FLOW SHOP SCHEDULING USING CLUSTERED DIFFERENTIAL EVOLUTION

Donald Davendra and Ivan Zelinka,
Department of Applied Informatics,
Tomas Bata University in Zlin,
Nad Stranemi 4511, Zlin 76001
Czech Republic.
Email: {davendra,zelinka}@fai.utb.cz

Godfrey Onwubolu,
Knowledge Management & Mining,
Richmond Hill,
Canada.
Email: onwubolu_g@dsgm.ca

KEYWORDS

Differential evolution, flowshop scheduling, clustering

ABSTRACT

A generic approach of population dynamics and clustering for permutative problems is presented in this paper. Diversity indicators are created from solution ordering and its mapping is shown as an advantage for population control in metaheuristics. Differential Evolution Algorithm is embedded with a population based on this approach and vetted with the Flow shop scheduling problem. Extensive experimentation is conducted on benchmark problems in this area with good results.

INTRODUCTION

Metaheuristics are algorithms, which are used for the optimization of complex systems. The vital attribute for these heuristics is that they have to operate without a-priori information of the system.

Metaheuristics operate on two ideological frameworks, firstly that a group of solutions provide a better platform to find optimal solution, and secondly that certain guiding concept leads the solutions towards the optimal solution, or nearby regions.

A number of different transformation concepts have evolved within the scope of Metaheuristics (Dreo et al., 2006). Genetic Algorithms, Differential Evolution, Particle Swarm Optimization and Self Organising Migrating Algorithm are some of the most potent heuristics available. Most of these algorithms have mimicked naturally occurring phenomena.

The principle concept of the population is to provide a pool of solutions from which the optimal solution evolves from, however during the subsequent transformation, the solutions cluster together in nearby neighbourhoods. This leads to the loss of diversity of the solutions, and later generations are usually mutating within a cloned gene pool. This phenomenon is generally termed as stagnation and the two most current correcting tools are local search within the hierarchical population and forfeiting of some better-placed solutions in lieu of worst ones (Onwubolu (2002)).

Stagnation is a major concern for evolutionary heuristics, since evolution is principally based on diversity of the existing population. The preconception that extended generations of a heuristic will lead to better solutions is nonviable for a stagnated population.

This paper is devoted to the concept of the population and its controlling dynamics. It is shown through experiment that population control is an effective tool in sustaining the diversity of the population, which in turn leads to more viable regions of search space for exploration.

1 POPULATION DYNAMICS

One of the most challenging optimization problems is *permutative based* combinatorial optimization. This class of problem, harbours some of the most famous optimization problems like travelling salesman (TSP) and vehicle routing problem (VRP).

What makes a permutative problem complex is that the solution representation is very concise, since it must have a discrete number of values, and each occupied variable in the solution is unique. Given a problem of size *n*, a representation can be described as $x = \{x_1, x_2, x_3, ..., x_n\}$, where each value x_1 in the solution is unique and the entire set of solutions is an integer representation from $\{1, n\}$.

From an optimization point of view, this represents a number of problems. Firstly, the search space is discrete and a number of validations inevitably have to be conducted in order to have a viable solution. Secondly, the search space is very large, to the scale of *n*!. Consequently, these problems are generally termed *NP* or *NP Hard* (Hochbam (1997)).

The usual approach is to explore the search space in the neighbourhood of good solutions in order to find better solutions. This unfortunately has the effect of converging the population, which then leads to stagnation. The usual term is *local optima convergence/stagnation*.

This research looks at the *diversity* of the population in order to aid the application of metaheuristics. A permutative solution and its representation present some advantages to this effect. The usual measure of a solution is its fitness, in respect to the problem being solved. In a permutative solution, the distinct ordering of values gives

Proceedings 23rd European Conference on Modelling and Simulation ©ECMS Javier Otamendi, Andrzej Bargiela, José Luis Montes, Luis Miguel Doncel Pedrera (Editors)
ISBN: 978-0-9553018-8-9 / ISBN: 978-0-9553018-9-6 (CD)

the opportunity to have other measures of diversity.

The following sections describe the approach developed within this scope of research. The first part discusses the need for having distinct initial population. The second part describes some fields of measure of diversity followed by the approach of clustering of the solutions. Differential Evolution Algorithm is then presented to which the clustered population has been applied. The subsequent sections describe the extensive experimentation conducted on Quadratic Assignment problems and analysis of the dynamic nature of clustering populations.

