4

Symbiotic Tabu Search

Ramin Halavati[1] and Saeed Bagheri Shouraki[2]
[1]Iranian Academic Centre for Education, Culture, & Research
[2]Sharif University of Technology
Iran

1. Introduction

Since the introduction of Tabu Search (Glover, 1989 & 1990), (Glover & Laguna, 1997), this simple yet effective heuristic has been used in many applications such as clustering (Liu et al., 2008), rule base generation (Bagis, 2007), feature selection (Oduntan et al., 2008), multi objective optimization (Zhu et al., 2007), (Vilcot, 2007), (Jaeggi et al., 2008), combinatorial optimization (Crainic et al., 2007), (Alvarez-Valdes et al., 2006), continuous optimization (Jaeggi et al., 2008), scheduling (Pan et al., 2007), (Chen et al., 2006), solving graph problems (Mermri et al., 2007), (Öncan et al., 2007), (Brandão & Eglese, 2006), and many more.

Tabu search metaheuristic has been used in two forms by now. The first form is the pure usage of tabu metaheuristic as the search algorithm. In this approach, the algorithm designer just focuses on implementing appropriate neighbour generation functions based on her/his specific task. Examples of this approach are (Pacheco et al., 2007) for urban transport optimization, (Liang & Chao, 2008) for facility layout optimization, (Exler et al., 2007) for control and system design, (Palubeckis, 2007) for maximum diversity problem, (Caserta & Uribe, 2007) for software system reliability, (Watcharasitthiwat et al., 2006) for design of FIR filters, and (Hallberg & Peng, 1996) for multicycle scheduling.

The second approach is to hybridize this idea with another search algorithm or another instance(s) of the same idea. Examples are hybridization with Memetic algorithms (El Fallahi, 2006), Particle Swarm Optimization (Shen et al. 2007), Genetic algorithms (Hou et al., 1999), (Vilcot & Billaut, 2007), and used in multiple/hierarchical form in (Oduntan et al., 2008), (Brandão, 2007), (Crainic et al., 2007), (Pothiya et al., 2007), (Watcharasitthiwat et al., 2006) or in parallel (Al-Yamani et al., 2003), (Bachelet et al., 1996), (Porto & Ribeiro, 1996).

Cited hybridizations can be categorized into two sub groups: either tabu search is the master search algorithm which makes use of another algorithm as slave for neighbour generation, as in (Hou et al., 1999), or the other algorithm is the master algorithm which uses tabu search for either initialization (Vilcot & Billaut, 2007) or local optimization (Gomes et al., 2003). In this chapter, we want to purpose another form of hybrid tabu search approach in which the tabu metaheuristic is neither the master part of the algorithm nor its slave part and two ingredients, tabu metaheuristic and symbiotic evolution, are intermixed so that none has a higher position than the other. The algorithm will be called Symbiotic Tabu Search (STS) and will be a general search and optimization algorithm that makes use of schemata theorem (Holland, 1975), automatically discovers and uses the linkages between solution parts and avoids local optima.

Source: Local Search Techniques: Focus on Tabu Search, Book edited by: Wassim Jaziri, ISBN 978-3-902613-34-9, pp. 278, October 2008, I-Tech, Vienna, Austria

In the rest of the chapter, we will first introduce the rationales behind symbiotic tabu search and why we were led to this idea. Then an implementation will be presented in section 3, followed by its comparison and test results in section 4, and finally the concluding remarks at section 5.

2. The backgrounds

In this section, we will first talk about the linkage problem, which is the main focus of Symbiotic Tabu Search algorithm, then will introduce symbiotic combination operator as an artificial tool inspired from the natural process of symbiogenesis. Then in the next section, we will present STS algorithm as the combination of symbiogenesis and tabu search as a remedy for the linkage problem.

2.1 The linkage problem

One of the earliest ideas in search algorithms was to use the building blocks of a problem (divide and conquer) and to combine the building blocks so that the good features of two partially good solutions would be added together and create a better solution. This was the main idea behind the recombination (cross over) operator of Holland's Genetic Algorithms (Holland, 1975) which be came the root of all evolutionary algorithms. But very soon three problems arouse in this field which were laterly called Linkage Problem: First, how to identify the good building blocks so that their combination will result in a good fitness value (Watson & Pollack, 1999); how to combine these blocks if they are conflicting (Watson & Pollack, 1999); and how to identify and what to do with bad genes which are stuck to good genes in one chromosome, the garbage genes (Forrest & Mitchell, 1993). The first problem results in requirment of prior domain knowledge for positioning genes so that recombination opertor would not break the building blocks, and the second and third problem results in slowing down, and sometime shutting down, the search process by approving and propagating bad genes.

These two problems result in a vast line of search methods based on building blocks or schemata theory with several different view points. The first ideas were based on designing more sophisticated recombination operators for simple genetic algorithms such as the ones with more number of cut points, random cut point positioning, uniform crossover, linear combination of genes, etc., see (Mitchell, 1999) for an extensive list. Regardless of the fact that some of these remedies totally neglect the idea behind schemata and building blocks, prior domain knowledge about the problem is still a serious requirement for selection/design of appropriate operator.

Another idea was to use chromosome reordering operators and repositioning of genes inside the chromosome on the fly such as Inversion operator (Bagley, 1967) and Linkage Learning Genetic Algorithm (Harrik, 1997). In such algorithms, each gene has a location indicator along with its value and both of these parameters change and propagate during evolution. Thus, gene values and locations are optimized together and the algorithm is supposed to rearrange gene locations so that those which have related effects on phenotype move together during application of simple recombination operators. The major reported problem of this approach was premature convergence or reaching a local optima before the genes were appropriately arranged (Newman, 2006), (Pelikan et al, 1999).

The third solution was to use partially specified chromosomes (PSCs) such as in Messy Genetic Algorithms (mGA) (Deb, 1991), (Goldberg et al, 1989), Cooperative Co-Evolutionary

Algorithms (CCEA) (Potter & De Jong, 1994), Symbiotic Evolutionary Adaptation Model (SEAM) (Watson & Pollack, 2000) and, and Incremental Commitment Genetic Algorithm (ICGA) (Watson & Pollack, 1999). In these approaches, the chromosomes have missing values for some locations and cooperation of several chromosomes (based on algorithm strategy) composes a solution. In mGA, this cooperation is minimal and each PSC is evaluated in the context of a template and is compared with other chromosomes which are similar enough to it. In CCEA, genome is split into several sub-genomes by algorithm designer and each pool is evolved in separation from other pools, improving the content of one of these sub-genomes. The cooperation between pools takes place during evaluations were a partial solution from one pool is concatenated to the bests of the other pools for evaluation. In SEAM and ICGA, PSCs are evaluated in the context of other chromosomes. In this approach, a context is a combination of some members of the chromosome pool that fully specify all chromosome locations together. To compute the fitness of a chromosome, all unspecified positions of the chromosome are filled with respective values from the the context and then the fitness value is computed.
The last school of dealing with linkage and building block problems tries to find estimation of distribution of good genes. In contrast with purely evolutionary approaches that put their direct focus on search for good solutions, the algorithms in this group try to estimate the distribution of good genes and construct good solutions based on these estimations. Generally, these approaches require a pre-selected distribution model and the process estimates the parameters of this distribution. To name some algorithms in this category, one can say Estimation of Distribution Algorithm (Larrañaga & Lozano, 2002), Population-Based Incremental Learning Algorithm (Baluja, 1994), Compact Genetic Algorithm (Harik et al, 1998), Extended Compact Genetic Algorithm (Sastry & Goldberg, 2000), Factorized Distribution Algorithm (Mühlenbein & Mahnig, 1999), Bayesian Optimization Algorithm (Pelikan et al, 1999), and Hierarchical Bayesian Optimization Algorithm (Pelikan et al, 2003). Each of these solutions has its own cons and pros and yet none is considered an ultimate remedy for the linkage problem. Some of the approaches require other source of domain knowledge like estimation of distribution of good genes in all fourth approach algorithms and appropriate gene groups in CCEA; Some need excessive computation power for extensive search as in mGA, see more details in (Kargupta, 1995); and at last, some are usable only in very specific purposes such as SEAM, see more details in (Halavati et al, 2007).

