSO as a population-based and stochastic technique needs a swarm of particles to carry out its optimization process [15, 16]. Using the search range defined for a problem, values are randomly generated for the decision variables. These values are then used to randomly distribute the particles in the solution search space before the technique begins its iterative process. During the optimization process, each particle communicates its new discoveries to others and this in turn determines subsequent moves of the particles in the search space. In each attempt of iteration, each particle makes use of two major pieces of information: its personal experience and the experiences of reachable neighbours to guide its search. Furthermore, the objective function of the problem being optimized is used to evaluate the quality of the discovery of each particle. Given an n-dimensional space, each particle is characterized by the position vector $X_i = (x_{i1}, \ldots, x_{in})$ and

```
Begin PSO Algorithm
  Input:   $f$: the function to optimize
           $s$: the swarm size
           $d$: the problem dimension
           $X_{\min}, X_{\max}$: decision variable search range
           $V_{\min}, V_{\max}$: particle velocity limits
  Output:  $x^*$: the best particle position found (global best)
           $f^*$: the best fitness value found
  Initialize: position $x_i = (x_{i1}, \ldots, x_{id})$ and velocity $v_i = (v_{i1}, \ldots, v_{id})$, for all particles in problem space
  evaluate $f(x_i)$ in $d$ variables and get $pbest_i$, $(i = 1, \ldots, s)$
  $gbest \leftarrow$ best of $pbest_i$
  While stopping criteria is false do
    Compute inertia weight ($\omega$) if it is not a constant
    Repeat for $s$ times
      Repeat for $d$ times
          update $v_i$ for particle using (2)
          validate for velocity boundaries using Algorithm 3
          update $x_i$ for particle using (3)
          validate for position boundaries using Algorithm 2
          compute $f(x_i)$
      End Repeat for $d$
      compute $f(x_i)$
      obtain new $pbest_i$
      If $f(x_i) < f(pbest_i)$ then $pbest_i \leftarrow x_i$
      If $f(x_i) < f(gbest)$ then
        $gbest \leftarrow x_i$
        $f(gbest) \leftarrow f(x_i)$
      end if
    End Repeat for $s$
  End while
  $x^* \leftarrow gbest$
  $f^* \leftarrow f(gbest)$
  Return $x^*$ and $f^*$
End PSO Algorithm
```

Algorithm 1

the velocity vector $V_i = (v_{i1}, \ldots, v_{in})$. When the particles are searching for optimum solution in the search space, their velocities and positions are updated using (2) and (3), respectively:

$$V_i(t+1) = \omega V_i(t) + c_1 r_1 (P_i - X_i) + c_2 r_2 (P_g - X_i), \quad (2)$$

$$X_i(t+1) = X(t) + V_i(t+1). \quad (3)$$

In (2), P_i and P_g are vectors representing the ith particle personal best position and swarm global best position, respectively; r_1 and r_2 are random numbers in the interval $[0, 1]$, while c_1 and c_2 are acceleration coefficients called cognitive and social scaling parameters which determine the extent of the random forces in the direction of P_i and P_g. The parameter t represents iteration index, while ω is the inertia weight which was introduced in [25]. It regulates the particle's velocity and helps to balance the global and local search abilities of PSO. In the original PSO algorithm, though not explicitly added to (2), the parameter ω implicitly had a value of 1. A general framework of PSO algorithm for continuous problems is presented in Algorithm 1.

Although the PSO was initially developed for continuous optimization problems, it has in several occasions been customized for discrete problems [19, 20]. Customizing PSO for discrete problems was initially proposed by Kennedy and Eberhart in [26] where they defined trajectories and velocities of particles in terms of changes of probabilities, where the particles move in a state space restricted to 0 and 1 on each dimension, with a certain probability computed using the sigmoid limiting transformation as shown in

$$x_{ij} = \begin{cases} 1, & \text{if } \mu \le S(v_{ij}) \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

where x_{ij} is the value in the jth dimension of the position of particle i, v_{ij} is the value in the jth dimension of the velocity of particle i, μ is a value selected from a uniform distribution in $[0, 1]$, and $S(v_{ij})$ is as given in

$$S(v_{ij}) = \frac{1}{1 + e^{-v_{ij}}}. \quad (5)$$

In this paper, some customization was made to the PSO algorithm before it was applied to the BAP. The different parts that were customized are as follows.

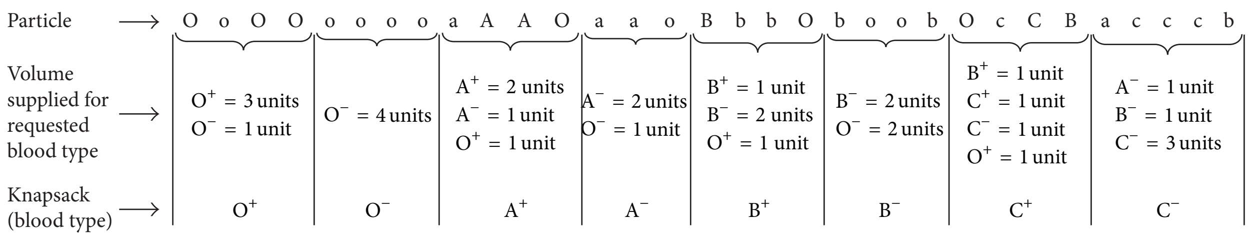


Figure 1: Exemplified particle representation in the proposed discrete PSO algorithm.

Table 3: Transformed particle used for computation in (2).

