AUTOMATING THE DRUG SCHEDULING OF CANCER CHEMOTHERAPY VIA EVOLUTIONARY COMPUTATION

K. C. Tan, T. H. Lee, J. Cai and Y. H. Chew
Department of Electrical and Computer Engineering
National University of Singapore
10 Kent Ridge Crescent Singapore 119260

Abstract: This paper presents the optimal control of drug scheduling in cancer chemotherapy using a distributed evolutionary computing software. Unlike conventional methods that often require gradient information or hybridization of different approaches in drug scheduling, the proposed evolutionary optimization methodology is simple and capable of automatically finding the near-optimal solutions for complex cancer chemotherapy problems. It is shown that different number of variable pairs in evolutionary representation for drug scheduling can be easily implemented via the software, since the computational workload is shared and distributed among multiple computers over the Internet. Simulation results show that the proposed evolutionary approach produces excellent control of drug scheduling in cancer chemotherapy, which are competitive or equivalent to the best solutions published in literature.

1. INTRODUCTION

An important problem in cancer chemotherapy is to determine the control policy such that after a fixed period of therapy, the tumour burden is minimized. This is essential when surgery or laser treatment is scheduled at some future date. Martin [16] approached this optimal control problem using non-linear programming techniques. The results were improved by Bojkov *et al.* [1], who used an intuitive approach coupled with direct search procedure proposed in [13]. Based on the approach of random numbers and search region contraction, the method of direct search optimization was applied to solve the problem [14]. Carrasco and Banga [3] proposed an adaptive stochastic algorithm to find the optimal control policy for the drug scheduling problem by removing the point constraints that are included to reduce the likelihood of the emergence of drug resistant cells.

While these methods have produced some successful results to the cancer chemotherapy problem, considerable amount of human effort and hybridization of different approaches in drug scheduling are often involved in the optimization process. These methods may also require the gradient information or multipass procedure in order to find an optimal control policy of drug scheduling, which could be tedious, subjective, and difficult to be applied to other cancer chemotherapy problems. Blessed with the rapid development of computer technology, the high computation and visualization power available at the desk nowadays allows the embedding of computer-automated scheduling (CAS) technology into a virtual problem-solving environment for sophisticated optimization problems. Building upon a high-performance optimization algorithm, such a computational framework promotes computer automation and supports the fundamental changes in conventional tuning process to automatic search of optimal schedules.

Addressing the needs of a modern computer-automated scheduling technology, this paper presents an evolutionary computing software named 'Paladin-DEC'. The software is build upon a distributed evolutionary algorithm which is available at http://evolab.ece.nus.edu.sg. It fully employs the resources of networked computers and inexpensive bandwidth to conquer complex scheduling problems, such as optimal control of drug scheduling in cancer chemotherapy, which are unsolvable or difficult-to-solve using a single computer in traditional approaches. Simulation results obtained show that the evolutionary approach produces excellent control of drug scheduling in cancer chemotherapy, which are competitive or equivalent to the best solutions ever published in literature according to the authors' best knowledge.

The paper is organized as follows: Section 2 describes the drug scheduling problem of cancer chemotherapy studied in this paper. Section 3 presents the architecture of a computer-automated drug scheduling technology. Section 4 describes the background of distributed evolutionary computing and presents a distributed evolutionary algorithm software which is used to facilitate the optimal control of cancer chemotherapy treatment. The results and performance comparisons to existing methods in literature are given in Section 5. Conclusions are drawn in Section 6.

2. PROBLEM FORMULATION

In several studies [15-16, 18-19], cancer chemotherapy protocols were chosen to minimize the tumour burden after a fixed period of treatment. There is no reward for reducing the tumour burden earlier rather than later, since the efficiency of the drug protocol is measured solely in terms of the final tumour burden. However, there is evidence suggesting that drug regimens maintaining a low intermediate tumour size could decrease the likelihood of the emergence of drug resistant cells [11]. The emergence of drug resistant cells is thought to be a significant factor in chemotherapeutic failure [8, 22].