1.1 Initial Population

The basic concept is to have a "clustured" population. This is an ideal position from which to guage the effectiveness of the heuristic. Also, this provides a better initial point for scheduling problems. An ideal explanation is the TSP, where adjacent cities usually provide minimal travel distance.

The initial population, *P*, for this heuristic is partially stochastic and partly deterministic. The population is divided into *n* sub-populations, *SPs*, n/2 randomly generated (SP_{rand}) and the rest structurally generated (SP_{struct}).

The formulation for SP_{rand} is fairly simple. A random permutative string is generated for each solution till a specified number given as P_{size}.

The structured population SP_{struct} is somewhat more complex. It is made of two parts. In the first part an initial solution is generated with ascending values given as $x_{ascending} = \{1, 2, 3, .., n\}$, where *n* is the size of the problem. In order to obtain a structured solution, the first solution is segmented and recombined in different orders to produce different combinations. The first segmentation occurs at $n/2$ and the two halvesś are swapped to produce the second solution. The second fragmentation occurs by the factor 3; $n/2$. Three regions of solutions now exist. The number of possible recombination′s that can exist is 3! = 6. At this point there are nine solutions in the SP_{struct}. The general representation is given as:

$$k \geq 1 + 2! + 3! + ... + z! \tag{1}$$

where *z* is the total number of permutations possible and *k* is P_{size}/n.

The second part of SP_{struct} is made from solution in descending order $x_{desending} = \{n, ..., 2, 1\}$. A similar approach is used for generating the rest of the possible permutations. The end result is that *n* separate populations exist and are used independently by the underlying heuristic.

1.2 Solution Dynamics

A solution represented as $x = \{x_1, x_2, ..., x_n\}$, where *n* is the number of variables, within a population has a number of attributes. Usually the most visible is its fitness value, by which it is measured within the population. This approach is not so viable in order to measure the diversity of the solution in the population. In retrospect, a single solution is assigned a number of attributes for measure, as given in Table 1. The most important attribute is the *deviation* (the difference between successive values in a solution). Since we are using only permutative solutions, deviation or *ordering* of the solution is important. This is due to the fact that each value in the solution is unique. Each value in the solution has a unique footprint in the search space. The formulation for deviation is given as:

Table 1: Solution Parameters

Parameter	Description	Activity
Deviation	Measure of deviation	Control
Spread	Alignment of solution	Control
Life	Generation cycles	Selection
Offspring	No. of offspring′s	Selection

Table 2: Spread generalization

Spread	Generalization
> 0	Forward spread
0	Even spread
< 0	Reverse spread

$$\delta = \left(\frac{\sum_{i=1}^{n-1} |x_i - x_{i+1}|}{n} \right) x_i \in \{x_1, x_2, ..., x_n\} \tag{2}$$

Spread of a solution gives the alignment of the solution. Each permutative solution has a specific ordering, whether it is *forward* aligned or *reverse* aligned. Whereas deviation measures the distance between adjacent solutions, spread is the measure of the hierarchy of subsequent solutions given as:

$$\partial = \begin{cases} +1 & if\,(x_{i+1} - x_i) \geq 1 \\ -1 & if\,(x_{i+1} - x_i) \leq 1 \\ i \in \{1, 2, ..., n\} \end{cases} \tag{3}$$

The generalisation of *spread* is given in Table 2. *Life* is the number of generations the solution has survived in the population and *Offspring* is the number of viable solutions that have been created from that particular solution. These two variables are used for evaluating the competitiveness of different solutions.

1.3 Clustering

Within the population, certain solutions exhibit attracting features. These points are usually local optima regions, which draw the solutions together. The approach utilized is to subdivide the population in clusters, each cluster a distinct distance from another. Two controlling parameters are now defined which control the clusters.

