NJERD, Vol. 6, No. 4, 2007

A SUBTOUR-FREE SET-SEQUENCING ALGORITHM BASED COMPUTER SOFTWARE FOR SOLVING THE MACHINE SET-UP PROBLEM

O. E. Charles-Owaba and V. O. Oladokun
Department of Industrial Engineering
University of Ibadan
Ibadan NIGERIA

ABSTRACT

The machine set-up problem (MSP) is that of determining an optimal sequence that a set of N operations will be performed by a general-purpose facility in order to minimize the total cost/time of re-setting the facility. Though the MSP is experienced in many small scale industrial situations, many of the existing algorithms are programmed to run on super computers or massively parallel computer processors which are often not available for small scale industrial environment in developing economies. In this study a Subtour-free Set Sequencing Algorithm based software for solving real life medium size machine set-up problem on personal computers was developed and tested. The computational time curve exhibited a polynomial growth for the range of problems solved and the solution system was shown to be practicable.

Keywords Combinatorial Optimization, Machine Set-up-Problem, Subtour, Software, Set-Sequencing-Algorithm

1.0 INTRODUCTION

The utilization of classical scheduling theory in most production environments, especially in a developing country like Nigeria, has been minimal even though scheduling theory has matured over the years (Maccarthy and Liu 1993, Woerner and Biefeld 1993, Pinedo 1995). Most often the scheduler is not familiar with complicated scheduling theories.

While manufacturing has been revolutionized in most developed countries where the concept of computer-integrated manufacturing (CIM) is widely used, it has been reported that the use of CIM is very low in Nigeria (Ogedengbe et al 2002). The development of low cost made in Nigeria industrial software is an important step towards realizing the full benefit of CIM. To increase the acceptability of any scheduling procedure and enhance its practical relevance such algorithm should be embedded in a user friendly software system (Maccarthy and Liu 1993).

The development of user friendly computer software for solving the Sequence Dependent Single Machine Set-up Problem (MSP) is the focus of this work.

2.0 THE MACHINE SET UP PROBLEM (MSP)

MSP is the problem of determining an optimal sequence that a set of N operations will be performed by a general purpose facility in order to minimize the total cost or time of re-setting the facility. In many real-life problems there are sequence dependent setup times S_{ij} (S_{ij} = setup time if job j is processed immediately after job i) and thus the makespan is a function of the schedule. Given N parts and a processor, an MSP algorithm finds the order (a sequence of the N parts) in which each part will pass through the processor or the processor will pass through each only once, so that the makespan is minimized.

It should be noted that MSP's rendition as a processor passing through the N parts (stations) is popularly interpreted as the Traveling Salesman Problem (TSP) (Dei'neko et al 2006, Gutin and Punnen, 2002, Lawler et al 1985, Kahng and Reda 2004, Balas and Simonetti 2001, Charles-Owaba 2001, Dantzig 1954, Little et al 1963). When S_{ij} = S_{ji} the problem is a special case, the Symmetric T (STSP). The Machine Setup Problem (MSP) is actually the equivalence of the more general Asymmetric TSP (ATSP) for which $S_{ij} \neq S_{ji}$ may hold. The Asymmetric TSP's cases are known to be more difficult to solve than their symmetric equivalent (Johnson et al 2002, Zhang 2004) and there have been indeed very few studies and algorithms on the

Asymmetric TSP (the MSP) reported in the literature (Kwon et al, 2005, Gutin and Punnen, 2002, Applegate et al, 2004, Walshaw, 2002)

3.0 THE SUB-TOUR FREE SET-SEQUENCING ALGORITHM SSA

Interest in new MSP algorithm development has been attracting much attention in recent times (see Balas and Simonetti ,2000, Baltz and Sviridenko, 20003, b For instance, the largest instance of the TSP solved to optimality is the 24, 978 Sweden cities carried out on a cluster of 96 Intel Xeon 2 8 GHz dual-processor machines requiring an equivalent estimated time of 91 9 CPU years of a single Intel Xeon 2 8 GHz processor The computation started in March 2003 and finished in May 2004 (Applegate, 2004)