2.2 Natural process of symbiogenesis and artificial symbiotic combination operator

The natural process of symbiogenesis (Merezhkovsky, 1909) is the creation of new species from the genetic integration of organisms, called symbionts. Symbiogenesis has enabled some of the major transitions in evolution (Maynard Smith & Szathmary, 1995), including the origin of eukaryotes which include all plants and animals. This kind of genetic integration is quite different from the transfer of genetic information in sexual reproduction. Sexual recombination occurs between similar organisms (i.e. of the same species) and involves the exchange of parts of the genome in a mutually exclusive manner so that every gene acquired from one parent is a gene that cannot be acquired from the other parent. In contrast, symbiotic combination may also occur between genetically unrelated organisms (i.e. different species) and involve the integration of whole genomes. The resultant composite may have all the genes from one symbiont and at the same time acquire any number of genes from the other symbiont (Watson & Pollack, 2000).

Based on this idea, symbiotic combination operator was introduced (Watson & Pollack, 1999), (Watson & Pollack, 2000) as an alternative for sexual recombination operator. This operator takes two PSCs and makes an offspring with the sum of their characteristics, see Figure 1 for an example. Therefore, in contrast to the standard crossover operator that is applied to fully specified chromosomes, symbiotic combination runs over partially specified representations and advances them towards fully specified ones. In the original introduction of this operator ((Watson & Pollack, 1999) and (Watson & Pollack, 2000)), when two chromosomes with conflicting genes were to be merged (i.e. they both had values for one/more specific location and the values contradicted), all these conflicts were resolved to the favour of the first donor. In this text, we do not need this assumption and we simply don't combine two chromosomes that have conflicts.

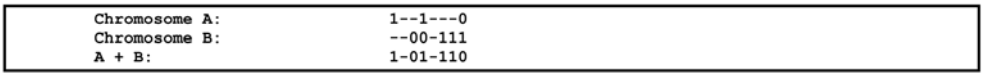


Fig 1. An example of symbiotic combination. Chromosomes A and B, each, have some unspecified locations, shown with '-' mark. Their combination has specified values for all locations that are specified in at least one of the donors. If there would be a conflict between the specified values, like the last gene of the above chromosomes, all conflicts are resolved in favor of one donor, here A.

3. Symbiotic tabu search

3.1 The idea behind symbiotic tabu search

The main target of STS is to use partially specied chromosomes (PSCs) as the core of the search approach, but make the selections using tabu prohibition heuristic on fully specified solutions. To do so, STS uses mutation operators, sometimes combine PSCs using symbiotic combination operator to create chromosomes with more specified locations, and makes the selection for reproduction using tabu heuristic and fitness value of fully specified solutions.

One requirement of a process that evolves PSCs is their evaluation during selection phase. In some specific problems, this is done using a direct evaluation function that can evaluate solutions with missing values; but when this is not available, the most common approaches are to evaluate a partially specified chromsome in the context of other members of the pool such as in SEAM and ICGA or to use a template for missing values as in mGA.

In the first approach, a context is a combination of some members of the pool that fully specify all chromosome locations. To compute the fitness of a chromosome in a context, all unspecified positions of the chromosome are filled with respective values from the context and then fitness value is computed. A major critique of this approach is the detachment between the context and the chromosome: When the combination of several chromosomes (the context) and the chromosome under evaluation results in a good fitness value, it means that the entire group has made a good cooperation and their being together results in a good outcome. So it would be good if the entire composition would get a higher chance of reemergence and survival, but in this class of algorithms, only the single individual under evaluation gets the reward, see (Halavati et al, 2007) for more details on this problem.

The local template solution of mGA also has this problem as selection is done at individual level. Also the process is limited to a local search based on the template that is used for

evaluations and although the template evolves through time, all evaluations are centered around its temporal value in each generation.
To overcome these problems, we propose evaluation of chromosomes in the context of other chromosomes, but the reproduction and selection must also be done at context level instead of individual level: if some chromosomes that are grouped for evaluation gain a high fitness value, all of them get a higher survival and reproduction chance and not just one of them. To implement this idea, we will use the term *assembly* hence forth for a set of none conflicting chromosomes that fully specify all locations; i.e., each location is specified by one and only one member of the assembly. Each member of an assembly will be called a *symbiont*. This way, whenever we need selection or evaluation, we can generate an assembly, evaluate it and if it gains enough credits for reproduction, reproduce its entire content. But after replication (and possible modifications) it can break again into its symbionts.
There are some considerations behind the recommended process:

3.1.1 Avoiding premature convergence

Creating schemata with more specified bits from the best assemblies in each iteration may result in a very fast creation of fully specified chromosomes and premature convergence of the search process. To prevent this, we limit the size of chromosomes that are created during the process to a value that gradually increases while the process goes on. This value will allow the creation of only single gene chromosomes at the beginning of the process and gradually reaches fully specified chromosomes. This will be called gradually cooling the process and the term is chosen to represent a force that prevents emergence of big chromosomes at the beginning of the process, when the pools is hot, and gradually cools the pool based on a predefined schedule, so that bigger chromosomes can emerge. In all of our implementations, this parameter is simply computed by a linear equation based on the generation number but more complicated functions can be used if the problem space is known better.

3.1.2 Avoiding local optima

STS algorithm creates schemata of the best assembly in each iteration by combining some symbionts of the best assembly. This is done to promote solutions similar to the best assembly, but if the best assembly is a local maximum, there would be a good chance that exactly the same assembly will be chosen again as the best assembly of the later steps, because all symbionts of this solution still exist in the pool and some combinations of them are also added. In this case, the local maximum will fill up the chromosome pool very fast with more and more copies of its subcomponents and this increases the possibility of its later creations and may cause the search to get stuck there. The key role of tabu prohibition is in this step where STS keeps a list of the latest best assemblies that have reproduced and avoids them during later assembly creations. This way, a local optimum can not repeat itself and this prohibition makes the algorithm search around the local maximum instead of repeating the exact assembly.

3.1.3 Exploiting partial evaluation function

If a partial evaluation function exists, STS can use it: Whenever such a function exists, STS can use it during assembly generation phase, by picking the first member of the assembly

randomly and then choosing the next members using a tournament selection approach. If the problem does not permit partial chromosome evaluation, this part can be replaced with a random selection of members for assembly build up.

3.1.4 Pruning none fitting chromosomes

Besides promoting and replicating good chromosomes, STS prohibits and decreases the reemergence of bad assemblies. To do so, removal of some chromosomes of assemblies with low fitness values is a good solution, but this must be done considering the fact that some of these chromosomes may also be in assemblies with high fitness values and they must be preserved. So, if a chromosome is part of both high fitness assemblies and low fitness assemblies, it must not be removed.