The effects of cancer chemotherapy, as outlined by Martin [16], are given by the following differential equations,

$$\frac{dx_1}{dt} = -\lambda x_1 + k(x_2 - \beta)H(x_2 - \beta) \qquad (1)$$

0-7803-7282-4/02/$10.00 ©2002 IEEE

$$\frac{dx_2}{dt} = u - \gamma x_2 \quad (2)$$

$$\frac{dx_3}{dt} = x_2 \quad (3)$$

with the initial state $x^T(0) = [ln(100)\ 0\ 0]$ and,

$$H(x_2 - \beta) = \begin{cases} 1 & if\ x_2 \geq \beta \\ 0 & if\ x_2 \leq \beta \end{cases} \quad (4)$$

where x_1 is a transformed variable that is inversely related to the mass of the tumour. The tumour mass is given by $N = 10^{12}\exp(-x_1)$ cells, and the initial tumour cell population is set at 10^{10} cells [16]. The variable x_2 is the drug concentration in the body in drug units [D], and x_3 is the cumulative effect of the drug. Eqn. 1 describes the net change in the tumour cell population per unit time. The first term on the right-hand side of the equation describes the increase in cells due to cell proliferation, and the second term describes the decrease in cells due to the drug. The parameter λ is a positive constant related to the growth function; k is the proportion of tumour cells killed per unit time per unit drug concentration which is assumed to be a positive constant. Eqn. 2 describes the net increase in the drug concentration at the cancer site. The variable u is the rate of delivery of the drug, and the half-life of the drug is $ln(2)/\gamma$.

It is assumed that the drug is delivered by infusion, and there is an instantaneous mixing of the drug with plasma as well as an immediate delivery of the drug to the cancer site. These assumptions represent approximation based on the relative amount of time it takes for the above activities to occur with respect to the total amount of time over which the treatment is administered. Eqn. 3 relates the cumulative toxicity of the drug to the drug concentration, e.g., the cumulative effect is the integral of the drug concentration over the period of exposure. The implication of the function described in eqn. 4 is that there is a threshold drug concentration level, β, below which no tumour cells are killed. The values of the parameters used are given in Table 1.

TABLE 1 PARAMETERS OF CANCER CHEMOTHERAPY MODEL

Parameter	Value
λ	9.9×10^{-4} days^{-1}
k	8.4×10^{-3} days^{-1} [D^{-1}]
β	10 [D]
γ	0.27 days^{-1}

The performance index [16] to be maximized is,

$$I = x_1(t_f) \quad (5)$$

where the final time $t_f = 84$ days. The control optimization is performed subject to constraints on the drug delivery,

$$u \geq 0 \quad (6)$$

and on the state variables,

$$x_2 \leq 50 \quad (7)$$

$$x_3 \leq 2.1 \times 10^3 \quad (8)$$

Cancer chemotherapy is a systemic treatment, so the action of the chemotherapeutic agent is not restricted to the tumour site. Any of the body organs are liable to injury. This is in contrast to localized treatments such as surgery or radiotherapy. The drug concentration and the time of exposure to the drug determine the degree of toxicity that is sustained in both the normal organs and the tumour [25]. Therefore the constraints on the drug concentration, x_2, and the cumulative drug effect, x_3, in eqns. 7 and 8 are to ensure that the patient can tolerate the toxic side effects of the drug. Drug resistance is considered to be a significant factor in chemotherapeutic failure [8, 22], and it has been shown that the drug resistant cells are likely to increase as the tumour burden increases [11]. In order to reduce the likelihood of the emergence of drug resistant cells, the tumour size is forced to reduce by at least 50 percent every three weeks, so that

$$x_1(21) \geq ln(200) \quad (9)$$

$$x_1(42) \geq ln(400) \quad (10)$$

$$x_1(63) \geq ln(800) \quad (11)$$

For this drug scheduling problem, the performance index I obtained by Martin [16] is 16.836, which corresponds to a final tumour size of $N = 4.878 \times 10^4$ cells. Martin used an established numerical solution technique known as control parameterization, and analytical gradients were constructed for all constraints so that the resulted non-linear programming can be solved via available software. Using an intuitive approach along with the direct search optimization procedure proposed in [13], a better performance index of 17.223 was obtained in [1], corresponding to $N = 3.31 \times 10^4$ cells. The performance index value was further improved to 17.476 [14], which corresponds to a final tumour size of $N = 2.57 \times 10^4$ cells. The direct search optimization procedure based on random numbers and search region contraction was used in [14], without the need of narrowing the search space according to intuitive feel of the problem.