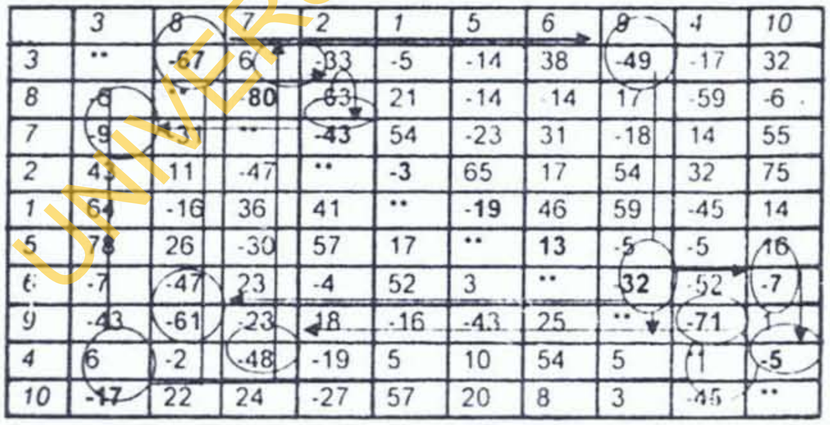
Unfortunately, super computing and massively parallel computing are often not available for industrial applications in many developing economies, (Oladokun 2006 Charles-Owaba, 2002) Therefore, the search for efficient MSP software suitable small end computing platform found in a typical Nigerian machine shop environment remains relevant

Implicit enumeration and heuristics are the MSP traditional solution approaches The former are inefficient, therefore impractical for industrial applications, while the later are efficient but many lack effectiveness For instance there are 2 4 x 10^{18} and 1 5 x10^{25} possible solutions for 20 parts problem and 25 parts problem respectively. The IBM built 280 6 teraflop Blue Gene/L supper computer (fastest modern computer) which could perform up to 280 6 trillion calculations per second (Fordahl 2005), using the complete enumeration algorithm, would take 2 days to find an optimal schedule for a 20 parts problem and 400 centuries for 25 parts problem Hence other more constructive methods are used to tackle this NP hard problem

3.1 The Set-Sequencing Algorithm (SSA)

Recently, Set Sequencing Algorithm (SSA) was proposed as a basis for the MSP solution approaches In the Set Sequencing paradigm (Charles-Owaba 2001, Charles-Owaba, 2002) a complete tour is viewed as comprising a set of N matrix elements (links) Given any initial MSP solution sequence S_i the SSA views a MSP matrix M( ) as comprising incumbent and candidate links E(M) Set Sequencing is defined as the transformation of a known sequence (S_i) to a new sequence (S_{i+1}) by feasibly replacing some incumbent link in E(S_i) with the same number (Na) of candidate links from set E(C_i)

Three distinct sets of links or edges. The set of incumbent links, Iu E(S_i). The set of replaced incumbent links, Lr E(S_i). The set of replacers or candidate links, Lc (C_i) are involved in the SSA process The SSA identifies the link in set Lc, bringing them into the new solution edge set E(S_{i+1}), one link at a time, and making sure that no pair comes from the same row or column of M( ) it continues until the path of search forms a closed circuit when $|Lc| = |Lr|$ holds A typical problem with closed circuit is shown in Fig 1



	3	8	7	2	1	5	6	9	4	10
3	**	-67	6	-33	-5	-14	38	-49	-17	32
8	-6	**	-80	-63	21	-14	14	17	-59	-6
7	-9	-31	**	-43	54	-23	31	-18	14	55
2	43	11	-47	**	-3	65	17	54	32	75
1	64	-16	36	41	**	-19	46	59	-45	14
5	78	26	-30	57	17	**	13	-5	-5	16
6	-7	-47	23	-4	52	3	**	32	52	-7
9	-43	-61	-23	18	-16	-43	25	**	-71	
4	6	-2	-48	-19	5	10	54	5	1	-5
10	-17	22	24	-27	57	20	8	3	-45	**

Fig. 1: SSA Process forming a closed circuit with feasible S_{i+1} = {8 2-1-5-6 10 4-7-3-9-8}