Cluster Attractor C_A: The distance that each segment of solution has to differ from each other. The C_A is given in (4).

$$C_A \in [0.1, 1+) \tag{4}$$

Table 3: Illustrative example

Solution	Dev	Spread	Clust	C_A	Fitness
1 2 3 4 5 6 7 8 9 10	0.9	+9	1	1.2	1592
1 2 5 6 9 10 3 4 7 8	2.1	+7	2	1.1	1559
10 9 8 7 6 5 4 3 2 1	0.9	-9	1	1.2	1567
10 9 6 5 2 1 8 7 4 3	2.1	-7	2	1.1	1547
6 4 8 2 3 9 1 5 7 10	3.7	+3	3	0.4	1765
9 4 3 7 5 2 10 1 6 8	4.1	-1	4	-	1788
8 7 4 1 3 6 5 2 10 9	2.5	-3	2	1.1	1678
2 5 7 9 1 8 10 3 6 4	3.6	+3	3	0.4	1686
6 1 3 9 7 10 5 2 8 4	3.6	-1	3	0.4	1654
5 9 2 6 8 3 1 7 10 4	3.9	+1	4	-	1545
C_E					90.13379

Within the population indexed by the *deviation*, solutions with similar deviation are clustered together, and each cluster is separated by at least a single C_A as seen in (5).

$$(\delta_1, \delta_2, ..., \delta_i) \overset{C_A}{\leftrightarrow} (\delta_{i+1}, \delta_{i+2}, ..., \delta_{2i}) \overset{C_A}{\leftrightarrow} \\ ... \overset{C_A}{\leftrightarrow} (\delta_{3i+1}, \delta_{3i2}, ..., \delta_{4i}) \tag{5}$$

An illustrative example is presented to showcase how this approach works. Assume a group of solutions as presented in Table 3.

Each solution is evaluated for its deviation and spread as given in columns 2 and 3. Column 4 gives the cluster to which the solution belongs, and column 5 gives the value of C_A by which each cluster is separated from each other. The graphical representation is given in Fig. 1.

The second controlling factor is the *Cluster Edge* C_E . Whereas C_A is the mapping of individual solutions, C_E is the measure of the entire population. C_E is the measure of the deviation of the fitness of the population and to prevent the population from stagnating to any fitness minima.

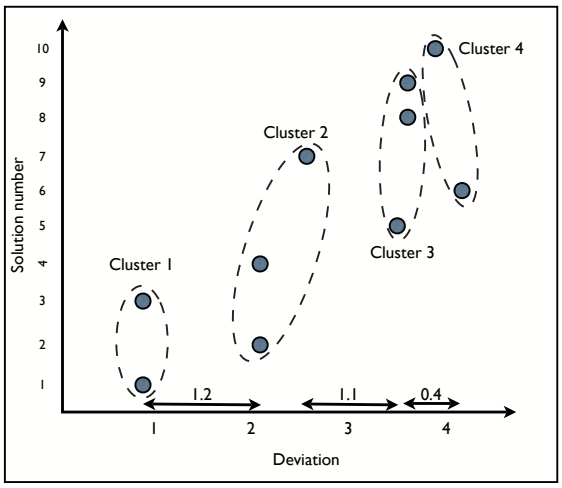

Figure 1: Graphical representation of population

Table 4: Selection criteria

Variables	Criteria
Fitness	Improves clusters best solution
C_A	Increases the value of C_A
C_E	Decreases

Table 5: Deletion criteria

Variables	Criteria
Life	High
Offspring	Low
C_A	Decreases

1.4 Selection and Deletion

Selection of the next generation is based on a tier-based system. If the new solution improves on the global minima, it is then accepted in the solution. Otherwise, competing clusters jockey for the new solution. Initially the solution is mapped for its deviation. This deviation is then mapped to the corresponding cluster.

Within the cluster, the placement of the solution is evaluated. If the new solution corresponds to an existing solution, or reduces the threshold C_A value of the cluster, then it is discarded.

The solution is accepted if it improves on the C_A value of the cluster (hence improving diversity) and also to some extent keeps the balance of the C_E . If the cluster has less than average solutions, then the new solution is admitted.

Table 4 gives the selection criteria. Once the solution is added to the cluster, another solution can be discarded. This solution is usually elected from the middle placed solutions in the cluster, whose fitness is not in the top 5% of the population. If no such solutions exist, then the average rated solution is removed. Solution with high *Life* and low *Offspring* are discarded, since they are considered dormant within the cluster.

Table 5 gives the deletion criteria.

2 DIFFERENTIAL EVOLUTION

Differential Evolution (DE) given by Price (1999), is the heuristic which was used in conjunction with the clustered population. DE uses a vector perturbation methodology for crossover.

Each solution is visualized as a vector in search space. A new vector is created by the combination of four unique vectors. A schematic of DE applied to clustering is given in Fig. 2.