For the simplified drug scheduling problem without three point constraints (eqns. 9-11), Carrasco and Banga [3] applied the adaptive stochastic algorithm to obtain a performance index of 17.742, which corresponds to a final tumour size of $N = 1.97 \times 10^4$ cells. Luus [12] applied the method of iterative dynamic programming (IDP) to further improve the value to 17.993, corresponding to $N = 1.534 \times 10^4$ cells. In this paper, a distributed evolutionary computing software is applied to solve the cancer chemotherapy problem. The concept, implementation, and results of the evolutionary drug scheduling methodology are given in the following sections.

3. DISTRIBUTED EVOLUTIONARY COMPUTING

3.1 Distributed Evolutionary Algorithms

Evolutionary algorithm (EA) based intelligent search techniques [9] that mimic the mechanics of natural selection and evolution have been found to be very efficient and effective in finding the global optimum in a noisy, poorly understood and/or non-differentiable search space. Unlike gradient-guided search techniques, EA requires no stringent conditions on the performance index, such as to be well-behaved or differentiable. An EA evaluates performance of candidate solutions at multiple points simultaneously. Before this simulated evolution process begins, an initial population of multiple coded chromosomes representing random control policy of drug scheduling is formed. Every such chromosome is assigned a performance index computed against the drug scheduling optimality as quantified by eqn. 5.

At each generation of search, multiple candidates are evaluated and the search is directed intelligently according to the Darwin's "survival-of-the-fittest" principle. Then useful search information and co-ordinates are exchanged and altered to reproduce the offspring for next generation. For better evolution diversity and convergence, the elite individuals at each generation are preserved for the next generation without going through the usual genetic operations [10, 24]. This evolution cycle will be repeated until the final generation is reached or the solution has been found. Obviously, the computation effort involved in such an evolutionary process is massive due to the inherent parallelism of EA, particularly in complex optimization problems such as drug scheduling of cancer chemotherapy.

Due to the intrinsic parallelism of EA, distributed evolutionary computing (DEC) was proposed to reduce the overhead computation time as well as to obtain better solutions for evolutionary optimization [6, 20]. Instead of evolving the entire population in a single computer, the concept of multiple inter-communicating subpopulations [7] is introduced in DEC to share and distribute the computational workload among multiple computers over the Internet. The DEC enables individuals to migrate among multiple subpopulations in order to simulate the model of niche and species in natural evolution environments as well as to induce diversity of elite individuals periodically. It has been applied to solve sophisticated optimization problems in various fields, such as image processing [5], data mining [2], and network design [23].

3.2 A Distributed Evolutionary Computing Software

This section presents a distributed evolutionary computing software named Paladin-DEC. The software implements a distributed evolutionary algorithm in a general framework of Java-based distributed system, which enhances the concurrent processing of evolution and allows intercommunications among multiple subpopulations. As shown in Figure 1, the Paladin-DEC software consists of two main blocks, i.e., servant and workshop blocks that are connected by RMI-IIOP (Remote Method Invocation over Internet Inter-ORB Protocol). The servant functions as an information-center and backup station through which peers can check their identifications or restore their working status. The workshop is a place where peers (free or occupied) work together in groups, e.g., the working peers are grouped together to perform specified task, while the free ones wait for the new jobs to be assigned.