The SSA process matrix in Fig. 1 is known as difference matrix D.( ) obtained from the MSP matrix as follows

For the candidate index

$D_I(x_i, y_i) = 2M(x_i, y_i) - (Vr + Vc)$

$$\text{Where } Vr = \begin{cases} M(x_i\ 1), & \text{if } x_i = N \\ M(x_i\ x_i+1), & \text{otherwise} \end{cases} \quad (1)$$

and

$$Vc = \begin{cases} M(N\ y_i), & \text{if } y_i = 1 \\ M(y_i-1\ y_i), & \text{otherwise.} \end{cases}$$

N, being the problem size

(2)

3.2 The Gain Function $\Delta_I v$ (Lr, Lc, Na)

The exact value of the gain Δv (Lr, Lc, Na) associated with a given set Lc is calculated directly from the current difference matrix, D_I( ) using equation 3, provided S_{I+1} forms a feasible or complete MSP tour

$$\Delta v\,(Lr.Lc.Na) = \frac{1}{2}\sum_{j=1}^{Na} D_I(u_j, v_j) \quad \forall\ u_j\text{-}v_j \in Lc \quad (3)$$

For the SSA procedure, the following equations are used to compute the value of the new sequence, S_I.

$$V_a(S_{I+1}) = V_a(S_I) + \Delta v\,(Lr,\ Lc,\ Na) \quad (4)$$

$$V_a(S_{I+1}) = V_a(S_I) + \frac{1}{2}\sum_{j=1}^{Na} D_I(u_j, v_j) \quad \forall\ u_j\text{-}v_j \in Lc \quad (5)$$

Observe in expressions (3) and (4.5) that, for the minimization MSP, it is desirable that the sum of terms, $\{D_I(u_j, v_j)\}$ be negative. This is actually an alternative MSP model equivalent to the traditional MSP model. Minimize $Va(S_I)$. See (Charles-Owaba 2001, Charles-Owaba, 2002 and Oladokun 2006) for further details

4.0 THE SOFTWARE DEVELOPMENT METHODOLOGY

The objectives of development process are to ensure that the software is user friendly and simple, that it is compatible with most common low-end computer platforms, that it has an open-ended interface amenable for future upgrading which can support and accommodate some other scheduling algorithms and that it contains adequate help facilities and documentation for easy maintenance.

4.1 Software Development Languages and Tools

Borland Delphi (version 6.0), based on object Pascal's syntax and programming logic, has been adopted. Pascal is powerful and very efficient in terms of time and space required to run a programme written in it. Borland Delphi, popular and well accepted among software developers, offers a practical and easy means of creating computer applications for the Microsoft Windows operating systems. It is a highly portable Integrated Development Environment (IDE) compatible with most computer platforms. These and many other features have informed our adoption this IDE, and object Pascal was used for the MSP algorithm code.

4.2 Software Design Methodology

The MSP software is designed as a window based programme with a mouse and keyboard driven input /output Graphical User Interface. The software system is made up of three subsystems as shown in Fig. 2. The three subsystems are as follows

- The graphical Interface (GUI)
- The processing or analysing subsystem and
- The external file and data management subsystem

4.2.1 The Graphical Interface (GUI)

The GUI is divided into three modules, which are the input module, the Help module and the Output module.

The Main interface

The MSP software interface is designed to resemble the familiar Microsoft based window interface. This is to take advantage of most users' familiarity with this type of interface.

Input Interface

The input module is capable of loading input problems online (interactively) and in batch (stored on external file). For file input the problem is stored in ASCII format using rich text file format. Both types of input options are designed to display the input problem in matrix format for any correction, Figs 3 & 4. It is the corrected version of the input problem that is used in solving the problem.