3.2 The implementation details of symbiotic tabu search

Figures 3 and 4 depict the implementation of STS. Figure 3 presents the main body of the algorithm and Figure 4 shows the assembly generation function which is in charge of creating assemblies when required. Assembly generation function (Figure 4) starts with a random chromosome and keeps adding chromosomes to the assembly so that new members have no conflicts with previous members and add some extra specified bits to the assembly. Two alternatives are shown in the diagram, once a partial evaluation function does not exist, new members are randomly selected from candidate members and when there is such a function, the best of each group of candidates is selected and added.

In the main body of the algorithm (Figure 3), as stated in 3.1.1 subsection and to avoid premature convergence, a size control mechanism is implemented using *MaxSize* variable that is initialized to 2 bit chromosomes at the beginning and is updated at the end of each iteration. A tabu list is also created in step 1 of Figure 3 and everytime an assembly is selected as the best of an iteration, re-creation of it in some further steps is prohibited by putting it in a queue of tabu answers in step 4. Note that assemblies are checked for being tabu in the final stage of assembly generation function and if an assembly with similar values for all locations of the generated assembly was found in tabu list, the newly generated assembly is discarded.

As stated in steps 3, 5, and 6 of Figure 3 diagram, only the best assembly of each generation is selected for mutation and symbiotic combination. In mutation step (5), with a certain probability, a mutated copy of each symbiont of the best assembly is created and added to the pool and in symbiotic combination step (6), combinations of each two symbionts of best assembly are created and added to the pool. At the end of step 6, it is checked that if the size of a combined symbiont exceeds maximum size threshold, it is randomly broken into some fragments of smaller sizes.

After replicating the best assembly, we prohibit the re-emergence of the worst assemblies in step 7 similar to the note at subsection 3.1.4: We separate the generated assemblies into two sets. The winners list includes all symbionts of all assemblies that stand in 25% highest ranks based on their fitness values and the losers list is made up of all symbionts of the worst 25% assemblies. Then, all symbionts of all members of the losers set are removed from the population, except the ones which are also a member of the winners set. Being in both sets is quite possible and frequent as the assembly generation phase may use a chromosome in

several assemblies with different fitness values. Using the above approach, we only remove chromosomes which haven't been able to take part in any good assembly.
After all stated steps, if the replication phase has created any duplicate chromosome in the population, the extra copies are removed in step 8 and if population exceeds a pre-specified threshold, some chromosomes are randomly selected and removed from the pool.

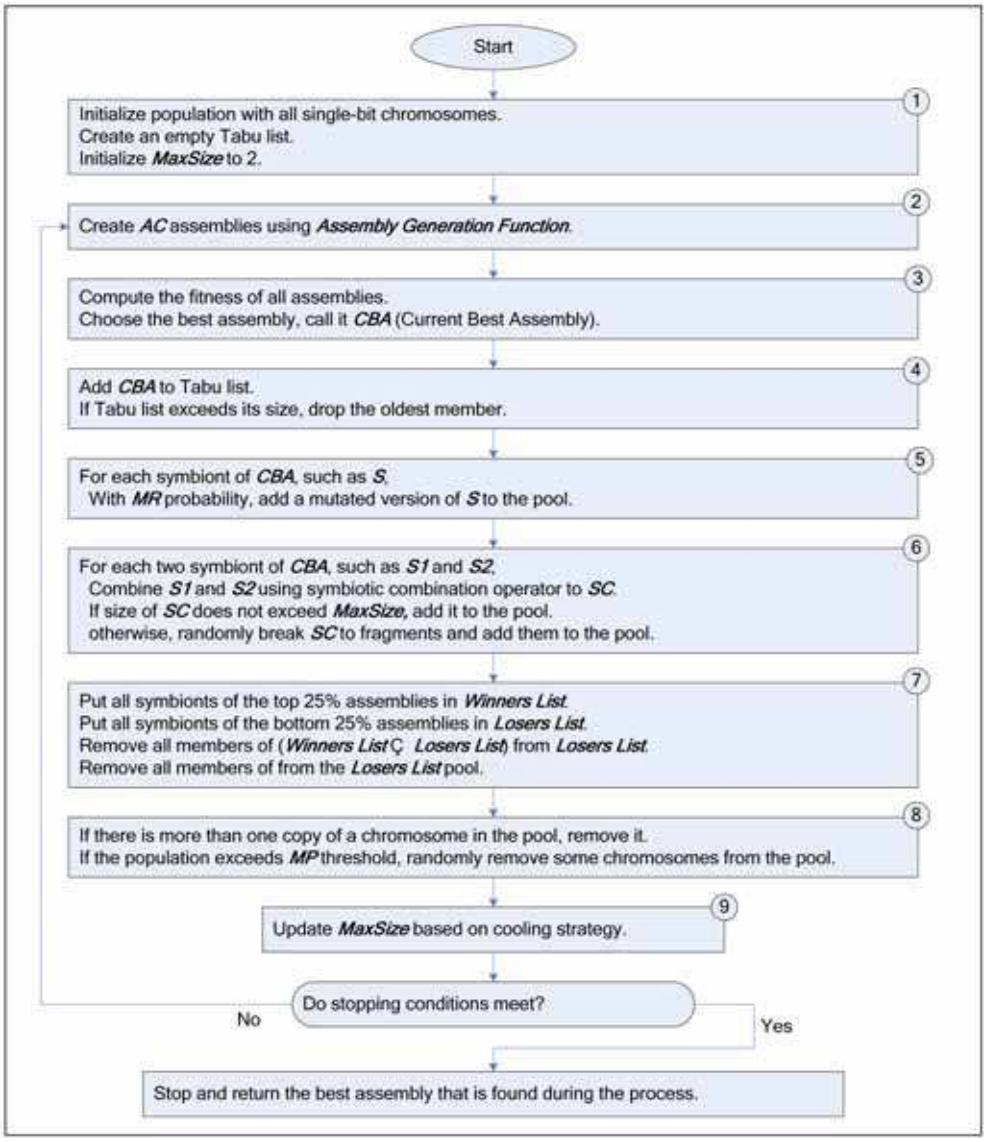


Fig. 2. Diagram of STS algorithm for optimization. The parameters are AC for Assemblies Count, MR for Mutation Rate, and MP for Maximum Population threshold.

3.3 Feature comparison of STS with some other algorithms

As stated before, there are generally three methods to deal with linkage or building block problems. One method is the estimation of gene distribution algorithms such as EDA (Larraaga & Lozano, 2002), PBIL (Baluja, 1994), cGA (Harik et al, 1998), ECGA (Sastry & Goldberg, 2000), FDA (Mühlenbein & Mahnig, 1999), BOA (Pelikan et al, 1999), and hBOA (Pelikan et al, 2003). As STS does not lie in this category, its features can not be easily compared with these algorithms because the cited algorithms require a previously selected distribution model for good genes while STS does not need it; but if such model exists, STS can not make use of it.