There are ten working strategies for DE, but the one selected for implementation is the DE/rand/2/bin represented as in (6):

$$U_{i,G+1} = x_{best,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G} - x_{j,r_3,G} - x_{j,r_4,G}) \tag{6}$$

This strategy was selected since it maps to the four unique clusters in the *SP*. The best solution is selected from the entire *SP* based on fitness value. Then, each random solution is selected from each distinct cluster. Again

1. Input: $D, G_{\max}, NP \geq 4, F \in (0,1+), CR \in [0,1]$, and initial bounds: $\vec{x}^{(lo)}, \vec{x}^{(hi)}$.

2. While $G < G_{\max}$

$\forall i \leq NP$:

3. Mutate and recombine:

3.1 $r_1, r_2, r_3, r_4 \in \{1,2,....,P_{size}\}$, randomly selected from each cluster

3.2 $j_{rand} \in \{1,2,...,D\}$, randomly selected once each i

3.3 $\forall j \leq D, u_{j,i,G+1} = \begin{cases} x_{best,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G} - x_{j,r_3,G} - x_{j,r_4,G}) \\ \quad \text{if } (rand_j[0,1] < CR \vee j = j_{rand}) \\ x_{j,i,G} \quad \text{otherwise} \end{cases}$

4. Select Criteria

$G = G + 1$

Figure 2: DE selection

the selected values are checked for opposing *spread*. If the spread is identical, then a second round of selection occurs. A schematic is given in Fig. 3. The selection

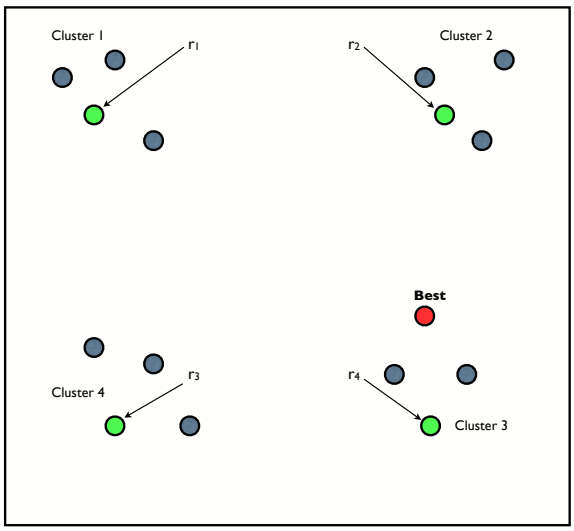

Figure 3: DE selection

of the cluster is random, so r_1 can be selected from any cluster with no preference. These values are subtracted given as $x_{j,r_1,G} - x_{j,r_2,G} - x_{j,r_3,G} - x_{j,r_4,G}$. The resulting value is multiplied by the scaling factor F and added to the best solution as given in Fig. 4. The resulting value is only accepted in the new solution if a generated random number is below the given threshold provided by the controlling parameter of *CR*. This procedure provides added stochasticity to the heuristic.

As can be envisioned, the resulting value obtained is a real number. The repairment routine of Onwubolu and Davendra (2009) is used for repairment of the solutions.

Since it is also possible to obtain identical values in the solution after DE crossover, the repairment routine is applied to repair the solutions.

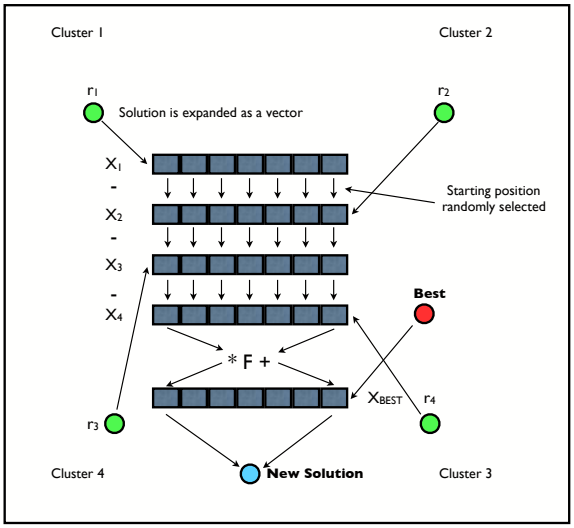

Figure 4: DE crossover

3 GENERAL TEMPLATE

Collating all the piecewise explanation, a general generic template is now described.