The Interactive input form allows users to input data directly in three possible formats. 1) As a N by N square matrix, 2) As a N by 2 matrix of the N station Cartesian coordinates and 3) As a N by 2 matrix of the N process states or levels

4.2.2 The Processing or Analysing subsystem

The processing subsystem consists of the computer codes behind the interface. Processing subsystem is made up of two independent modules. The first module is the default module, which contain the object Pascal codes of the MSP algorithm and embedded directly into the GUI subsystem. The second module contains the set of codes for random problems generator and the input data verification code

4.2.3 Programme structure

Fig. 5 shows a simplified structure of the MSP software Modules

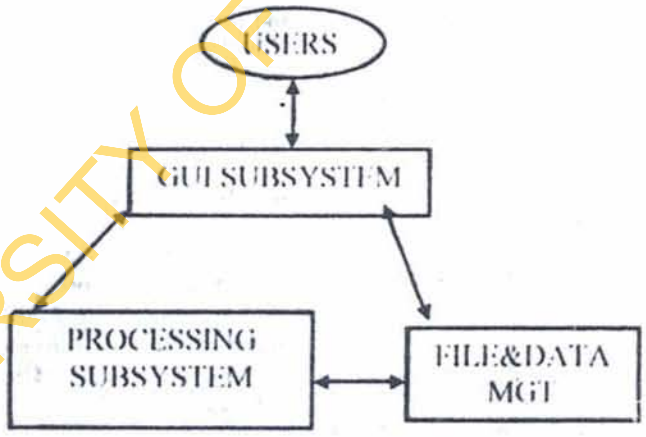
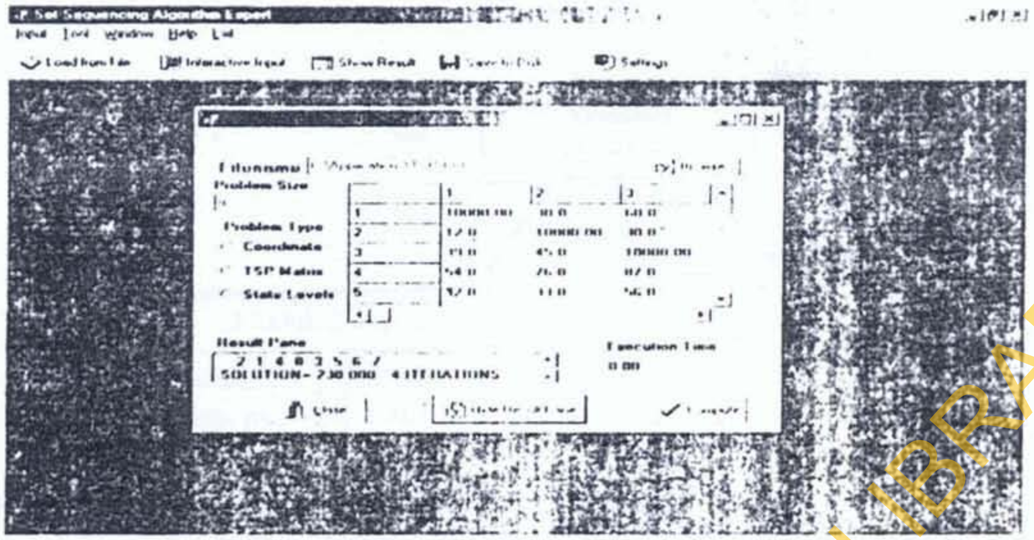