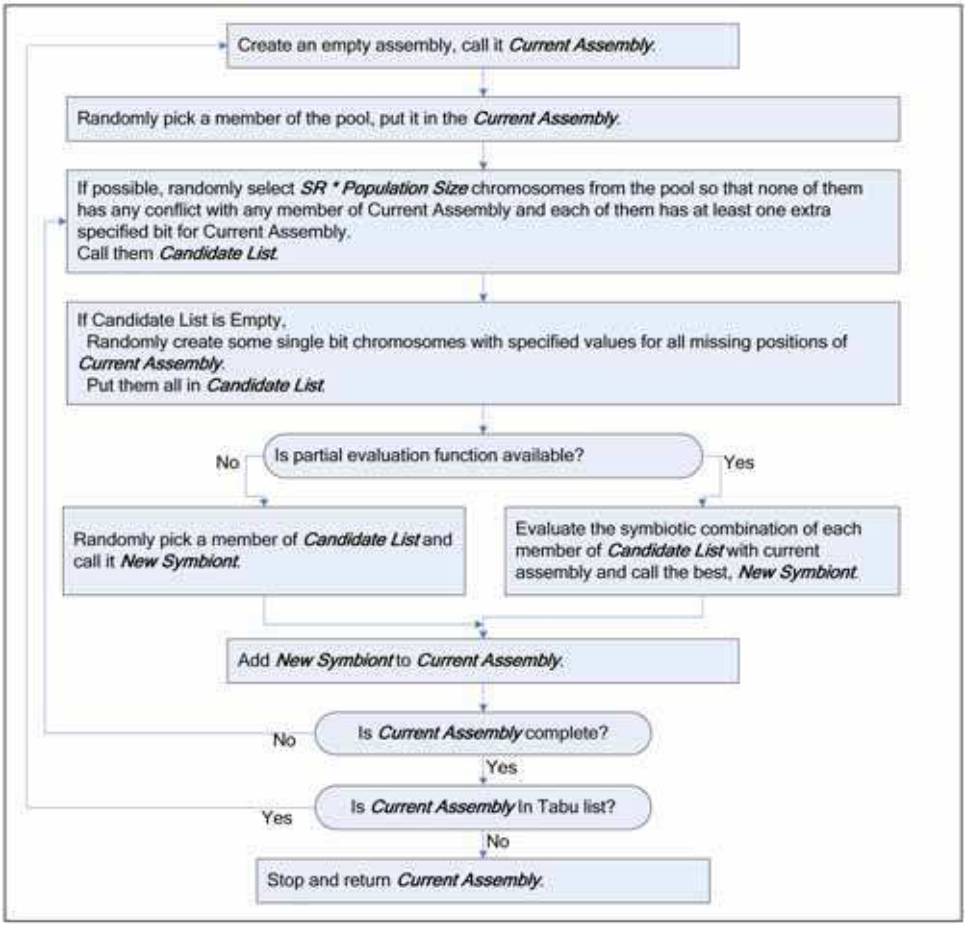


Fig. 3. Diagram of Assemly Generation Function. The parameters are *SR* for Selection Rate and *Population Size* for the number of chromosomes in the pool.

In Table 1, we have compared STS with some major algorithms of *partial chromosome specification* and *chromosome reordering* schools. As stated there, STS does not use fully specified chromosomes, so does not have the garbage genes problem; it doesn't work under the bias of any conflict resolution method as it does not combine conflicting chromosomes; it

is not under the influence of size penalty function as it lets the chromosomes grow only if they prove have high fitness value in cooperation with each other; it requires no prior domain knowledge or partial evaluation function, but it can make use of both if available; it uses substructures and can gradually evolve and reform them and is not bound to algorithm designer for substructure definition.

	SGA	mGA	SEAM	CGA	LLGA	CCEA	STS
Fully Specified Chromosomes?	Yes	No	No	No	Yes	No	No
Fixed Chromosome Locations?	Yes	Yes	Yes	Yes	No	Yes	Yes
Sensitive to Genome Order?	Yes	No	No	No	No	No	No
Carries Garbage?	Yes	No	No	No	Yes	No	No
Conflict Resolution Method?	Random	In Favor of First Parent	In Favor of First Parent	Most Fit Parent	Remove Originals	Not Needed	Not Needed
Diversity Preservation Method?	Mutation	Thresholding	Deterministic Crowding	Mutation	Mutation	Mutation	Mutation
Requires Size Penalty Function?	No	No	No	Yes	No	No	No
Requires Domain Knowledge?	Yes	Yes	Yes	Yes	No	Yes	No
Uses Substructures?	No	Yes	Yes	Yes	No	Yes	Yes
Substructures Evolve?	No	Yes	Yes	Yes	No	No	Yes

Table 1. Feature Comparison of STS and some other algorithms

4. Experimental results

4.1 Benchmarks problems

We used three benchmark problem sets. The first one is the Hierarchical If and Only I (HIFF) function (Watson & Pollack, 1999) with fully deceptive behaviour. The function takes an N-bit input and computes the fitness as stated in equation (1).

$$F(B)=\begin{cases}1, & \text{if } |B|=1\\ |B|+F(B_L)+F(B_R), & \text{if } (|B|\rangle 1) \text{ and } (\forall i,\{b_i=0\} \text{ or } \forall i,\{b_i=1\})\\ F(B_L)+F(B_R), & \text{otherwise}\end{cases} \quad (1)$$

B_L and B_R are respectively the left and the right half of B bit string.

The second benchmark is the concatenation of multiple 8-Queen problems (M8Q) (Eiben et al, 1995). In each instance, M separate problems of putting 8 queens on an 8×8 chessboard must be solved, so that no two queens on a board can attack each other. The chromosome includes the rows of the queens and columns are all assumed distinct and fixed.

The third benchmark is the KN-Trap function (Kargupta, 1995). The chromosomes are concatenations of K sections of N bits. For each N bit section, the fitness is computed as stated in equation (2) and the summation of all fitness values is assigned to the whole chromosome. The Trap function has a deceptive behaviour with the global maximum for an all-zero bit string and a negative gradient towards this point.

The chromosomes in all problems are shuffled (Watson & Pollack, 1990), so that adjacency data may not be used by any of the algorithms.

$$F(B) = \begin{cases} \text{SizeOf(B)}, & \text{if } Ones(B) = 0 \\ \text{Ones(B) -1}, & Otherwise \end{cases} \quad (2)$$

Ones(B) equals to the number of bits in B with value 1.

4.2 Benchmarks algorithms

We compared STS with Hill Climbing (HC) (Russle and Norvig, 2002), Symbiotic Evolutionary Adaptation Model (SEAM), simple Genetic Algorithm (SGA), and plain Tabu Search algorithm (TSA). HC was chosen as the most primitive local search algorithm, SEAM from symbiotic algorithms family, TSA from tabu search family, and SGA as the basic evolutionary algorithm.

To select appropriate parameters for SEAM, TSA, SGA, and STS on each instance of the benchmark problems, we used a greedy optimization algorithm with 100 steps and 20 random restarts. The measures for selecting appropriate parameters where first, number of solved problems, and second, the time needed to solve the problems. The best parameter settings, for each algorithm/problem instance is specified in Tables 2-5. For each algorithm, problems which could not be solved with any set of parameters are marked with N/A for not available.

Each instance of the benchmark problems was tried with each algorithm in 30 independent runs. Each algorithm/problem was given a maximum number of allowed fitness function calls based on problem complexity. These maximum values were chosen based on our initial experiments and were the same for all algorithms (10 times the fastest result that we found on that problem).

The cooling function of STS is implemented as a linear function, by dividing the iteration number per *CoolingRate* parameter. For each algorithm/problem, the success rates (number of times that each algorithm could find the optimum value (Forrest & Mitchell, 1993b)) and the average number of fitness computations for cases in which each algorithm has been able to solve the problem are depicted in respective diagrams.