1. *Initialize*: Assign the problem size n, population size P_{size}, sub population sizes SP_{struct} SP_{rand}, and the control parameters of C_A and C_E .

2. *Generate:* Randomly create SP_{rand}, halves the size of P_{size}, and then structurally create SP_{struct}. These two form the basis of the population.

3. *Calculate:* Calculate the *deviation* and *spread* of each solution. Taking *deviation* values, configure the population into four clusters. The minimal separation value between the clusters is assigned as C_A. Taking the entire *SP*, the standard deviation of the *fitness* is computed. This is labelled as the C_E.

4. *Generation*

   (a) DE is applied to each *SP* in turn.

   (b) The new solution is calculated for its *deviation* and *spread*.

   (c) Using the selection criteria, the solution is placed within the cluster corresponding to its deviation. If replicated solutions exist, then it is discarded. Selection is based on *fitness* and the move of the C_A and C_E.

5. *Re-calculation:* The SP is re-calculated for its cluster boundaries.

6. *Dynamic clustering:* If the value of C_A has decreased, then the boundary solutions are reconfigured. The C_E value is calculated for the new population.

4 FLOW SHOP SCHEDULING

One of the common manufacturing tasks is *scheduling*. Often in most manufacturing systems, a number of tasks have to be completed on every *job*. Usually all these jobs have to follow the same route through the different *machines*, which are set up in a series. Such an environment is called a *flow shop* (FSS) Pinedo (1995).

The standard three-field notation Lawler et al. (2006) used is that for representing a scheduling problem as $\alpha|\beta|F(C)$, where α describes the machine environment, β describes the deviations from standard scheduling assumptions, and *F(C)* describes the objective *C* being optimised. This research solves the generic flow shop problem represented as $n/m/f||F(C_{\max})$.

Stating these problem descriptions more elaborately, the minimization of completion time (makespan) for a flow shop schedule is equivalent to minimizing the objective function $\Im$:

$$\Im = \sum_{j=1}^{n} C_{m,j} \tag{7}$$

$$s.t. \quad C_{i,j} = \max\left(C_{i-1,j}, C_{i,j-1}\right) + P_{i,j} \tag{8}$$

where $C_{m,j} =$ is the completion time of job j, $C_{1,1} = k$ (any given value), $C_{i,j} = \sum_{k=1}^{j} C_{1,k}$; $C_{j,i} = \sum_{k=1}^{i} C_{k,1}$, $i \Rightarrow$ machine number,$j \Rightarrow$ job in sequence, $P_{i,j} \Rightarrow$ processing time of job *j* on machine *i*. For a given sequence, the mean flow time, $MFT = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}$, while the condition for tardiness is $c_{m,j} > d_j$. The constraint of Equation 8 applies to these two problem descriptions.

5 EXPERIMENTATION

The first sets of Flowshop scheduling benchmark problems are Car Carlier (1978), Rec Reeves (1995) and Hel Heller (1960) benchmark sets. A total of 31 instances exist, each of varying size and difficulty Ponnambalam et al. (2001).

Comparison is done with other published heuristics on the same problem instances. Comparison of the DE_{Clus} heuristics is done with the Improved Genetic Algorithm (IGA) and Multiagent Evolutionary Algorithm (MAEA) Hu et al. (2006) and the Hybrid Genetic Algorithm (HGA) and Othogonal Genetic Algorithm (OGA) of Tseng and Lin (2006). The results are given in Table 6.

The results of DE were obtained from Davendra and Onwubolu (2007). In 11 instances, the optimal value was obtained, and on average the percentage increase was below 1%. Using clustering, DE_{clus} markedly improves all the solutions. This is clearly seen in the large problems sizes of 50 jobs and more. The improvement is clearly in excess of 1.5%.

