We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

15

Hybrid Differential Evolution – Scatter Search Algorithm for Permutative Optimization

Donald Davendra[1], Ivan Zelinka[1] and Godfrey Onwubolu[2]
[1]*Tomas Bata University in Zlin, Nad Stranemi 4511, Zlin 76001*
[2]*Knowledge Management & Mining*
[1]*Czech Republic*
[2]*Canada*

1. Introduction

Adaptive memory programming approaches have proven effective in finding high quality solutions to many real world intractable problems. Therefore, over the years, researches have attempted to combine the best features from different adaptive memory approaches to derive more powerful hybrid heuristics (Onwubolu, 2002). Combining the best features of different heuristics will give a new heuristic that is superior to the individual systems from which these features are derived.

Differential evolution (DE) algorithm (Price, 1999) is an evolutionary approach which does not inhibit any adaptive memory features. It is however a very powerful and robust heuristic for continuous optimization. Continuous optimization is a very important aspect of optimization; however a heuristics application to permutative optimization is imperative if it is to be generic. Permutative optimization encompasses many aspects of engineering. In practical settings, it is common to observe features which are discrete, such as the different discrete nut and bolt sizes, fixed number of machines in a manufacturing plant or discrete number of buses in a fixed route. All these problems are practical and challenging, which utilize discrete values. The purpose of this paper is then to introduce an enhanced different evolution algorithm for discrete optimization which is hybridized by the adaptive memory heuristic of scatter search (SS) (Glover, 1998).

SS is a highly effective heuristic which is the superset of tabu search (TS) (Glover, 1998). It has been successfully applied to many permutative optimization problems. The objective of the proposed hybrid optimization approach is then to isolate its highly effective intensification and diversification routines and embed it in the EDE structure. The result is a highly evolved hybrid enhanced differential evolution scatter search (HEDE-SS) heuristic.

The hybrid optimization scheme is applied to two difficult permutative optimization problems of quadratic assignment problem (QAP) and the flow shop scheduling problem (FSS). The results generated by the hybrid scheme are then compared with the heuristics of EDE and SS in order to show that the hybrid scheme is an improvement over the original heuristics. Additionally, the results of the hybrid scheme is compared with the optimal results from the operations research (OR) library and with the results obtained by other heuristics for the same problem instances from the literature.

Source: Evolutionary Computation, Book edited by: Wellington Pinheiro dos Santos,
ISBN 978-953-307-008-7, pp. 572, October 2009, I-Tech, Vienna, Austria

This chapter is divided into the following sections; section two presents the two different discrete optimization problems, section three introduces the EDE, SS and the developed hybrid approach, section four gives the experimentation and analysis and section five concludes the research.

2. Permutative optimization

2.1 Quadratic assignment problem

The QAP is a combinatorial optimization problem stated for the first time by Koopmans and Beckman (1957) and is widely regarded as one of the most difficult problem in this class. The approach is to have two matrices of size *n* x *m* given as:

$$A = \left(a_{ij}\right) \tag{1}$$

$$B = \left(b_{ij}\right) \tag{2}$$

The objective is then to find the permutation which minimizes

$$\min_{\pi \in \prod(n)} f(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}\ b_{\pi(i)\pi(j)} \tag{3}$$

where $\prod(n)$ is a set of permutations of *n* elements. QAP is considered a *NP* hard problem (Shani & Gonzalez, 1976) and problem sizes of larger than 20 are considered intractable.

Many application have been identified for QAP, which include amongst others, the allocation of plants to candidate locations; layout of plants; backboard wiring problem; design of control panels and typewriter keyboards; balancing turbine runners; ordering of interrelated data on a magnetic tape; processor-to-processor assignment in a distributed processing environment; placement problem in VLSI design; analyzing chemical reactions for organic compounds; and ranking of archaeological data. The details and references for these and additional applications can be found in Burkard (1991) and Malucelli (1993).

Two approaches have been identified to solve QAP; exact and heuristic algorithms. Exact algorithms for QAP include approaches based on dynamic programming by Christofides and Benavent (1989); cutting planes by Bazaraa and Sherali (1980); and branch and bound by Lawler (1963) and Pardalos and Crouse (1989). Among these, the branch and bound algorithms obtain the best solution, but are unable to solve problems of size larger than n = 20.

For larger sized problems, heuristic approaches have been developed. Some of the most notable are: simulated annealing by Connolly (1990), tabu searches of Taillard (1991), Battiti and Tecchiolli (1994) and Sondergeld and Voβ (1996), the hybrid genetic-tabu searches of Fleurent and Ferland (1994) and more recently the ant colony approach by Gambardella, Taillard and Dorigo (1999).

2.2 Flow shop scheduling

In many manufacturing and assembly facilities a number of operations have to be done on every job. Often, these operations have to be done on all jobs in the same order, which implies that the jobs have to follow the same route. The machines are assumed to be set up

and the environment is referred to as flow shop (Pinedo, 1995). The flow shop can be formatted generally by the sequencing on n jobs on m machines under the precedence condition. The general constraints that are assessed for a flow shop system is the time required to finish all jobs or makespan, minimizing of average flow time, and the maximizing the number of tardy jobs.
When searching for an optimal schedule for an instance of the $Fm||C_{\max}$ problem, the question can arise weather it suffices merely to determine a permutation in which the job traverses the entire system, or more logically to check the possibility for one job to bypass another while waiting in queue for a engaged machine. Changing the sequence of jobs waiting in a queue between two machines may, at times, result in a smaller makespan. Where the number of jobs is small, finding the optimal sequence of jobs which results in the minimal makespan is relatively easy. But more often the number of jobs to be processed is large, which leads to big-O order of $n!$ Consequently, some type of algorithm is essential in these large problems in order to avoid combinatorial explosion (Onwubolu, 2002).
The minimization of completion time for a flow shop schedule is equivalent to minimizing the objective function $\Im$.

$$\Im = \sum_{j=1}^{n} C_{m,j} \tag{4}$$

where $C_{m,j}$ is the completion time of job j. To calculate $C_{m,j}$ the recursive procedure is followed for any i^{th} machine j^{th} job as follows:

$$C_{i,j} = \max\left(C_{i-1,j}, C_{i,j-1}\right) + P_{i,j} \tag{5}$$

Where, $C_{1,1} = k$ (any given value) and $C_{i,j} = \sum_{k=1}^{j} C_{1,k};;\ C_{i,j} = \sum_{k=1}^{i} C_{k,1}$ where i is the machine number, j is the job in sequence and $P_{i,j}$ is the processing time of job j on machine i.

3. A hybrid approach to discrete optimization

3.1 Enhanced differential evolution

Developed by Price and Storn (2001), differential evolution (DE) algorithm is a very robust and efficient approach to solve continuous optimization problems. A discrete optimization approach for DE was initially explored by Davendra (2001) and since then by many other researchers (see for details Onwubolu (2001), Onwubolu (2004) and Lampinen and Storn (2004)). The EDE approach by Davendra and Onwubolu (2009) is used as the DE approach for this hybrid system.
Onwubolu and Davendra (2004) developed the approach of a discrete DE, which utilized the highly effective approach of *Forward Transformation* and *Backward Transformation* by Onwubolu (2001) in the conversion of a discrete solution into a continuous solution and vice versa, for the operation of the DE internal mutation schema. This approach was highly effective in solving the scheduling problem of flow shop (FSS). EDE was proposed as an enhancement of the discrete DE in order to improve the quality of the solutions perturbed by DE (Davendra & Onwuolu, 2009).
The basic outline of EDE is given in Figure 1.