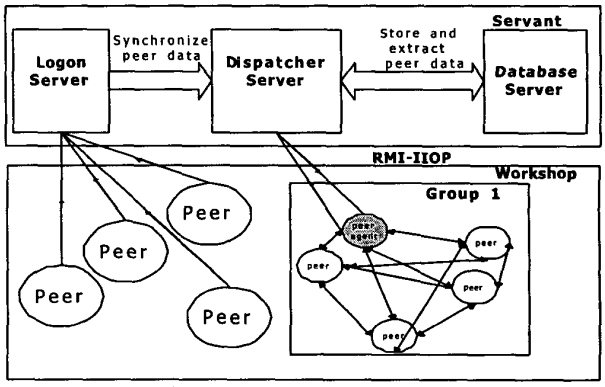

Figure 1 Schematic framework of Paladin-DEC software

The servant contains three different servers, i.e., logon server, dispatcher server, and database server. The graphical user interface (GUI) of a logon server is shown in Figure 2, which is a receptionist that assigns identification to any registered peers. The logon server also removes the information and identification of a peer when it is logged off as well as synchronizes the peer's information to the dispatcher server. The dispatcher server is responsible for choosing the tasks to be executed, the group of peers to perform the execution, and to transfer the peers' information to/from the database server. The dispatcher server also synchronizes the information, updates the peer's list, and informs the database server for any modification. Whenever there is a task available, the dispatcher server will transfer the task to a group of selected peers.

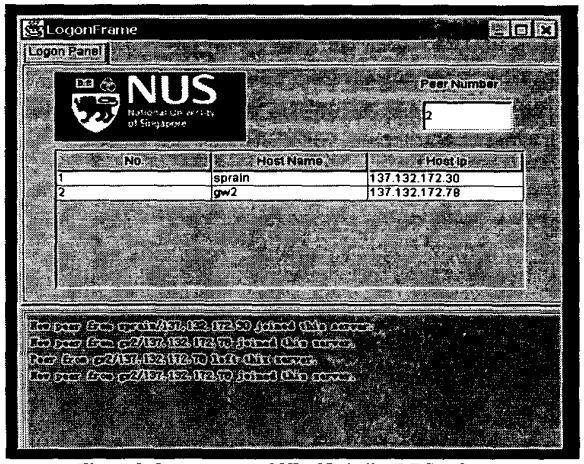

Figure 2 Logon server GUI of Paladin-DEC software

The GUI of the dispatcher server is shown in Figure 3, which contains all relevant information such as the working group's number and name, the number of free/total peers, working environments within the group, and the jobs that are available for execution. Through the GUI, administrator can easily decide the job to be executed, the number of peers to perform the job, and termination of any working group if necessary. The database server stores the peers' information from the dispatcher server and extracts the information for evaluation when requested. After a peer gets the name of a task, it loads the corresponding class remotely from the file server and begins the job execution.

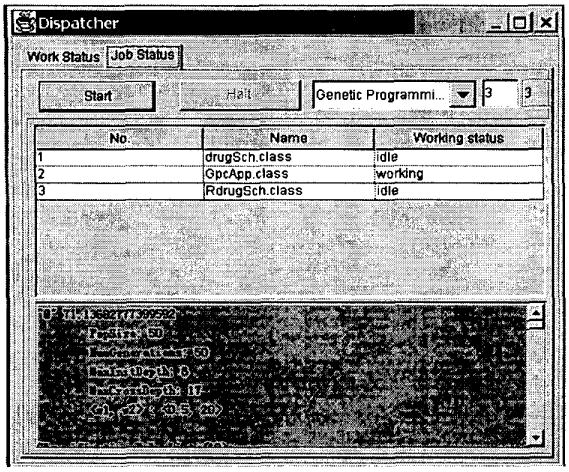

Figure 3 Dispatcher server GUI of Paladin-DEC software

The GUI of a peer in Paladin-DEC software is shown in Figure 4, which allows users to easily access the working process of any jobs under execution, the time elapsed, and the environment types. During the simulation, peers make inter-communication and transfer the intermediate results to dispatcher server for backup as shown in Figure 5. In the environment of DEC, each peer corresponds to a subpopulation. The migration interval determines the instance that a peer should initiate a migration process during the evolution, and the migration rate determines the number of individuals to be migrated in a subpopulation. If a peer meets the requirement of migration, a reference to another peer in the same working group will be assigned for migration. Once a peer finishes the job, it transfers the results to the dispatcher server and sets its status to 'free'. Similarly, a group will be dismissed by the dispatcher server once all peers in the group have finished the job.