Table 7: Comparison of Taillard Instances

	PSO_{spv}		DE_{spv+ex}		DE		DE_{clus}	
Problem	Δavg	Δstd	Δavg	Δstd	Δavg	Δstd	Δavg	Δstd
20x5	1.71	1.25	0.69	0.64	0.98	0.66	**0.55**	0.71
20x10	3.28	1.19	2.01	0.93	1.81	0.77	**1.32**	0.98
20x20	2.84	1.15	1.85	0.87	1.75	0.57	**0.98**	1.32
50x5	1.15	0.7	0.41	0.37	0.4	0.36	**0.33**	0.76
50x10	4.83	1.16	**2.41**	0.9	3.18	0.94	3.13	0.77
50x20	6.68	1.35	**3.59**	0.78	4.05	0.65	3.67	0.56
100x5	0.59	0.34	**0.21**	0.21	0.41	0.29	0.38	0.54
100x10	3.26	1.04	1.41	0.57	1.46	0.36	**1.31**	0.32
100x20	7.19	0.99	3.11	0.55	3.61	0.36	**2.23**	0.45
200x10	2.47	0.71	1.06	0.35	0.95	0.18	**0.69**	0.54

In general comparison with published results, the clustered approach of DE_{clus} is one of the top two performing heuristics. MAEA approach is the best comparative heuristic and in comparison with DE_{clus}, even though MAEA obtains more optimal solutions, DE_{clus}, performs more consistently in large problems.

The second set of benchmark problems is referenced from Taillard (1993). These sets of problems have been extensively evaluated by Reeves and Yamada (1998). This benchmark set contains 120 particularly hard instances each of 10 different size, selected from a large number of randomly generated problems.

The results are tabulated as quality solutions with the percentage relative increase in makespan with respect to the upper bound provided by Taillard (1993). To be specific the formulation is given as:

$$\Delta_{avg} = \frac{(H - U) \times 100}{U} \tag{9}$$

where *H* denotes the value of the makespan that is produced by the utilized algorithm and *U* is the upper bound or the lower bound as computed.

Comparison is done with the results obtained for DE_{clus} with those produced by Particle Swarm Optimization PSO_{spv} and DE with local search $DE_{spv+exchange}$ as in Tasgetiren et al. (2004b) and Tasgetiren et al. (2004a) and compared in Table 7.

DE_{clus} obtains better results in the instances of (20x5, 20x10, 20x20, 50x5,100x10, 100x20, 200x10) and $DE_{spv+exchange}$ performing best in the remaining (50x10, 50x20, 100x5) instances. The advantage of $DE_{spv+exchange}$ is the fact that it employs local search, whereas DE_{clus} is employing 2-opt local search as given in Onwubolu and Davendra (2009).

DE with clustering produces better results than DE without clustering, validating the approach of embedding population dynamics.

6 ANALYSIS AND CONCLUSION

A detailed operational analysis of the heuristic is now presented. A sample program output is presented to illustrate the heuristic behaviour during experimentation. Fig. 5 and Fig. 6 gives the first and final populations mapped on the deviation space.

Table 6: Comparison of clustered heuristics with other published heuristics

Name	n x m	Cost	H-GA	OGA	IGA	MAEA	DE	DE_{Clus}
Car1	11x5	7038	**0**	**0**	**0**	**0**	**0**	**0**
Car2	13x4	7166	**0**	**0**	**0**	**0**	**0**	**0**
Car3	12x5	7312	**0**	**0**	**0**	**0**	**0**	**0**
Car4	14x4	8003	**0**	**0**	**0**	**0**	**0**	**0**
Car5	10x6	7720	**0**	**0**	**0**	**0**	**0**	**0**
Car6	8x9	8505	**0**	**0**	**0**	**0**	**0**	**0**
Car7	7x7	6590	**0**	**0**	**0**	**0**	**0**	**0**
Car8	8x8	8366	**0**	**0**	**0**	**0**	**0**	**0**
Rec01	20x5	1247	0	0.04	**0**	**0**	**0**	**0**
Rec03	20x5	1109	**0**	**0**	**0**	**0**	**0**	**0**
Rec05	20x5	1242	0.08	0.21	**0**	**0**	**0**	**0**
Rec07	20x10	1566	**0**	0.79	**0**	**0**	**0**	**0**
Rec09	20x10	1537	**0**	0.35	**0**	**0**	**0**	**0**
Rec11	20x10	1431	**0**	0.91	**0**	**0**	**0**	**0**
Rec13	20x15	1930	0.52	1.08	0.62	**0**	0.45	0.31
Rec15	20x15	1950	0.92	1.23	0.46	**0**	0.32	0.28
Rec17	20x15	1902	1.26	2.08	1.73	**0**	0.29	0.28
Rec19	30x10	2093	0.38	1.76	1.09	**0.28**	0.42	0.338
Rec21	30x10	2017	0.89	1.64	1.44	0.44	0.39	**0.38**
Rec23	30x10	2011	0.45	1.9	0.45	0.44	**0.21**	**0.21**
Rec25	30x15	2513	1.03	2.67	1.63	0.43	0.32	**0.29**
Rec27	30x15	2373	1.18	2.09	0.8	0.56	0.42	**0.27**
Rec29	30x15	2287	1.05	3.28	1.53	0.78	0.61	**0.34**
Rec31	50x10	3045	0.56	1.49	0.49	**0.1**	0.7	0.32
Rec33	50x10	3114	**0**	1.87	0.13	**0**	0.84	0.28
Rec35	50x10	3277	**0**	**0**	**0**	**0**	0.91	0.27
Rec37	75x20	4951	2.54	3.41	2.26	2.72	1.32	**0.33**
Rec39	75x20	5087	1.79	2.28	1.14	1.61	1.56	**0.29**
Rec41	75x20	4960	2.82	3.43	3.27	2.7	1.98	**0.28**
Hel01	100x10	513	-	-	-	**0.38**	2.1	0.53
Hel02	20x10	135	-	-	-	**0**	1.97	**0**