- **Initial Phase**
  1. *Population Generation:* An initial number of discrete trial solutions are generated for the initial population.
- **Conversion**
  2. *Forward Transformation:* This conversion scheme transforms the parent solution into the required continuous solution.
  3. *DE Strategy:* The DE strategy transforms the parent solution into the child solution using its inbuilt crossover and mutation schemas.
  4. *Backward Transformation:* This conversion schema transforms the continuous child solution into a discrete solution.
- **Mutation**
  5. *Relative Mutation Schema:* Formulates the child solution into the discrete solution of unique values.
- **Improvement Strategy**
  6. *Mutation:* Standard mutation is applied to obtain a better solution.
  7. *Insertion:* Uses a two-point cascade to obtain a better solution.
  8. *Repeat:* Execute steps 2-7 until reaching a specified cutoff limit on the total number of iterations.
- **Local Search**
  9. *Local Search:* Is initiated if stagnation occurs

Fig. 1. EDE outline

3.2 Scatter search

Scatter search (SS) and its generalized form path relinking (PR) are heuristics which are build on the principles of surrogate constraint design (Glover, 1977). In particular they are designed to capture information not contained separately in the original solutions, and take advantage of auxiliary heuristic solution methods to evaluate the combinations produced and generate new solutions.

The two principles that govern SS are;

(1) Intensification
(2) Diversification

Intensification refers to the role of isolating the best performing solutions from the populations in order to obtain a group of good solutions. Diversification in turn isolates the solutions which are the furthest from the best solutions and combined them with the best solutions. This new pool of solutions is the reference set where crossover occurs in order to create solutions from new solution regions by the combination of the intensified solutions and diversified solutions. Intensification and diversification are commonly termed as adaptive memory programming.

Many applications of discrete programming have emerged from SS. Some of these are: Vehicle Routing (Taillard, 1996), Quadratic Assignment (Cung *et al,* 1997), Job Shop Scheduling (Yamada & Nakano, 1996), Flow Shop Scheduling (Yamada & Reeves, 1997) and Linear Ordering (Laguna, *et al.*, 1997).

3.3 Hybrid approach

EDE is a highly randomized algorithm (Davendra, 2003), which utilizes a high number of randomly generated values for its operation. SS on the other hand is one of the new

optimization algorithms which have adaptive memory programming, enabling it to retain its long term memory in order to find good search space (Glover, 1998). This hybrid approach brings together the highly effective intensification and diversification aspects of SS (Glover, 1998) into the operational domain of EDE.
Figure 2 gives the outline for the hybrid structure.

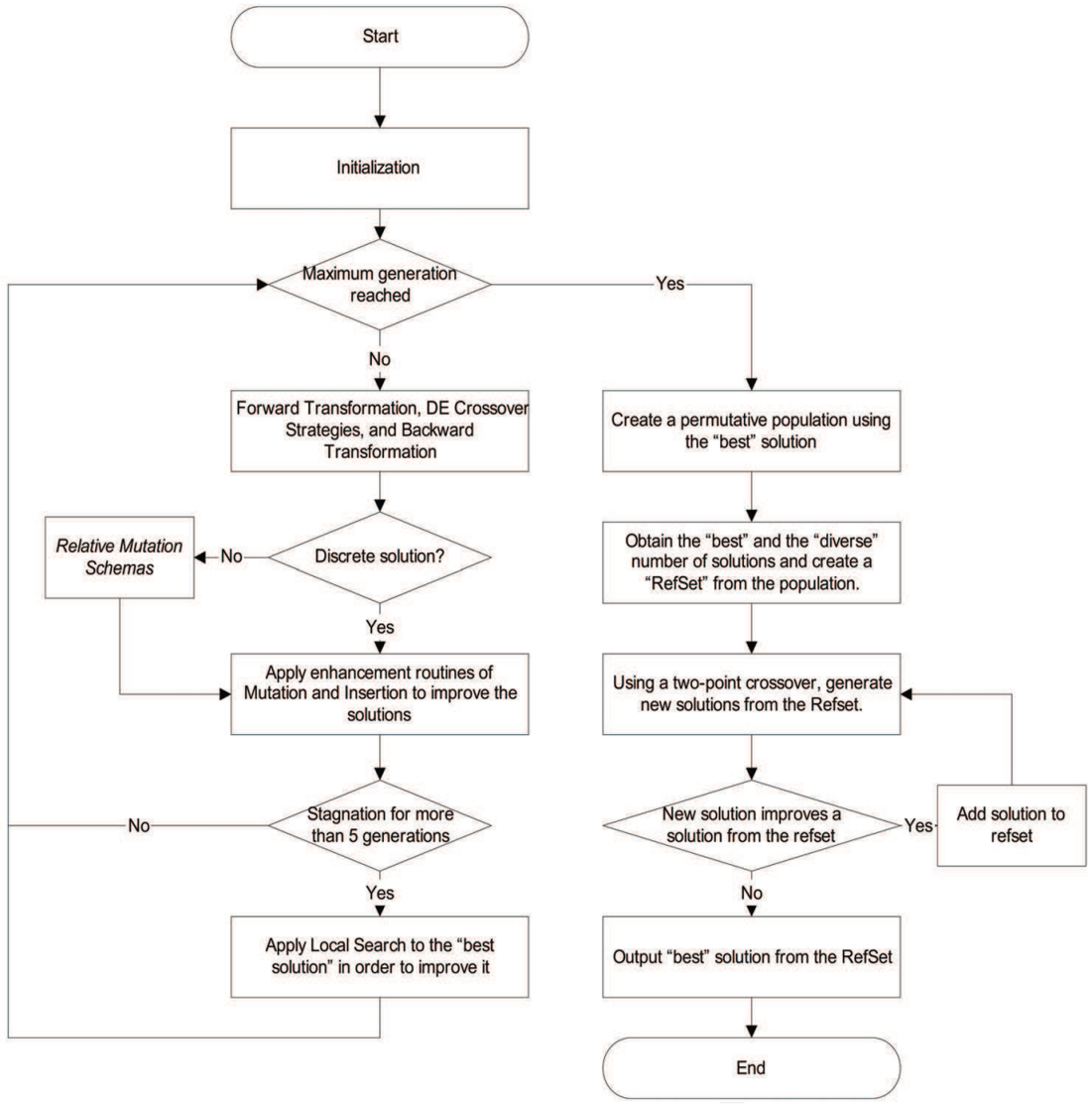

Fig. 2. Hybrid EDE/SS outline

3.3.1 Initialization

The first process in the hybrid EDE SS (HEDE-SS) is the initialization of the population and the operational parameters. HEDE-SS has several operational parameters as given in Table 1.
The lower $x_j^{(lo)}$ bound and the upper $x_j^{(hi)}$ bound specify the range of the solution. The size of the population *NP* is usually in the range of 100 to 500 solutions. The size of each solution *D* is dependent on the problem size at hand. EDE has two tuning parameters of *CR* and *F* which are also initialized; $CR \in [0,1]$ and $F \in (0,1+)$. The Strategy number refers to the type

of DE crossover employed for the problem. The Mutation refers to the type of *Relative Mutation schema* used for the discrete solution. The RefSet size is the number of solutions that are combined for the intensification and diversification.