The Paladin-DEC software allows the evolutionary optimization results and parameter settings in different groups to be saved in log files (plain-text format) via the dispatcher server. Due to the different computational capability of computers over the network, the distributed system should be able to balance the workload in order to make all peers evolve in a similar pace. The reason is that if some peers finish the evolution too fast than others, the whole population size will be reduced and the desired effect of distributed evolution in multiple subpopulations will be diminished. One simple approach to the problem is to adjust the size of subpopulation adaptively along the evolution based upon the peer's computational capability, e.g., a larger subpopulation size is assigned to peers with higher computational capabilities and vice versa, as adopted in the Paladin-DEC software.

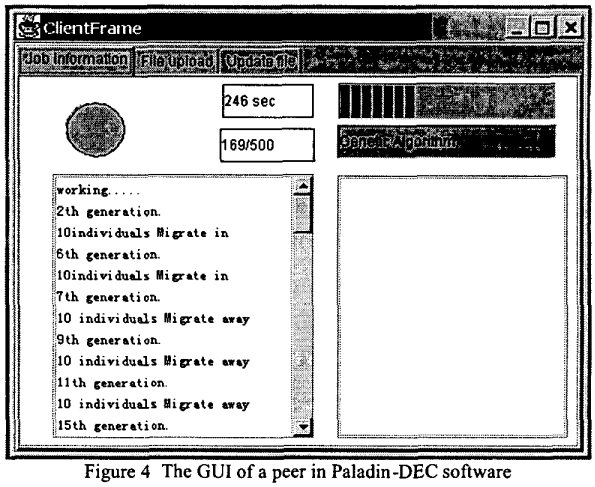

Figure 4 The GUI of a peer in Paladin-DEC software

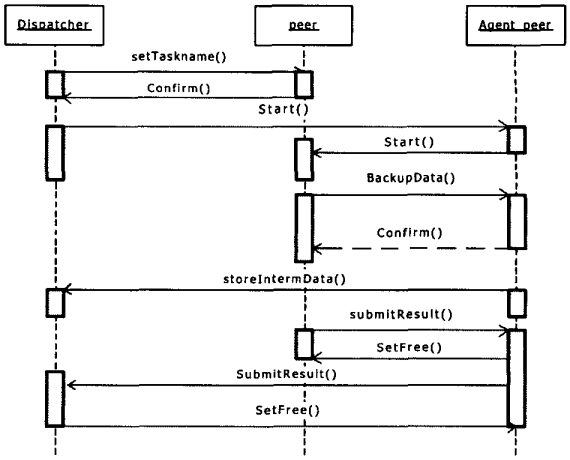

Figure 5 The working process of Paladin-DEC software

4. EVOLUTIONARY DRUG SCHEDULING VIA PALADIN-DEC SOFTWARE

For the drug scheduling problem in cancer chemotherapy as described in Section 2, there are 84 control variables to be optimized which represent the dosage levels for the 84 days. Due to the large number of variables and fine accuracy involved, a few representation schemes of variables in evolutionary optimization were investigated. Among all schemes examined, the pair-wise variable representation was found to be the most accurate and efficient. The information of dosage level and start-day are coded as variable pairs in

such representation, for instance, (45.1, 21) means the starting of drug schedule from day 21st with a dosage level of 45.1 [D]. Clearly, the number of variable pairs to be examined in the representation determines the length of the chromosome and the degree-of-freedom in the scheduling process, e.g., a larger number of variable pairs means a longer chromosome length and more scheduling freedom, and vice versa. To study the tradeoff between the number of variable pairs and the scheduling performance, a few different settings of variable pairs (between 3-20 pairs) for the drug scheduling problem were simulated using the Paladin-DEC software with the following parameters: subpopulation size = 5000; generation size = 2000; tournament selection size = 2 [24]; crossover rate = 0.7; mutation rate = 0.1; migration rate = 25 (0.5% of the subpopulation); migration interval = 10 (0.5% of the generation size); number of preserved elite individuals = 50; and number of peers (subpopulations) = 6.