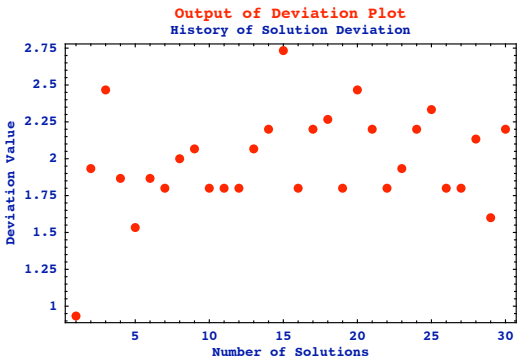

Figure 5: DE_{clust} initial population

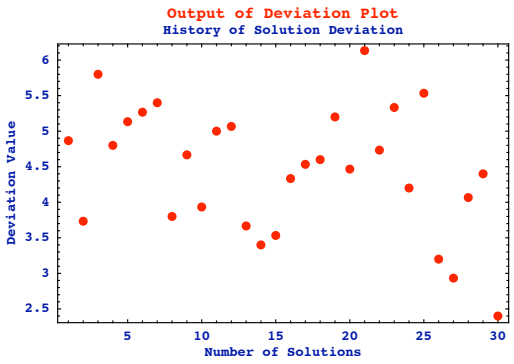

Figure 6: DE_{clust} final population

The evident feature of the two plots is that the diversity of the solutions increased over the generations. Another feature is that separate search spaces were explored since the deviation value increased over the generations.

The second controlling parameter of C_E, which controls the grouping of the population, is given in Fig. 7. As seen from 7, there is a general decline of C_E within DE_{clust}. This is evident for a major migration of the cluster.

The history of the best solution is presented in Fig. 8. Clustering improves solution diversity, however the importance of C_E can be easily seen when comparing the two plots of C_E in Fig. 7 and best solution in Fig. 8. A correlation is observed between the two plots, since any major activity in C_E has a flow on impact to the best individual. The explanation is that once the cluster drifts into new solution spaces, a rapid divergence occurs, which leads to possibly discovery of new optimal solutions.

Once the local space is searched, a different directional search occurs, which generally results in an increase of the value of C_E, as observed from the two different plots.

Another unique feature of the experimentation as the fact that approximately 45% of solutions were obtained from the structured population. This validates the approach of utilising a structured initial population alongside a random population.

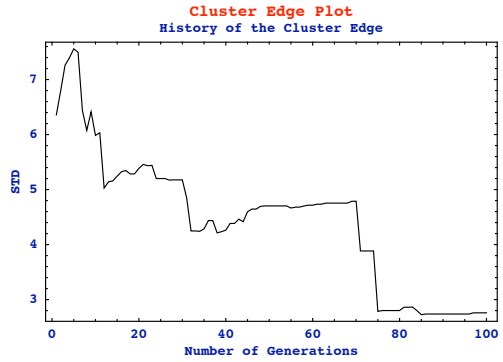

Figure 7: DE_{clust} C_E values over the generations

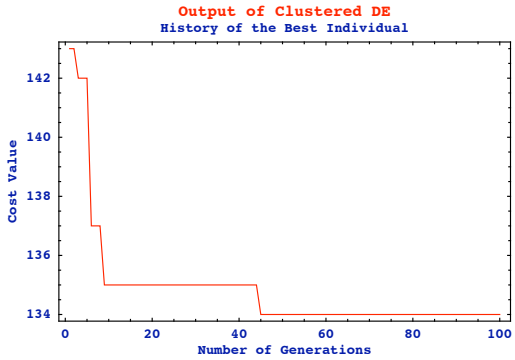

Figure 8: Best solution history of DE_{clust}

REFERENCES

Carlier, J. (1978). Ordonnancements a contraintes disjonctives. *Operations Research*, 12:333–351.