Parameter	Syntax	Range	Description
Population size	NP	NP ≥ 4	The population size for HEDE-SS
Solution size	D	D ≥ 4	The size of each solution
Lower bound	$x_j^{(lo)}$	$x_j^{(lo)} \geq 1$	The lower bound of the solution
Upper bound	$x_j^{(hi)}$	$x_j^{(hi)} = D$	The upper bound of the solution
Crossover constant	CR	$CR \in [0,1]$	The crossover used for DE perturb
Scaling factor	F	$F \in (0,1+)$	Scaling factor for perturb
Strategy number	Strategy	$Str \in \{1,2,...,10\}$	The strategy to employ
Relative mutation	Mutation	$Mut \in \{1,2,3\}$	Mutation to employ
Reference Set	RefSet	RefSet ≥ 4	Size of RefSet for SS
Generations	$G_{\max}$	$G_{\max} = 500$	Number of generations of EDE

Table 1. Operational parameters for HEDE-SS

The solution created is discrete and given as:

$$x_{j,i}^{(G=0)} = \begin{cases} (\text{int})\left(rand_j[0,1] \cdot \left(x_j^{(hi)} + 1 - x_j^{(lo)}\right) + \left(x_j^{(lo)}\right)\right) \\ \text{if} \quad x_{j,i} \notin \{x_{0,i}, x_{1,i}, ... x_{j-1,i}\} \end{cases} \tag{6}$$

Each solution created is random, and multiple identical solutions can exist within an EDE population as long as it is discrete.

3.3.2 Conversion

The conversion routine is used to transform each discrete solution into a continuous solution since the canonical form of DE only operates in the continuous domain. These solutions after DE internal crossover are retransformed into a discrete solution. The two conversion schemas were developed by Onwubolu (2001). For implementation details please see Onwubolu and Davendra (2004), Onwubolu (2004), Davendra (2001) and Davendra (2003).

The forward transformation is given as:

$$x_{j,i} = x_{j,i}' \tag{7}$$

where $x_{j,i}$ represents a discrete solution and $x_{j,i}'$ represents a continuous solution.

Price and Storn (2001) described ten different working crossover schemas for DE. Each schema developed has a different approach to optimization through the crossover utilized.

The ten strategies are divided into two groups of different crossover schemas; *exponential* and *binomial.*

Exponential refers to the fact that crossover will only occur in one loop until it is within the *CR* bound. The first occurrence of a random number selected between *0* and *1*, going beyond the parameter set by *CR* stops the crossover schema and all the values remaining are left intact.

Binomial on the other hand states that crossover will occur on each of the values whenever a randomly generated number between *0* and *1*, is within the *CR* bound.

The ten different strategies are given in Table 2.

Convention (DE/x/y/z)*	**Representation**
DE/best/1/exp	$u_i = x_{best} + F \bullet \left(x_{r1} - x_{r2} \right)$
DE/rand/1/exp	$u_i = x_{r1} + F \bullet \left(x_{r2} - x_{r3} \right)$
DE/rand-to-best/1/exp	$u_i = x_i + F \bullet \left(x_{best} - x_i \right) + F \bullet \left(x_{r1} - x_{r2} \right)$
DE/best/2/exp	$u_i = x_{best} + F \bullet \left(x_{r1} - x_{r2} - x_{r3} - x_{r4} \right)$
DE/rand/2/exp	$u_i = x_{r5} + F \bullet \left(x_{r1} - x_{r2} - x_{r3} - x_{r4} \right)$
DE/best/1/bin	$u_i = x_{best} + F \bullet \left(x_{r1} - x_{r2} \right)$
DE/rand/1/bin	$u_i = x_{r1} + F \bullet \left(x_{r2} - x_{r3} \right)$
DE/rand-to-best/1/bin	$u_i = x_i + F \bullet \left(x_{best} - x_i \right) + F \bullet \left(x_{r1} - x_{r2} \right)$
DE/best/2/bin	$u_i = x_{best} + F \bullet \left(x_{r1} - x_{r2} - x_{r3} - x_{r4} \right)$
DE/rand/2/bin	$u_i = x_{r5} + F \bullet \left(x_{r1} - x_{r2} - x_{r3} - x_{r4} \right)$

Table 2. DE Crossover schemas

* x – type of solution used for crossover, y – number of solutions used, z – type of crossover used.

Each problem class has to be tuned as to what are its optimal operating parameters.

Once the DE internal crossover schemas have operated, the *backward transformation* changes the values generated into discrete values. The *backward transformation* is the reverse of the *forward transformation* and is given as:

$$x_{j,i}{}' = x_{j,i} \tag{8}$$

3.3.3 Relative mutation

The solution obtained from the *backward transformation* is not always discrete. In Onwubolu and Davendra (2004) up to 80 per cent of all solutions generated were infeasible. A new approach has been developed in order to retain discrete solutions after transformation. Two types of infeasible solution may exist:

(1) Out-of-bound values
(2) Repetitive values

Out-of-bound values are easily handled by HEDE-SS. All bound offending values are simply dragged to the bound they violate.

$$x_{j,i} = \begin{cases} x_j^{(lo)} & \text{if } x_{j,i} < x_j^{(lo)} \\ x_j^{(hi)} & \text{if } x_{j,i} > x_j^{(hi)} \end{cases} \tag{9}$$

In order to remove repetition from the solution, three relative mutation schemas have been developed; *front mutation* (FM), *back mutation* (BM) and *random mutation* (RM).

3.3.3.1 Front Mutation

Front mutation (FM) is the schema which transforms the solution into a discrete solution through the ascending order of index. The solution is firstly sorted into feasible and infeasible values starting from index one. This implies that the value which occurs first in the solution is considered feasible, whereas its next occurrence is infeasible.

$$x_{j,i} = \begin{cases} x_{j,i} & \text{if } x_{j,i} \notin \{x_{1,i},...,x_{j-1,i}\} \\ \tilde{x}_{j,i} & \text{if } x_{j,i} \in \{x_{1,i},...,x_{j-1,i}\} \end{cases} \tag{10}$$

In FM all infeasible values are replaced with feasible values starting from index one. A random value is generated between the lower and upper bound for each infeasible value, and checked to see whether it already exists in the solution. If repetition is detected, then another random value is generated.

$$x_{rand} = \begin{cases} x_{rand} = rnd[1, D] \\ \quad \text{where } x_{rand} \notin \{x_{1,i},...,x_{D,i}\} \end{cases} \tag{11}$$

Each infeasible value within the solution is thus replaced and the solution in now discrete.
FM converts any integer solution into a discrete solution, however a forward bias is shown. FM converts from the front, starting with index one of the solutions. A reverse mutation process is also developed termed the *back mutation.*

3.3.3.2 Back Mutation

Back mutation (BM) is a mutation schema which is the complete opposite of the FM. In BM, all the values in the solutions are sorted from the back of the solution starting with the index *D*. The value occurring last within the solution is considered feasible whereas its earlier placement is considered infeasible.

$$x_{j,i} = \begin{cases} x_{j,i} & \text{if } x_{j,i} \notin \{x_{j+1,i},...,x_{D,i}\} \\ \tilde{x}_{j,i} & \text{if } x_{j,i} \in \{x_{j+1,i},...,x_{D,i}\} \end{cases} \tag{12}$$

As in FM, for each infeasible value, a random number between the lower and upper bounds is obtained and checked against all the values in the solution. If it is unique then it replaces the infeasible value starting from index *D*.
BM is the opposite of FM; however a reverse bias now exists. A bias is not favorable since it limits the search space of a population, and again does not represent a realizable system. The third mutation schema *random mutation* is totally random, with both its sorting and replacement.