0.07	0.04	0.07	0.07	0.04	0.04	0.04	0.04	0.07	0.15	0.15	0.07	0.07	0.07	0.04	0.15	0.07	0.07	0.07	0.07	0.04	0.04	0.07	0.07	0.15	0.3	0.15	0.07	0.15	0.15	0.15	0.07

Table 4: Values and proportions of the different blood types.

Blood type	A^+	A^-	B^+	B^-	C^+	C^-	O^+	O^-
Value	4/27 = 0.15	2/27 = 0.07	4/27 = 0.15	2/27 = 0.07	8/27 = 0.3	4/27 = 0.15	2/27 = 0.07	1/27 = 0.04
Proportion (%)	32	5	12	2	3	1	39	7

3.1.1. Particle Representation. Each particle is encoded by a long string of characters representing each of the blood types. The string is divided into groups, with each group representing the content of each of the 8 knapsacks for each blood type. The characters in each group show which units of blood have been placed into each knapsack. Furthermore, each character represents 1 unit of positive blood type if it is uppercase (e.g., "A" for "A^+"), but represents 1 unit of negative blood type if it is lowercase (e.g., "a" for "A^-"). An example is shown in Figure 1.

The example in Figure 1 shows an interpretation of the representation of a typical particle in the swarm. The particle is of size (dimension) 32 and it represents the combination of all the units of requested blood types. The first 4 dimensions represent the number of units (i.e., 4) of O^+ that was requested and the supply is made up of 3 units of O^+ and 1 unit of O^-. The last 5 dimensions also represent the number of units (i.e., 5) of C^- that was requested and the supply is made up of 1 unit of A^-, 1 unit of B^-, and 3 units of C^-. The same idea of interpretation applies to the remaining dimensions of the particle. In the course of implementation of the technique, the dimension of the particles and the size of the various knapsacks vary each day relative to the changing requests received by the blood bank.

3.1.2. Transformation of Particles. From Figure 1, it is clear that the values of the dimensions of the particle are alphabets. Therefore, X_i, P_i, and P_g will contain alphabets too and these cannot be directly used to compute $V_i(t+1)$ in (2). For the velocity of each particle to be computed, these alphabets should be converted to numbers. There may be different methods to do this; however, the method used was to replace the alphabet (blood type) in each dimension of the particle with its value (see Table 4). In other words, the particle represented in Figure 1 becomes as represented in Table 3.

P_i and P_g are also transformed likewise, depending on the contents of their respective dimensions; hence, (2) can be computed to obtain a value for $V_i(t+1)$.

3.1.3. Obtaining the Positions of Particles. The value obtained for $V_i(t+1)$ is then transformed using (5) to get a value between 0 and 1. This value is then used to determine the blood type to be imported into the blood bank when there is a shortage of blood to meet requests. In other words, the different compatible blood types (including the blood type requested) that could be used to meet the request are given equal chances to be imported from external source(s) into the blood bank. This means that if there are two compatible blood types, each is given 50% chance to be imported, if there are four compatible blood types, each is given 25% chance of being imported, and so forth. Before the beginning of next iteration, each particle is reverted back to a string of alphabets of blood types representing the new potential solution.

3.2. Multiple Knapsack Model. The multiple knapsack problems involve selecting any of n items and packing them into m knapsacks of different capacity c in order to obtain the largest sum of profit [11]. The standard model for multiple knapsack problems is stated in (6)–(9). Consider

$$\text{maximize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} p_j x_{ij} \tag{6}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} w_j x_{ij} \le c_i \quad i = 1, \dots, m, \tag{7}$$

$$\sum_{i=1}^{m} x_{ij} \le 1 \quad j = 1, \dots, n, \tag{8}$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \dots, m; \; j = 1, \dots, n. \tag{9}$$

Equation (6) is the objective function, which is to maximize the profit of the number of items placed in all the knapsacks; (7) is the constraint that ensures that the capacity of each knapsack is not exceeded; (8) is the constraint that ensures that each item can only be placed in any of the knapsacks once

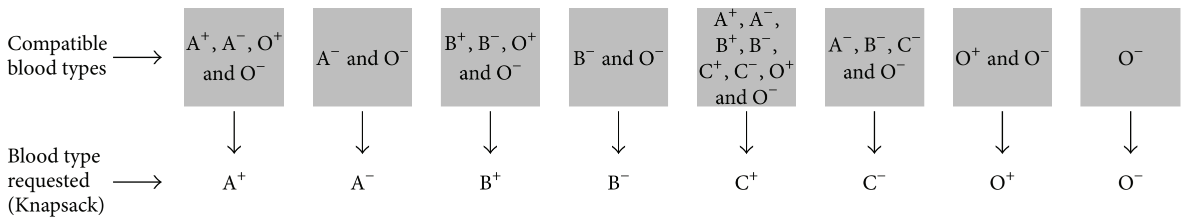


Figure 2: Representing blood type compatibilities as knapsack problem.

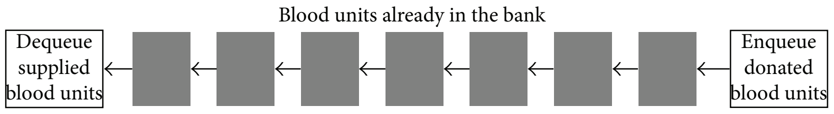

Blood units already in the bank
Dequeue supplied blood units
Enqueue donated blood units

Figure 3: Representation of how blood units are stored in the blood bank.

and (9) indicates whether an item j is placed into knapsack i or not.