	HIFF-5	HIFF-6	HIFF-7	1x4 Trap	4x4 Trap	2x8 Trap	1x8 Queens	5x8 Queens	10x8 Queens	15 x 8 Queens
Number of Contexts	50	100	150	50	50	100	50	N/A	N/A	N/A

Table 2. Parameters of benchmark problems for SEAM algorithm.

	HIFF-5	HIFF-6	HIFF-7	1x4 Trap	4x4 Trap	2x8 Trap	1x8 Queens	5x8 Queens	10x8 Queens	15 x 8 Queens
Tabu List Size	100	N/A	N/A	100	100	100	N/A	100	N/A	N/A

Table 3. Parameters of benchmark problems for TSA algorithm.

	HIFF-5	HIFF-6	HIFF-7	1x4 Trap	4x4 Trap	2x8 Trap	1x8 Queens	5x8 Queens	10x8 Queens	15 x 8 Queens
Population Size	300	N/A	N/A	100	100	N/A	100	100	100	500
Mutation Rate	0.75	N/A	N/A	0.65	0.65	N/A	0.7	0.65	0.95	0.75
Cross Over Rate	0.7	N/A	N/A	0.7	0.65	N/A	0.6	0.65	0.5	0.75
Elitism Percentage	0.1	N/A	N/A	0.1	0.3	N/A	0.15	0.3	0.1	0.3
Selection Method[1]	RW	N/A	N/A	TS	RW	N/A	TS	RW	TS	RW

Table 4. Parameters of benchmark problems for SGA Algorithm

	HIFF-5	HIFF-6	HIFF-7	1x4 Trap	4x4 Trap	2x8 Trap	1x8 Queens	5x8 Queens	10x8 Queens	15 x 8 Queens
Number of Assemblies	25	50	75	25	25	50	25	100	75	90
Cooling Rate x 1000	15	17.5	20	10	12.5	15	11	12.5	12.5	20
Selection Rate	0.001	0.001	0.001	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Mutation Rate	0.5	0.5	0.5	0.7	0.7	0.6	1	1	1	1
Number of Tabus	200	200	200	100	100	100	100	100	100	100

Table 5. Parameters of benchmark problems for STS Algorithm.

4.3 Experimental results of HIFF problems

The success rates and performance of the five tested algorithms on HIFF problems are depicted in Figure 5 diagrams. As it is seen, the only algorithm which was not able to solve all instances of 32 bit HIFF was SGA; TSA and HC totally failed to solve the 64 bit instances of HIFF while SGA still solved 20% of instances of 64 bit HIFF; and on 128 bit problems, only 30% were solved by SEAM while STS succeeded in 95% of cases.

Also it must be noted that in 32 bit problems, TSA and HC used almost double th computation time that STS required to solve the problem and generally, STS required much more computation time in compare with HC an TSA.

This problem has a fully deceptive behaviour, and identification and using building blocks is a serious requirement of the task. That's why SGA failed from the beginning. HC and TSA where successful at the smallest case where the search space was still not very large and found the solutions with checking almost $1/10000^{th}$ of all possible solutions, but with bigger search spaces they totally failed to find the solution by plain tracing and without using schemata. The only two algorithms that survive till 128 bit problems are STS and SEAM while STS is ahead with 65% more success rate, but exploiting much more computation time, even in smaller problems. This is because STS takes much more time analysing different cases (using cooling mechanism) before it composes a bigger building block but SEAM makes building blocks much faster and in the first success of a building block. Therefore SEAM can converge much faster than STS, but as it is seen, in big problems where there are much more local optima this may lead to a wrong turn to a local pick.

[1] RW for Rolette Wheel / TS for Tournoment Selection

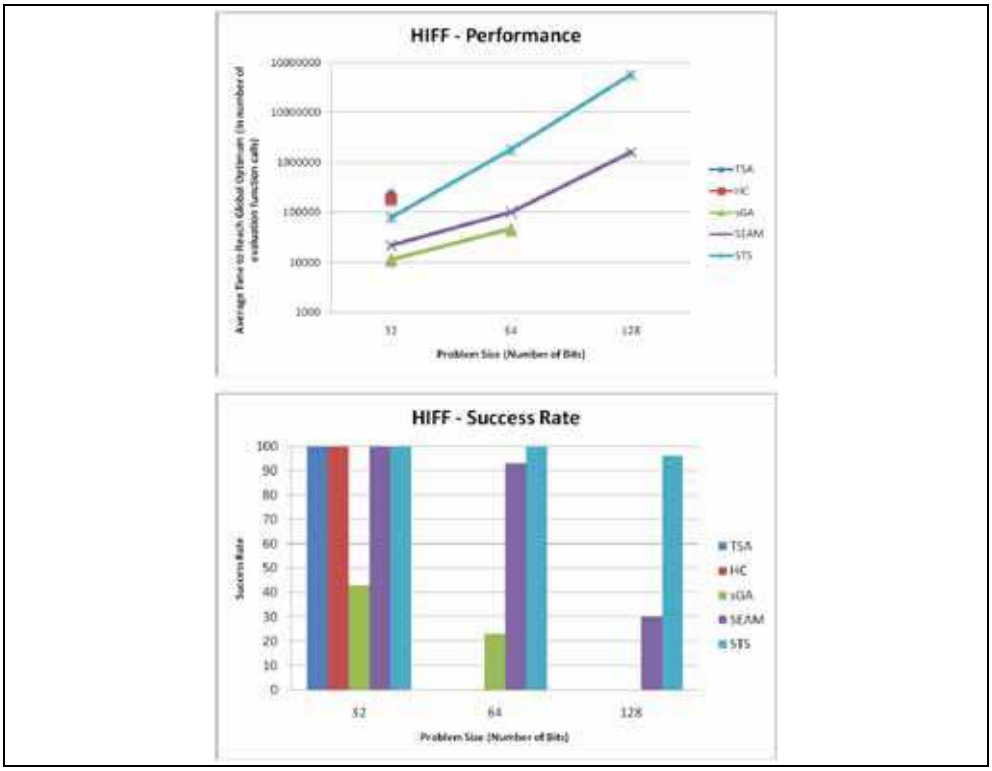


Fig. 5. Performance and Success rates of HC, TSA, SGA, SEAM, and STS algorithms for HIFF Problems.

4.4 Experimental results of M8Q problems

As depicted in Figure 6, the smallest instance of M8Q problem is solved completely by all algorithms, but the only ones that keep solving the bigger instances are SGA and STS. The reason behind the failure of HC and TSA can be in the big search space of 5 and more number of boards (120 bit for 5 boards, 240 bits for 10 boards and 360 bits for 15 boards) and the excessive number of local optima of M8Q problems.

SEAM also failed soon due to its nature of partial chromosome evaluation which creates a bigger chromosome, only if it gains better fitness value in compare with both its parents, in the context of many other chromosomes. This evaluation fails in M8Q problem as there are many global optima whose components are not a good solution of other global optima and therefore, the combination of each two chromosomes may be a good subcomponent of one solution and a bad subcomponent for another solution.

The only algorithms that could solve the 5, 10, and 15 board problems where SGA and STS and as depicted in Figure 6, the computation time of STS was somewhat lower than SGA and the success rate, more than 50% better in the biggest problem. Again we believe that the better success rate of STS in compare with SGA is due to its ability to create appropriate building blocks and therefore optimization of different boards almost separately, while SGA can not do this due it its fixed structured recombination operators.