3.3.3.3 Random Mutation

Random mutation (RM) is a total random based mutation schema. There exists no bias in this schema as it does with FM and BM. The first process in this schema is to create a random array which contains the indexes for the solution to be checked for repetition.

$$\begin{aligned} \Re_1 &= \{y_1, y_2, y_3, ..., y_D\} \\ &\text{where } \vec{x}^{(lo)} \geq y \leq \vec{x}^{(hi)} \\ &j = 1, 2, ..., D \end{aligned} \tag{13}$$

The value y_1 points to the first value to be checked, and so on until all values within the solutions are checked for repetition.

$$x_{y_j,i} = \begin{cases} x_{y_j,i} & \text{if } x_{y_j,i} \notin \{x_{y_1,i}, ..., x_{y_{j-1},i}\} \\ \tilde{x}_{y_j,i} & \text{if } x_{y_j,i} \in \{x_{y_1,i}, ..., x_{y_{j-1},i}\} \end{cases} \tag{14}$$

Repetitive values are now isolated, however their replacement is yet to be determined. Another random array is now created which will index the infeasible values in their order of replacement.

$$\begin{aligned} \Re_2 &= \{z_1, z_2, z_3, ..., z_D\} \\ &\text{where } \vec{x}^{(lo)} \geq z \leq \vec{x}^{(hi)} \\ &j = 1, 2, ..., D \end{aligned} \tag{15}$$

The value in the solution pointed by the index z_1 is checked for repetition. If repetition is indicated from the previous step, then the repetitive value in replaced by a unique random value as given in equation 13. Using the index array $\Re_2$, all the values in the solution which are infeasible are replaced with feasible values.

RM is truly random, from the point of selection of infeasible values to its replacement. The random arrays enforce random initiation of infeasible values and its replacement. All three mutation schemas; FM, BM and RM are able to convert an integer solution into a discrete solution.

3.3.4 Improvement strategies

Each discrete solution obtained is improved by the two improvement routines of *mutation* and *insertion.* Mutation is referenced from GA, where it has been highly successful in finding local optimal solutions (Goldberg, 1989). Mutation refers to the application of a single swap of values in a solution, in order to exploit better search space. In order for mutation to operate two random numbers are generated.

$$\begin{aligned} r_1, r_2 &\in rand[1, D] \\ &\text{where as } r_1 \neq r_2 \end{aligned} \tag{16}$$

These random numbers are the indexes of the positions of the two values in the solution which are to be swapped. The value in the solution $x_{r_1,i}$ indexed by r_1 is swapped by the value $x_{r_2,i}$ indexed by r_2. The new solution is evaluated with the objective function. Only if a

better objective function is achieved, then the new solution is retained, else the reverse process occurs to obtain the original solution.

$$x_i = \begin{cases} x_i'' & \text{if } f(x_i'') < f(x_i) \\ x_i & otherwise \end{cases} \tag{17}$$

The inclusion of mutation introduces some probability of local improvement of a solution.
Insertion also works along the same lines as mutation. Insertion is the cascade of values between two randomly generated positions in the solution. Insertion is regarded as more robust than mutation since a number of values are involved in its operation. As in mutation, two random values are generated given by equation 18. The value in the solution $x_{r_2,i}$ indexed by r_2, is removed and all the values from the value $x_{r_1,i}$ indexed by r_1 till $x_{r_2-1,i}$ are shifted one index up. The value $x_{r_2,i}$ is inserted in the place indexed by r_1. The new solution is evaluated with the objective function. Only if a better objective function is achieved, then the new solution is retained, else the reverse process occurs to obtain the original solution.

3.3.5 Local search

Stagnation is common amongst population based heuristics. The population converges to local optima and is unable to find new search regions in order to find the global optima. In order to help a heuristic just out of the local optima, a *local search* (LS) routine is embedded in the heuristic. LS is a brute force approach to find a better solution. It is also time and memory expensive. In order to minimize time, a non improving population of ten generations, is classified as stagnation. LS operates on the "best" solution in the population, since the best solution has the highest probability of being in the vicinity of a better solution. The LS employed in HEDE-SS is the 2-Opt algorithm by Onwubolu (2002).

3.3.6 Permutative population

A second population is created in order for the *intensification* and the *diversification* strategies to operate. As stipulated by Glover (1998), for a heuristic to employ memory adaptive programming, each solution in the population should be unique. The "best" solution in the population is isolated and another population is created using the best solution as the permutation base given by:

$$\mathrm{P(h)} = \left(P(h:h), P(h:h-1), ..., P(h:1)\right) \tag{18}$$

The size of the population h is dependent on the size of the solution D and the index $h \le D$ specified. For details see Glover (1998).

3.3.7 Reference set

The reference set is generated by two aspects of the population; intensified solutions and diversified solution. The size of the reference set *refset* is defined at the beginning. It is usual to have half the solutions in the population as intensified and the rest as diversified. Intensified solutions are obtained by evaluating the population and removing the specified *refset/2* best solutions from the population into the refset.

Campos *et al.* (2001) outlined how the diverse solutions are obtained from a population. The way diverse solutions are computed is through the computation of the minimum distances

of each solution in the population to the solutions in *refset*. Then the solution with the maximum of these minimum distances is selected. Population solutions are included in *refset* according to the *maxmin* criterion which maximizes the minimum distance of each candidate solution to all the solutions currently in the reference set. The method starts with *refset*= *Best* and at each step *refset* is extended with a solution P_j from the population $\Im$ to be *refset* = *refset* $\cup\{P_j\}$, and consequently $\Im$ is reduced to $\Im = \Im \setminus \{P_j\}$. Then the distance of solution *P* to every solution currently in the reference set is computed to make possible the selection of a new population solution according to the *maxmin* criterion. More formally, the selection of a population solution is given by

$$P_j = \arg\max \min_{i=1,..,|refset|} \{\zeta_{ij} : j = 1,..,|\Im|\} \tag{19}$$

where ζ is the diversity measure which is the distance between solutions P_i and P_j, which differ from each other by the number of edges which follows as:

$$\zeta_{ij} = \left|\left(P_i \cup P_j\right) \setminus \left(P_i \cap P_j\right)\right| \tag{20}$$

For details see Campos *et al.* (2001).

3.3.8 Combine solutions

The combination method is a key element in scatter search implementation (Campos *et al.*, 2001). For the combination method in HEDE-SS, the GA two-point crossover schema is used. The crossover is similar to the mutation used in EDE. Two random values are generated which are mutually exclusive and also not equal to any of the bounds.

$$\begin{aligned} r_1, r_2 \ &\in \ rand[2, D-1] \\ &\text{where as } r_1 \neq r_2 \end{aligned} \tag{21}$$

Two distinct solutions P_1 and P_2 from the *refset* are selected starting from the first two solutions and using the two random values r_1 and r_2 as indexes to the solutions, the regions between the two bounds in the two solutions are swapped as follows:

$$P_1 = \{x_{11}, x_{12}, ..., x_{1n}\} \text{ Solution 1}$$

$$P_2 = \{x_{21}, x_{22}, ..., x_{2n}\} \text{ Solution 2}$$

Using the two random numbers as indexes the two solutions are now represented as:

$$P_1 = \left\{x_{11}, x_{12}, \underset{r_1}{|}\, x_{13}, x_{14}, x_{15}, \underset{r_2}{|}\, x_{16}, ..., x_{1n}\right\}$$

$$P_2 = \left\{x_{21}, x_{22}, \underset{r_1}{|}\, x_{23}, x_{24}, x_{25}, \underset{r_2}{|}\, x_{26}, ..., x_{2n}\right\}$$

The swap between the regions denoted by the two random numbers in now represented as:

$$P_1 = \left\{ x_{11}, x_{12}, \underset{r_1}{|} x_{23}, x_{24}, x_{25}, \underset{r_2}{|} x_{16}, \ldots, x_{1n} \right\}$$

$$P_2 = \left\{ x_{21}, x_{22}, \underset{r_1}{|} x_{13}, x_{14}, x_{15}, \underset{r_2}{|} x_{26}, \ldots, x_{2n} \right\}$$

The resulting solutions are not discrete and the RM schema is used to transform the solution into a discrete form.