Application of the Technique from Multiple Knapsack Problems (MKP). Inspired by the various successes in the application of MKP to different problems, the technique was applied to the BAP considered in this paper with some modifications. Equations (10)–(13) represent the way MKP was applied. In the case of the assignment of blood, units of blood types stored in the bank are the items, while different requests for the blood types are the knapsacks and the number of blood types compatible with each blood type is the capacity of each knapsack. Consider

$$\text{minimize} \quad \sum_{i=1}^{m}\sum_{j=0}^{n} p_j x_{ij} \tag{10}$$

$$\text{Subject to} \quad T = S + \sum_{j=0}^{n} p_j \tag{11}$$

$$x_{ij} \in \{0,1\} \quad i = 1,\ldots,m; \; j = 1,\ldots,n \tag{12}$$

$$S, x_{ij} \geq 0, \quad i = 1,\ldots,m; \; j = 1,\ldots,n, \tag{13}$$

where T is the total units of blood supplied from the blood bank and/or imported from external source(s), S is the total unit of blood supplied from the bank in each day, p_j is units (or volume) of blood type j coming from external sources, m is the total number of requests (number of knapsacks) to be met, and n is the total number of blood types available in the bank for supply (meet requests).

Equation (11) certifies that the total blood units supplied are given by the amount of blood units from the bank and external sources. Equation (12) shows whether there is a supply of blood "1" or not "0." Equation (13) is a nonnegativity constraint; that is, all the values S and x_{ij} are positive integers.

In applying this technique to the BAP, some additional constraints are to be satisfied. These constraints are stated as follows.

(i) In each day, there are 8 sets of requests for the various blood types (A^+, A^-, B^+, B^-, C^+, C^-, O^+, and O^-) which correspond to 8 knapsacks.

(ii) To satisfy requests, units of blood within the bank or imported from external source(s) must be placed in each knapsack.

(iii) Each knapsack can only be satisfied with a compatible unit of blood. This is illustrated in Figure 2.

It should be noted that the weight w_j of every unit of blood of any of the blood type is equal to 1. As a result, selecting the most profitable units that would satisfy each request in order to maximize profit is of uttermost interest.

3.3. Queuing Technique. In the daily management of the blood bank, it is ensured that the shelf life of each unit of blood is not exceeded in order to minimize wastage. An efficient way which can be monitored is to use the queue technique. Queue is a list-like structure that provides restricted access to its elements. Elements may only be inserted at the back (enqueue) and removed from the front (dequeue). It implements the First-In, First-Out (FIFO) operations; thus, queues release their elements in order of arrival. This means that when units of blood are supplied from the bank, they are dequeued but enqueued when units of blood are donated into the bank. This is illustrated in Figure 3, using a single-linked list. It should be noted that each of the blood type is represented separately.

Represented in (14)–(16) is the way the stack in Figure 3 is operated in the algorithm:

$$S_j = B_{\text{tot}} - \bigcup_{i=0}^{r} R_{ij} \quad j = 1,\ldots,n \tag{14}$$

$$B_{\text{tot}}(t+1) = B_{\text{tot}}(t) \cup \bigcup_{x=0}^{d} D_{xy} \quad y = 1,\ldots,m \tag{15}$$

$$R_{ij}, D_{xy} \geq 0. \tag{16}$$

In (14) and (15), S_j is the total blood units of type j supplied by the bank, r is the total number of requests of all the blood

Table 5: Various values used for defining the different datasets used in the simulation.

Dataset	Initial total volume of blood in bank	Running time (days)	Upper and lower bounds for requests (%)	Upper and lower bounds for donations (%)	Remark
1	500	90	[25, 75]	[25, 75]	These datasets were used to test the algorithm with different initial volume
2	1000	90	[25, 75]	[25, 75]	
3	2000	90	[25, 75]	[25, 75]	
4	1000	90	[25, 75]	[30, 75]	These datasets were used to test the algorithm when the ratios of requests to donations are unequal
5	1000	90	[30, 75]	[25, 75]	
6	500	365	[25, 75]	[25, 75]	These datasets were used to test the volume of blood units that will expire when the bank is run for a year
7	1000	365	[25, 75]	[25, 75]	

types, R_{ij} is the total blood units of type j requested, B_{tot} is the total units of all the blood types in the bank, t represents the current day of blood bank operation, d is the total units of all the blood types donated, and D_{xy} is the total blood units of type y donated.

Expiration of Blood Units. As mentioned earlier, the queue is used to monitor the shelf life of each unit of the different blood type. Any of the blood units of a particular blood type that exceeds its shelf life are removed from the queue. This is represented in (16). Consider

$$B_{tp}(t+1) = B_{tp}(t) - \bigcup_{i=1}^{z} (E_i, f), \quad \forall f > \text{life}, \qquad (17)$$

where $tp \in \{A^+, A^-, B^+, B^-, C^+, C^-, O^+, O^-\}$ is the blood type, B_{tp} is the total blood units of type tp on queue, t is the current day of operation, and E_i is the expired blood unit i if its life, f, on the queue is greater than its shelf life, life.

3.4. Bottom-Up Technique. Some blood types are compatible while others are not, as shown in Figure 1. Thus, when there is a request for some units of a particular blood type, the request could be met using the same blood type if there are enough of its units in the bank. If there are not enough units (shortage) of the type requested, available units of compatible types are used to meet up the remaining units. In this case, a bottom-up approach was used in the course of assigning blood to meet the request. This means that, whenever blood is to be assigned, the requested blood type is considered first; when there is a shortage the next available compatible blood type is assigned, and so forth. Blood type O^- is the last option, being the universal donor; if it is not available, then blood will be imported into the bank from external source(s) relative to the request. This approach is exemplified as follows.