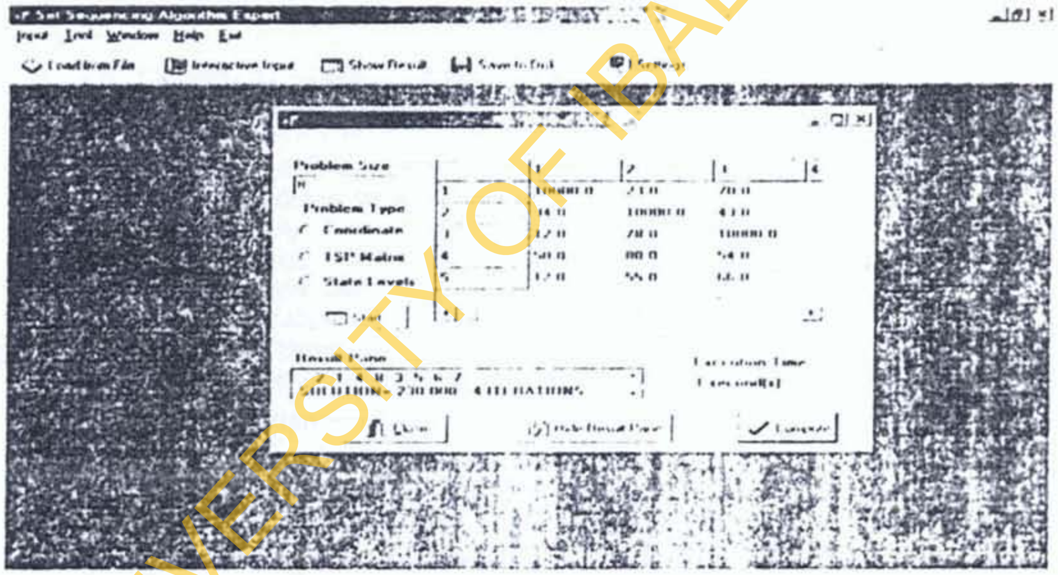
Fig. 2: The MSP Software system



	1	2	3
1	10000.00	[illegible]	[illegible]
2	12.0	10000.00	30.0
3	[illegible]	45.0	10000.00
4	54.0	76.0	87.0
5	52.0	11.0	56.0

Result Pane
```
 2 1 4 8 3 5 6 7
SOLUTION= 230.000   4 ITERATIONS
```
Execution Time
0.00

Fig. 3: File loading input menu interface



	1	2	3
1	10000.0	21.0	70.0
2	34.0	10000.0	43.0
3	12.0	78.0	10000.0
4	50.0	80.0	54.0
5	12.0	55.0	66.0

Result Pane
```
 2 1 4 8 3 5 6 7
SOLUTION= 230.000   4 ITERATIONS
```
Execution Time
0 second(s)

Fig. 4 Interactive problem loading menu

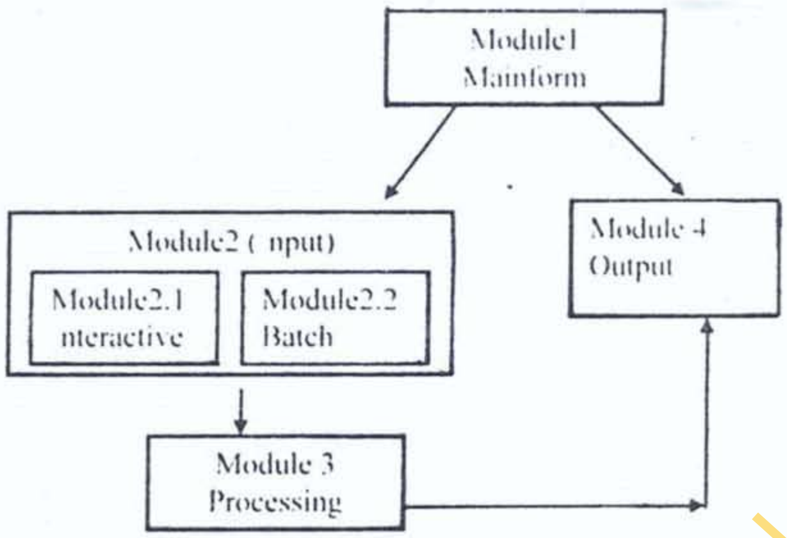


Fig. 5: A simplified structure of the MSP software modules

5.0 COMPUTATIONAL EXPERIENCE AND DISCUSSION OF RESULTS

The new SSA algorithm was also evaluated using a set of 280 randomly generated Asymmetric MSP problems. Similarly, 7 Asymmetric TSP instances of 17 to 65 vertices from TSPLIB (Reneilt, 1991, Reneilt, 1995) were used for comparative analysis. A real life application involving a 60-part machine set-up problem carried out on a general-purpose lathe machine was also solved.