The LS schema is applied to each new solution as part of the improvement routine of HEDE-SS. The new solution is evaluated and compared to the worst solution in the *refset*. If the new solution improves on the worst solution, it then replaces the worst solution in the *refset*. The whole process iterates with solutions selected from the solutions iteratively. On each iteration from the first solution to the last, the amount of addition to the *refset* of new improved solution is recorded. If no new solution is recorded in any iteration, then the *refset* has reached a point of stagnation, and the best value in the *refset* is printed as the best solution for the HEDE-SS.

3.4 Hybrid pseudo-code

A pseudo code representation of hybrid is given in order for the reader to understand how all the different routines are combined together.

/ Initial parameters are first obtained */*

GET *NP, D,* $G_{\max}$ *, CR, F, Strategy Number, Mutation Type,* $x_j^{(lo)}$ $x_j^{(hi)}$ and *refset*

/ Operational parameters are initialized */*

SET $x_{\min}$ *, best_sol, ObjFun, ObjDist, RefSet*

/ Create the initial population of solutions */*

FOR (i = 1 to i ≤ *NP*)

FOR (j = 1 to j ≤ *D*)

GET $x_{j,i}^{(G=0)} = \begin{cases} (\text{int})\left(rand_j[0,1] \bullet \left(x_j^{(hi)} + 1 - x_j^{(lo)}\right) + \left(x_j^{(lo)}\right)\right) \\ \text{if} \quad x_{j,i} \notin \left\{x_{0,i}, x_{1,i}, \ldots x_{j-1,i}\right\} \end{cases}$

ENDFOR

ENDFOR

/ Find the best solution and solution cost */*

$x_{\min} = x_i$ */* The best solution is initialized as the first solution of the population */*

best_sol = 1 */* The best solution index is set to one for the initial solution. */*

FOR (i = 1 to i ≤ *NP*)

IF ($f(x_i) < f(x_{\min})$) */* If the current solution has a less functional value*

SET $x_{\min} = x_i$ *than* $x_{\min}$ *, it replace* $x_{\min}$ *as the best and the*

SET *best_sol* = i *index is appropriately updated. */*

ENDIF

ENDFOR

/ Iterate through the generations */*

FOR (k=1; k ≤ $G_{\max}$)

/ Iterate through the solutions */*

FOR (i = 1 to i ≤ *NP*)

/ Apply Forward Transformation to each value in the solution */*

FOR (j = 1 to j ≤ *D*)

SET $x_{j,i}{}' = -1 + (\alpha \bullet x_{j,i})$ /* α is a constant */

ENDFOR

/ The objective function of the parent solution is calculated */*

SET Parent_ObjFun = $f(x_i)$

/ Select two random solutions from the population other than the current one* x_i' */*

GET $r_1, r_2 \begin{cases} \in rand[1, NP] \\ \text{where as } r_1 \neq r_2 \neq i \end{cases}$

/ Perform D binomial trials, change at least one parameter of the trial solution* x_i' *and perform mutation */*

GET $t = rand[1, D]$ */* A random starting point is obtained */*

FOR (z =1 to z ≤ *D*)

/ If a randomly generated value is less than CR or the counter is within the specified limit */*

IF (($rand[0,1]$ < *CR*) **OR** (z = *D*-1)) **THEN**

/ DE's internal mutation schema operates */*

$$u_i = x_i' + F \bullet (x_{best} - x_i') + F \bullet (x_{r1}' - x_{r2}')$$

/ If condition is not correct the original solution is retained for the new generation */*

ELSE

SET $u_i = x_i'$

ENDIFELSE

/ Increment the counter t */*

$t = z + 1$

ENDFOR

/ Apply Backward Transformation to each value in the solution to obtain the original */*

FOR (j = 1 to j ≤ *D*)

SET $x_{j,i} = (1 + x_{j,i}{}') \bullet \beta$ /* β is a constant */

ENDFOR

/ Check if the solution is feasible or not */*

FOR (j = 1 to j ≤ D)

/ If infeasible solutions are found */*

IF ($x_{j,i} \in \{x_{1,i}, x_{2,i}, x_{2,i}, ..., x_{D,i}\} / x_{j,i}$ **OR** $x^{(hi)} > x_{j,i} < x^{(lo)}$)

/ Relative Mutation Schema first drags all out of bound values to the bound it violates */*

FOR (j = 1 to j ≤ D)

$$\textbf{SET } x_{j,i} = \begin{cases} x_j^{(lo)} & \text{if } x_{j,i} < x_j^{(lo)} \\ x_j^{(hi)} & \text{if } x_{j,i} > x_j^{(hi)} \end{cases}$$

ENDFOR

/ Front mutation is chosen to show how the solution is made discrete */*

FOR (j = 1 to j ≤ D)

/ If a value within the solution is found to be repetitive, a unique random value is created to replace it */*

IF ($x_{j,i} \in \{x_{1,i}, x_{2,i}, x_{2,i}, ..., x_{D,i}\} / x_{j,i}$)

$$\textbf{GET } x_{rand} = \begin{cases} x_{rand} = rnd[1, D] \\ \text{where } x_{rand} \notin \{x_{1,i}, ..., x_{D,i}\} \end{cases}$$

SET $x_{j,i} = x_{rand}$

ENDIF

ENDFOR

ENDIF

ENDFOR

/ Standard mutation is applied to the solution in the hope of getting a better solution*/*

$$\textbf{GET } r_1, r_2 \begin{cases} \in rand[1, D] \\ \text{where as } r_1 \neq r_2 \end{cases}$$

$$u_i = \{x_{1,i}, .., x_{r_1,i}, .., x_{r_2,i}, .., x_{D,i}\}$$

$$u_i'' = \{x_{1,i}, .., x_{r_2,i}, .., x_{r_1,i}, .., x_{D,i}\}$$

/ If the objective function of the new solution is better than the original solution, the new solution is retained in the population */*

IF($f(x_i'') < f(x_i)$)

SET $x_i = x_i''$

ENDIF

/ Insertion is applied to the solution in the hope of getting a better solution*/*

$$\textbf{GET } r_1, r_2 \begin{cases} \in rand[1, D] \\ \text{where } r_1 \neq r_2 \text{ and } r_1 < r_2 \end{cases}$$

$$u_i = \{x_{1,i}, .., x_{r_1,i}, .., x_{r_2,i}, .., x_{D,i}\}$$

$$u_i'' = \{x_{1,i}, .., x_{r_2,i}, x_{r_1,i}, x_{r_1+1,i}, .., x_{r_{2-1},i}, .., x_{D,i}\}$$

/ If the objective function of the new solution is better than the original solution, the new solution is retained in the population */*

IF($f(u_i'') < f(u_i)$)

SET $u_i = u_i''$

ENDIF

/ The objective function of the solution is calculated */*

SET Child_ObjFun = $f(u_i)$

/ If the child improves on the parent, then the child is included in the population */*

IF (Child_ObjFun < Parent_ObjFun)

SET $x_i = u_i$

/ The new solution is also compared against the best solution*/*

IF($f(u_i) < f(x_{\min})$)

SET $x_{\min} = u_i$

SET *best_sol* = i

ENDIF

ENDIF

ENDFOR

ENDFOR

/ Using the best solution* $x_{\min}$ *, generate a permutative population */*

SET $h = D/2$

WHILE(*h* >1)