Given that typREQ is the number of units requested of blood type typ and typVOL is the number of units of the same blood type that is available in the bank to meet the request, if $typ\text{REQ} > typ\text{VOL}$, then there is a shortage of the blood type to meet the request.

4. Numerical Simulations

The proposed PSO algorithm proposed for solving the considered problem was investigated, while implementing the modified multiple knapsack problem, with a number of simulations with different datasets. As a result of the nonavailability of real-life data, randomly generated data were used. The stochastic nature of blood donation and requests in real-life scenario informed the usage of the fictitious randomly generated data. Whenever real-life data becomes available, they can be substituted for the fictitious data. The application software was developed in Microsoft Visual C# programming language.

4.1. Data Generation. All the data used were randomly generated based on the uniform random number generator. Given a specified (by the user) upper and lower bounds, random volume of requests and donation for each blood type is computed in each day. This provided the opportunity of testing the blood assignment system developed under various scenarios of varied volumes of requested and donated-blood types. Furthermore, an initial volume of all the blood types in the bank was also specified and used to calculate respective volume for each blood type using some blood type proportion. In this case, the blood type proportion in South Africa was used (see Table 1). All the parameters and their respective values as used in the algorithm to generate the testing data are presented in Table 5.

4.2. Parameter Setting. Presented in this section are the definitions of all the parameters used in the course of implementing the proposed model and algorithm. In Table 4, the different blood types and their respective value and proportion are presented. In Table 4, "Value" was computed based on the information in Figure 1. The numerator is the number of compatible blood types that can be received by the respective "Blood type," while the denominator is the sum of all the compatible blood types.

For the PSO technique, the values for c_1 and c_2 were, respectively, set to 1.7 and parameter ω was set to 0.715, as recommended in [27]. The parameters r_1 and r_2 were randomly generated using the uniform random number generator. A swarm size of 50 particles was used and the maximum number of iteration was set to 1000.

4.3. Results and Discussion. Presented in Tables 6, 7, 8, 9, 10, 11, and 12 are the average blood units that were available in, requested for, supplied by, and imported into the blood

Table 6: Average volume of blood units during a runtime of 90 days with 500 blood units as initial volume (using dataset 1).

Average volume	O^+	O^-	A^+	A^-	B^+	B^-	C^+	C^-
Available in bank	114.06	17.36	109.57	19.47	41.63	8.31	21.02	4.96
Requested	66.79	11.87	54.43	8.41	19.87	3.31	5.08	1.77
Supplied	65.33	11.97	54.32	8.49	19.87	3.42	5.08	1.57
Imported into bank	0.42	1.03	0.00	0.01	0.00	0.01	0.00	0.00

Table 7: Average volume of blood units during a runtime of 90 days with 1000 blood units as initial volume (using dataset 2).

Average volume	O^+	O^-	A^+	A^-	B^+	B^-	C^+	C^-
Available in bank	231.34	34.59	216.72	40.36	80.79	12.56	45.67	15.77
Requested	134.59	23.87	109.73	16.88	40.10	6.68	10.23	3.31
Supplied	131.66	24.19	109.47	17.09	39.96	6.81	10.23	2.97
Imported into bank	0.90	2.08	0.00	0.02	0.00	0.00	0.00	0.02

Table 8: Average volume of blood units during a runtime of 90 days with 2000 blood units as initial volume (using dataset 3).

Average volume	O^+	O^-	A^+	A^-	B^+	B^-	C^+	C^-
Available in bank	460.20	69.17	430.09	80.11	159.83	26.31	90.77	30.54
Requested	267.53	47.46	218.10	33.68	79.71	13.24	20.42	6.54
Supplied	261.86	48.09	217.62	33.91	79.47	13.67	20.42	5.90
Imported into bank	1.69	4.01	0.00	0.06	0.00	0.00	0.00	0.00

bank over the period of 90 and 365 days, using all the datasets. Figures 4–7 show the various curves associated with the management of the blood bank as observed during the simulations.

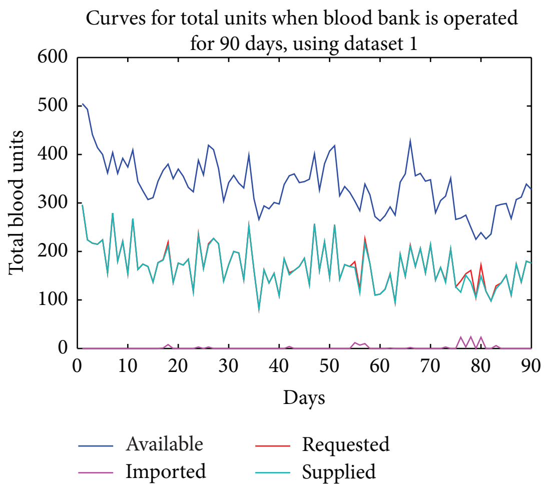
From the results, it is evident that managing the blood bank with each of the datasets over the number of specified periods (90 and 365 days) led to average importation of very few blood units from external sources. The importation became necessary when there were shortages of compatible blood types to meet requests. In Table 10, A^+ blood type was imported into the blood bank when dataset 5 was used (with unequal ratio of requests to donations of blood units). For A^- blood type, apart from Table 9 (using dataset 4), some of its units were imported into the bank when the other datasets were used and the average number of units imported was in increasing order with small variations. Using any of the datasets, no units of B^+ were imported. Some units of B^- were imported in Tables 6, 10, 11, and 12 (using datasets 1, 5, 6, and 7). No C^+ blood type was imported into the bank using any of the datasets. Some units of C^- blood type were imported in Tables 7, 11, and 12 (using datasets 2, 6, and 7).