5.1 Computer Platform Used

The specification of the computer facility used is summarized below.

i Computer Processor 667MZ Pentium III MMX processor.
ii Computer Memory, 128Mb SDRAM and
iii Operating system, Microsoft Windows 98, window 2000 or window XP

5.2 Computational Time Analysis

For the randomly generated problems with problem sizes ranging between 10 and 140 the problems matrix cost elements were in four ranges (1) 5-50, (2) 10-50, (3) 20-50 (4) 100 000-500 000. The computational time increased from 0.012 minute (for the 20 x 20) to 11 minutes for the (100 x 100 problems). Table 1 gives the summary of the time. The Time vs Problem Size curve is shown in Fig 6.

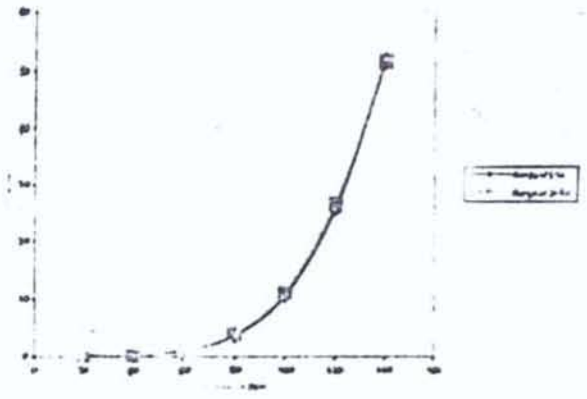


Fig. 6 Time vs Problem Size Graph

5.3 Effect of Range Difference

In order to evaluate the sensitivity of the proposed algorithm to problem range the test of mean of time of the two ranges was carried out using t test of mean. The size cost elements has no effect on the computational time at 5% level.

Table 1: Summary of Results

Size	No of problems	Mean Time (min)	Standard Deviation
20	15	0 0794	0 256339
40	10	0 2026	0 023244
60	10	1 042	0 10051
80	10	3 7268	0 352296
100	10	10 7609	0 67852
120	10	26 1492	0 901776
140	10	51 4897	1 583409

Table 2: Computational time result under range 5-50

Size	No of Problems	Mean Time (min)	Variance	Standard Error of mean
20	5	0 2112	0 444307	0 2980
40	5	0 2042	0 016254	0 0570
60	5	1 0908	0 111909	0 1496
80	5	3 5892	0 346121	0 2631
100	5	10 5286	0 669107	0 3658
120	5	25 6866	0 707048	0 3760
140	5	51 6348	2 082	0 6453

Table 3: Computational Time Result under Range 20-50

Size	No of Problems	Mean Time(min)	Variance	Standard Error of mean
20	5	0 0134	0 00114	0 0150
40	5	0 201	0 030741	0 0784
60	5	0 9932	0 065213	0 1142
80	5	3 8644	0 334842	0 2587
100	5	10 9932	0 673248	0 3658
120	5	26 6118	0 891507	0 42225
140	5	51 3446	1 119734	0 4732

Table 4: Test of Means for the Two Ranges 5-50 and 20-50

Size	t-calculated	t- tabulated		Df	Comment
		α=0 05	α=0 01		
20	0 662694	2 776	4 604	4	There is no significant difference in all the problem sizes
40	0 033007	2 30	3 355	8	
60	0 518559	2 30	3 355	8	
80	-0 74571	2 30	3 355	8	
100	-0 89667	2 30	3 355	8	
120	-1 63628	2 30	3 355	8	
140	0 362652	2 30	3 355	8	

5.4 Regression Analysis

The plots of experimental data obtained as shown in Fig6 were curve fitted into a polynomial function assuming a function of the form $Y \quad Ax^{B}$

Where Y =T the computing time. X n the problem size and B & A are the regression coefficients Applying least square fitting gives the following function

$$Y=1\ 099E\text{-}6X^{3\ 5} \text{ or } T=1\ 09\times10^{-6}n^{3\ 5} \qquad (6)$$

Expression 1 seems to agree with the estimated number of computations (NC) derived as follows

(a) Number of Transition Process units (TP) per replaceable incumbent = 2(N-2)

(b) Possible number of feasible terms in a TP= 3.4 N (arithmetic progression)