The fitness function [10] used in the evolutionary optimization for the drug scheduling problem with and without point constraints is denoted by F_{point} and F, respectively,

$$F_{point} = I - p_{state} - p_{point} \quad (12a)$$

$$F = I - p_{state} \quad (12b)$$

where,

$$p_{state} = \sum_{i=2}^{3} w(x_i) \cdot (x_i - \bar{x}_i) \quad (13a)$$

$$p_{point} = w(x_1) \sum_{j=1}^{3} (x_1(j) - \bar{x}_1(j)) \quad (13b)$$

and $\bar{x}_1(1) = \ln(200)$; $\bar{x}_1(2) = \ln(400)$; $\bar{x}_1(3) = \ln(800)$; $w(x_1) = 2$; $w(x_2) = 1$; $w(x_3) = 0.3$; $\bar{x}_2 = 50$; $\bar{x}_3 = 2100$, and I is the performance index as given in eqn. 5. The weight w was set in such a way that $w(x_i) \cdot \bar{x}_i$ is about equal for i = 1, 2, and 3, in order to distribute the penalty resulting from each constraint. The cancer chemotherapy model was simulated using numerical differentiation method of Runga-Kutta [21], with a small time interval of 0.01 day for good accuracy.

The evolutionary drug scheduling results are illustrated in Figure 6, which show the performance index I (inversely related to the final mass of the tumour) versus the number of variable pairs for the problem with and without point constraints. Clearly, the scheduling results for the problem without point constraints are better, since it has a larger flexibility of minimizing the tumour size. As shown in Figure 7, the best performance index found for the drug scheduling problem without point constraints is 17.993, with a dosage schedule of {(0, 0), (32.1, 41), (13.484, 43), (13.21, 83)}. With all constraints satisfied in the scheduling problem, the best performance index is 17.472, with a dosage schedule of {(0, 0), (57.05, 18), (13.5, 19), (4.3, 21), (0, 22), (54.125, 48), (15.4, 49), (13.688, 50), (13.5, 51)}, as depicted in Figure 8. Tables 2 and 3 compare the evolutionary scheduling results of Paladin-DEC software to other approaches in literature for the problem with and without point constraints, respectively. Among all methods studied, the Paladin-DEC and Luus [12] produced the best drug scheduling results for the problem without point constraints. When the three point constraints are considered, the performance index of Paladin-DEC is very close to the best solution found in literature [14]. From these findings, it is believed that the control policies reported in [12] and [14] are very close to or at the global optimum, and there may be different control policies giving the same optimal performance index. However, it should be noted that the optimal solution obtained by the method of iterative dynamic programming (IDP) [12] required the multipass procedure and many control parameters. Moreover, a quadratic penalty function with shifting term was applied to deal with the constraint of x_3, and the method of Mekarapiruk and Luus [17] was used to handle the inequality constraint of x_2. As shown in Tables 2 and 3, the drug scheduling results obtained by the Paladin-DEC software are also significantly better than other stochastic optimization approaches, such as simulated annealing (SA), tabu search (TS), and reactive tabu search (RTS) [4].

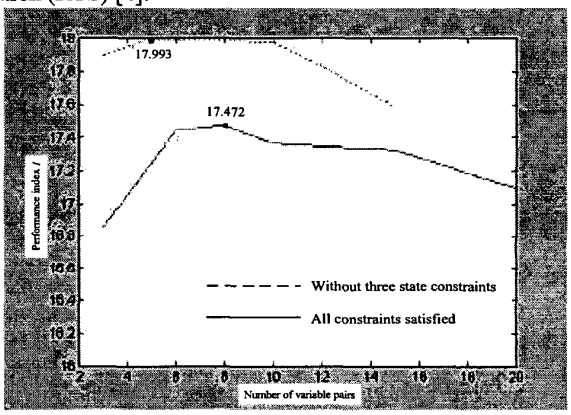

Figure 6 The performance index I versus the number of variable pairs for the drug scheduling problem with and without point constraints