Much importation was observed with O^+ and O^- blood types, but with O^-, being a universal donor, the importation was more compared to others. Also, when there is a shortage of O^- type, no other blood types could be used as alternative(s) for it and thus it must be imported into the bank. The very low level of average importation of blood units from external source(s) is an indication that the bank was efficiently managed. This efficient management was a result of combining the efforts of the multiple knapsack problem technique used to implement the cross-matching between blood types and PSO technique used to spread importation of blood types (where necessary) in assigning blood types to meet requests. Using all the tested datasets, no blood units exceeded their shelf life; thus, no wastage of blood products was experienced. The reason for this is that the cross-matching method helps to quickly use the older blood units which have been on queue to meet requests while the new ones donated into the blood bank are enqueued.

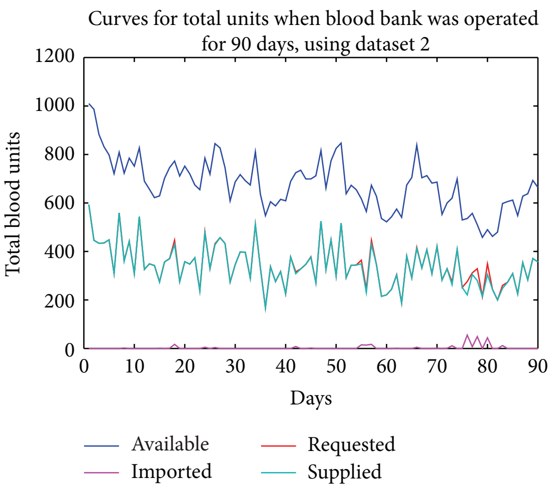
Presented in Figures 4 and 5 are the curves showing the various volume levels of the different blood types in the blood bank when the bank was operated for 90 and 365 days, respectively. The curves show the corresponding total units of blood available, requested, supplied, and imported in each day of operating the bank. In figures, it is evident that minimal units of blood were imported from external source(s) into the bank. In the first few days no units were imported, but between 75th and 85th days a higher number of blood units were imported because the number of units requested was higher than the available blood units relative to the blood types requested. From the graphs, it appears that at the point of importation the volume of available blood in the bank was high and there could be the question, why should there be importation? The reason for importation is that the entire volume of blood in the bank comprised all the units of all blood types and the other units of blood available at the point of importation were not compatible with the blood types requested; therefore, they could not be supplied or used to meet the requests.

Also, in Figure 5, the curves show the corresponding total units of blood available, requested, supplied, and imported in each day of operating the bank for a period of 365 days. In this case, the total units of blood imported (though little) were higher than when the bank was operated for 90 days. In the graphs, it is observed that the highest units of blood were imported between the 340th and 350th days. At this period, some of the available blood types were already exhausted and there were needs for importation of blood units from external source(s) to meet the requests.

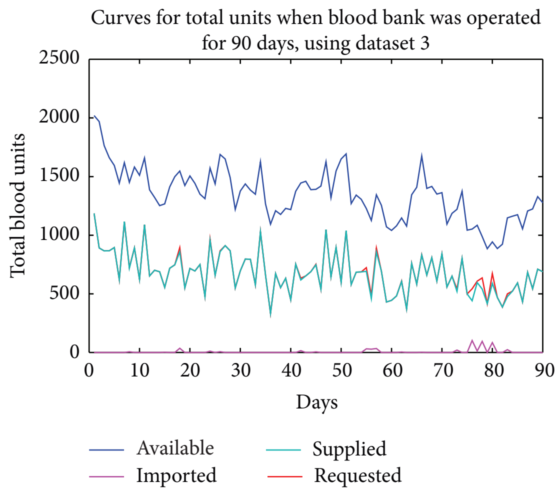
Presented in Figure 6 are the curves showing the number of blood units imported when the algorithm was tested with all the datasets. The highest importation of blood units occurred when dataset 4 was used (Figure 6(a)) and when dataset 7 was used (Figure 6(b)); these occurred around the 80th and 350th days, respectively. In Figure 7, the ratios of total units of blood imported to total units of blood requested for both 90 and 365 days of operating the blood bank are presented. For the graphs, the algorithm appears not to have performed very well with datasets 5 and 7.



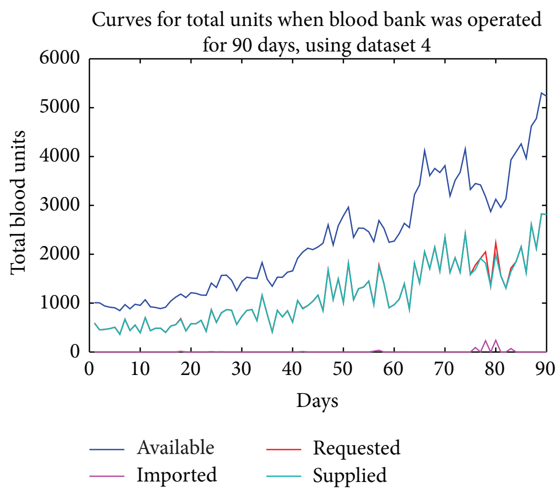
(a)