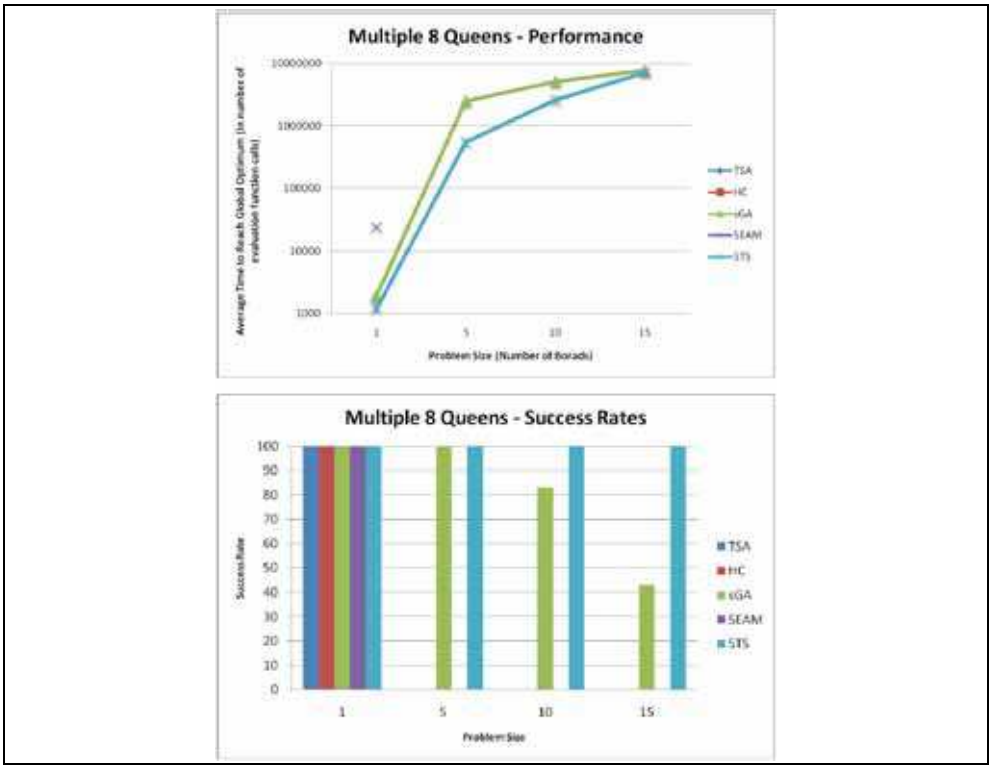


Fig. 6. Performance and Success rates of HC, TSA, SGA, SEAM, and STS algorithms for Multiple 8 Queens Problems.

4.5 Experimental results of KnTrap problem

The results of KnTrap problems are presented in Figure 7. This problem has a relatively very small size (16 bits in the biggest instance), but due to its deceptive behaviour and positive gradient only towards local optima, it is a very hard problem for greedy algorithms. HC and TSA were quite successful in all instances of this problem, as the search space is very small and due to the internal restart and choosing new random start point of both algorithms, they find the global optimum after enough number of restarts. SGA fails in 4×4 and 2×8 instance of this problem as the selection force always leads it to local optima and distances it from the best answer, and SEAM fails for the same reason on the biggest instance. But STS still keeps solving all instances, with more computation time in compare with HC and TSA, as its cooling mechanisms tries to find building blocks of each size, and not just based on previously found building blocks. So it ultimately finds partial optimum solutions of the size of trap functions and by combining them, finds the global solution.

5. Concluding remarks

To overcome the linkage problem, we proposed the usage of partially specified chromosomes (PSCs) along with evaluation and selection at group level. The main point

behind this idea was to select and replicate PSCs at group level instead of individual level, because once a group of PSCs show a good fitness value together, it does not implicate that each of them is a good subsolution in general, but it means that they are good together and can form a good cooperation; So, once we evaluate a group and they show good fitness value, all members of the group receive a higher change for survival and replication. During replication, to increase the chance of regrouping for some members of a successful group, we stick some of them together using symbiotic combination operator and build partial solutions with more specified bits. If these partial solutions show good fitness values in their future combinations with other members of the pool, they would be replicated again and grow bigger and if not, they will be destroyed. As this template is quite prone to getting stuck in local optima, it is augmented with tabu prohibition mechanism to avoid this problem and perform a better global search. The major advantages behind this idea are:

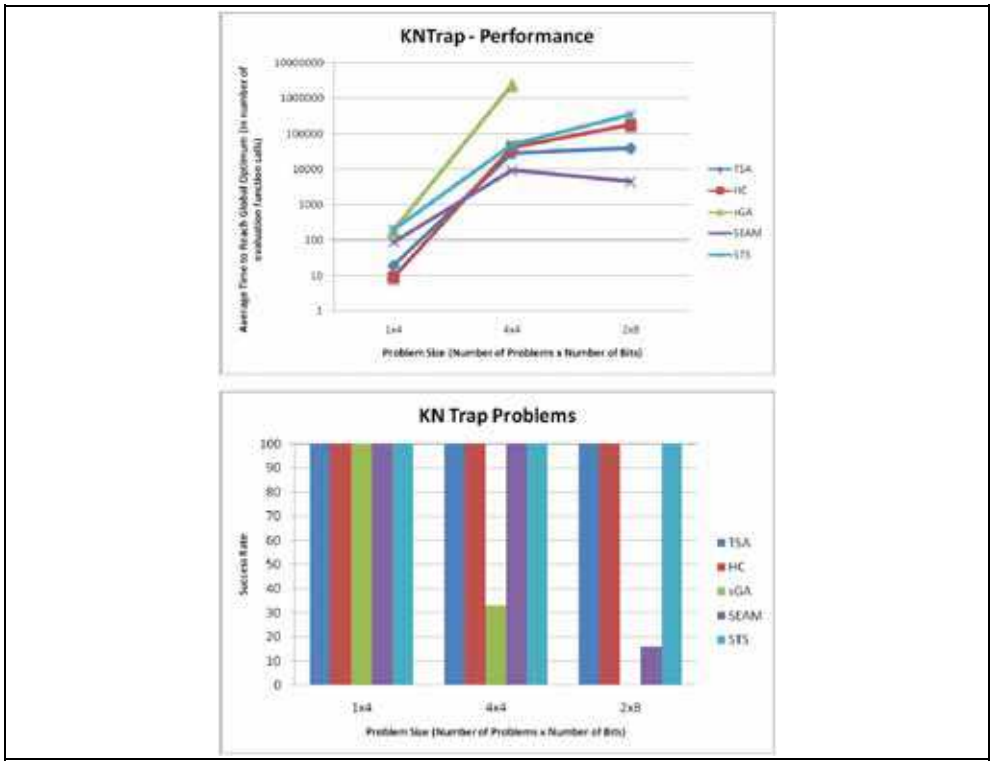


Fig. 7. Performance and Success rates of HC, TSA, SGA, SEAM, and STS algorithms for KnTrap Problems.

1. The evolutionary algorithm does not need prior domain knowledge for chromosome or recombination operator design for preservation of building blocks during recombination, as there is no recombination any more.
2. The process uses schemata and sub structures during evolution and can find and use building blocks, while the building blocks have no predefined structure or size

limitations. Moreover, growth control is done only based on fitness and therefore there would be no bias due to size penalty function or anything similar.
3. As the process uses PSCs and creates bigger chromosomes only if the group of chromosomes shows a high fitness value, garbage genes are produced less than approaches based on fully specified chromosomes.
4. Tabu prohibition prevents getting stuck in local optima and Cooling mechanism ensures a wide global search before narrowing down to areas limited by schemata.