SET *s* = *h*

SET $r \in \begin{cases} s + rh \le D \\ r > 0 \end{cases}$

WHILE(*s* >1)

$$P(h{:}s) = (s, s+h, s+2h, ..., s+rh)$$

$$s = s - 1$$

ENDWHILE

/ All the sub solutions are appended together for the full solution.*

$$P(h) = (P(h:h), P(h:h-1), ..., P(h:1))$$

$$h = h - 1$$

ENDWHILE

/ Evaluate the population and store the objective function. */*
FOR(i=1 to i < *h*)
SET $ObjFun_i = f\left(P_i\right)$
ENDFOR

/ Remove the best refset/2 solutions from the population and store in refset.*/*
FOR(i=1 to i < *refset*/2)
SET best = $ObjFun_1$
FOR(j=1 to j < *h*)
IF($ObjFun_j \leq$best)
SETbest = $ObjFun_j$
ENDIF
ENDFOR

/ Remove the solution indexed from the population into the refset. */*
MOVE $P_{best} \rightarrow refset_i$
SET $h = h - 1$
ENDFOR

/ Remove the diverse refset/2 solutions from the population and store in refset.*/*
FOR(i= *refset*/2 to i < *refset*)
FOR(j=1 to j < i)

/ Calculate the distance from each solution in refset to the population. */*
FOR(k=1 to k < h)
GET $evaluate\left(refset_j \xrightarrow{\text{distance}} P_k\right)$
ENDFOR

/ Store the maximum of the distance for a particular solution in refset. */*
$ObjDist_j = \max(evaluate)$
ENDFOR

/ Select the minimum of the values in ObjDist and move the corresponding solution to refset. */*
MOVE $P_{\min(ObjDist_j)} \rightarrow refset_i$
ENDFOR

/ Combine each of the solutions in refset with each other to obtain better solutions. */*
FOR(i=1 to i < *refset*-1)
FOR(j=i +1 to j < *refset*)
GET $r_1, r_2 \in rand\left[2, D-1\right]$
where as $r_1 \neq r_2$
FOR(k= r_1 to k≤ r_1)

/ Swap the values between the two solutions */*

$$ref\,set_{i,k} \overset{swap}{\leftrightarrow} ref\,set_{j,k}$$

ENDFOR

/ A relative mutation schema is applied to the solutions to make it discrete. The pseudo code for it is described in the first section */*

/ Local search is applied to the solution to improve it. */*

$$refset_i \overset{Local\ Search}{\rightarrow} refset'_i,\ refset_j \overset{Local\ Search}{\rightarrow} refset'_j$$

/ If this solution improves on the worst solution in Refset, it then replaces that solution in refset. */*

IF($f\left(refset'\right) < f\left(refset_{refset\_size}\right)$)

MOVE $refset' \rightarrow refset_{refset\_size}$

ENDIF

ENDFOR

/ Output the best solution from the refset as the best solution in the heuristic. */*

PRINT $refset_{best}$

4. Experimentation and validation

The validation of this hybrid approach is conducted on the two demanding problems of QAP and FSS. Each experiment is conducted in two phases. The first phase is to experimentally obtain the operating parameters of HEDE-SS. The second phase is the comparison of the hybrid with other established heuristics reported in the literature.

4.1 QAP

The problem instances selected for the QAP are from the Operation Research (OR) library and reported in Gambardella *et al*. (1999). There are two separate problem modules; *regular* and *irregular*. The difference between regular and irregular problems is based on the *flow-dominance* (*fd*), which is used to differentiate among the classes of QAP instances. It is defined as a coefficient of variation of the flow matrix entries multiplied by 100. That is:

$$fd = 100\sigma/\mu \tag{22}$$

where:

$$\mu = \frac{1}{n^2} \bullet \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} \tag{23}$$

$$\sigma = \sqrt{\frac{1}{n^2 - 1} \bullet \sum_{i=1}^{n}\sum_{j=1}^{n}\left(b_{ij} - \mu\right)} \tag{24}$$

4.1.1 Irregular problems

Irregular problems have a flow-dominance statistics larger than 1.2 (Taillard, 1995). Most of the problems come from practical applications or have been randomly generated with non-uniform laws, imitating the distributions observed on real world problems.

The operational parameters of EDE, found through extensive experimentation are given in Table 3 along with the size of the *refset* and the relative mutation schema selected which is RM.

Parameter	Strategy	CR	F	NP	$G_{\max}$	RefSet	Mut
Values	1	0.9	0.3	500	500	30	3

Table 3. HEDE-SS QAP operational values.

Eighteen problem instances are evaluated of four different types; *bur, chr, els, kra* and *tai.* Comparisons were made with other heuristics of the tabu searches of Battiti and Tecchiolli (RTS), Taillard (TT) and the genetic hybrid method of Fleurent and Ferland (GH). A simulated annealing due to Connolly (SA) that is cited as a good implementation by Burkard and Celia was also included. Finally the work covered by Gambardella, Thaillard and Dorigo with Ant Colony (HAS-QAP) is compared as the best results for these instances of QAP.

Table 4 compares all the methods on long execution of $G_{\max}$ =500.

Instance	flow dom	n	Optimal	TT	RTS	SA	GH	HAS-QAP	HEDE-SS
bur26a	2.75	26	5246670	0.208	-	0.1411	0.0120	0	**0**
bur26b	2.75	26	3817852	0.441	-	0.1828	0.0219	0	**0**
bur26c	2.29	26	5426795	0.170	-	0.0742	0	0	**0**
bur26d	2.29	26	3821225	0.249	-	0.0056	0.002	0	**0**
bur26e	2.55	26	5386879	0.076	-	0.1238	0	0	**0**
bur26f	2.55	26	3782044	0.369	-	0.1579	0	0	**0**
bur26g	2.84	26	10117172	0.078	-	0.1688	0	0	**0**
bur26h	2.84	26	7098658	0.349	-	0.1268	0.0003	0	**0**
chr25a	4.15	26	3796	15.969	16.844	12.497	2.6923	3.0822	**0.023**
els19	5.16	19	17212548	21.261	6.714	18.5385	0	0	**0**
kra30a	1.46	30	88900	2.666	2.155	1.4657	0.1338	0.6299	**0**
kra30b	1.46	30	91420	0.478	1.061	1.065	0.0536	0.0711	**0**
tai20b	3.24	20	122455319	6.700	-	14.392	0	0.0905	**0**
tai25b	3.03	25	344355646	11.486	-	8.831	0	0	**0**
tai30b	3.18	30	637117113	13.284	-	13.515	0.0003	0	**0**
tai35b	3.05	35	283315445	10.165	-	6.935	0.1067	0.0256	**0**
tai40b	3.13	40	637250948	9.612	-	5.430	0.2109	0	**0**
tai50b	3.10	50	458821517	7.602	-	4.351	0.2124	0.1916	**0**

Table 4. HEDE-SS comparison with other heuristics for irregular problems.

The average quality of the solutions produced by the methods is shown, measured in per cent above the best solution value known from the OR Library. The best results obtained are indicated in boldface. From Table 4 it is shown that methods involving tabu search and simulated annealing are not well adapted for irregular problems. The well performing heuristics are able to produce solutions with at less than 1% with the same computing power and time. For the *bur...* problem instances the HAS-QAP heuristic shows optimal results, however HEDE-SS outperforms HAS-QAP by obtaining the optimal results. The most challenging problem instance is *chr25a.* All heuristics apart from HEDE-SS obtain very poor results, especially HAS-QAP getting over 3 over cent to the optimal. HEDE-SS outperforms all other heuristic by getting 0.023 per cent to the optimal. The *kra…* problem instances are also dominated by EDE, which obtains the optimal result and outperforms all other heuristics. For the *tai* problems, HEDE-SS obtains the optimal result for all problem instances while HAS-QAP fails to obtain consistent results.