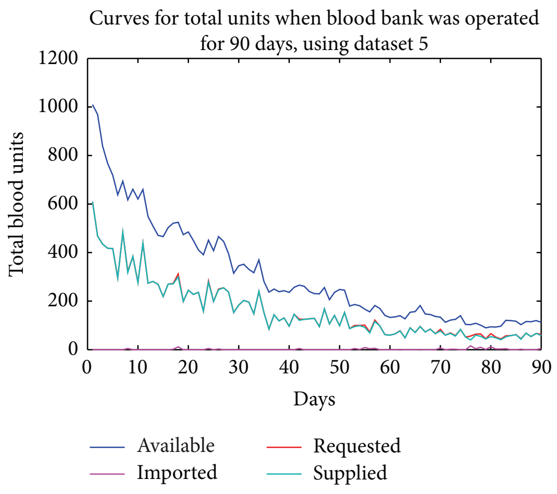
(b)



(c)



(d)

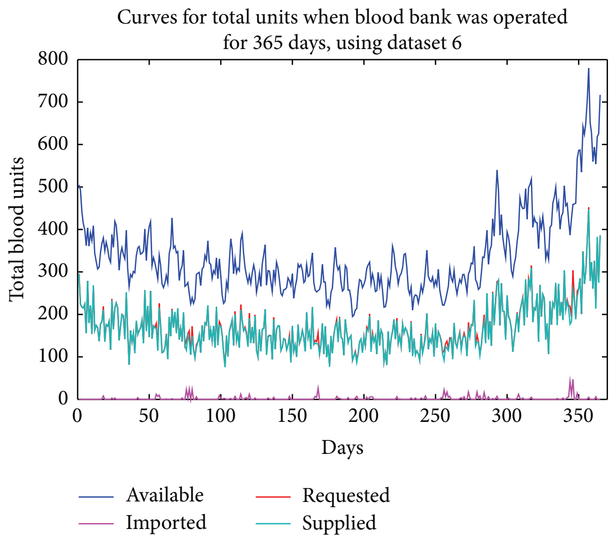


(e)

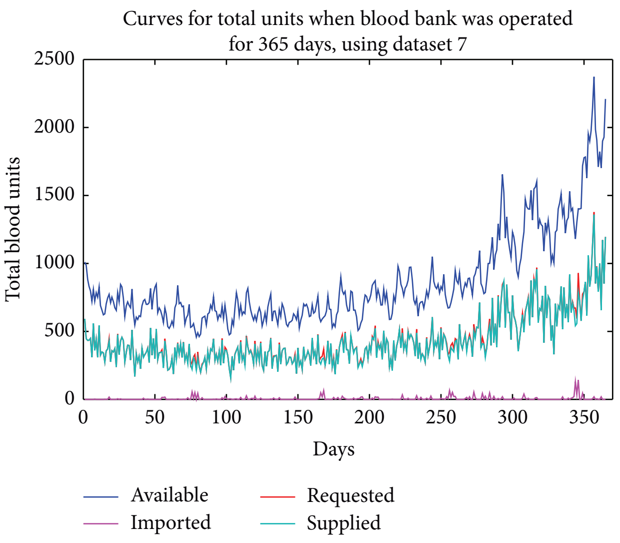
Figure 4: Various curves showing the different blood levels in the blood bank over 90 days, using datasets from 1 to 5.

Table 9: Average volume of blood units during a runtime of 90 days with 1000 blood units as initial volume (using dataset 4).

Average volume	O^+	O^-	A^+	A^-	B^+	B^-	C^+	C^-
Available in bank	759.21	126.86	773.10	146.51	260.02	46.24	112.67	62.60
Requested	449.09	80.66	372.94	56.67	135.39	22.91	34.90	10.08
Supplied	439.63	82.37	372.94	56.63	135.39	23.17	34.90	9.59
Imported into bank	2.29	5.72	0.00	0.00	0.00	0.00	0.00	0.00



(a)



(b)

Figure 5: Importation rate of blood units using datasets 6 and 7.

Table 10: Average volume of blood units during a runtime of 90 days with 1000 blood units as initial volume (using dataset 5).

Average volume	O^+	O^-	A^+	A^-	B^+	B^-	C^+	C^-
Available in bank	108.80	15.39	82.01	14.78	41.89	6.63	29.31	6.51
Requested	64.07	11.33	51.89	8.11	18.93	3.12	4.79	1.69
Supplied	63.24	11.24	51.52	8.07	18.84	3.28	4.79	1.50
Imported into bank	0.33	0.91	0.02	0.17	0.00	0.01	0.00	0.00

5. Conclusion

The problem of assigning of blood units by blood banks to meet requests for blood transfusion in hospitals has been considered in this paper and an efficient method of handling this problem was proposed. The method that was proposed combined the efforts of the PSO technique and multiple knapsack problem. The multiple knapsack problem was modified to reflect some additional constraints that needed to be satisfied for the blood bank to be efficiently managed. Using the queue, multiple knapsack problem, and optimization (PSO) techniques, the total units of blood types imported into the bank were greatly minimized and no wastage was experienced.

Table 11: Average volume of blood units during a runtime of 365 days with 500 blood units as initial volume (using dataset 6).

Average volume	O^+	O^-	A^+	A^-	B^+	B^-	C^+	C^-
Available in bank	116.71	16.04	88.33	14.69	54.68	6.43	28.35	10.55
Requested	66.05	11.79	53.84	8.37	20.01	3.36	5.07	1.72
Supplied	65.27	11.77	53.04	8.38	19.89	3.37	5.07	1.61
Imported into bank	0.41	1.03	0.00	0.30	0.00	0.05	0.00	0.01

Table 12: Average volume of blood units during a runtime of 365 days with 1000 blood units as initial volume (using dataset 7).