The Symbiotic Tabu Search (STS) algorithm was presented in section 3 and was compared with hill climbing (HC), plain tabu search algorithm (TSA), simple genetic algorithm (SGA), and symbiotic evolutionary adaptation model (SEAM) in section 4 on three families of benchmark problems. The problems were chosen from 3 categories, a fully deceptive problem (HIFF), a combinatorial optimization problem (M8Q), and concatenation of some deceptive functions (KnTrap).

Our test results totally complied with our assertions about STS and showed that STS was quite more successful in reaching optimum solutions using less computational power, except in cases that search for building blocks was useless and solutions were quite trivial. In these cases, the gadgets of STS were just unnecessary and only time consuming, but yet STS could find the optimal solution with more computations.

As a final statement, we believe that evaluation and selection at group level can be a major advantage to search and optimization methods that are based on partially specified chromosomes and symbiotic combination is an appropriate tool for building schemata in such algorithms. Along with these two, tabu prohibition is a mandatory mechanism to prevent local optima as this search strategy is quite prune to getting stuck in the first local optima. Putting these along the facts that this algorithm does not require prior domain knowledge for operator or chromosome design, does not require any sort of partial evaluation function or size limit functions but can use them if available, and can create building blocks with no pre-specified limit, STS can be introduced and recommended as a general search and optimization algorithm that can automatically discover and use building blocks to cope with any form of internal relation between different parts of the solution.

6. Acknowledgements

Authors wish to give their sincerest thanks to Professor Caro Lucas for his valuable comments during this task, and Ms. Mojdeh Jalali Heravi & Ms. Bahareh Jafari Jashmi for their help through implementation and tests.

7. References

Al-Yamani, A.; Sait, S.M.; Barada, H. & Youssef, H. (2003). Parallel tabu search in a heterogeneous environment, *Proceedings of Parallel and Distributed Processing Symposium*, 8 pp. on CDROM, ISBN: 0-7695-1926-1, Nice, France, 22-26 April 2003.

Alvarez-Valdes, R.; Parreño, F. & Tamarit, J.M. (2006). A tabu search algorithm for a twodimensional non-guillotine cutting problem. *European Journal of Operational Research*, Vol. 183, No. 3, 16 December 2007, 1167-1182.

Bachelet, V.; Preux, P. & Talbi E-G. (1996). Parallel Hybrid Meta-Heuristics: Application to the Quadratic Assignment Problem. *Proceedings of the Parallel Optimization Colloquium*, pp. 233-242, Versailles, France, March 1996.

Bagis, A. (2007) Fuzzy rule base design using tabu search algorithm for nonlinear system modeling, *ISA Transactions*, Vol. 47, No. 1, January 2008, 32-44.

Bagley, J.D. (1967). *The Behaviour of Adaptive Systems Which Employ Genetic and Correlation Algorithms*, PhD Dissertation, University of Michigan.

Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning*, Tech. Rep. No. CMU-CS-94-163. Pittsburgh, PA, Carnegie Mellon University.

Brandão, J. (2007). A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem, *European Journal of Operational Research*, in Press, available via Science Direct, doi:10.1016/j.ejor.2007.05.059.

Brandão, J. & Eglese, R. (2006). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research*, Vol. 35, No. 4, April 2008, 1112- 1126.

Caserta, M. & Uribe, A.M. (2007). Tabu search-based metaheuristic algorithm for software system reliability problems, *Computers & Operations Research*, in Press, available via Science Direct, doi:10.1016/j.cor.2007.10.028

Chen, L.; Bostel, N.; Dejax, P.; Cai, J. & Xi, L. (2006). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. *European Journal of Operational Research*, Vol. 181, No. 1, 16 August 2007, 40-58.

Crainic, T.G.; Perboli, G. & Tadei, R. (2007). TS_2PACK: A two-level tabu search for the threedimensional bin packing problem, *European Journal of Operational Research*, in Press, available via Science Direct, doi:10.1016/j.ejor.2007.06.063.

Deb, K. (1991). *Binary and floating point function optimization using messy genetic algorithms* (IlliGAL Report No. 91004). Urbana: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Eiben, A.E; Raué, P.E. & Ruttkay, Z. (1995). *GA-easy and GA-hard Constraint Satisfaction Problems*. In: Constraint Processing, Manfred Meyer (Ed.), Springer-Verlag LNCS 923, 267-283.

El Fallahi, A.; Prins, C. & Calvo, R.W. (2006). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem, *Computers & Operations Research*, Vol. 35, No. 5, May 2008, 1725-1741.

Exler, O.; Antelo, L.T.;. Egea, J.A.; Alonso, A.A. & Banga, J.R. (2007), A Tabu search –based algorithm for mixed-integer nonlinear problems and its application to integrated process and control system design, *Computers and Chemical Engineering*, in Press, available via Science Direct, doi:10.1016/j.compchemeng.2007.10.008.

Forrest, S. & Mitchell, M. (1993). Relative Building-block fitness and the Building-block Hypothesis, *In Foundations of Genetic Algorithms 2*, L. D. Whitley (Ed.), 109-126. Morgan Kaufmann, San Mateo, CA.

Glover, F. & Laguna, M. (1997). Tabu Search. Kluwer Kluwer Academic Publishers, Boston.

Glover, F. (1989). Tabu Search, Part I, *ORSA Journal on Computing*, 1, 190-206.

Glover, F. (1990). Tabu Search, Part II, *ORSA Journal on Computing*, 2, 4-32.

Goldberg, D.E.; Korb, B. & Deb, K. (1989) Messy Genetic Algorithms: Motivation, analysis, and first results. *Computer Systems*, 3, 5, 493-530.

Gomes, L. C. T.; de Sousa, J. S. ; Bezerra, G. B.; de Castro, L. N. & Von Zuben, F. J. (2003) Copt-aiNet and the gene ordering problem, *Proceedings of Second Brazilian Workshop on Bioinformatics*, Rio de Janiro, Brazil, 3-5 December 2003.

Halavati, R.; Shouraki, S.B.; Jashmi, B.J. & Heravi, M.J. (2007). SEAM+ Evolutionary Optimization Algorithm, *Proceedings of the 7th International Conference on Intelligent Systems Design and Applications*, IEEE Computational Intelligence Society , Rio de Janiro, Brazil.

Hallberg, J. & Peng, Z. (1996). Multicycle Scheduling under Local Timing Constraints using Genetic Algorithms and Tabu Search. Proceedings of 22nd Euromicro Conference'96, Beyond 2000: Hardware and Software Strategies, Prague, Czech Republic, 2-5 Sep. 1996.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

Harik, G.R. (1997). *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithm*, PhD Dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Harik, G.R.; Lobo, F.G. & Goldberg, D.E. (1998). The compact genetic algorithm. *Proceedings of the IEEE Conference on Evolutionary Computation*, 523-528.

Hou, T.; Wang, J.; Chen, L. & Xu, X. (1999). Automatic docking of peptides and proteins by using a genetic algorithm combined with a tabu search. *Protein Engineering*, Vol. 12, No.8, 639-647.