Davendra, D. and Onwubolu, G. (2007). Flow shop scheduling using enhanced differential evolution. In *Proc.21 European Conference on Modeling and Simulation*, pages 259–264, Prague, Czech Rep.

Dreo, J., Petrowski, A., Siarry, P., and Taillard, E. (2006). *Metaheuristics for Hard Optimization*. Springer-Verlag, Germany.

Heller, J. (1960). Some numerical experiments for an mj flow shop and its decision- theoretical aspects. *Operations Research*, 8:178–184.

Hochbam, D., editor (1997). *Approximation Algorithms for NP Hard Problems*. PWS Publishing Company., Washington, USA.

Hu, K., Li, J., Liu, J., and Jiao, L. (2006). Permutation flow-shop scheduling based on multiagent evolutionary algorithm. In Sattar, A. and Kang, B., editors, *AI 2006: Advances in Artificial Intelligence*. Springer-Verlag, Berlin, Germany.

Lawler, E. et al. (2006). Sequencing and scheduling: algorithms and complexity. In Graves, S., Kan, A. R., and Zipkin, P., editors, *In Logistics of Production and Inventory*. North Holland, Amsterdam, Netherlands.

Onwubolu, G. (2002). *Emerging Optimization Techniques in Production Planning and Control*. Imperial Collage Press, London, England.

Onwubolu, G. and Davendra, D., editors (2009). *Differential Evolution: A Handbook for Global Permutation-based Combinatorial Optimization*. Springer, Germany.

Pinedo, M. (1995). *Scheduling: theory, algorithms and systems*. Prentice Hall, New Jersey.

Ponnambalam, S., Aravindan, P., and Chandrasekhar, S. (2001). Constructive and improvement flow shop scheduling heuristic: an extensive evaluation. *Production Planning and Control*, 12:335–344.

Price, K. (1999). An introduction to differential evolution. In Corne, D., Dorigo, M., and Glover, F., editors, *New ideas in Optimisation*. McGraw Hill, UK.

Reeves, C. (1995). A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, 22:5–13.

Reeves, C. and Yamada, T. (1998). Genetic algorithms, path relinking and flowshop sequencing problem. *Evolutionary Computation*, 6:45–60.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operations Research*, 64:278–285.

Tasgetiren, M., Liang, Y.-C., Sevkli, M., and Gencyilmaz, G. (2004a). Particle swamp optimization algorithm for permutative flowshops sequencing problems. In *Differential Evolution Algorithm for Permutative Flowshops Sequencing Problem with Makespan Criterion*, pages 442–452, Sakaraya, Turkey.

Tasgetiren, M., Sevkli, M., Liang, Y.-C., and Gencyilmaz, G. (2004b). Particle swamp optimization algorithm for permutative flowshops sequencing problems. In *4th International Workshops on Ant Algorithms and Swamp Intelligence*, pages 389–390, Brussel, Belgium.

Tseng, L. and Lin, T. (2006). A hybrid genetic algorithm for the flow-shop scheduling problem. *Lecture Notes in Computer Science*, pages 218–227.

AUTHOR BIOGRAPHIES

DONALD DAVENDRA has a BSc, PGD and MSc in Computing Science from the University of the South Pacific. He is currently completing his PhD in Technical Cybernatics at Tomas Bata University. His email is `davendra@fai.utb.cz`.

IVAN ZELINKA is Assoc. Professor of Informatics and Head of Department of Applied Informatics at Tomas Bata University. His email is `zelinka@fai.utb.cz`.

GODFREY ONWUBOLU is a Professor of Manufacturing and Director of Knowledge Managament and Mining division of DaySpring Global Multinational Inc. His email is `onwubolu_g@dsgm.ca`.