Average volume	O^+	O^-	A^+	A^-	B^+	B^-	C^+	C^-
Available in bank	286.50	39.22	220.99	35.68	140.24	14.68	80.41	43.34
Requested	169.22	30.23	137.99	21.49	51.36	8.65	12.93	4.32
Supplied	166.98	30.05	135.69	21.36	51.05	8.58	12.93	4.09
Imported into bank	1.22	3.08	0.00	0.98	0.00	0.17	0.00	0.01

Only the red blood cells were considered in the paper and no real-life data were used but randomly generated data.

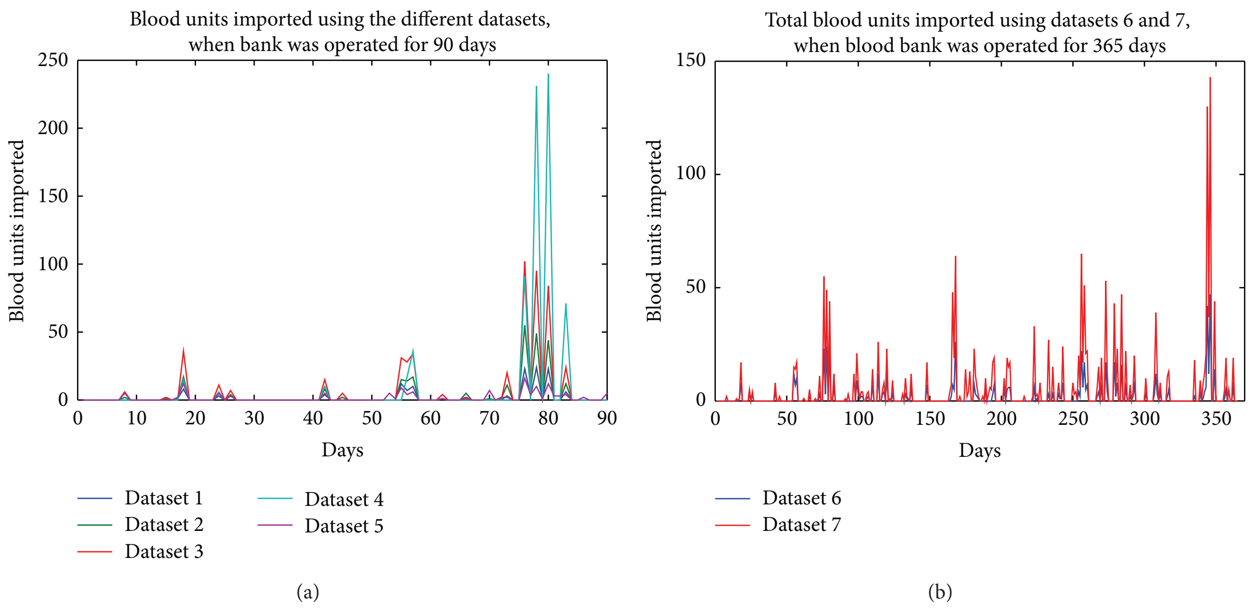


FIGURE 6: Importation rate of blood units using the different datasets.

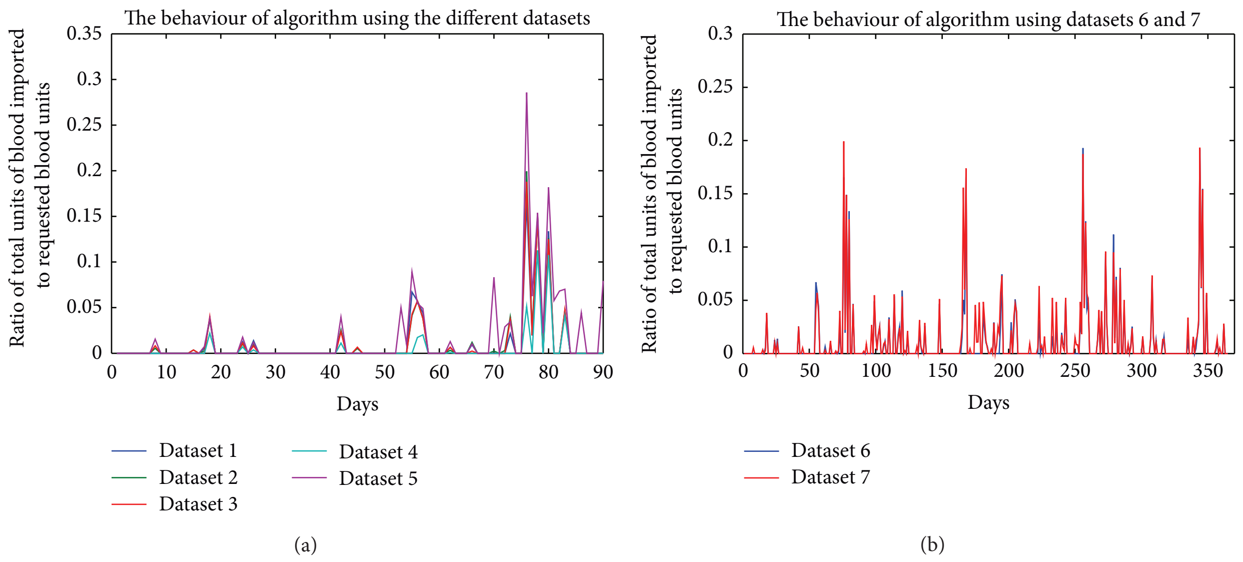


FIGURE 7: Performance measurement of the algorithm using the different datasets.