Jaeggi, D.M.; Parks, G.T.; Kipouros; T. & Clarkson, P.J. (2008) The development of a multiobjective Tabu Search algorithm for continuous optimization problems. European *Journal of Operational Research*, Vol. 185, No. 3, 16 March 2008, 1192-1212.

Kargupta, H. (1995). *SEARCH Polynomial Complexity And The Fast Messy Genetic*, PhD Dissertation, University of Illinois at Urbana-Champaign, Urbana, IL.

Larrañaga, P. & Lozano, J.A. (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

Liang L.Y. & Chao, W.C. (2008) The strategies of tabu search technique for facility layout optimization, *Automation in Construction*, in press, available via Science Direct, doi:10.1016/j.autcon.2008.01.001.

Liu, Y.; Yia, Z.; Wu, H.; Ye, M. & Chen, K. (2008). A tabu search approach for the minimum sum-of-squares clustering problem. *Information Sciences*, in press, available via Science Direct, doi:10.1016/j.ins.2008.01.022.

Maynard Smith, J. & Szathmary, E. (1995) *The Major Transitions in Evolution*, WH Freeman: Oxford UK, 1995.

Merezhkovsky, K.S. (1909). The Theory of Two Plasms as the Basis of Symbiogenesis, a New Study or the Origins of Organisms. *Proceedings of the Studies of the Imperial Kazan University*, Publishing Office of the Imperial University, (In Russian).

Mermri, E.B; Katagiri, H.; Sakawa, M. & Kato, K. (2007). Remarks on the application of genetic algorithm and tabu search method to nonlinear spanning tree problems. *Applied Mathematics and Computation*, Vol. 188, No. 2, 15 May 2007, 1071-1086.

Mitchell, M. (1999). *An Introduction to Genetic Algorithms*, MIT Press, 0−262−13316−4, London, England.

Mühlenbein, H. & Mahnig, T. (1999). Convergence theory and application of the factorized distribution algorithm. *Journal of Computing and Information Technology*, Vol. 7, No. 1, 19-32.

Newman, D.R. (2006). *The Use of Linkage Learning in Genetic Algorithms*, http://www.ecs.soton.ac.uk/~drn05r/ug/irp/, (Last Seen: September 2006).

Oduntan, I.O.; Toulouse, M.; Baumgartner, R.; Bowman, C.; Somorjai, R. & Craini, T.G (2007), A multilevel tabu search algorithm for the feature selection problem in biomedical data. *Computers & Mathematics with Applications*, Vol. 55, No. 2, March 2008, 1019-1033.

Öncan, T.; Cordeau, J.F. & Laporte, G. (2007) A tabu search heuristic for the generalized minimum spanning tree problem. *European Journal of Operational Research*, in Press, available via Science Direct, doi:10.1016/j.ejor.2007.08.021.

Pacheco, J.; Alvarez, A.; Casado, S. & González-Velarde, J.L. (2007). A tabu search approach to an urban transport problem in northern Spain, *Computers & Operations Research*, in press, available via Science Direct, doi:10.1016/j.cor.2007.12.002.

Palubeckis, G. (2007). Iterated tabu search for the maximum diversity problem, *Applied Mathematics and Computation*, Vol. 189, No. 1, 1 June 2007, 371-383.

Pan, N.H.; Hsaio, P.W. & Chen, K.Y. (2007). A study of project scheduling optimization using Tabu Search algorithm, *Engineering Applications of Artificial Intelligence*, in Press, available via Science Direct, doi:10.1016/j.engappai.2007.11.006.

Pelikan, M.; Goldberg, D.E. & Cantu-Paz, E. (1999). BOA: The Bayesian optimization algorithm. Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, I, Orlando, FL. Morgan Kaufmann Publishers, San Francisco, CA., 525- 532.

Pelikan, M. & Goldberg, D.E. (2003). Hierarchical BOA solves using spin glasses and MAXSAT. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, II, 1275-1286.

Porto, S.C.S. & Ribeiro, C.C. (1996). Parallel tabu search message-passing synchronous strategies for task scheduling under precedence constraints. *Journal of Heuristics*, Vol. 1, No. 2, March 1996, 207-223.

Pothiya, S.; Ngamroo, I. & Kongprawechnon, W. (2007) Application of multiple tabu search algorithm to solve dynamic economic dispatch considering generator constraints. *Energy Conversion and Management*, Vol. 49, No. 4, April 2008, 506-516.

Potter, M.A., De Jong, K.A. (1994). A Cooperative Coevolutionary Approach to Function Optimization. *In: Parallel Problem Solving from Nature (PPSN III)*, Y. Davidor, H.-P. Schwefel and R. Manner (Eds.). Berlin: Springer-Verlag, 249-257.

Russle, S.J. & Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*, *2nd Edition*, Prentice-Hall, 111-112.

Sastry, K. & Goldberg, D. (2000). *On Extended Compact Genetic Algorithm* (IlliGAL Report No. 2000026). Urbana, IL: University of Illinois at Urbana-Champaign.

Shen, Q.; Shi, W.M. & Kong, W. (2007) Hybrid particle swarm optimization and tabu search approach for selecting genes for tumor classification using gene expression data, *Computational Biology and Chemistry*, Vol. 32, No. 1, February 2008, 53-60.

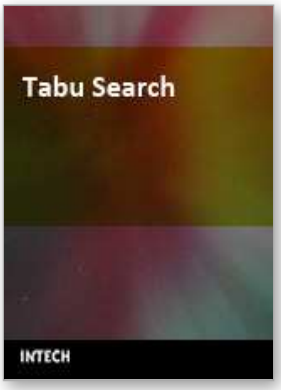
Vilcot, G. & Billaut, J.C. (2007). A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem, *European Journal of Operational Research*, in Press, available via Science Direct, doi:10.1016/j.ejor.2007.06.039.

Watcharasitthiwat, K.; Koseeyaporn, J. & Wardkein, P., (2006). Designing Digital FIR Filters Using Multiple Tabu Search Algorithm, *Proceedings of International Conference on Communications, Circuits and Systems*, pp. 171-175. ISBN: 0-7803-9585-9, Guilin, June 2006.

Watson, R.A. & Pollack, J.B. (1999), Incremental Commitment in Genetic Algorithms, *Proceedings of GECCO'99.*, Morgan Kaufmann, 710-717.

Watson, R.A. & Pollack, J.B. (2000). Symbiotic Combination as an Alternative to Sexual Recombination in Genetic Algorithms, *Proceedings of Parallel Problem Solving from Nature (PPSN VI)*, 425-436.

Zhu, Z.C.; Ng, K.M. & Ong, H.L. (2007) An Application of Tabu Search Algorithm on Costbased Job Shop Problem with Multiple Objectives, *Proceedings of IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 912-916, ISBN: 978-1-4244-1529-8, Singapore, 2-4 Dec. 2007.



Tabu Search

Edited by Wassim Jaziri

ISBN 978-3-902613-34-9
Hard cover, 278 pages
Publisher I-Tech Education and Publishing
Published online 01, September, 2008
Published in print edition September, 2008

The goal of this book is to report original researches on algorithms and applications of Tabu Search to real-world problems as well as recent improvements and extensions on its concepts and algorithms. The book' Chapters identify useful new implementations and ways to integrate and apply the principles of Tabu Search, to hybrid it with others optimization methods, to prove new theoretical results, and to describe the successful application of optimization methods to real world problems. Chapters were selected after a careful review process by reviewers, based on the originality, relevance and their contribution to local search techniques and more precisely to Tabu Search.

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