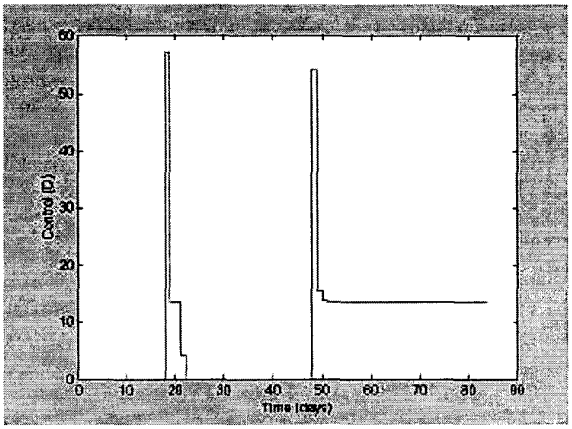

Figure 7 The optimal drug schedule without three point constraints

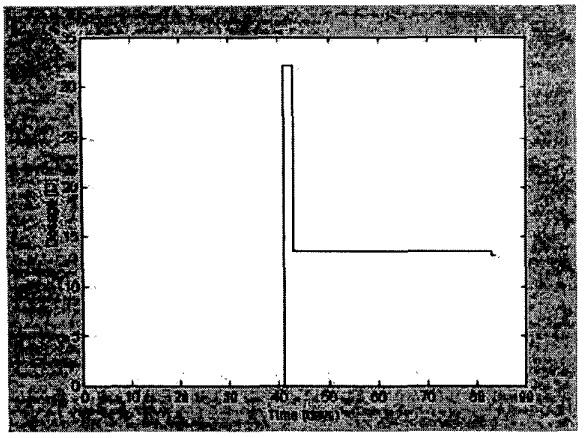

Figure 8 The optimal drug schedule with all constraints satisfied

TABLE 2 RESULTS FOR THE DRUG SCHEDULING PROBLEM WITH THREE POINT CONSTRAINTS

Approaches	Performance index	Percentage difference as compared to the best-known
Martin [16]	16.836	3.66%
Bojkov [1]	17.223	1.45%
Luus [14]	17.476	(Best-known)
SA [4]	16.726	4.29%
TS [4]	16.895	0.03%
RTS [4]	16.899	3.30%
Paladin-DEC	17.472	0.02%

TABLE 3 RESULTS FOR THE DRUG SCHEDULING PROBLEM WITHOUT THREE POINT CONSTRAINTS

Approaches	Performance index	Percentage difference as compared to the best-known
Carrasco and Banga [3]	17.742	1.41%
Luus [12]	17.993	(Best-known)
SA [4]	16.411	8.82%
TS [4]	16.558	7.99%
RTS [4]	17.567	2.38%
Paladin-DEC	17.993	(Best-known)

6. CONCLUSIONS

This paper has presented the optimal control of drug scheduling in cancer chemotherapy using a Paladin-DEC distributed evolutionary computing software. It has been shown that the evolutionary drug scheduling approach is simple and capable of solving complex cancer chemotherapy problems by sharing and distributing the computational workload among multiple computers over the Internet. Simulation results for the problem with and without point constraints confirmed the effectiveness of the proposed approach by producing excellent control policies of the drug scheduling in cancer chemotherapy, which are competitive or equivalent to the best solutions published in literature. On-going work includes the application of Paladin-DEC software to other cancer chemotherapy problems, which may be highly sophisticated or difficult-to-solve using existing methods in a single computer.

REFERENCES

[1] Bojkov, B., Hansel, R., and Luus, R. (1993). Application of direct search optimization to optimal control problems, *Hung. J. Ind. Chem.*, vol. 21, pp. 177-185.

[2] Brameier, M., and Banzhaf, W. (2001). A comparison of linear genetic programming and neural networks in medical data mining, *IEEE Trans. on Evolutionary Computation*, vol. 5, no. 1, pp. 17-26.