(c) Average number of terms in TP= (N + 3)/2

(d) Number of replaceable incumbents = N

(e) Number differences to be computed per iteration N(N-2)

$$NC= (a)(c)(d) +e = 2(N\text{-}2)(\frac{N+3}{2})(N) +N(N\ 2)$$

Considered over k iterations

$$NC= k[2(N\text{-}2)(\frac{N+3}{2})(N) +N(N\text{-}2)]= kN(N\text{-}2)(N+4)$$

Assumed k is a constant, actually k averages six iterations across the problem sizes solved (Oladokun, 2006)

$$\text{Then } NC= N(N\text{-}2)(N+4) \qquad (7)$$

This does not include the number computations required for testing for the feasibility of each circuit This perhaps explains why expression 6 is a polynomial of degree 3 5 instead of 3 as expected in Eqn (7)

The algorithm of Christofides (1976) has a computational complexity of O (N^3) (Papadimitriou and Steiglitz 1982) On the other Assignment Based Tour construction approach's computational complexity is dominated by the solution of the linear assignment problem for which the currently best known approach takes time of $O(N^3)$ (Johnson et al. 2002) While branch and bound algorithms are exponential even for very small sized problems and the time complexity of the 4-opt algorithm is $O(N^4)$

5.5 Solution Quality Assessment

Comparing with the result of some traditional methods we assess the quality of solution obtained from the proposed method Two methods Nearest Neighbour with varying origin (multi start) and the Set Sequencing Algorithm with Cycles Removal were used The solutions quality were consistently better

Some set of asymmetric TSP were collected from the TSPLIB and solved with the new software Table 5 contains the solutions obtained from the new algorithm and the best solution reported by other algorithms for each problem on the website The problems selected were those arranged in format compatible with the software input format This confirms the effectiveness of the RSSA compared with existing MSP solution procedures

Table 5: ATSPLIB Test Data

S/N	TSPLIB Problem's Identity	Problem's Size	Input value	Best Reported Solution	SSA's Solution	Computational Time (min)
1	Ftv17	17	167	39	39	0 012
2	Ftv38	39	2331	1530	1459	0 109
3	P43	43	5843	5620	5576	0 281
4	Ftv44	45	2678	1613	1613	0 270
5	Ftv47	48	4289	1776	1776	0 283
6	Ftv55	56	3974	1608	1608	0 73
7	Ftv64	65	4783	1839	1839	1 53

Sources: (Reneilt. 1991. Reneilt. 1995)

6.0 EXPERIENCES AND CONCLUSIONS USING THE SOFTWARE

The following can be concluded from the experience of test users of the MSP software during the process of solving both real life problems and randomly generated problems.

1 The MSP software is easy to use, as it does not require memorizing any command. It is mouse driven presenting users with adequate guidelines.

2 Also the software has been shown to be compatible with most common low-end computer platform. Tests were carried out on the popular window 98 and window XP operating system running on Pentium III processor.

It is concluded that efficient subtour-free Set Sequencing Algorithm based software for solving real life medium size MSPs on personal computers is practicable. The computational time curve exhibited a polynomial growth for the range of problems solved.

REFERENCES

Applegate D., Bixby R., Chvatal V., Cook W. and Helsghaun K. (2004): The Sweden Cities TSP Solution' Available at www.tsp.gatech.edu//sweeden/cities/ Cities.htm

Balas E. and Simonetti N. (2001): 'Linear time Dynamic Programming Algorithms for New Classes of Restricted TSPs: a Computational Study' INFORMS Journal on Computing Vol 13, No 1, pp56-75.

Baltz A. and Srivastav A. (2005): "Approximation algorithms for the Euclidean bipartite TSP" Operations Research Letters, Vol. 33, No 4, pp 403-410.

Charles–Owaba O. E. (2001): 'Optimality Conditions to the Acyclic Traveling Salesman Problem' Operational Research Society of India Vol 38, No 5.