Therefore, with respect to future work, other components of the blood could be considered and the proposed method could be tested with real-life data.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] G. P. Prastacos, *Blood-Management System: An Overview of Theory and Practice*, International Institute for Applied Systems Analysis, Laxenburg , Austria, 1980.

[2] J. P. F. Charpin and A. O. Adewumi, "Optimal assignment of blood in a blood banking System," Tech. Rep., Mathematics in Industry Study Group (MISG), 2011.

[3] L. Jean, R. P. Myriam, and A. A. Celso, "Particle swarm optimization applied to task assignment problem," in *Proceedings of the 10th Brazilian Congress on Computational Intelligence (CBIC '11)*, Fortaleza, Brazil, November 2011.

[4] A. O. Adewumi, J. O. A. Ayeni, and E. P. Fasina, "Some comments on task allocation problem in distributed computing systems," *Journal of Computer Science and Its Application, Nigeria*, vol. 9, no. 1, pp. 136–145, 2003.

[5] A. O. Adewumi, *Some improved genetic-algorithms based heuristics for global optimization with innovative applications [Ph.D. thesis]*, School of Computational and Applied Mathematics,

University of Witwatersrand, Johannesburg, South Africa, 2010.

[6] M. Yusuff, J. Ariffin, and A. Mohamed, "Solving vehicle assignment problem using evolutionary computation," in *Advances in Swarm Intelligence*, vol. 6145 of *Lecture Notes in Computer Science*, pp. 523–532, 2010.

[7] J. C. Bansal and K. Deep, "A modified binary particle swarm optimization for knapsack problems," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11042–11061, 2012.

[8] B. Ye, J. Sun, and W.-B. Xu, *Solving the Hard Knapsack Problems with a Binary Particle Swarm Approach*, vol. 4115 of *Lecture Notes in Computer Science*, 2006.

[9] Z. Ren and J. Wang, "A discrete particle swarm optimization for solving multiple knapsack problems," in *Proceedings of the 5th International Conference on Natural Computation (ICNC '09)*, vol. 3, pp. 166–170, Tianjin, China, August 2009.

[10] V. Angelis, N. Ricciardi, and G. Storchi, *Optimizing Blood Assignment in a Donation-Transfusion System*, Departimento di Statistica, Probabilita e Statistiche Applicate, Universita A degli Studi di Roma 'La Sapienza', Rome, Italy, 2001.

[11] A. Adewumi, N. Budlender, and M. Olusanya, "Optimizing the assignment of blood in a blood banking system: some initial results," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '12)*, pp. 751–756, Brisbane, Australia, June 2012.

[12] E. Dufourq, M. O. Olusanya, and A. O. Adewumi, "Studies in metaheuristics for the blood assignment problem," in *Proceedings of the Operation Research Society of South Africa (ORSSA '12)*, September 2012.

[13] K. Igwe, M. Olusanya, and A. Adewumi, "On the performance of GRASP and dynamic programming for the blood assignment problem," in *Proceedings of the IEEE Global Humanitarian Technology Conference (GHTC '13)*, pp. 221–225, San Jose, Calif, USA, October 2013.

[14] M. O. Olusanya and A. O. Adewumi, "Using Metaheuristic techniques to optimize the Blood Assignment Problem," in *Proceedings of the 4th IEEE International Advance Computing Conference (IACC '14)*, pp. 1331–1336, February 2014.

[15] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.

[16] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, Nagoya, Japan, October 1995.

[17] B. Khokhar, K. P. S. Parmar, and S. Dahiya, "An efficient particle swarm optimization with time varying acceleration coefficients to solve economic dispatch problem with valve point loading," *Energy and Power*, vol. 2, no. 4, pp. 74–80, 2012.

[18] J. D. L. Silva, *Metaheuristic and multiobjective approaches for space allocation [Ph.D. thesis]*, University of Nottingham, Nottingham, UK, 2003.

[19] M. M. Mansour, S. F. Mekhamer, and N. E.-S. El-Kharbawe, "A modified particle swarm optimizer for the coordination of directional overcurrent relays," *IEEE Transactions on Power Delivery*, vol. 22, no. 3, pp. 1400–1410, 2007.

[20] V. N. Dieu, P. Schegner, and W. Ongsakul, "A newly improved particle swarm optimization for economic dispatch with valve point loading effects," in *Proceedings of the IEEE Power and Energy Society General Meeting*, pp. 1–8, San Diego, Calif, USA, July 2011.

[21] Y. Ma, C. Jiang, Z. Hou, and C. Wang, "The formulation of the optimal strategies for the electricity producers based on the particle swarm optimization algorithm," *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1663–1671, 2006.

[22] A. A. Martins and A. A. Oluyinka, "An Adaptive Velocity Particle Swarm Optimization for high-dimensional function optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 2352–2359, Cancún, Mexico, June 2013.

[23] M. A. Arasomwan and A. O. Adewumi, "Improved particle swarm optimization with a collective local Unimodal search for continuous optimization problems," *The Scientific World Journal*, vol. 2014, Article ID 798129, 23 pages, 2014.

[24] M. A. Arasomwan and A. O. Adewumi, "On the performance of linear decreasing inertia weight particle swarm optimization for global optimization," *The Scientific World Journal*, vol. 2013, Article ID 860289, 12 pages, 2013.

[25] Y. H. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[26] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation*, vol. 5, pp. 4104–4108, Orlando, Fla, USA, October 1997.

[27] M. Jiang, Y. P. Luo, and S. Y. Yang, "Particle swarm optimization-stochastic trajectory analysis and parameter selection," in *Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*, F. T. S. Chan and M. K. Tiwari, Eds., pp. 179–198, InTech, 2007.