[3] Carrasco, E. F., and Banga, J. R. (1997). Dynamic optimization of batch reactors using adaptive stochastic algorithms, *Industrial and Engineering Chemistry Research*, vol. 36, pp. 2252-2261.

[4] Chang, M. Y. (2000). *Optimal Scheduling of Cancer Chemotherapy*, B.Eng. Thesis, Dept. of Electrical and Computer Engineering, National University of Singapore.

[5] Chen, Y. W., Nakao, Z., and Xue, F. (1996). A parallel genetic algorithm based on the island model for image restoration neural networks for signal processing, *Proc. of IEEE Signal Processing Society Workshop*, pp. 109 -118.

[6] Chong, F. S. (1997). *A Java Based Distributed Genetic Programming on The Internet*, M.Eng. Thesis, School of Computer Science, University of Birmingham.

[7] Cristea, V., and Godza G. (2000). Genetic algorithms and intrinsic parallel characteristics, *Proc. IEEE Cong. on Evolutionary Computation*, vol. 1, pp. 431-436.

[8] Crowther, D. (1974). Rational approach to the chemotherapy of human malignant disease – II, *Br. Med. Journal*, vol. 4, pp. 216-218.

[9] Fogel, D. B. (2000). What is evolutionary computation? *IEEE Spectrum*, pp. 26-32.

[10] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Massachusetts.

[11] Goldie, J. H., and Coldman, A. J. (1978). A mathematical model for relating the drug sensitivity of tumours of their spontaneous mutation rate, *Can. Treat. Rep.*, vol. 63, pp. 1727-1733.

[12] Luus, R. (1998). Comments on "Dynamic optimization of batch reactors using adaptive stochastic algorithms", *Industrial and Engineering Chemistry Research*, vol. 37, pp. 305.

[13] Luus, R., and Jaakola, T. H. (1973). Optimization by direct search and systematic reduction of the size of search region, *AICHe Journal*, vol. 19, pp. 760-766.

[14] Luus, R., Hartig, F., and Keil, F. J. (1995). Optimal drug scheduling of cancer chemotherapy by direct search optimization, *Hung. J. Ind. Chem.*, vol. 23, pp. 55-58.

[15] Martin, R. B. (1991). *Optimal Control of Drug Administration in Cancer Chemotherapy*. Ph.D. Thesis, University of Western Australia.

[16] Martin, R. B. (1992). Optimal control drug scheduling of cancer chemotherapy, *Automatica*, vol. 28, pp. 1113-1123.

[17] Mekarapiruk, W., and Luus, R. (1997). Optimal control of inequality state constrained systems, *Ind. Eng. Chem. Res.*, vol. 36, pp. 1686-1694.

[18] Murray, J. M. (1990a). Optimal control of a cancer chemotherapy problem with general growth and loss functions, *Math. Biosci.*, vol. 98, pp. 273-287.

[19] Murray, J. M. (1990b). Some optimal control problems in cancer chemotherapy with a toxicity limit, *Math. Biosci.*, vol. 100, pp. 49-68.

[20] Paechter, B., and Back, T. (2000). A distributed resources evolutionary algorithm machine (DREAM), *Proc. of IEEE Cong. on Evolutionary Computation*, vol. 2, pp. 951-958.

[21] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing*, Second Edition, Cambridge University Press, Cambridge, New York.

[22] Skipper, H. E. (1978). Adjuvant Chemotherapy, *Cancer*, vol. 41, pp. 936-940.

[23] Sleem, A., Ahmed, M., Kumar, A., and Kamel, K. (2000). Comparative study of parallel vs distributed genetic algorithm implementation of ATM network environment, *Proc. of Fifth Int. Sym. on Computers and Communications*, pp. 152-157.

[24] Tan, K. C., Lee, T. H., Khoo, D., and Khor, E. F. (2001). A multi-objective evolutionary algorithm toolbox for computer-aided multi-objective optimization, *IEEE Trans. on Systems, Man and Cybernetics: Part B (Cybernetics)*, vol. 31, no. 4, pp. 537-556.

[25] Wheldon, T. E. (1998). Mathematical models in cancer research, *Adam Hilger*, Bristol.