Charles–Owaba O. E. (2002): "Set-sequencing Algorithm to the Cyclic TSP" Nigerian Journal of Engineering Management. Vol 3, No 2, pp47-64

Christofides N. (1972):"Bounds for the Traveling-Salesman Problem", *Operations Research* Vol 20, pp 1044-1056

Dantzig G. B., Fulkerson D.R. and Johnson S. M. (1954): "Solution of a Large-Scale Traveling-Salesman Problem". *Operations Research* Vol 2 393-410

Okamoto Y. and Woeginger G. J. (2006): The Traveling Salesman Problem with Few Inner Points" *Operations Research Letters*. Vol 34, No 1, pp 106-110

Gutin G. and Punnen J. (2002)(eds): "The TSP and its Variations" Kluwer Academic Publishers

Johnson D. S. , Gutin G., McGeoch L. A. , Yeo A., Zhang W. and Zverovitch A. (2002): 'Experimental Analysis of Heuristics for the Asymmetric Traveling Salesman Problem" in Gutin G and Punnen H (eds) - 'The Traveling Salesman Problem and its Variations Kluwer Academic Publishers

Kahng A. B. and Reda S. (2004): 'Match twice and Stitch: a New TSP Tour Construction Heuristic "*Operations Research Letters* Vol 32, No 6, pp 499-509

Kwon S., Kim H. and Kang M. (2005): "Determination of the Candidate Arc Set for the Asymmetric Traveling Salesman Problem" Computers and Operations Research. Vol 32, No 5, pp 1045-1057

Lawler E. J., Lenstra J. K., Rinnoy Kan A. H. G. and Shmoys D. B. (1985): "The Travelling Salesman Problem: A guided tour of Combinatorial Optimization" John Wiley & Sons

Lewenstein M. and Sviridenko M.(2003): 'A 5/8 Approximation Algorithm for the Maximum Asymmetric TSP SIAM Journal on Discrete Mathematics Vol 17, No 2, pp 237-238

Little J. D. C., Murty K. G., Sweeney D. W. and Karel C. (1963): "An Algorithm for the Traveling Salesman Problem". *Operations Research* Vol 11, pp 972-989

Maccarthy B. L. and Liu J. (1993): 'Addressing the Gap in Scheduling Research: a Review of Optimization and Heuristic Methods in Production Scheduling', *International Journal of Production Research*. Vol 31, No 1, pp 59-79

Ogedengbe T. I., Adejuyigbe S. B. and Aderoba A. A. (2002): Adoption of Computer Integrated

Manufacturing in Nigeria" Nigerian Journal of Engineering Management Vol 3. No 3. pp10-16

Oladokun V. O. (2006): "The Development of A Subtour-Free Set Sequencing Algorithm and The Software For Solving The Machine Set-Up Problem A Ph D Thesis at the University Of Ibadan Ibadan

Papadimitriou C. and Steiglitz K. (1982): "Combinatorial Optimization Algorithms and Complexity" Prentice Hall, Englewood Cliffs, NJ

Pinedo M. (1995): "Scheduling Theory, Algorithm and Systems", Prentice Hall Pub New Jersey

Reneilt G. (1991): "TSPLIB- A Traveling Salesman Problem Library ORSA Journal of Computing Vol 3, No 4. pp 376-384

Reneilt G. (1995): "TSPLIB95 Tech Rep Inst Angewandte Math, University of Heidelberg

Saadani N. E., Guinet A. and Moalla M. (2005): "A Travelling Salesman Approach to Solve The F/No Idle/Cmax Problem", European Journal of Operations Research, Vol 161, No 1 pp 11-20

Vairaktarakis G. L. (2006): "Simple Algorithms for Gilmore-Gomory's Traveling Salesman and Related Problems" Journal of Scheduling, Vol 6 No 6 499-520

Walshaw C. A. (2002): "Multilevel Approach to the Travelling Salesman Problem" Operations Research Vol 50, No 5. pp 862-877

Woerner I W. and Biefeld E. (1993): "Hypertext Based Design of a User Interface for Scheduling" Proceedings of the AIAA Computing in Aerospace 9 San Diego Califonia

Zhang W. (2004): Phase Transitions and Backbones of the Asymmetric Traveling Salesman Problem", Journal of Artificial Intelligence Research Vol 21. pp 471-497