4.1.2 Regular problems

Regular problems also know as unstructured problems are identified as having the flow-dominance statistics less than 1.2 (Taillard, 1995). These instances are randomly generated, and have good solutions spread over the whole solution set.

A comparison with the established algorithms from the literature is also done for the regular problems. The same heuristics as for irregular problems are retained for the comparison as shown in Table 5.

Instance	flow dom	n	Optimal	TT	RTS	SA	GH	HAS-QAP	HEDE-SS
nug20	0.99	20	2570	0	0.911	0.070	0	0	**0**
nug30	1.09	30	6124	0.032	0.872	0.121	0.007	0.098	**0**
sko42	1.06	42	15812	0.039	1.116	0.114	0.003	0.076	**0**
sko49	1.07	49	23386	0.062	0.978	0.133	0.040	0.141	**0**
sko56	1.09	56	34458	0.080	1.082	0.110	0.060	0.101	**0**
tai20a	0.61	20	703482	0.211	0.246	0.716	0.628	0.675	**0**
tai25a	0.60	25	1167256	0.510	0.345	1.002	0.629	1.189	**0**
tai30a	0.59	30	1818146	0.340	0.286	0.907	0.439	1.311	**0**
tai35a	0.58	35	2422002	0.757	0.355	1.345	0.698	1.762	**0**
tai40a	0.60	40	3139370	1.006	0.623	1.307	0.884	1.989	**0**
tai50a	0.60	50	4941410	1.145	0.834	1.539	1.049	2.800	**0**
wil50	0.64	50	48816	0.041	0.504	0.061	0.032	0.061	**0**

Table 5. HEDE-SS comparison with other heuristics for regular problems.

HEDE-SS obtains the optimal result for all instances. The performance of HAS-QAP, which was the closest heuristic in irregular problems, has decreased in regular problems. The results obtained by HAS-QAP for *nug* and *sko* are within tolerable limits, however for *tai* problem instances the results are in excess of 1 per cent to the optimal. HEDE-SS manages to

obtain optimal results for the *nug, sko, tai* problem instances. The only serious competition is seen from GH, which on average outperforms HAS-QAP for the *nug* and *sko* problem instances and RTS which performs best for *tai* problem instances. The conclusion that can be drawn is that no one heuristic performs optimally for all problem instances tested apart from HEDE-SS, which outperforms all other tested heuristics for the regular problems. By the performance of the compared heuristics it can be observed that regular problems are more difficult to solve than irregular problem, yet HEDE-SS manages to perform exceptionally well for both (Davendra & Onwubolu, 2007).

4.2 FSS results

The flow shop experimentation was conducted with the Thaillard benchmark problem sets (Taillard, 1993). These sets of problems have been extensively evaluated: see Nowicki et al (1996), Reeves et al (1998). This benchmark set contains 120 particularly hard instances of 12 different sizes, selected from a large number of randomly generated problems. Of these 100 problem instances were evaluated by HEDE-SS and compared with published work. These instances are: *jobs – machines* (*n* x *m*); 20 x 5, 20 x 10, 20 x 20, 50 x 5, 50 x 10, 50 x 20, 100 x 5, 100 x 10, 100 x 20, 200 x 10, a sample of 10 instances for each set was provided in the OR Library.

A maximum of ten iterations was done for each problem instance. The population was kept at 500, and 500 generations were specified for EDE, and the RefSet was kept at 30 for the SS heuristic as shown in Table 6.

Parameter	**Strategy**	**CR**	**F**	**NP**	$G_{\max}$	**RefSet**	**Mut**
Values	7	0.9	0.4	500	500	30	3

Table 6. HEDE-SS FSS operational values.

The results represented in Table 7 are as quality solutions with the percentage relative increase in makespan with respect to the upper bound provided by Thaillard (1993). To be specific the formulation is given as:

$$\Delta_{avg} = \frac{(H-U)\times 100}{U} \tag{25}$$

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

The results are compared with those produced by Particle Swarm Optimization (PSO*spv*) (Tasgetiren et al, 2004), DE (DE*spv*), DE with local search (DE*spv*+*exchange*) as in Tasgetiren et al (2004) and Enhansed DE (EDE) of Davendra and Onwubolu (2009).

As seen in Table 7, HEDE-SS compares very well with other algorithms. HEDE-SS outperforms PSO and DE*spv*. The only serious competition comes from the new variant of DE*spv*+*exchange*. HEDE-SS outperforms DE*spv*+*exchange* in all but two data sets of 50x20 and 100x20.

The main indicator of the effectiveness of HEDE-SS is its comparison with EDE. The hybrid system is better performing than the canonical approach. SS enhances the application of EDE and thus the hybrid approach can be viewed a s superior heuristic.

	PSO_{spv}		DE_{spv}		$DE_{spv+exchange}$		EDE		HEDE-SS	
	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}	Δ_{avg}	Δ_{std}
20x5	1.71	1.25	2.25	1.37	0.69	0.64	0.98	0.66	0.54	0.51
20x10	3.28	1.19	3.71	1.24	2.01	0.93	1.81	0.77	1.51	0.64
20x20	2.84	1.15	3.03	0.98	1.85	0.87	1.75	0.57	1.62	0.59
50x5	1.15	0.70	0.88	0.52	0.41	0.37	0.40	0.36	0.32	0.21
50x10	4.83	1.16	4.12	1.10	2.41	0.90	3.18	0.94	2.21	1.32
50x20	6.68	1.35	5.56	1.22	3.59	0.78	4.05	0.65	3.79	0.81
100x5	0.59	0.34	0.44	0.29	0.21	0.21	0.41	0.29	0.21	0.33
100x10	3.26	1.04	2.28	0.75	1.41	0.57	1.46	0.36	1.33	0.42
100x20	7.19	0.99	6.78	1.12	3.11	0.55	3.61	0.36	3.12	0.56
200x10	2.47	0.71	1.88	0.69	1.06	0.35	0.95	0.18	0.88	0.29

Table 7. HEDE-SS Taillard problem performance comparisons

5. Conclusion

The hybrid approach of HEDE-SS is highly effective in permutative optimization. This conclusion is reached through the experimentation that has been conducted in order to validate this new approach. Optimal results have been achieved by the HEDE-SS on all but one instance of QAP both regular and irregular problem, and on that instance of *chr25*, HEDE-SS outperforms all other listed heuristics.

In FSS problem instances, the hybrid approach is also a strong performer. It easliy improves the results of EDE and other varients of DE and PSO.

Overall, hybridization can be seen as the next evolution of meta-heuristics. With improving hardware technologies, it thus becomes viable to have multiple heuristics combined for better performace. The main concept of using two unique paradigm based systems such as DE and SS is justified as both complement each other and improve the results.

6. Acknowledgement

The author would like to acknowledge the following research grants for financial support for this research.

1. Grant Agency of the Czech Republic **GARC 102/09/1680**
2. Grant of the Czech Ministry of Education **MSM 7088352102**

7. References

Bazaraa, M.S., &. Sherali., M.D (1980), Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem. *Naval Research Logistics Quarterly* 27, 29-41.

Battitti, R., & Tecchiolli, G. (1994), The reactive tabu search. *ORCA Journal on Computing*, 6, 126-140

Burkard, R. E. (1991), Location with spatial interactions: The quadratic assignment problem, In Mirchandani, P.B. and Francis, R. L. (Eds), *Discrete Location Theory*, John Wiley.

Campos, V., Glover, F., Laguna, M. & Marti´, R. (2001), An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem. *Journal of Global Optimization* 21, 397-414.

Christofides, N. & Benavent E. (1989), An exact algorithm for the quadratic assignment problem. *Operations Research* 37, 760-768,

Chung, V., Mautor, T., Michelon, P. & Tavares, A. (1997), A scatter search based approach for the quadratic assignment problem, In Baeck, T., Michalewick, Z., & Yao, X. (Eds) *Proceedings of ICEC'97* 165-170, IEEE Press

Connolly, D. T. (1990), An improved annealing scheme for the QAP, *European Journal of Operation Research* 46, 93-100.

Davendra, D. (2001), Differential Evolution Algorithm for Flow Shop Scheduling, *Unpublished Bachelor of Science Degree Thesis*, University of the South Pacific.

Davendra, D. (2003), Hybrid Differential Evolution Algorithm for Discrete Domain Problems, *Unpublished Master of Science Degree Thesis*, University of the South Pacific.

Davendra, D. & Onwubolu, G (2007) Enhanced Differential Evolution hybrid Scatter Search for Discrete Optimisation. *Proceeding of the IEEE Congress on Evolutionary Computation*, Sept 25-28, Singapore. Pp. 1156–1162

Davendra, D. & Onwubolu, G. (2009) Forward Backward Transformation. In *Differential Evolution – A handbook for the combinatorial-based permutitive optimization*. Onwubolu G. and Davendra (Eds), D. Spriner, Germany.

Dorigo, M. & Gambardella, L. M. (1997), Ant Colony System: A Co-operative Learning Approach to the Traveling Salesman Problem. *IEEE Transaction on Evolutionary Computations* 1, 53-65.

Fleurent, C., & Ferland, J. A. (1994), Genetic Hybrids for the Quadratic Assignment Problem, *Operations Research Quarterly* 28, 167 – 179.

Gambardella, L. M., Thaillard, E. D., & Dorigo, M. (1999), Ant Colonies for the Quadratic Assignment Problem, *Journal of Operational Research* 50, 167-176.

Glover, F. (1998), A Template for Scatter Search and Path Relinking, In Hao, J. K., Lutton, E., Schoenauer, M. & Snyers, D. (Eds) *Lecture Notes in Computer Science* 12363, 13 – 54.

Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc.

Koopmans, T. C. & Beckmann, M. J. (1957), Assignment problems and the location of economic activities. *Econometrica* 25, 53-76.

Laguna, M., Martí, R. & Campos, V. (1997), Tabu Search with Path Relinking for the Linear Ordering Problem. *Research Report*, University of Colorado.

Lampinen, J. & Storn, R. (2004). Differential Evolution, In Onwubolu, G. C., Babu, B. (eds.), *New Optimization Techniques in Engineering*, Springer Verlag, Germany, 123-163.

Lawler, E. L. (1963), The quadratic assignment problem. *Management Science* 9, 586-599.

Malucelli, F. (1993), Quadratic assignment Problems: Solution Methods and Applications. *Unpublished Doctoral Dissertation*, Departimento di Informatica, Universita di Pisa, Italy.

Nowicki, E. & Smutnicki, C. (1996). A fast tabu search algorithm for the permutative flow shop problem. European Journal of Operations Research, 91, 160-175

Onwubolu, G., C. (2001), Optimization using Differential Evolution Algorithm, *Technical Report TR-2001-05*, IAS, October 2001.

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

Onwubolu, G., C., & Clerc. M. (2004), Optimal path for automated drilling operations by a new heuristic approach using particle swamp optimization. *International Journal of Production Research*, 42, 3, 473-491.

Onwubolu, G., C. & Davendra, D. (2006). Scheduling flow shops using differential evolution algorithm. *European Journal of Operations Research*, 171,674-679

Onwubolu G., C. & Davendra D. (2009) *Differential Evolution – A handbook for the combinatorial-based permutitive optimization*. Spriner, Germany.

OR Library: http://mscmga.ms.ic.ac.uk/info.html

Pardalos, P. M., & Crouse, J. (1989), A parallel algorithm for the quadratic assignment problem, *Proceedings of the Supercomputing 1989 Conference*, ACM Press, 351-360.

Ponnambalam, S. G., Aravindan, P. & 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., & Glover, F. (Eds.), *New Ideas in Optimization*, McGraw Hill International, UK, 79-108.

Price, K., & Storn, R. (2001), Differential Evolution homepage (Website of Price and Storn) as at 2001. http://www.ICSI.Berkeley.edu/~storn/code.html

Reeves, C. & Yamada, T. (1998). Genetic Algorithms, path relinking and flowshop sequencing problem. Evolutionary Computation 6, 45-60.

Sahni, S. & Gonzalez, T. (1976), P-complete approximation problems, *Journal of the ACM,* 23, 555-565.

Sondergeld, L., & Voβ. S. (1996), A star-shaped diversification approach in tabu search, in Osman, I., and Kelly, J. (eds) *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers: Boston, 489-502.

Taillard, E. (1991), Robust taboo search for the quadratic assignment problem, *Parallel Computing,* 17, 443-455

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

Taillard, E. (1995), Comparison of Iterative Searches for the Quadratic Assignment Problem. *Location Science*, 3, 87–105.

Taillard, E., D. (1996), A heuristic column generation method for the heterogeneous VRP. *CRT-96-03*, Centre de recherché sur les transports, Université de Montréal.

Tasgetiren, M. F., Sevkli, M. Liang, Y-C., & Gencyilmaz, G. (2004). Particle swamp optimization algorithm for permutative flowshops sequencing problems, *4th*

International Workshops on Ant Algorithms and Swamp Intelligence, ANTS2004, LNCS 3127, Brussel, Belgium. September 5-8, 389-390.

Tasgetiren, M. F., Liang, Y-C., Sevkli, M. & Gencyilmaz, G. (2004). Differential Evolution Algorithm for Permutative Flowshops Sequencing Problem with Makespan Criterion, *4th International Symposium on Intelligent Manufacturing Systems, IMS2004*, Sakaraya, Turkey. September 5-8, 442-452.

Yamada, T. & Nanako, R. (1996), Scheduling by Genetic Local Search with Multi-Step Crossover. *4th International Conference on Parallel Problem Solving from Nature*, 960-969.

Evolutionary Computation

Edited by Wellington Pinheiro dos Santos

ISBN 978-953-307-008-7
Hard cover, 572 pages
Publisher InTech
Published online 01, October, 2009
Published in print edition October, 2009

This book presents several recent advances on Evolutionary Computation, specially evolution-based optimization methods and hybrid algorithms for several applications, from optimization and learning to pattern recognition and bioinformatics. This book also presents new algorithms based on several analogies and metafores, where one of them is based on philosophy, specifically on the philosophy of praxis and dialectics. In this book it is also presented interesting applications on bioinformatics, specially the use of particle swarms to discover gene expression patterns in DNA microarrays. Therefore, this book features representative work on the field of evolutionary computation and applied sciences. The intended audience is graduate, undergraduate, researchers, and anyone who wishes to become familiar with the latest research work on this field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Donald Davendra, Ivan Zelinka and Godfrey Onwubolu (2009). Hybrid Differential Evolution – Scatter Search Algorithm for Permutative Optimization, Evolutionary Computation, Wellington Pinheiro dos Santos (Ed.), ISBN: 978-953-307-008-7, InTech, Available from: http://www.intechopen.com/books/evolutionary-computation/hybrid-differential-evolution-scatter-search-algorithm-for-permutative-optimization

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen