Chapter 1
A Unified Taxonomy of Hybrid Metaheuristics with Mathematical Programming, Constraint Programming and Machine Learning

El-Ghazali Talbi

Abstract. Over the last years, interest on hybrid metaheuristics has risen considerably in the field of optimization. The best results found for many real-life or classical optimization problems are obtained by hybrid algorithms. Combinations of algorithms such as metaheuristics, mathematical programming, constraint programming and machine learning techniques have provided very powerful search algorithms. Four different types of combinations are considered in this chapter:

- Combining metaheuristics with (complementary) metaheuristics.
- Combining metaheuristics with exact methods from mathematical programming approaches which are mostly used in operations research.
- Combining metaheuristics with constraint programming approaches developed in the artificial intelligence community.
- Combining metaheuristics with machine learning and data mining techniques.

1.1 Introduction

This chapter deals with the design of hybrid metaheuristics and their implementation. A taxonomy of hybrid algorithms is presented in an attempt to provide a common terminology and classification mechanisms. The goal of the general taxonomy given here is to provide a mechanism to allow comparison of hybrid algorithms in a qualitative way. In addition, it is hoped the categories and their relationships to each other have been chosen carefully enough to indicate areas in need of future work as well as to help classify future work. Among existing classifications in other domains, one can find examples of flat and hierarchical classifications schemes [74]. The taxonomy proposed here is a combination of these two schemes - hierarchical

El-Ghazali Talbi
University of Lille 1, CNRS, INRIA, Lille - France
e-mail: talbi@lifl.fr

E.-G. Talbi (Ed.): Hybrid Metaheuristics, SCI 434, pp. 3–76.
springerlink.com

as long as possible in order to reduce the total number of classes, and flat when the descriptors of the algorithms may be chosen in an arbitrary order. The same classification is used for all types of combinations. For each type of hybrids, the main ideas in combining algorithms are detailed. Each class of hybrids is illustrated with some examples. A critical analysis is also carried out.

In fact, the taxonomy could usefully be employed to classify any hybrid optimization algorithm (specific heuristics, exact algorithms). The basic classification is extended by defining the space of hybrid metaheuristics as a grammar, where each sentence is a method that describes a combination of metaheuristics, mathematical programming and constraint programming. In this chapter, a "high-level" description of hybrid metaheuristics is proposed. The internal working and the algorithmic aspects of a given metaheuristic are not considered.

The chapter is organized as follows. First, in section 1.2, our concern is hybrid algorithms combining metaheuristics. The design and implementation issues of hybrid metaheuristics are detailed. A taxonomy is presented to encompass all published work up to date in the field and to provide a unifying view of it. A grammar which generalizes the basic hybridization schemes is proposed. In section 1.3, the combination of metaheuristics with mathematical programming approaches is considered. Section 1.4 deals with the combination of metaheuristics with constraint programming techniques. Then, in section 1.5 the combination of metaheuristics with machine learning and data mining algorithms is addressed. Hybrid metaheuristics for multi-objective optimization are addressed in section 1.6.

1.2 Hybrid Metaheuristics

Hybridization of metaheuristics involves a few major issues which may be classified as design and implementation. The former category concerns the hybrid algorithm itself, involving issues such as functionality and architecture of the algorithm. The implementation consideration includes the hardware platform, programming model and environment on which the algorithm is to be run. In this chapter, a difference is made between the design issues used to introduce hybridization and implementation issues that depend on the execution model of the algorithms.

1.2.1 Design Issues

The taxonomy will be kept as small as possible by proceeding in a hierarchical way as long as possible, but some choices of characteristics may be made independent of previous design choices, and thus will be specified as a set of descriptors from which a subset may be chosen.

1.2.1.1 Hierarchical Classification

the structure of the hierarchical portion of the taxonomy is shown in figure 1.1. A discussion about the hierarchical portion then follows. At the first level, one may distinguish between low-level and high-level hybridizations. The low-level hybridization addresses the functional composition of a single optimization method. In this hybrid class, a given function of a metaheuristic is replaced by another metaheuristic. In high-level hybrid algorithms, the different metaheuristics are self-contained. There is no direct relationship to the internal workings of a metaheuristic.

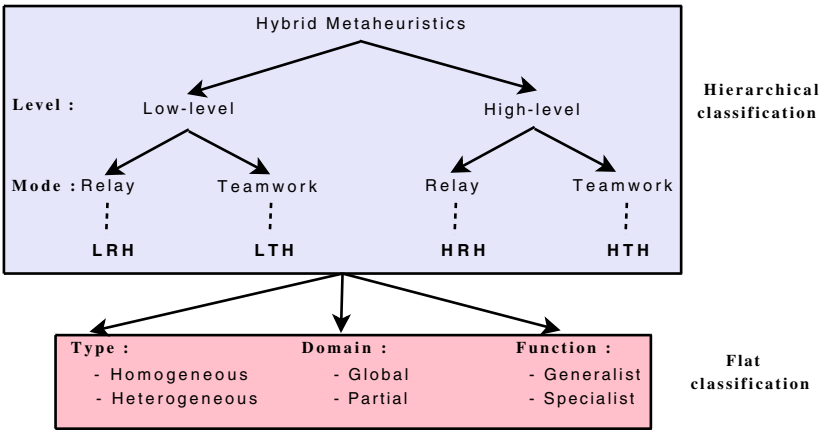

Fig. 1.1 Classification of hybrid metaheuristics in terms of design issues

In relay hybridization, a set of metaheuristics is applied one after another, each using the output of the previous as its input, acting in a pipeline fashion. Teamwork hybridization represents cooperative optimization models, in which many cooperating agents evolve in parallel; each agent carries out a search in a solution space.

Four classes are derived from this hierarchical taxonomy:

- **LRH (Low-level Relay Hybrid):** this class of hybrids represents algorithms in which a given metaheuristic is embedded into a S-metaheuristic (Single solution based metaheuristic) [**?**]. Few examples of hybrid metaheuristics belong to this class.

  Example 1.1. **Embedding local search into simulated annealing:** the main idea is to incorporate deterministic local search techniques into simulated annealing so that the Markov chain associated to simulated annealing explores only local optima [124]. The algorithm proceeds as follows: suppose the configuration is currently locally optimal. This is labeled `Start` in figure 1.2. A perturbation or a "kick" is applied to this configuration which significantly changes the current solution `Start`. After the kick, the configuration labeled `Intermediate` in the figure is reached. It is much better to first improve `Intermediate` by

a local search and apply the accept/reject test of simulated annealing only afterwards. The local search takes us from `Intermediate` to the configuration labeled `Trial`, and then the accept/reject test is applied. If `Trial` is accepted, one has to find an interesting large change to `Start`. If `Trial` is rejected, return to `Start`. Many of the barriers (the "ridges") of the fitness landscape are jumped over in one step by the hybrid metaheuristic.

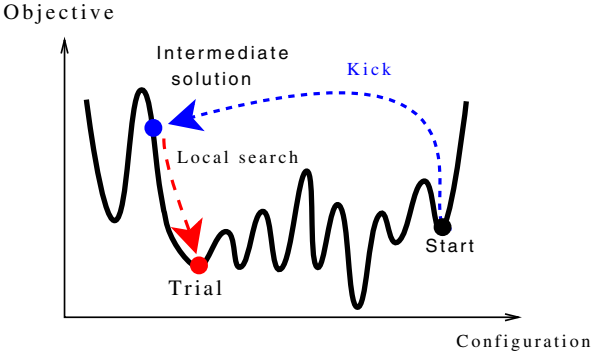

Fig. 1.2 An example of LRH hybridization embedding local search into simulated annealing. The figure gives a schematic representation of the objective function and the configuration modification procedure used in the LRH hybrid algorithm.

To implement the above hybridization, the choice for an appropriate "kick" should be adapted to both the optimization problem and the local search method used. For the traveling salesman problem, if the local search algorithm used is the 2-opt local search heuristic, the "kick" move must apply a k-change with $k > 2$ to prevent cycles. The "kick" operator must attain solutions which are always outside the neighborhood associated to the local search algorithm.

- **LTH (Low-level Teamwork Hybrid):** two competing goals govern the design of a metaheuristic: exploration and exploitation. Exploration is needed to ensure that every part of the space is searched enough to provide a reliable estimate of the global optimum. Exploitation is important since the refinement of the current solution will often produce a better solution. P-metaheuristics (Population based metaheuristics) [?] (e.g. evolutionary algorithms, scatter search, particle swarm, ant colonies) are powerful in the exploration of the search space, and weak in the exploitation of the solutions found.

  Therefore, most efficient P-metaheuristics have been coupled with S-metaheuristics such as local search, simulated annealing and tabu search, which are powerful optimization methods in terms of exploitation. The two classes of algorithms have complementary strengths and weaknesses. The S-metaheuristics will try to optimize locally, while the P-metaheuristics will try to optimize globally. In LTH hybrid, a metaheuristic is embedded into a

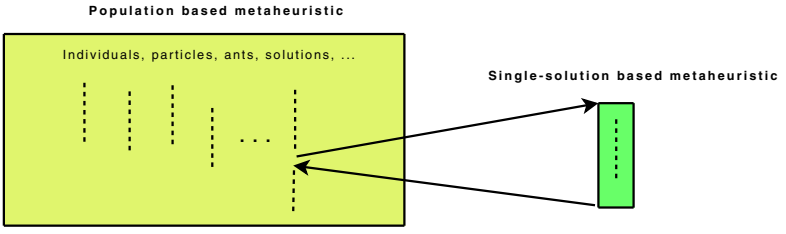

Fig. 1.3 Low-level Teamwork Hybrid (LTH). S-metaheuristics are embedded into P-metaheuristics.

P-metaheuristic[1] (Fig. 1.3). This class of hybrid algorithms is very popular and has been applied successfully to many optimization problems. Most of the state-of-the art of P-metaheuristics integrate S-metaheuristics.

Example 1.2. **Embedding S-metaheuristics into evolutionary algorithms:** when an evolutionary algorithm is used as a global optimizer, its standard operators may be augmented with the ability to perform local search. Instead of using a blind operator acting regardless of the fitness of the original individual and the operated one, an operator which is a heuristic that considers an individual as the origin of its search applies itself, and finally replaces the original individual by the enhanced one (see figure 1.4). The use of local search with evolutionary algorithms is also inspired by biological models of learning and evolution. EAs take many cues from mechanisms observed in natural evolution. Similarly, models of learning are often equated with techniques for local optimization [148]. Research on the interaction between evolution and learning had naturally led computer scientists to consider interactions between evolutionary algorithms and local optimization [20].

The genetic operators replaced or extended are generally mutation[2] and crossover.

– **mutation:** the local search algorithm may be a simple local search [157] [174] [100], tabu search [70] [111] [171], simulated annealing algorithm [36] [180] or any S-metaheuristic (e.g. threshold accepting, guided local search). This kind of operators is qualified *lamarckian*[3]. In the lamarckian model, an individual is replaced by the local optima found, contrary to the *baldwin* model where the local optima is just used to evaluate the individual. In several occasions, LTH has provided better results than other methods on difficult problems. For instance, good results have been obtained on the graph coloring problem combining a genetic algorithm with tabu search [71]. A local search algorithm which uses problem-specific knowledge may be incorporated into

[1] This class of hybrid metaheuristics includes *memetic algorithms*.

[2] Also known as evolutionary local search algorithms.

[3] The name is an allusion to Jean Batiste de Lamarck's contention that phenotype characteristics acquired during lifetime can become heritable traits.

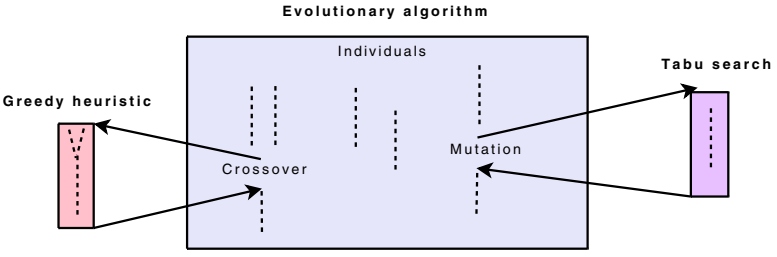

Fig. 1.4 Illustration of a LTH hybrid. For instance, a tabu search is used as a mutation operator and a greedy heuristic as a crossover operator into a genetic algorithm.

the genetic operators [37]. Questions concerning the best use of local search with a genetic algorithm have been addressed in [86].

– **Crossover:** classical crossover operators do not use any heuristic information about a specific application domain. They are blind operators. One can introduce heuristic crossover in order to account for problem-specific information [82]. For instance, greedy heuristics for the crossover operator have shown to improve EAs results when applied to job-shop scheduling, set covering, and traveling salesman problems [52].

  Many crossover operators including heuristic information have been proposed for continuous optimization:

  · Heuristic crossover where the offspring has the following form $x' = u(x_2 - x_1) + x_2$ where u is a uniform random value in $[0,1]$, x_1 and x_2 are the two parents with the condition that x_2 is better than x_1 [181]. This heuristic crossover uses the objective function in determining the direction of the search.

  · Simplex crossover where more than two parents are selected, the worst (resp. the best) individuals x_2 (resp. x_1) are determined. The centroid of the group c is then computed without taking into account the solution x_2. The offspring has the following form $x' = c + (c - x_2)$ [143].

This hybrid model can be used to improve any P-metaheuristic: ant colonies [161] [156], genetic programming [135], particle swarm optimization, and so forth. The S-metaheuristic has been introduced to intensify the search. Let us notice that the scatter search metaheuristic already includes an improvement procedure which is based on S-metaheuristics [49].

The main problem encountered in this class of hybrids is *premature convergence*. Indeed, if the hybridization is applied at each iteration, very competitive solutions will be generated in the beginning of the search which will cause an eventual premature convergence. Conditional hybridization is carried out to prevent this phenomenon by applying the combination:

– **Static manner:** for instance the combination is performed at a given frequency. The hybridization is applied once for a given number of iterations.

– **Adaptive manner:** when a given event occurs during the search the hybridization is performed. For instance, if there is no improvement of the search for a given number of iterations.

- **HRH (High-level Relay Hybrid):** in HRH hybrids, self-contained metaheuristics are executed in a sequence. For example, the initial solution of a given S-metaheuristic may be generated by another optimization algorithm. Indeed, the initial solution in S-metaheuristics has a great impact on their performances. A well known combination scheme is to generate the initial solution by greedy heuristics which are in general of less computing complexity than iterative heuristics (Fig. 1.5).

  This scheme may be also applied to P-metaheuristics, but a randomized greedy heuristic must be applied to generate a diverse population (Fig. 1.5). Greedy heuristics are in general deterministic algorithms and then they generate always the same solution. On the other hand, the diversity of the initial population has a great impact on the performance of P-metaheuristics. This hybrid scheme is carried out explicitly in the scatter search metaheuristic.

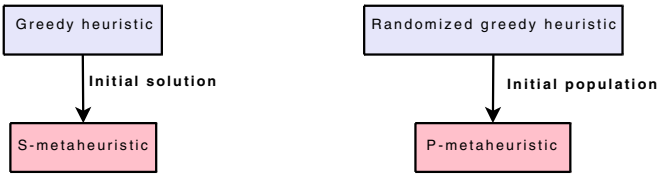

Fig. 1.5 High-level Relay Hybridization. (Left) Generation of the initial solution of a S-metaheuristic by a greedy algorithm. (Right) Generation of the initial population of a P-metaheuristic by a randomized greedy heuristic.

Combining in the HRH scheme P-metaheuristics with S-metaheuristic is also largely applied. It is well known that P-metaheuristics are not well suited for fine-tuning structures which are very close to optimal solutions. Indeed, the strength of P-metaheuristics is in quickly locating the high performance regions of vast and complex search spaces. Once those regions are located, it may be useful to apply S-metaheuristics to the high performance structures evolved by the P-metaheuristic.

A fundamental practical remark is that after a certain amount of time, the population is quite uniform and the fitness of the population is no longer decreasing. The odds to produce fitter individuals are very low. That is, the process has fallen into a basin of attraction from which it has a low probability to escape.

The exploitation of the already found basin of attraction to find as efficiently as possible the optimal point in the basin is recommended. It is experimentally clear that the exploitation of the basin of attraction that has been found may be more efficiently performed by another algorithm than by a P-metaheuristic. Hence, it is much more efficient to use a S-metaheuristic such as a hill-climbing or tabu search (see figure 1.6). The HRH hybridization may use a greedy heuristic to

generate a good initial population for the P-metaheuristic (see figure 1.6). At the end of a simulated annealing search, it makes sense to apply local search on the best found solution to ensure that it is a local optima.

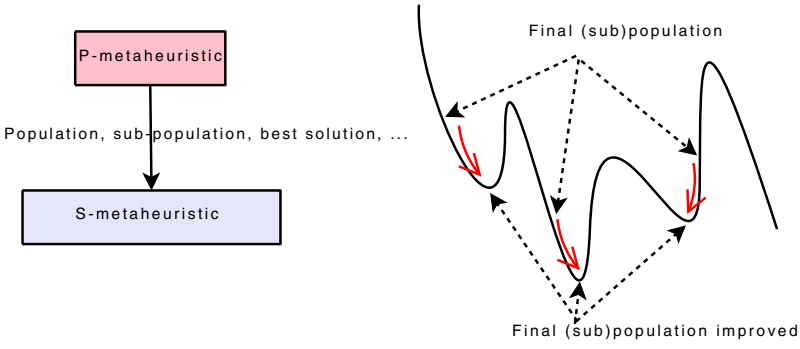

Fig. 1.6 High-level Relay Hybridization. There may be more than two algorithms to be pipelined.

In this hybrid scheme, the S-metaheuristics may be applied to:

- **The whole population:** this will leads to the best final solutions but with a more important computational cost of the search.
- **A sub-population:** the selection of the subpopulation may be based on the diversity of the solutions. This is a good compromise between the complexity of the search and the quality of the final results.
- **The best solution of the population:** the S-metaheuristic is applied once on the best solution of the obtained population. This procedure will reduce the search time but does not ensure to find the best solution.

A path relinking strategy may be applied to a population or a set of elite solutions found by a metaheuristic [6]. Path relinking may be seen as an intensification task over a given population of solutions.

Example 1.3. **HRH hybrid evolutionary algorithms:** many research works of the literature have used the idea of HRH hybridization for EAs. In [121] [166], the considered hybrid scheme introduces respectively simulated annealing and tabu search to improve the population obtained by a GA. In [132], the author introduces hill-climbing to improve the results obtained by an ES. In [118], the algorithm proposed starts from simulated annealing and uses GAs to enrich the solutions found. Experiments performed on the graph partitioning problem using the tabu search algorithm exploiting the result found by a GA give better results than a search performed either by the GA, or the tabu search alone [166].

- **HTH (High-level Teamwork Hybrid):** the HTH[4] scheme involves several self-contained algorithms performing a search in parallel, and cooperating to find an optimum. Intuitively, HTH will ultimately perform at least as well as one algorithm alone, more often perform better, each algorithm providing information to the others to help them.

Example 1.4. **Island model for genetic algorithms:** the first HTH hybrid model has been proposed for genetic algorithms (GAs). This is the well known island model[5]. The population in this model is partitioned into small subpopulations by geographic isolation. A GA evolves each subpopulation and individuals can migrate between subpopulations (Fig. 1.7). This model is controlled by several parameters: the topology that defines the connections between subpopulations, the migration rate that controls the number of migrant individuals, the replacement strategy used, and a migration interval that affects how often migration occurs. In some island models, the individuals really migrate and therefore leaves empty space in the original population. In general, the migrated individuals remain in the original population (i.e. pollination model [155]).

Let us present some pioneering island models for GAs. Tanese proposed a GA based HTH scheme that used a 4-D hypercube topology to communicate individuals from one subpopulation to another [170]. Migration is performed at uniform periods of time between neighbor subpopulations along one dimension of the hypercube. The migrants are chosen probabilistically from the best individuals of the subpopulation and they replace the worst individuals in the receiving subpopulation.

Cohoon, Hedge, Martin and Richards proposed a HTH based on the theory of "punctuated equilibria" [41]. A linear placement problem was used as a benchmark and experimented using a mesh topology. They found that the algorithm with migration outperforms the algorithm without migration and the standard GA. This work was later extended using a VLSI design problem (graph partitioning) on a 4-D hypercube topology [42] [43].

Belding in [19] attempted to extend the Tanese's work using the Royal Road continuous functions. Migrants individuals are sent to a random selected subpopulation, rather than using a hypercube topology. The global optimum was found more often when migration (i.e. cooperating GAs) occurred than in completely isolated cases (i.e. non-cooperating GAs).

Afterwards, the HTH hybrid model has been generalized to other P-metaheuristics and S-metaheuristics. Indeed, the HTH hybrid model has also been applied to simulated annealing [61], genetic programming [114], evolution strategies [178], ant colonies [123], scatter search [50], tabu search [62], and so on.

[4] HTH hybrids is referred as *multiple interacting walks* [176] *multi-agent algorithms* [24], and *cooperative search algorithms* [40] [39] [90] [92] [172].

[5] Also known as migration model, diffusion model, and coarse grain model.

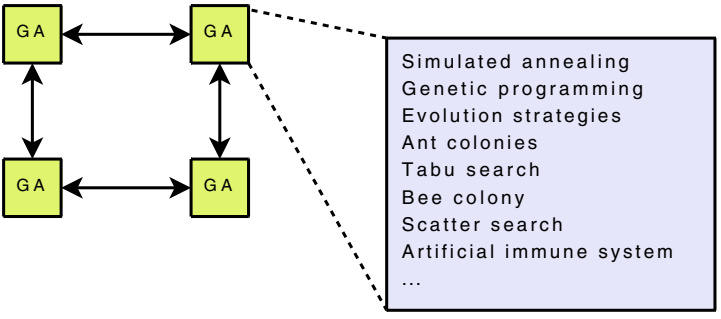

Fig. 1.7 The island model of genetic algorithms as an example of High-level Teamwork Hybrid (HTH). The same model has been used with different topologies for simulated annealing, genetic programming, evolution strategy, ant colony, tabu search, bee colony, artificial immune system, etc.

1.2.1.2 Flat Classification

Homogeneous/Heterogeneous: in homogeneous hybrids , all the combined algorithms use the same metaheuristic. Hybrid algorithms such as the island model for GAs belong to this class of hybrids. The homogeneous metaheuristics may differ in the initialization of their (Fig. 1.8):

- **Parameters:** in general, different parameters are used for the algorithms. For instance, in the HTH hybrid scheme which is based on tabu search, the algorithms may be initialized with different tabu list sizes [179]; different crossover and mutation probabilities may be used in evolutionary algorithms, etc.
- **Search components:** given a metaheuristic, one can use different strategies for any search component of the metaheuristic, such as the representation of solutions, objective function approximations [59] [151], initial solutions, search operators (neighborhood, mutation, crossover, ...), termination criteria, etc.

Using different parameters or search components into a given metaheuristic will increase the robustness of the hybrid algorithm.

Example 1.5. **Heterogeneous hybrids:** in heterogeneous algorithms , different metaheuristics are used (Fig. 1.9). A heterogeneous HTH algorithm based on genetic algorithms and tabu search has been proposed in [46] to solve a network design problem. The population of the GA is asynchronously updated by multiple tabu search algorithms. The best solutions found by tabu search algorithms build an elite population for the GA.

The GRASP method (Greedy Randomized Adaptive Search Procedure) may be seen as an iterated heterogeneous HRH hybrid, in which local search is repeated from a number of initial solutions generated by a randomized greedy heuristic [66] [65]. The method is called adaptive because the greedy heuristic takes into account the decisions of the precedent iterations [64].

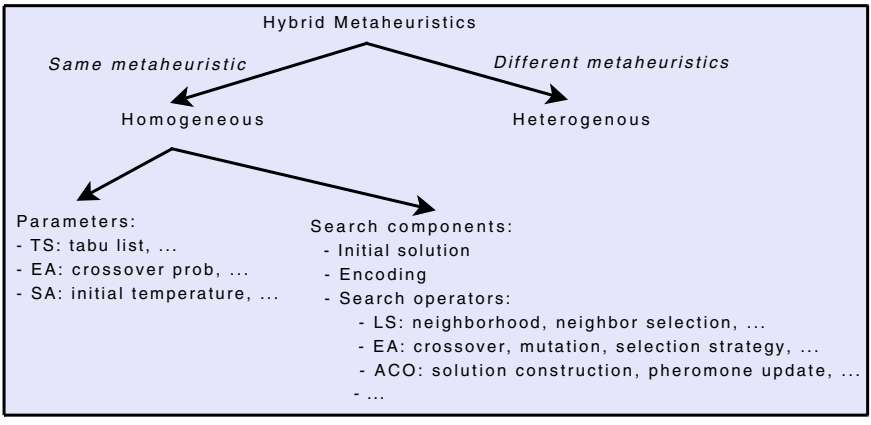

Fig. 1.8 Homogeneous versus heterogeneous hybrid metaheuristics. Some illustrative examples of parameters and search components are illustrated.

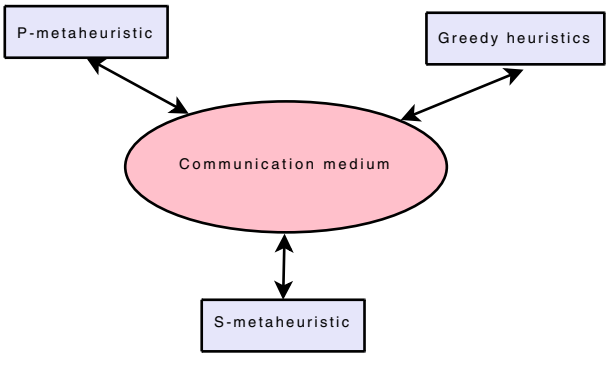

Fig. 1.9 High-level Teamwork Hybridization HTH (heterogeneous, global, general). Several search algorithms cooperate, co-adapt, and co-evolve a solution.

Global/Partial: from another point of view, one can also distinguish two kinds of cooperation: global and partial. In global hybrids , all the algorithms explore the same whole search space. The goal is here to explore the space more thoroughly. All the above mentioned hybrids are *global* hybrids, in the sense that all the algorithms solve the whole optimization problem. A global HTH algorithm based on tabu search has been proposed in [47], where each tabu search task performs a given number of iterations, then broadcasts the best solution. The best of all solutions becomes the initial solution for the next phase.

In partial hybrids, the problem to be solved is decomposed a priori into subproblems, each one having its own search space (Fig. 1.10). Then, each algorithm is dedicated to the search in one of these sub-spaces. Generally speaking, the subproblems are all linked with each others, thus involving constraints between optima

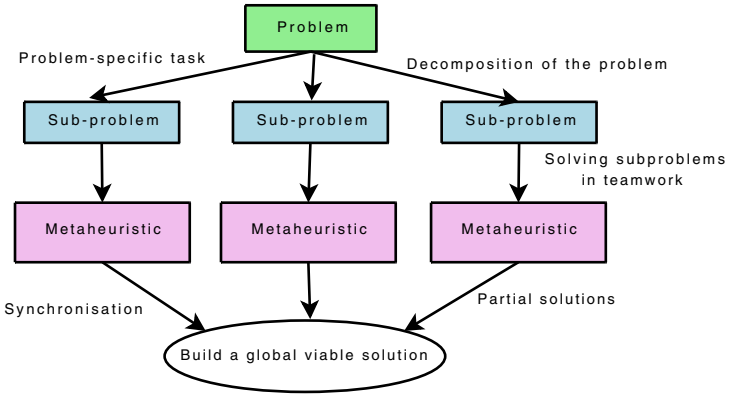

Fig. 1.10 Partial hybrid schemes. Several search algorithms cooperate in solving subproblems. A synchronization is performed to build a global solution from the partial solutions found.

found by each algorithm. Hence, the algorithms communicate in order to respect these constraints and build a global viable solution to the problem.

Example 1.6. **Partial hybrids:** this approach has been applied for many specific metaheuristics such as simulated annealing and tabu search algorithms [158]. It is also a part of more general search framework such as POPMUSIC (Partial OPtimization Metaheuristic Under Special Intensification Conditions [160] and VNDS (Variable Neighborhood Decomposition Search) [85].

Asynchronous teams (A-Teams) represent a general model for a HTH heterogeneous partial hybrid strategy [168]. They manipulate a set of solutions, which may be global or partial solutions. A set of agent cooperate via a blackboard system, a shared memory structure. An agent may be any search algorithm or operator, which consists in picking a (partial) solution from the blackboard, transforming it and sending back the result.

An example of application of partial homogeneous HTH has been done for the job-shop scheduling problem [93]. The search algorithm is a GA. Each GA evolves individuals of a specie which represent the process plan for one job. Hence, there are as many cooperating GAs as there are jobs. The communication medium collects fitted individuals from each GA, and evaluates the resulting schedule as a whole, rewarding the best process plans.

Decomposition techniques based on partitioning time have been used to solve many problems such as the production lot-sizing (partitioning of time) [63]. Decomposition techniques based on partitioning a geographical region have been largely applied to optimization problems associated with Euclidean distances such as the TSP [110], the VRP, and the P-median problem [159].

Example 1.7. **Partitioning a continuous objective function :** a function f is separable if:

$$(argmin_{x_1} f(x_1,...),...,argmin_{x_n} f(...,x_n)) = argmin(f(x_1,x_2,...,x_n))$$

It follows that the function f can be optimized in a parallel way using n independent algorithms. Each algorithm will solve a 1-D optimization problem.

Generalist/Specialist: all the above mentioned hybrids are *general hybrids* , in the sense that all the algorithms solve the same target optimization problem. *Specialist hybrids* combine algorithms which solve different problems. The COSEARCH generic model belongs to this class of hybrids (Fig. 1.11). COSEARCH manages the cooperation of a search agent (a local search), a diversifying agent and an intensifying agent. The three agents exchange information via a passive coordinator called the adaptive memory[6]. A main key point of the COSEARCH approach is the design of this memory which focus on high quality regions of the search and avoid attractive but deceptive areas. The adaptive memory contains a history of the search; it stores information about the already visited areas of the search space and about the intrinsic nature of the good solutions already found. When diversifying, the local search agent receives starting solutions in unexplored regions; when intensifying, the search agent receives an initial solution in a promising region. The diversifying agent refers to the adaptive memory (information about the explored areas) to yield a solution from an unexplored region. The intensifying agent refers to the adaptive memory (information about promising regions) to produce a promising starting solution.

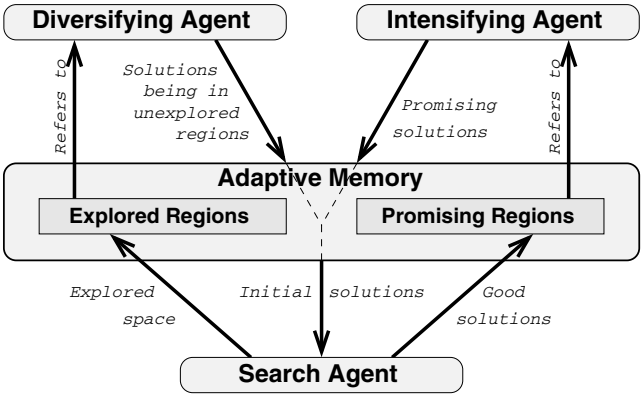

Fig. 1.11 The COSEARCH HTH specialist hybrid model for metaheuristics. Several search algorithms solve different problems.

Example 1.8. **COSEARCH for the quadratic assignment problem:** an example of the application of the COSEARCH approach has been developed in [164] to solve the quadratic assignment problem (QAP). A parallel tabu search is used to solve the QAP, while a genetic algorithm makes a diversification task, which is formulated as

[6] The concept of adaptive memory has been proposed in the domain of combinatorial optimization [162]. It is similar to the concept of blackboard in the field of Artificial Intelligence [60].

another optimization problem. The frequency memory stores information relative to all the solutions visited during the tabu search. The genetic algorithm refers to the frequency memory to generate solutions being in unexplored regions.

Another approach of specialist hybrid HRH heuristics is to use a heuristic to optimize another heuristic, i.e. find the optimal values of the parameters of the heuristic (Fig. 1.12). This approach is known as *meta-optimization*. For instance, it has been used to optimize simulated annealing and noisy methods (NM) by GA [115], ant colonies (AC) by GA [1], simulated annealing based algorithms by a GA [79], and a GA by a GA [153][7]. In [153], the three parameters optimized are the crossover rate, inversion rate, and mutation rate. The individuals of the population associated to the optimizer consist of three integers representing the mutation rate, inversion rate, and crossover rate. The fitness of an individual is taken to be the fitness of the best solution that the GA can find in the entire run, using these parameters.

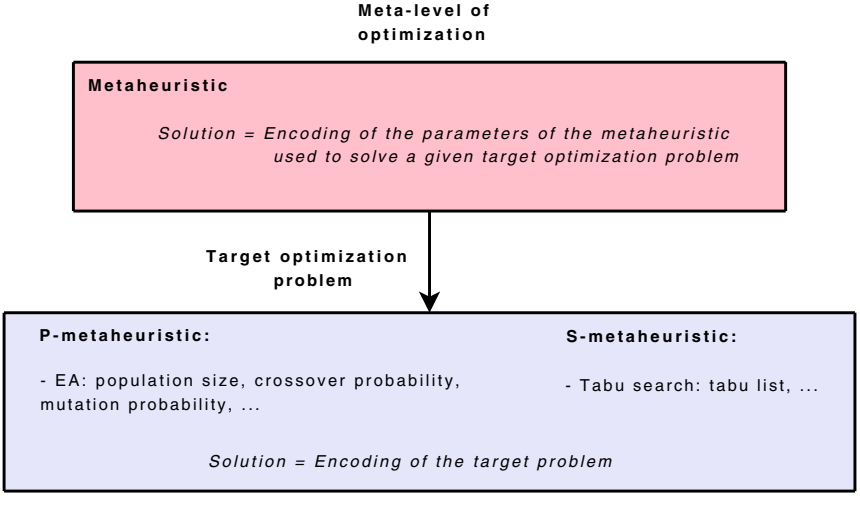

Fig. 1.12 Meta-level of optimization in metaheuristics. Metaheuristics are used to optimize the parameters of another metaheuristic.

1.2.2 Implementation Issues

The structure of the taxonomy concerning the implementation issues is shown in figure 1.13. A discussion about this taxonomy then follows.

1.2.2.1 Dedicated versus General-Purpose Computers

Application specific computers differ from general purpose ones in that they usually only solve a small range of problems, but often at much higher rates and lower cost.

[7] This procedure is also called meta-evolution.

Their internal structure is tailored for a particular problem, and thus can achieve much higher efficiency and hardware utilization than a processor which must handle a wide range of tasks.

In the last years, the advent of programmable logic devices has made easier to build specific computers for metaheuristics such as simulated annealing [3] and genetic algorithms [149]. A general architecture acting as a template for designing a number of specific machines for different metaheuristics (SA, TS, etc) may be constructed [2]. The processor is built with XILINX FPGAs and APTIX interconnection chips. Experiments evaluating a simulated annealing algorithm to solve the traveling salesman problem achieved a speedup of about 37 times over an IBM RS6000 workstation. To our knowledge, this approach has not been yet proposed for hybrid metaheuristics.

Nowadays, the use of GPU (Graphical Processing Unit) devices is more and more popular in many application domains. Indeed, those devices are integrated into many workstations to deal with visualization tasks. The idea is to exploit those available resources to improve the effectiveness of hybrid metaheuristics.

1.2.2.2 Sequential versus Parallel

Most of the proposed hybrid metaheuristics are sequential programs. According to the size of problems, parallel implementations of hybrid algorithms have been considered. The easiness to use a parallel and distributed architecture has been acknowledged for the HTH hybrid model.

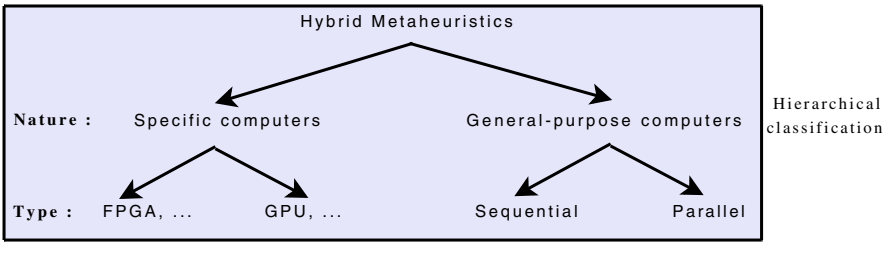

Fig. 1.13 Classification of hybrid metaheuristics (implementation issues)

1.2.3 A Grammar for Extended Hybridization Schemes

Given a set of metaheuristics A_i, a classification of basic hybridizations has been presented, in which the following notations can be described:

- $LRH(A_1(A_2))$ (homogeneous, heterogeneous) (partial, global) (specialist, general): the metaheuristic A_2 is embedded into the single-solution metaheuristic A_1.
- $HRH(A_1 + A_2)$ (homogeneous, heterogeneous) (partial, global) (specialist, general): the self-contained metaheuristics A_1 and A_2 are executed in sequence.

- $LTH(A_1(A_2))$ (homogeneous, heterogeneous) (partial, global) (specialist, general): the metaheuristic A_2 is embedded into the population-based metaheuristic A_1.
- $HTH(A_1,A_2)$ (homogeneous, heterogeneous) (partial, global) (specialist, general): the self-contained metaheuristics A_1 and A_2 are executed in parallel and cooperate.

These hybridizations should be regarded as primitives that can be combined in different ways. The grammar given in figure 1.14 generalizes the basic hybridization schemes. One of the practical importance of the grammar is to specify the hybrid heuristic to use, if a metaheuristic problem solving tool is used.

<hybrid-metaheuristic> ⟶ <design-issues><implementation-issue>
<design-issues> ⟶ <hierarchical><flat>
<hierarchical> ⟶ <LRH> | <LTH> | <HRH> | <HTH>
<LRH> ⟶ LRH(<S-metaheuristic>(<metaheuristic>))
<LTH> ⟶ LTH(<P-metaheuristic>(<metaheuristic>))
<HRH> ⟶ HRH(<metaheuristic> + <metaheuristic>)
<HTH> ⟶ HTH(<metaheuristic>)
<HTH> ⟶ HTH(<metaheuristic>, <metaheuristic>)
<flat> ⟶ (<type> , <domain> , <function>)
<type> ⟶ homogeneous | heterogeneous
<domain> ⟶ global | partial
<function> ⟶ general | specialist
<implementation-issue> ⟶ *<specific computers>* | *<general-purpose computers>*
<specific computers> ⟶ FPGA | GPU | ...
<general-purpose computers> ⟶ sequential | parallel
< metaheuristic > ⟶ <S-metaheuristic> | <P-metaheuristic>
<S-metaheuristic> ⟶ LS | TS | SA | TA | NM | GDA | ILS | GRASP | ...
<P-metaheuristic> ⟶ EA | SS | ACO | PSO | AIS | BC | EDA | CA | CEA | ...
<metaheuristic> ⟶ <hybrid-metaheuristic>

Fig. 1.14 A grammar for extended hybridization schemes

Example 1.9. **Extended hybridization schemes:** let us present some examples of extended hybridization schemes (Fig. 1.15). Boese et al. [25] suggested an adaptive multi-start (AMS) approach, which may be seen as a HRH(LS + LTH(GA(LS))) scheme. First, AMS generates a set of random starting solutions and runs an LS algorithm for each solution to find corresponding local optima. Then, AMS generates new starting solutions by combining features of the T best local optima seen so far, with T being a parameter of the approach. This mechanism bears some resemblance to GAs, but differs in that many solutions (instead of just two) are used to generate the new starting solutions. New local optima are obtained by running the LS algorithm from these new starting solutions, and the process iterates until some stop criterion is met.

D. Levine has used a HTH(HRH(GH+LTH(GA(LS)))) hierarchical scheme in his PhD to solve set partitioning problems. Efficient results have been obtained with a

parallel static implementation in solving big sized problems in real world applications (airline crew scheduling) [117]. At the first level, a HTH hybrid based on the island model of parallel genetic algorithms is used. The initial population of each GA was generated by a greedy heuristic (the Chvatal heuristic [38]), and a local search algorithm was used to improve the solutions at each generation of the GA. The same hybrid scheme with a sequential implementation has been used in [26] to solve the traveling salesman problem. The local search algorithms used are the well known 2-opt and or-opt heuristics. The author reported some interesting results on the 442 and 666-city problems. He found the optimum of the 442-city problem, and a solution within 0.04% of the optimum for the 666-city problem.

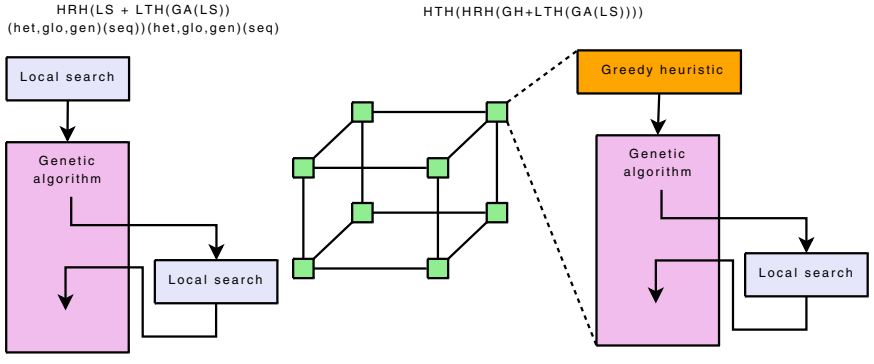

Fig. 1.15 Extended hybridization schemes

The objective of this chapter is far from providing an exhaustive list of research works using hybrid metaheuristics. Following this grammar, more than 125 annotated hybrid metaheuristics may be found in [163]. This shows the usefulness of the taxonomy.

1.3 Combining Metaheuristics with Mathematical Programming

Metaheuristics and exact algorithms are complementary optimization strategies in terms of the quality of solutions and the search time used to find them. In the last few years, solving exactly important optimization problems using for example integer programming techniques has improved dramatically. Moreover, the availability of efficient optimization software, libraries and frameworks for mathematical programming and high-level modeling languages will lead to more hybrid approaches combining metaheuristics and exact optimization algorithms. In the next section, the main mathematical programming exact approaches that can be used to solve optimization problems are presented. Then, an instantiation and extension of our classification to hybrid schemes combining mathematical programming approaches and metaheuristics is presented.

1.3.1 Mathematical Programming Approaches

The main mathematical programming approaches may be classified as follows:

- **Enumerative algorithms:** this class of algorithms contains tree search algorithms such as branch and bound and dynamic programming. They are based on a divide and conquer strategy to partition the solution space into subproblems and then optimizing individually each subproblem.
- **Relaxation and decomposition methods:** this class of methods are based on relaxation techniques such as the Lagrangian relaxation [69], and decomposition methods such as the Bender's decomposition and the continuous semi-definite programming .
- **Cutting plane and pricing algorithms:** this class of algorithms is based on polyhedral combinatorics in which the search space is pruned.

1.3.1.1 Enumerative Algorithms

Enumerative methods include branch and bound, dynamic programming, A*, and other tree search algorithms. The search is carried out over the whole search space, and the problem is solved by subdividing it in simpler subproblems.

Branch and bound algorithm is one of the most popular method to solve optimization problems in an exact manner. The algorithm is based on an implicit enumeration of all solutions of the considered optimization problem. The search space is explored by dynamically building a tree whose root node represents the problem being solved and its whole associated search space. The leaf nodes are the potential solutions and the internal nodes are subproblems of the total solution space. The size of the subproblems is increasingly reduced as one approaches the leaves.

The construction of such a tree and its exploration are performed using two main operators: *branching* and *pruning* (Fig. 1.16). The algorithm proceeds in several iterations during which the best found solution is progressively improved. The

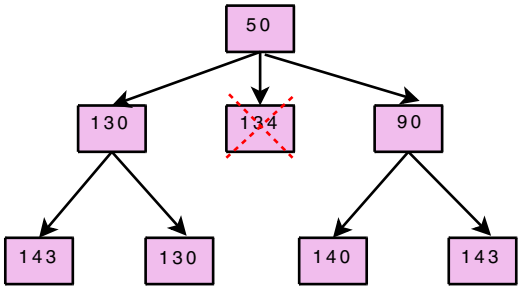

Fig. 1.16 The branch and bound algorithm. This figure shows the nodes actually explored in the example problem, assuming a depth-first and left-to-right search strategy. The subtree rooted at the second node on level 2 is pruned because the cost of this node (134) is greater than that of the cheapest solution already found (130).

generated nodes and not yet treated are kept in a list whose initial content is limited to only the root node. The two operators intervene at each iteration of the algorithm. The branching strategy determines the order in which the branches are explored. Many branching strategies may be applied such as the *depth-first*, the *breadth-first*, and the *best-first* strategies. The *pruning* strategy eliminates the partial solutions that do not lead to optimal solutions. This is done by computing the lower bound associated to a partial solution. If the lower bound of a node (partial solution) is greater than the best solution found so far or a known upper bound of the problem, the exploration of the node is not needed. The algorithm terminates if there are no more nodes to branch or all nodes are eliminated. Hence, the most important concepts in designing an efficient branch and bound algorithm are the quality of the bounds and the branching strategy.

Example 1.10. **Branch and bound algorithm on the TSP:** let us consider the TSP problem. A straightforward method for computing a lower bound on the cost of any solution may be the following:

$$\frac{1}{2}\sum_{v\in V} \text{sum of the costs of the two least cost edges adjacent to } v$$

For the example shown in figure 1.17, the lower bound is associated to the edges $(A,D),(A,B),(B,A),(B,E),(C,B),(C,A),(D,A),(D,C),(E,B),(E,D)$ and then is equal to 17.5. In the search tree (Fig. 1.17), each node represents a partial solution. Each partial solution is represented by the set of associated edges (i.e. edges that must be in the tour) and the non associated edges (i.e. set of edges that must not be on the tour). The branching consists in generating two children nodes. A set of additional excluding and including edges is associated to each child. Two rules may be applied. An edge (a,b) must be included if its exclusion makes it impossible for a or b to have two adjacent edges in the tour. An edge (a,b) must be excluded if its inclusion causes for a or b to have more than two adjacent edges in the tour or would complete a non-tour with edges already included. The pruning consists first in computing the lower bounds for each child. For instance, if the edge (A,E) is included and the edge (B,C) is excluded, the lower bound will be associated to the following selected edges $(A,D),(A,E),(B,A),(B,E),(C,D),(C,A),(D,A),(D,C),(E,B),(E,A)$ and is equal to 20.5. If the lower bound associated to a node is larger than the known upper bound, the node is proved to be unable to generate an optimal solution and then is not explored. A best-first search heuristic is considered in which the child with the smaller lower bound is explored first. The upper bound is updated each time a new complete solution is found with a better cost.

The *dynamic programming* (DM) approach is based on the recursive division of a problem into simpler subproblems. This procedure is based on the *Bellman's principle* which says that "the sub-policy of an optimal policy is itself optimal with regard to the start and end states" [21].

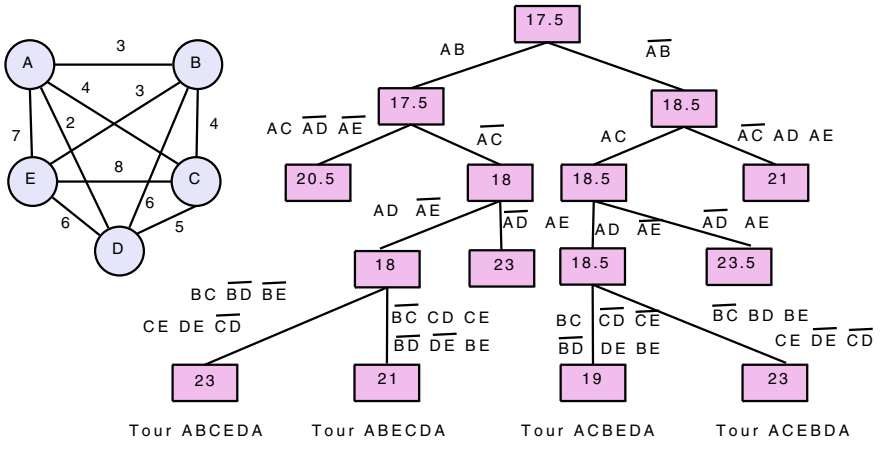

Fig. 1.17 Illustration of the branch and bound algorithm on the traveling salesman problem

Designing a dynamic programming procedure for a given problem needs the definition of the following components [23]:

- Define the *stages* and the *states*. A problem can be divided into a number of stages N. A number of states are associated to each stage.
- Define the cost of the initial stage and states. There is an initial state of the system x_0.
- Define the recursive relation for a state at stage k in terms of states of previous stages. The system takes the state x_k at the stage k. At the k stage, the state of the system change from x_k to x_{k+1} using the following equation

$$x_{k+1} = f_k(x_k, u_k)$$

  where u_k is a control that takes values from a given finite set, which may depends on the stage k. The transition from the state k to $k+1$ involves a cost $g_k(x_k, u_k)$. The final transition from $N-1$ to N involves the terminal cost $G(x_N)$. The functions f_k, g_k and G must be determined.

Given a control sequence $(u_1, u_2, ..., u_{N-1})$, the corresponding state sequence will be $(x_0, ..., x_N)$ which is determined from the initial state x_0 using the equation below. In dynamic programming, the objective is to find the optimal control sequence minimizing the total cost:

$$G(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$

DP have been successfully applied to knapsack, planning and routing-type problems, in which it is easy to define efficient recursive relationships between stages.

Example 1.11. **Dynamic programming for the $\{0,1\}-$knapsack problem:** let us consider the following instance for the knapsack problem with a total capacity equal to 5 (see table 1.1).

Table 1.1 An instance for the knapsack problem with a capacity of 5

Item (i)	**Weight** (w_i)	**Utility** (u_i)
1	2	65
2	3	80
3	1	30

The stages are represented by the items. The number of stages are then equal to the number of items (3). The state y_i at stage i represents the total weight of items i and all following items in the knapsack. The decision at stage i corresponds to how many items i to place in the knapsack. Let us call this value k_j. This leads to the following recursive formulas: let $f_j(y_j)$ be the value of using y_j units of capacity for items j and following. Let $\lfloor a \rfloor$ represents the largest integer less than or equal to a.

$$f_3(y_i) = 30.y_i$$

$$f_j(y_i) = max_{k_i \leq \lfloor \frac{y_i}{w_i} \rfloor} \{u_i k_i + f_{i+1}(y_i - w_i k_i)\}$$

1.3.1.2 Relaxation and Decomposition Methods

Relaxation methods consist in relaxing a strict requirement in the target optimization problem. In general, a given strict requirement is simply dropped completely or substituted by another one which is more easily satisfied. The most used relaxation techniques are the LP-relaxation and the Lagrangian relaxation. In addition to their use on solving optimization problems, relaxation methods are also used to generate bounds.

Linear programming relaxation: linear programming relaxation (LP-relaxation) is a straightforward approach which consists in ignoring the integrity constraints of an integer program (IP). Once the integrity constraints are dropped, the problem can be solved using LP solvers. This gives a lower bound for the problem. If the solution found satisfies the integer constraints (generally not true), it will be considered as the optimal solution for the IP program. If the relaxed problem is infeasible, then so is the IP program. LP-relaxation is widely used in branch and bound algorithms to solve IP problems in which the branching is performed over the fractional variables.

Lagrangian relaxation: Lagrangian relaxations are widely used to generate tight lower bounds for optimization problems. The main idea is to remove some constraints and incorporate them in the objective function. For each constraint, a penalty function is associated. The choice of which constraints are handled in the objective

function is important. More complicated constraints to satisfy are preferable as they generate an easiest problem to solve. Given the following LP problem:

Max $c^T x$
s.t. $Ax \leq b$
with $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m,n}$

The set of constraints A is split into two sets: $A_1 \in R^{m_1,n}$ and $A_2 \in R^{m_2,n}$, where $m_1 + m_2 = m$. Then, the subset of constraints A_2 is integrated into the objective function which gives the following Lagrangian relaxation of the original problem:

Max $c^T x + \lambda^T (b_2 - A_2 x)$
s.t. $A_1 x \leq b_1$
with $x \in \mathbb{R}^n$, $A_1 \in \mathbb{R}^{m_1,n}$ and $A_2 \in \mathbb{R}^{m_2,n}$

where $\lambda = (\lambda_1, ..., \lambda_{m_2})$ are non negative weights which penalize the violated constraints A_2. The efficiency of Lagrangian relaxation depends on the structure of the problem; there is no general theory applicable to all problems. Lagrangian relaxation may find bounds which are tighter than the LP-relaxation. The problem is solved iteratively until optimal values for the multipliers are found. One of the main issues in the Lagrangian relaxation is the generation of the optimal multipliers. This difficult problem can be solved by metaheuristics.

In practice, decomposition methods are used to solve large IP problems. Among the numerous decomposition approaches one can refer to Bender's decomposition and Dantzig-Wolfe decomposition.

Bender's decomposition: the Bender's decomposition algorithm is based on the notion of complicated variables. It consists in fixing the values of complicated variables and solves the resulting reduced problem iteratively [22]. Given a MIP problem :

Max $c^T x + h^T y$
s.t. $Ax + Gy \leq b$
with $x \in \mathbb{Z}_+^n$ and $y \in \mathbb{R}_+^p$

If the set of variables x is fixed, the following linear program is obtained

$$z_{LP}(x) = max\{hy / Gy \leq b - Ax\}$$

and its dual

$$min\{u(b - Ax) / uG \geq h, u \in R_+^m\}$$

If the dual polyhedron is assumed to be not empty and bounded, the MIP model can be formulated as follows:

$$z = max_{x \in Z_+^n} (cx + min_{i \in 1,...,T} (u^i (b - Ax)))$$

This model can be reformulated as:

$$z = max\{\eta / \eta \leq u^i (b - Ax), i \in 1, ...T, x \in Z_+^n\}$$

Then, the algorithm finds cutting planes based on the dual problem. The cutting planes are added to the problem and the problem is re-solved.

1.3.1.3 Branch and Cut and Price Algorithms

The objective of the following popular techniques is to generate tighter IP relaxations.

Cutting plane: cutting plane approaches have been proposed in 1958 by Gomory [81]. The use of cuts can improve greatly branch and bound algorithms. In general, cutting plane algorithms consist in iteratively adding some specific constraints to the LP-relaxation of the problem. Those constraints represent restrictions to the problem so that the linear programming polytope closely approximates the polyhedron represented by the convex hull of all feasible solutions of the original IP problem. A good survey of branch and cut algorithms and their use for different families of optimization problems may be found in [108] [131].

Column generation: column generation has been first applied by Gilmore and Gomory [77]. Column generation (i.e. Dantzig-Wolfe decomposition) generates a decomposition of the problem into a master and subproblems (Fig. 1.18). A good survey may be found in [12].

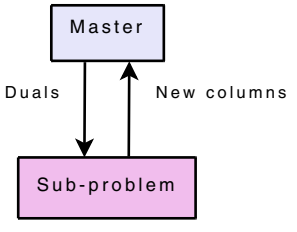

Fig. 1.18 Branch and price approach

1.3.2 Classical Hybrid Approaches

Exact MP algorithms are known to be time and/or memory consuming. In general they cannot be applied to large instances of difficult optimization problems. On one hand their combination with metaheuristics may improve the effectiveness of heuristic search methods (i.e. getting better solutions). On the other hand, this type of combination allows the design of more efficient exact methods (i.e. finding optimal solutions in shorter time). The following sections illustrate, for each class of hybrids belonging to the presented taxonomy, some hybridization schemes combining exact MP algorithms and metaheuristics.

1.3.2.1 Low-Level Relay Hybrids (LRH)

This class of algorithms represents hybrid schemes in which a metaheuristic approach (resp. exact approach) is embedded into an exact approach (resp. S-metaheuristic approach) to improve the search strategy. In this usual combination, a given metaheuristic or exact algorithm solves a problem of a different nature of the considered optimization problem.

Embedding S-metaheuristics into exact algorithms: indeed, metaheuristics may solve many search problems involved in the design of an exact method such as the node selection strategy, upper bound generation, column generation (Fig. 1.19):

- **Bounding:** providing an upper bound associated to a node of the branch and bound algorithm can be designed using a metaheuristic. Indeed, the partial solution is completed by a given metaheuristic and then a local upper bound is provided.
- **Cutting:** in the branch and cut algorithm, the cutting plane generation problem is a crucial part of the algorithm: the part that looks for valid inequalities that cut off the current non-feasible linear program (LP) solution. Metaheuristics may be used in this *separation procedure*. For instance, this approach has been proposed for the CVRP (Capacitated Vehicle Routing Problem) [10]. Some metaheuristics (e.g. tabu search, greedy heuristics) have been designed to extract a set of violated capacity constraints of the relaxed problem.
- **Pricing:** in the branch and price algorithm, the pricing of columns may be carried out by a metaheuristic [67].

Some metaheuristic ingredients may also be used in tree search algorithms such as the concepts of tabu lists and aspiration criteria [139].

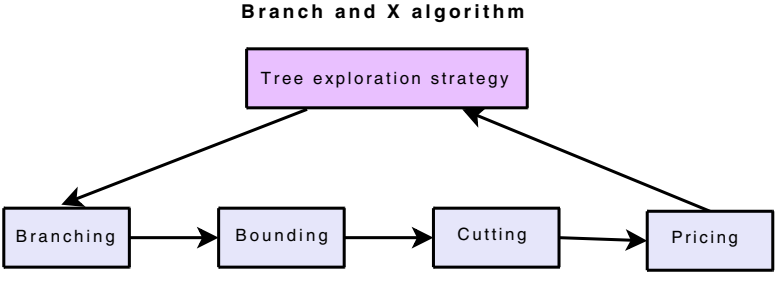

Fig. 1.19 LRH cooperation in which a metaheuristic can be used in the design of some search components of branch and X family of algorithms (e.g. branch and bound, branch and cut, branch and price): selection of the node to explore, generation of an upper bound, cutting plane generation, column generation selection, etc.).

Embedding exact algorithms into S-metaheuristics: many combinations may be designed in which exact algorithms are embedded into search components of S-metaheuristics.

Very large neighborhoods: S-metaheuristics may be improved using very large neighborhoods. The concept of large neighborhoods is also used in the ILS (perturbation operator [120]) and the VNS metaheuristics. Mathematical programming approaches may be used to search efficiently those large neighborhoods to find the best or an improving solution in the neighborhood (Fig. 1.20). Algorithms such as branch and bound, dynamic programming [138], network flow algorithms [56], and matching algorithms [83] have been used to explore large neighborhoods defined for different important optimization problems. Some hybrid schemes explore the *whole* neighborhood while other neighborhood search algorithms explore a *subset* of the neighborhood. If no polynomial time algorithm exists to search the whole neighborhood, a partial search is generally performed.

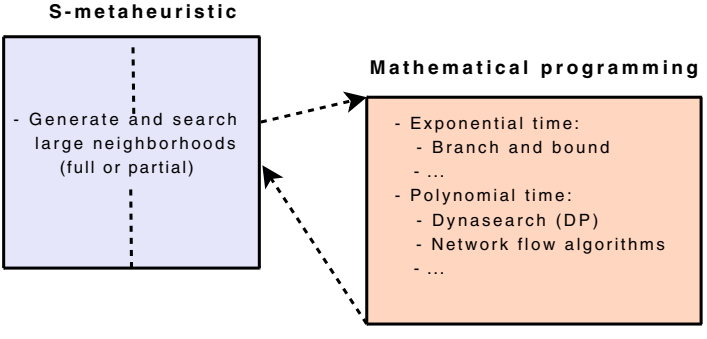

Fig. 1.20 LRH cooperation where a mathematical programming approach can be used for the efficient search of a very large neighborhood into S-metaheuristics.

Example 1.12. **Hyperopt:** the hyperopt S-metaheuristic explores only a subset of a very large neighborhood [27]. The hybrid algorithm has been used to solve the asymmetric traveling salesman problem. The move operator is based on hyperedges which represent subpaths of the tour. A hyperedge $H(i,j)$ is represented by its start node i, end node j and length k. A k-hyperopt move consists in deleting two hyperedges $H(i_1,i_{k+1})$ and $H(j_1,j_{k+1})$ of length k. It is supposed that $H(i_1,i_{k+1}) \bigcap H(j_1,j_{k+1}) = \phi$, i.e. the hyperedges share no common edges. Then, the move operator adds edges to the hyperedges $H(i_{k+1},j_1)$ and $H(j_{k+1},i_1)$ to construct a feasible tour (Fig. 1.21). The size of the hyperedge neighborhood grows exponentially with k. The neighborhood search algorithm is reduced to a smaller TSP problem. The algorithms used are based on enumeration for small k and a dynamic programming algorithm for medium values of k.

Some search concepts of exact algorithms may be used in S-metaheuristics. Efficient mathematical programming approaches that generate "good" lower bounds exist

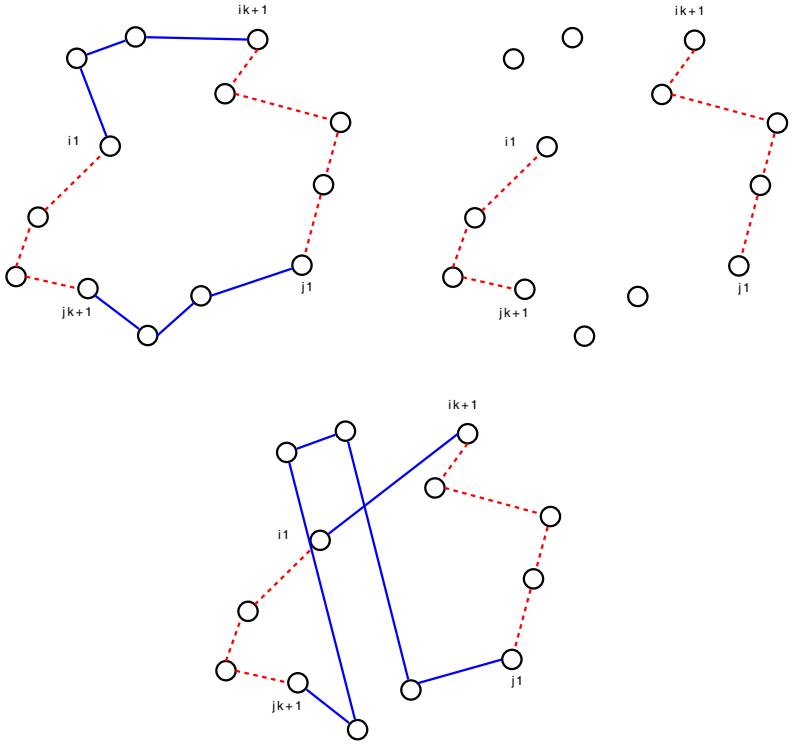

Fig. 1.21 Large neighborhood based on hyperedge for the asymmetric traveling salesmen problem

for many optimization problems. Information related to lower bounds, obtained for example by Lagrangian relaxation, can be exploited into a metaheuristic to intensify the search in promising regions of the search space. For instance, information based on Lagrangian multipliers are exploited to guide the metaheuristic in solving the set covering problem [17]. Lower bounds have been used in S-metaheuristics such as tabu search to improve the search [88] [56].

1.3.2.2 Low-Level Teamwork Hybrids (LTH)

Recall that in this class of hybrid algorithms, a search component of a P-metaheuristic is replaced by another optimization algorithm. Concerning the combination of P-metaheuristics and MP algorithms, two main hybrid approaches may be considered: exact search hybrid algorithms in which a P-metaheuristic is embedded into an exact algorithm, and heuristic search algorithms in which an exact algorithm is embedded into a P-metaheuristic.

Embedding a P-metaheuristic into an Exact Algorithm: as mentioned previously in this chapter, the questions arising in designing a branch and bound algorithm are:

- **Branch ordering:** how the problem to solve (node of the tree) is decomposed into subproblems? On which variable the next branching is applied? Indeed, near-optimal solutions obtained by metaheuristics may guide the branch and bound to apply an efficient branch ordering by giving preference to branches which share common values with near-optimal solutions.
  The node selection problem in tree search based on branch and bound may be solved by metaheuristics. For instance, genetic programming approaches have been used to deal with the node selection problem [113].
- **Variable selection:** in which subproblem (child node) the search will be performed in the next step? What value should first be assigned to the branching variable? Information obtained from branch and bound tree can be used by heuristic algorithms to determine a better strategy for variable selection [113].

An exact algorithm constructs partial solutions which are used to define a search space for a metaheuristic. Then, the obtained results are exploited in order to refine the bounds or generate the columns into a branch and cut algorithm.

Example 1.13. **Local branching :** the *local branching* exact approach has been proposed in [68]. It uses the principle of local search heuristics. The search space is partitioned by introducing branching conditions expressed through (invalid) linear inequalities called local branching cuts. Let us consider a MIP problem with $\{0,1\}$ variables. The $k-opt$ neighborhood is considered. The main principle of the local branching method is to iteratively solve a subproblem corresponding to the neighborhood $k-opt$ of a partial solution s. Two partitions are then considered: $p_1 = \{x \in \{0,1\}^n / \Delta(x,s) \leq k\}$ and $p_2 = \{x \in \{0,1\}^n / \Delta(x,s) \geq k+1\}$, where Δ represents the Hamming distance, and n the size of the problem. The problem associated to p_1 is solved. A new subproblem is generated if an improved solution is found. Otherwise, the other problem is solved using the standard procedure.

Embedding an exact algorithm into P-metaheuristic: in this hybrid scheme, some search components of a P-metaheuristic induce optimization problems which are solved by exact algorithms (Fig. 1.23).

Example 1.14. **Exact algorithms into recombination operators:** exact algorithms may be integrated into recombination operators of P-metaheuristics such as evolutionary algorithms to find the best offspring from a large set of possibilities. The induced problem $\texttt{Recombination}(S_1, S_2)$ is defined to generate the best offsprings from the parents S_1 and S_2. A common idea is to keep the common elements of the parents and explore all the other possibilities to generate better offsprings (Fig. 1.22). For instance, a branch and bound algorithm (resp. dynamic programming) has been used into the crossover operator of a genetic algorithm in solving permutation problems [48] (resp. [182]). For some specific problems, polynomial exact algorithms may also be used such as minimum spanning tree algorithms [95], matching algorithms in a bipartite graph [4] [11] for optimized crossover operators.

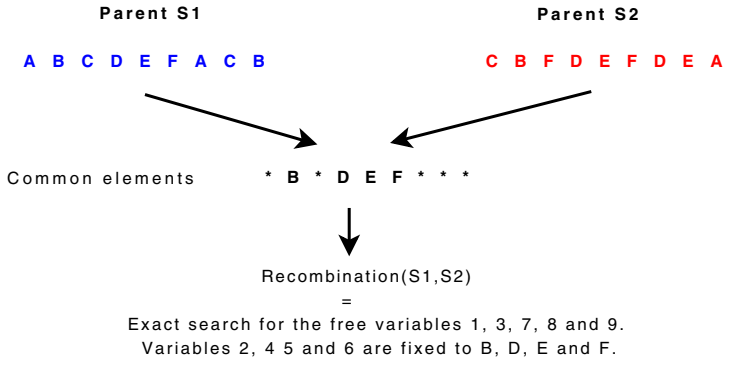

Fig. 1.22 Using exact algorithms into recombination operators (e.g. crossover) of P-metaheuristics

Large neighborhood search algorithms integrated in P-metaheuristics belong typically to the LTH class of hybrids. For instance, the mutation operator in EAs can also be substituted by MP algorithms which explore large neighborhoods (Fig. 1.23).

Exact decoding: exact algorithms can also be used as decoders of incomplete solutions carried out by metaheuristics. This hybrid strategy is applied in the case where metaheuristics use an incomplete encoding for the problem. Once a good incomplete solution is found, exact algorithms complete optimally the missing part of the encoding.

Exact search ingredients: some search ingredients of exact algorithms can also be used in P-metaheuristics:

- **Lower bounds:** the use of lower bounds into a P-metaheuristic can improve the search. Lower bounds have been used in the construction phase of the ant colonies P-metaheuristic to solve the quadratic assignment problem (QAP) [122]. The well known Gilmore-Lawler lower bound and the values of the dual variables are used to order the locations during the construction phase. The impact of the location in a given QAP instance depends on the value of its associated dual variable. The concept of bounds has been used into evolutionary algorithms for the mutation and the crossover operators [169] [57]. Indeed, partial solutions that exceed a given bound are deleted. The bounds are computed using the linear and Lagrangian relaxation, and tree search methods.
- **Partial solutions :** the evaluated partial solutions (subproblems) maintained by the branch and bound family of algorithms may provide interesting initial solutions to improve. The evaluation of those partial solutions will guide the metaheuristics to more promising regions of the search space [122]. The partial solution with the least cost lower bound suggests a promising region by giving additional information.

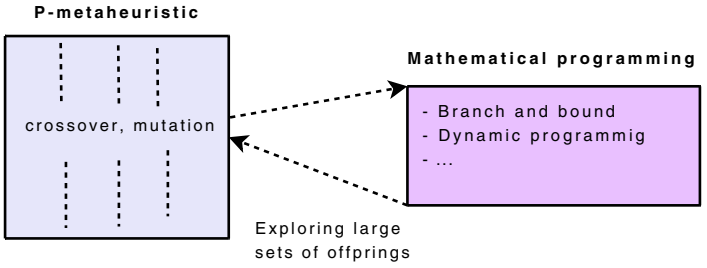

Fig. 1.23 LTH heuristic cooperation: exact algorithms are used as search components of a P-metaheuristic (e.g. recombination, mutation).

1.3.2.3 High-Level Relay Hybrids (HRH)

This class of cooperation, where self-contained algorithms are used in sequence, is the most popular in practice. This may be seen as a pre-processing or a post-processing step. Some information is provided in sequential between the two families of algorithms (metaheuristics and MP algorithms) (Fig. 1.25).

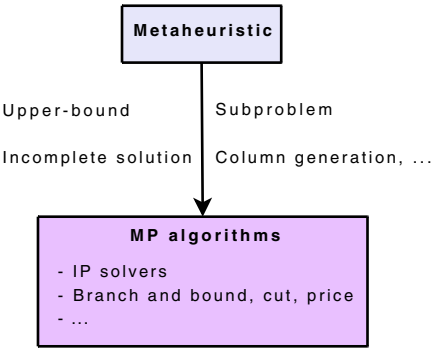

Fig. 1.24 HRH cooperation: information provided by metaheuristics to MP algorithms

Information provided by metaheuristics: in the case where the information is provided by the metaheuristics, the most natural and trivial hybrid approach is to start with a metaheuristic to find a "good" *upper bound* which will be used by a MP algorithm in the bounding phase (Fig. 1.25). Indeed, the efficiency of the search (pruning phase) is largely dependent on the quality of the upper bound.

Using the characteristics of generated high quality solutions, metaheuristics can be used to reduce the size of the original problem. Then, the exact method can be applied to solve the reduced problem. This approach is interesting for optimization problems where "good solutions" share many components [8]. This allow to reduce the problem into a much smaller problem which can be solved exactly by state-of-the-art mathematical programming algorithms. The reduction phase may concern:

- **Partitioning of decision variables:** in this strategy, the decision variables are partitioned into two sets X and Y. The metaheuristic will fix the variables of the set X and the exact method will optimize the problem over the set Y. Hence, the generated subproblems are subject to free variables in the set Y and freezed variables in the set X. Those subproblems are solved exactly.

  A set of high quality solutions may be obtained by a P-metaheuristic or an iterated S-metaheuristic. The characteristics of this set can be exploited to define smaller problems by fixing some variables and solve the resulting subproblems by exact algorithms. An example of such strategy is the Mimausa method for the quadratic assignment problem [125]. The method builds at each iteration a subproblem by fixing k decision variables and solves it by a branch and bound algorithm.

  Example 1.15. **Reducing problems by metaheuristics to be solved by MP algorithms:** analyzing the landscape for the TSP problem, one can observe that local optimum solutions share many edges with the global optimum and they are concentrated in the same region of the search space (big valley structure) [165]. This characteristic has been exploited in [44] to design one of the most efficient heuristic for the TSP: the tour merging heuristic. The tour merging heuristic consists of two phases: the first phase generates a set T of "good" tours using the Lin-Kernigham algorithm on the input graph $G = (V,E)$. Then, a dynamic programming algorithm is applied on a restricted graph $G^{'} = (V,E')$, where $E' = \{e \in E / \exists t \in T, e \in t\}$. The exact algorithm solves instances up to 5000 cities.

  For the p-median problem, the same remark holds in the analysis of its landscape [146]. The first phase is based on an iterated S-metaheuristic using different random initial solutions. The problem is reduced in terms of the number of nodes (location facilities) using the *concentration set* (CS) concept. The integer programming model of the restricted problem is solved using respectively a linear programming relaxation (CPLEX solver) and a branch and bound. The authors exploit the fact that more than 95% of linear programming relaxation optimal solutions are integers.

  For continuous optimization, HRH hybrid schemes are very popular. For instance, a hybrid method combining tabu search and the simplex algorithm provides interesting results in solving complex continuous functions [35].
- **Domain reduction:** this strategy consists in reducing the domain of values that the decision variables can take. The metaheuristic will perform a domain reduction for the decision variables and then an exact method is used over the reduced domains. For instance, a GA may be used to find promising ranges for decision variables and then tree search algorithms are considered to find the optimal solution within those ranges [129].

Information provided by exact algorithms: in the case where the information is provided by an exact algorithm, many hybrid approaches may be designed:

- **Partial solutions:** partial solutions are first provided by an exact algorithm which are then completed by a metaheuristic.

- **Problem reduction:** in this strategy, a problem reduction is carried out by an exact algorithm. For instance, a tree search algorithm has been used to reduce the size of a nurse scheduling problem [58]. Then, a tabu search strategy is applied to solve the problem within a simplified objective function [58].
- **Relaxed optimal solutions and their duals:** the optimal solutions for relaxed formulation (e.g. LP-relaxation, Lagrangian relaxation) of the problem and its duals may be exploited by metaheuristics.

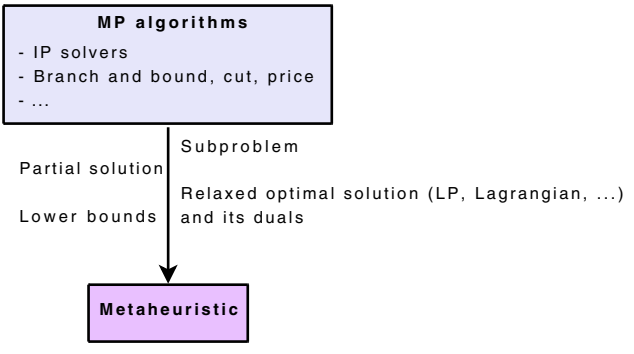

Fig. 1.25 HRH cooperation: information provided by MP algorithms to metaheuristics

Example 1.16. **LP-relaxations as an input for metaheuristics:** information gathered from solutions obtained by LP-relaxations of MIP problems may be used as an input for a metaheuristic. A straightforward approach is the "dive and fix" strategy, where the value of a subset of the integer variables are fixed and the resulting LP problem is solved. This strategy iterates until the LP finds an integer solution. This will restrict the search space of metaheuristics in promising regions. This idea has been used to design an efficient hybrid approach for the 0-1 multidimensional knapsack problem [175]. Many linear relaxation of the MIP formulation of the problem including different constraints on the number of elements of the knapsack are solved exactly. The obtained solutions are exploited to generate initial solutions for multiple tabu search metaheuristics.

1.3.2.4 High-Level Teamwork Hybrids (HTH)

Few strategies belong to this class of hybrids which combines metaheuristics and MP algorithms in a parallel cooperative way. However this is a promising class of hybrids. A set of agents representing metaheuristics and MP algorithms are solving global, partial or specialist optimization problems and exchanging useful informations. The majority of proposed approaches fall in the class of partial and specialist hybrids. Indeed, the search space is generally too large to be solved by an exact approach. One of the main issues in the HTH hybrid is the information exchanged

between metaheuristics and MP algorithms. The different complementary algorithms solving different problems may exchange any information gathered during the search to improve the efficiency and the effectiveness of the hybrid approach: solution(s), subproblems, relaxed optimal solutions and its duals, upper bounds, lower bounds, optimal solutions for subproblems, partial solutions, etc.

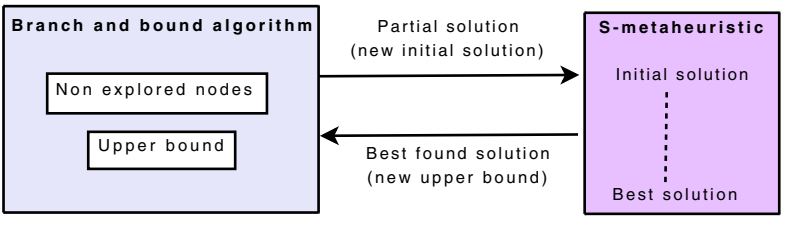

Fig. 1.26 HTH cooperation between metaheuristics and MP algorithms

Example 1.17. **Parallel cooperation between a branch and bound and a S-metaheuristic:** in a parallel cooperation between branch & bound algorithms and a S-metaheuristic, the following information may be exchanged (Fig. 1.26):

- **From a branch & bound algorithm to a S-metaheuristic:** a subproblem of the branch and bound (node of the tree, partial solution) with least-cost lower bound may be used by a S-metaheuristic to generate an initial solution. The lower bound is used to predict potential interesting search regions. This process may be initiated as a diversification search, when the classical "intensification" process is terminated. Indeed, this partial solution provides a promising area for a S-metaheuristic to explore. The non explored node list maintained by a branch and bound provides a metaheuristic with new initial solutions.
- **From a S-metaheuristic to a branch & bound algorithm:** the best solution identified so far by a metaheuristic may be used in branch and bound algorithms for a better pruning of the search tree. Indeed, better is the upper bound, more efficient is the pruning of the search tree. This information is exchanged each time the best solution found is improved.

In generalist and global hybrids, where all the algorithms are solving the same target problem, the space of design is reduced. For instance, a parallel HTH hybrid which consists in combining a branch and bound algorithm with simulated annealing has been designed [134]. The SA algorithm sends improved upper bounds to the exact algorithm. Any integer bound obtained by the B&B execution is passed to SA and used as an alternative reheated solution.

In specialist hybrids, where the algorithms are solving different problems, many strategies may be proposed (Fig. 1.27). For instance, a parallel cooperation between a local search metaheuristic and a column generation (branch and price) algorithm

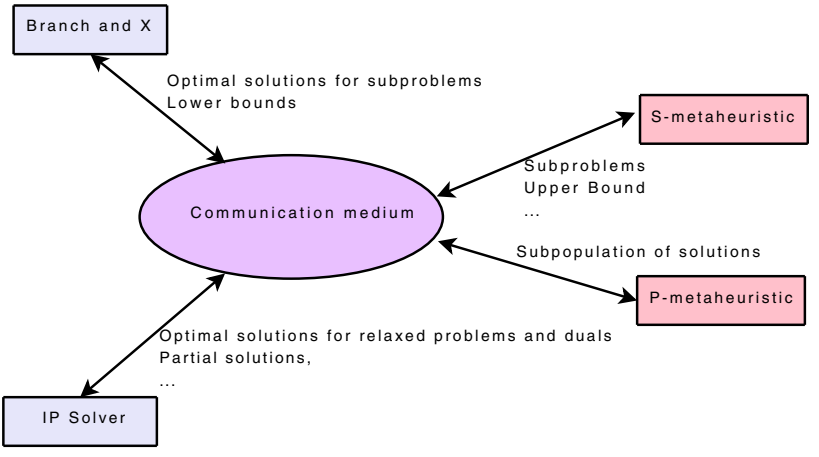

Fig. 1.27 Specialist HTH cooperation between S-metaheuristics and MP algorithms

to solve the VRP problem has been proposed [33]. The local search algorithm is used to generate new columns for a branch and cut algorithm.

Extending the grammar, presented in section 1.14, with hybrid schemes combining metaheuristics with exact optimization algorithms has been presented in [102]. More than 60 annotated hybrid approaches are detailed in the paper. Other examples of combining metaheuristics with exact algorithms may be found in the survey papers [140] [55] [102].

1.4 Combining Metaheuristics with Constraint Programming

Constraint programming (CP) is a modeling and an exact[8] search paradigm based on constraint satisfaction techniques which are largely used in the artificial intelligence community [9]. CP has been applied successfully to many combinatorial optimization problems with tightly-constrained search problems, while metaheuristics perform well for under-constrained optimization problems.

Nowadays, more and more hybrid approaches combining metaheuristics and constraint programming are used to solve optimization problems. Indeed, metaheuristics and constraint programming are complementary search and modeling approaches, which may be combined naturally to solve optimization problems in a more efficient manner [72]. One of the main advantages of using constraint programming is its flexibility. Models are based on a declarative programming paradigm. Hence, the addition/deletion of new constraints in the model is straightforward.

[8] The term complete is always used in the CP community.

1.4.1 Constraint Programming

Optimization problems in constraint programming are modeled by means of a set of variables linked by a set of constraints. The variables take their values on a finite domain of integers. The constraints may have mathematical or symbolic forms. *Global constraints* refer to a set of variables of the problem. An example of such global constraints is `all_different`$(x_1, x_2, ..., x_n)$ which specifies that all the variables $x_1, x_2, ..., x_n$ must be different.

Solving a feasibility problem in CP is based on interleaving the *propagation* and the *search* processes in order to find a feasible solution for the problem. Minimizing an objective function may be reduced to solve a given number of feasibility problems.

A propagation algorithm is associated to each (or a set of) constraint(s). It consists in filtering (or reducing) from variable domains the values that cannot lead to feasible solutions. The propagation algorithm is terminated once no more values can be eliminated from the variable domains.

Once the propagation phase is finished, there may remain some inconsistent values in the variable domains. Therefore, a *search* algorithm is launched. The search algorithm is based on a tree search procedure where a branching step is applied by partitioning the current problem into subproblems. Branching may be done by instantiating a given variable to a feasible value of its domain or adding a new constraint.

The questions arising in designing a search algorithm in CP are more or less similar to those of branch and bound algorithms:

- **Branch ordering:** how the problem to solve (node of the tree) is splitted into subproblems when the propagation algorithm is inconclusive? On which variable the branching is applied next?
- **Variable selection:** in which subproblem (child node) the search continue next? What value should be first assigned to the branching variable?

Example 1.18. **A CP model for Sudoku:** nowadays, the Sudoku logic game is very popular. The principle of the game is to fill a 9×9 grid so that each row and each column contains the numbers from 1 to 9. Moreover, each of the nine 3×3 boxes contains the numbers from 1 to 9. A partially completed grid is provided as an input for each game setting. A CP model using the Gecode solver may be the following:

The `must_be_distinct` constraint has been used to model the three constraints of the problem (row, column, 3*3 boxes). The rest of the model represents the input setting of the game. It assigns the predefined values to squares of the grid (otherwise by default it is 0). Figure 1.28 illustrates a solution for a given game input.

Algorithm 1. CP model for Sudoku

```
class Sudoku < Gecode::Model
def initialize(predefined_values)
# Create the squares representing the integer variables
@squares = int_var_matrix(9, 9, 1..9)
# Distinctness constraint
9.times do |i|
# All rows must contain distinct numbers
@squares.row(i).must_be.distinct
# All columns must contain distinct numbers
@squares.column(i).must_be.distinct
# All 3x3 boxes must contain distinct numbers
@squares.minor((i % 3) * 3, 3, (i / 3) * 3, 3).must_be.distinct
end
# Place the constraints from the predefined squares on them
predefined_values.row_size.times do |i|
predefined_values.column_size.times do |j|
unless predefined_values[i,j].zero?
@squares[i,j].must == predefined_values[i,j]
end
end
```

	2	6				8	1	
3			7		8			6
4				5				7
	5		1		7		9	
		3	9		5	1		
	4		3		2		5	
1				3				2
5			2		4			9
	3	8				4	6	

Input game

7	2	6	4	9	3	8	1	5
3	1	5	7	2	8	9	4	6
4	8	9	6	5	1	2	3	7
8	5	2	1	4	7	6	9	3
6	7	3	9	8	5	1	2	4
9	4	1	3	6	2	7	5	8
1	9	4	8	3	6	5	7	2
5	6	7	2	1	4	3	8	9
2	3	8	5	7	9	4	6	1

Solution of the game

Fig. 1.28 Illustration of the Sudoku game

1.4.2 Classical Hybrid Approaches

Many combination schemes show that the hybridization of metaheuristics and CP is fruitful for some optimization problems. The following sections illustrate, for each class of hybrids belonging to the presented taxonomy, some hybridization schemes between constraint programming algorithms and metaheuristics. Some illustrative examples may also be found in [72].

1.4.2.1 Low-Level Relay Hybrids (LRH)

As within mathematical programming approaches, constraint programming may be used to explore large neighborhoods in S-metaheuristics (full or partial). Indeed, when the propagation tends to reduce the search space, CP is an efficient approach in modeling the expression of neighborhoods and exploring very large neighborhoods with side constraints [154]. Two different types of exploration may be applied:

- **Neighborhoods with expensive testing of feasibility:** neighborhoods around the current solution are defined by adding side constraints to the original problem. Checking the feasibility for all side constraints by CP may be efficient. Indeed, the feasibility test of solutions may be an expensive task. The propagation algorithms of CP reduce the size of neighborhoods.
- **Large neighborhoods:** optimizing the exploration of the neighborhood with inlined constraint checks. For instance, the problem of searching very large neighborhoods is tackled with a constraint programming solver in the resolution of vehicle routing problems [137]. A CP model has been proposed for the neighborhood, where every feasible solution represents a neighbor. A given subset of decision variables may also be fixed [8]. A CP search has been carried out over the uninstantiated variables to solve a scheduling problem. A similar approach has been proposed in [154] for a vehicle routing problem, and in [30] for a job-shop scheduling problem.

1.4.2.2 Low-Level Teamwork Hybrids (LTH)

In this class of LTH hybrids between metaheuristics and CP, two main categories may be distinguished: exact search hybrid algorithms in which a metaheuristic is embedded into constraint programming, and heuristic search algorithms in which constraint programming is embedded into a P-metaheuristic.

Embedding metaheuristics into constraint programming: metaheuristics may be used to improve the search algorithm in CP. The following hybrid approaches may be applied to converge more quickly to the optimal solution or approximating "good" solutions:

- **Node improvement:** metaheuristics may be applied to partial solutions of the tree to improve or repair the nodes of the search tree. A greedy approach may also explore a set of paths from a node of the search tree. Then, CP search continue from the improved solutions [31].
- **Discrepancy-based search algorithms:** this approach generates near-greedy paths in a search tree. This approach has been used in limited discrepancy search [87] and dynamic backtracking [78]. Lookahead evaluation of greedy algorithms may also be used over the nodes of the search tree [31].
- **Branch ordering:** metaheuristics may be applied to answer the following question: which child node to investigate first when diving deeper into the node. Metaheuristics may be used to give a preference to a branch that is consistent

with the near-optimal solution. Indeed, the use of metaheuristics produces a better variable ordering and then will speedup the tree search. This approach has been used for solving satisfiability problems [152].

Metaheuristics may also be considered to solve relaxed problem. At each node, the subproblem is relaxed by removing some constraints. The violated constraints in the obtained solution will form the basis for branching. for instance, this approach has been used to solve a scheduling problems [109] [130].

- **Variable selection:** metaheuristics are used for variable selection at each node of the tree. This strategy consists in reducing the list of candidates. A straightforward strategy has been used in [64]. Let $v_1, v_2, ..., v_n$ be the possible branches in a decreasing order of preference (lower bound $h(v_i)$). The strategy consists in selecting the v_i branches such as $h(v_i) \leq h(v_1) + \alpha(h(v_n) - h(v_1))$ where $\alpha \in [0,1]$ is a parameter. Those branches constitute the RCL list (Restricted Candidate Lists), whereas the other branches are not explored.
- **Branching restriction:** metaheuristics may be used to filter the branches of the tree-search node. This hybrid scheme has been proposed to solve a scheduling problem [32].

CP can construct partial solutions which are used to define a search space for a metaheuristic. Then, the results obtained are used in order to refine the bounds or columns to generate in a branch and cut algorithm.

Embedding constraint programming into P-metaheuristics: some search components of a P-metaheuristic induce optimization problems which are solved by CP. For instance, some recombination operators such as crossover in EAs may be optimized using CP. In addition to the recombination operators, large neighborhood search algorithms based on CP can be integrated into unary operators of P-metaheuristics such as the mutation in EAs.

Some search ingredients of constraint programming algorithms can also be used into P-metaheuristics. For instance, the use of lower bounds into a P-metaheuristic can improve the search. The partial solutions (subproblems) maintained by CP may provide to metaheuristics interesting initial solutions to metaheuristics. The evaluation of those partial solutions will guide the metaheuristics to more promising regions of the search space. The partial solution with the least cost lower bound suggests a promising region.

CP algorithms can also be used as decoders of indirect representations carried out by metaheuristics. This strategy may be applied once the metaheuristics use indirect encoding which represent incomplete solutions of the problem. This strategy is efficient when the decoding involves complex constraints to satisfy.

1.4.2.3 High-Level Relay Hybrids (HRH)

In this class of hybrids, self-contained metaheuristics are used in conjunction with CP in a pipeline manner. Metaheuristics are considered as a pre-processing or a post-processing step for CP.

Information provided by metaheuristics: in the case where the informations are provided by metaheuristics, similar information exchanges as with mathematical programming algorithms may be used: upper bounds, incomplete solutions, subproblems, etc.

Information provided by constraint programming: in the case where the information is provided by CP, the same information as with mathematical programming[9] may be considered: partial solutions (i.e. subproblems), optimal solutions for relaxed problems, etc.

For instance, heuristic search-based hybrid scheme may be applied in solving some generated subproblems by CP. A subset of variables are assigned values using a complete search approach. This approach has been used for scheduling problems [133] and routing problems [154]. This hybrid scheme has been also proposed to solve satisfiability (SAT) problems, where a depth-bounded tree search is carried out and a local search procedure is applied at nodes reaching the depth-limit [84]. CP can be also applied to an incomplete formulation of the problem. For instance, all (or a set of) feasible solutions are generated by a CP strategy. Then, a metaheuristic will be applied to improve feasible solutions represented by the leaves of the CP tree.

1.4.2.4 High-Level Teamwork Hybrids (HTH)

Few hybrid HTH strategies combining CP and metaheuristics have been investigated. This class constitutes a promising way to develop efficient solvers and optimization algorithms. The architecture of this class of hybrids may be viewed as a set of agents implementing different strategies (CP, metaheuristics, MP) in solving the target problem, and different subproblems and relaxed problems. Those agents

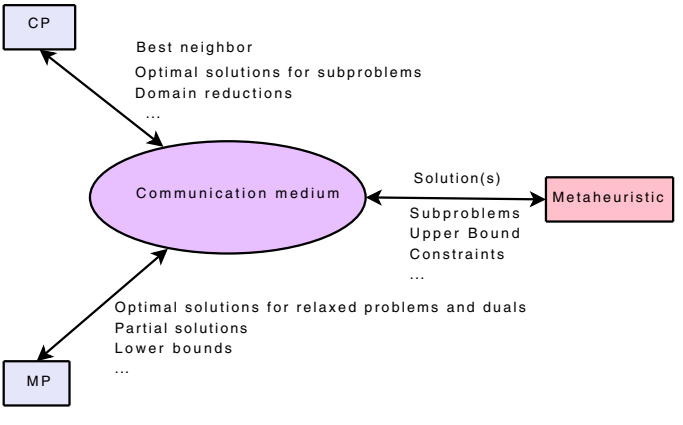

Fig. 1.29 HTH cooperation between metaheuristics, MP and CP strategies

[9] However, the duals cannot be considered.

will exchange information on the search. For an exact approach, the objective is to speedup the search in obtaining an optimal solution (efficiency). For a heuristic strategy, the objective is also to improve the quality of the obtained solutions (effectiveness). The information exchanged may include: solution(s), subproblems, relaxed optimal solutions, upper bounds , lower bounds , optimal solutions for subproblems, partial solutions, etc (Fig. 1.29).

1.5 Hybrid Metaheuristics with Machine Learning and Data Mining

Combining metaheuristics with data mining and machine learning techniques represents another way to improve the efficiency and effectiveness of the optimization algorithms based on metaheuristics.

1.5.1 Data Mining Techniques

Data mining (DM), also known as knowledge discovery in databases (KDD), is the process of automatically exploring large volumes of data (e.g. instances described according to several attributes), to extract interesting knowledge (patterns). In order to achieve this goal, data mining uses computational techniques from statistics, machine learning and pattern recognition .

Various data mining tasks can be used depending on the desired outcome of the model. Usually a distinction is made between supervised and unsupervised learning. Classical tasks of supervised learning are (Fig. 1.30):

- **Classification:** examining the attributes of a given instance to assign it to a predefined category or class.
- **Classification rule learners:** discovering a set of rules from the data which forms an accurate classifier.

The most common tasks of unsupervised learning are:

- **Clustering:** partitioning the input data set into subsets (clusters), so that data in each subset share common aspects. The partitioning is often indicated by a similarity measure implemented by a distance.
- **Association rule learners:** discovering elements that occur in common within a given data set.

The feature selection task objective consists in reducing the number of attributes (i.e. dimensionality of the data set). Feature selection is often considered as a necessary preprocessing step to analyze data characterized by a large number of attributes. It allows to improve the accuracy of the extracted models. Two models of feature selection exist depending on whether the selection is coupled with a learning scheme or not. The first one, the *filter model* , which carries out the feature subset selection and the learning (e.g. classification, clustering) in two separate phases, uses a

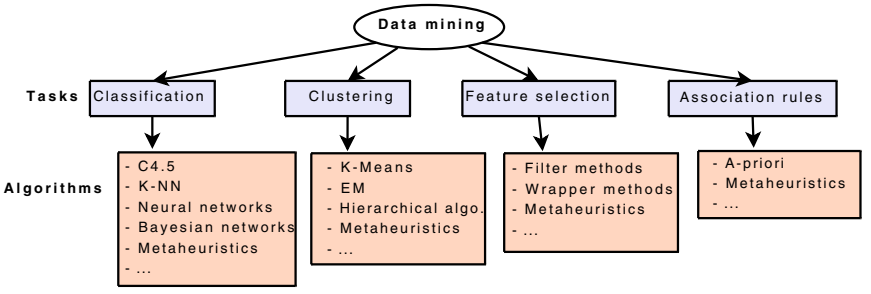

Fig. 1.30 Some data mining tasks and associated algorithms

measure that is simple and fast to compute. The second one, the *wrapper method*, which carries out the feature subset selection and learning in the same process, engages a learning algorithm to measure the accuracy of the extracted model. From the effectiveness point of view, wrapper methods are clearly advantageous, since the features are selected by optimizing the discriminate power of the finally used learning algorithm. However, their drawback is a more important computational cost.

Metaheuristics have been largely used to solve data mining tasks with a great success. However, using data mining techniques to improve the efficiency and effectiveness of metaheuristics, which is our concern in this chapter, is less studied. This hybridization scheme can be viewed as knowledge extraction and integration into metaheuristics. This knowledge may take different forms. Figure 1.31 describes some ways to integrate knowledge into metaheuristics.

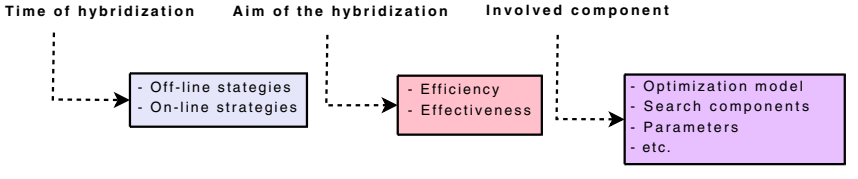

Fig. 1.31 Some ways integrating knowledge into metaheuristics

Three criteria will be used to refine our classification [104]:

- **Time of extracting the knowledge:** two kinds of hybridizations can be distinguished depending on the time of extracting the used knowledge. Hybridizations which extract the knowledge before the search starts are called *off-line knowledge* strategies and combinations where the knowledge is extracted dynamically during the search are described as *on-line knowledge* strategies.
- **Aim of the hybridization:** either the combination allows to improve the efficiency of the algorithm by reducing the search time, or the combination is used to improve the effectiveness of the algorithm leading to better quality of solutions. The efficiency may be carried out by approximating the objective function

or reducing the size of the search space. The effectiveness may be improved by incorporating some knowledge into the search components or by updating the parameters of the metaheuristics in an adaptive way. Of course, a given hybridization may improve both criteria: efficiency and effectiveness.

- **Involved component:** a metaheuristic is composed of different search components. Hybridization can occur in any search component such as encoding of solutions, initialization of solutions, search variation operators (e.g. mutation, crossover, neighborhood), etc. It may also be used to fix the parameters of the algorithm or defining the optimization problem to solve (e.g. objective function).

1.5.2 Main Schemes of Hybridization

In the following sections, some hybridization schemes between metaheuristics and data mining techniques are presented according to each class of the general taxonomy.

1.5.2.1 Low-Level Relay Hybrid (LRH)

Traditional S-metaheuristics, greedy or multi-start strategies (e.g. GRASP algorithm) do not use any information on the search of previous iterations to initialize the next search even if the tabu search algorithm uses the concept of memory to guide the search. Hence, some knowledge may be introduced in those families of metaheuristics.

Optimization model: the extracted knowledge may be used to transform the target optimization problem. For instance, in the ART (Adaptive Reasoning Technique) on-line approach, the search memory is used to learn the behavior of a greedy algorithm [136]. Some constraints are added to the problem. Those constraints are generated from the non interesting visited solutions according to the values associated to their decision variables. Similar to the tabu list strategy, those constraints are dropped after a given number of iterations.
Parameters setting: another LRH hybrid approach provides a dynamic and adaptive setting of the parameters of a S-metaheuristic. Indeed, knowledge extracted during the search may serve to change dynamically at run time the values of some parameters such as the size of the tabu list in tabu search, the temperature in simulated annealing.

This dynamic setting may also concern any search component of a S-metaheuristic such as the neighborhood and the stopping criteria.

1.5.2.2 Low-Level Teamwork Hybrids (LTH)

This hybrid scheme is very popular in P-metaheuristics.

Search components: a straightforward LTH hybrid approach consists in using data mining techniques in recombination operators of P-metaheuristics. In this class of hybrids, the knowledge extracted during the search is incorporated into the recombination operators for the generation of new solutions (Fig. 1.32). From a set of solutions (e.g. current population, elite solutions), some models are extracted which may be represented by classification rules, association rules, decision trees, etc. Those models (patterns) will participate in the generation of new solutions to intensify or diversify the search.

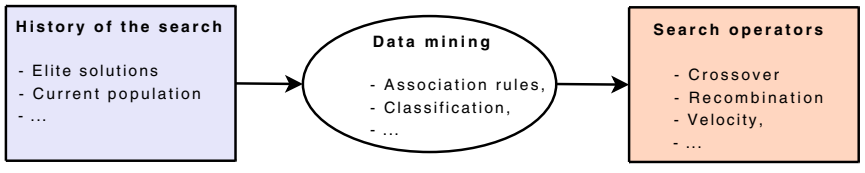

Fig. 1.32 Extracting knowledge from the history of the search and its use into search operators.

Example 1.19. **Integrating knowledge into recombination operators:** in this hybrid scheme, a set of decision rules describing the generated solutions are extracted. For instance, classification rules describing the best and worst individuals of the current population are extracted [127]. Those rules are generated using the *AQ learning algorithm*, a general decision rules learning algorithm (Fig. 1.33). The extracted rules are incorporated into the crossover operator of an evolutionary algorithm to reduce the search space for the offsprings (Fig. 1.34). The obtained results indicate that those learnable evolution models allow to speedup the search and improve the quality of solutions [127].

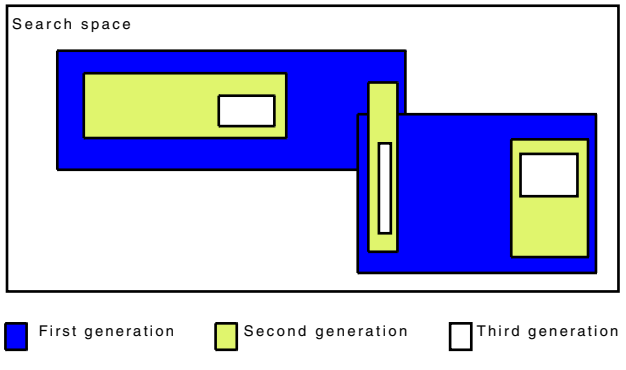

Fig. 1.33 Reduction of the search space for the offsprings using a learnable evolution model

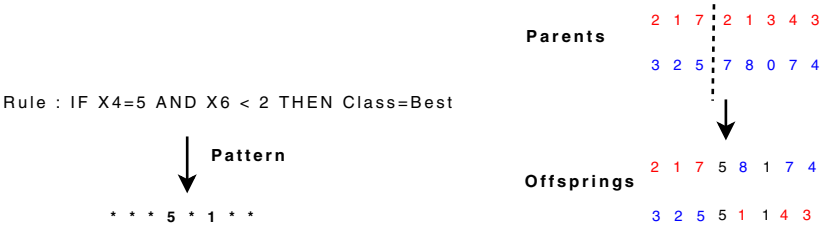

Fig. 1.34 Crossover operator using the induced rule as a pattern. For instance, the extracted pattern (...5.1..) is included into the offsprings.

EDA (Estimation of Distribution Algorithms) can also be considered as LTH hybrids using estimated probability distributions to generate new solutions. Similarly, *cultural algorithms* use high quality individuals to develop beliefs constraining the way in which individuals are transformed by genetic operators [144]. In cultural algorithms, beliefs are formed based on each entity's individual experiences. The reasoning behind this is that cultural evolution allows populations to learn and adapt at a rate faster than pure biological evolution. Importantly, the learning which takes place individually by each entity is passed on the remainder of the group, allowing learning to take place at a much faster rate.

Civilized genetic algorithms constitute another LTH hybrid approach integrating concepts from machine learning [150]. They differ from Darwinian evolution as they keep information of the population in order to avoid doing the same errors. The knowledge is dynamically updated during the successive generations. They have been applied to binary encodings in which a preprocessing step using a genetic algorithm is carried out to obtain a diverse population.

Parameter setting: a dynamic setting of the parameters of a P-metaheuristic can be carried out by a data mining task. Any parameter of a P-metaheuristic, such as the mutation and crossover probabilities in evolutionary algorithms, the pheromone update in ant colonies, and the velocity update in particle swarm optimization, can be modified dynamically during the search. Indeed, knowledge extracted during the search may serve to change dynamically at run time the values of those parameters. For instance, the initialization of the mutation rate may be adjusted adaptively by computing the progress of the last applications of the mutation operator [173] [91]. Hence, it becomes possible to determine the probabilities of application of a given operator in an adaptive manner where more efficient an operator is, more important its probability of application will be. Another approach could be to analyze in details the new individuals generated by operators (in terms of quality and diversity) using clustering algorithms. This would give valuable information that can help to set the new application probabilities.

Optimization model: many optimization problems such as engineering design problems are concerned by expensive objective functions. In this hybrid scheme, supervised classification algorithms can be used to approximate the objective function during the search. The number of solutions to evaluate according to the real

objective function can also be reduced. In this case, already evaluated solutions will represent the predefined classes. A non evaluated solution is classified, using for example the k-nearest neighbor classification algorithm. The objective function of a given solution is then approximated using the evaluated solution of the associated class.

This process may be also carried out by clustering algorithms using *fitness imitation*. A clustering algorithm is applied on a population of solutions to be evaluated. Each cluster will have a representative solution. Only the solution that represents the cluster is evaluated [142] [112] [99]. Then, the objective function of other solutions of the cluster is estimated in respect to its associated representative[10] (Fig. 1.35). Different clustering techniques may be used such as K-means and fuzzy c-means.

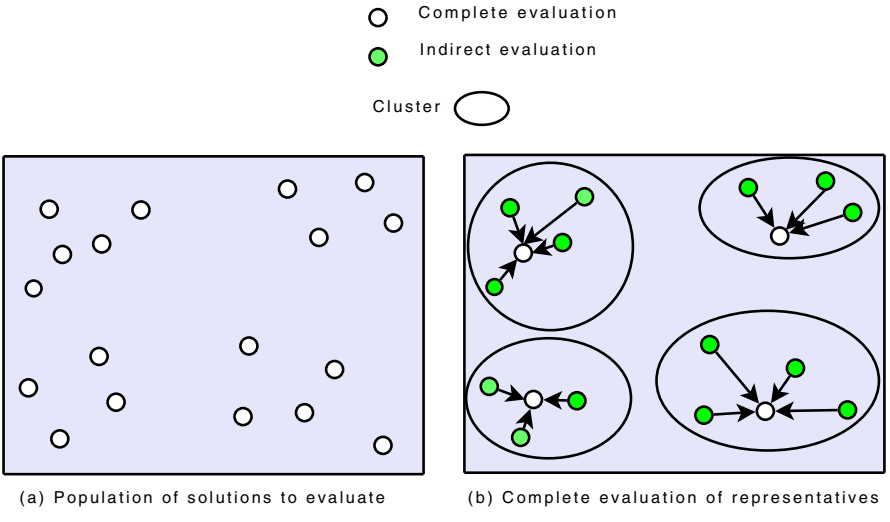

Fig. 1.35 Evaluating a solution by using the representative of its cluster (fitness imitation)

1.5.2.3 High-Level Relay Hybrid (HRH)

In this HRH hybrid approach, a priori knowledge is first extracted from the target optimization problem. Then, this knowledge is used into the metaheuristic for a more efficient search. The previously acquired knowledge may be obtained from previous experimentations, an expert, landscape analysis, etc. Many schemes may be introduced into this traditional hybrid class.

Search components: for instance, data mining algorithms may be applied for the initialization of solutions. Instead of generating the initial solutions randomly, problem knowledge can be used to generate solutions which integrate "good" patterns.

[10] This scheme is called fitness imitation or fitness inheritance.

Example 1.20. **Any-time learning algorithm in dynamic optimization:** a genetic algorithm has been initialized with a case-based reasoning in a tracker / target simulation with a periodically changing environment [141]. Case-based initialization (learning agent) allows the system to automatically bias the search of the GA toward relevant areas of the search space in a changing environment (dynamic optimization problem). This scheme may be seen as a general approach to continuous learning in a changing environment. The learning agent continuously tests search strategies using different initial solutions. This process allows the update of the knowledge base on the basis of the obtained results. This knowledge base generated by a simulation model will be used by any search agent.

Parameter setting: the same hybrid scheme may be used within the initialization of the parameters of any metaheuristic. A difficult part in designing metaheuristics deals with the setting of their parameters. Indeed, many parameters compose metaheuristics such as the probability of application of a given operator, the tabu list, the size of the population or the number of iterations? An empirical approach consists in both running several times the metaheuristic with different parameters values and trying to select the best values. If the number of trials or the number of parameters is important, determining the best set of parameters may require some statistical analysis. This may be seen as a data mining help (Fig. 1.36).

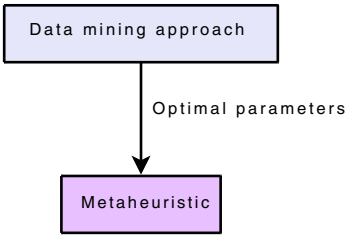

Fig. 1.36 Setting the parameters of a metaheuristic using a data mining approach

Optimization model: data mining techniques can also be used in decomposing the optimization problem handled by a metaheuristic. For instance, in optimization problems dealing with Euclidean distances, such as vehicle routing and the P-median optimization problems, clustering algorithms may be used to decompose the input space into subspaces. Metaheuristics are then used to solve those subproblems associated to the subspaces. Finally, a global solution is built using partial final solutions.

Example 1.21. **Clustering routing problems:** some efficient techniques in solving routing problems (e.g. TSP, VRP) decompose the operational space into subspaces using clustering algorithms such as the K-means or the EM (Expectation Maximization) algorithm (Fig. 1.37). Indeed, a metaheuristic is then used to solve the different subproblems. This approach is interesting for very large problems instances.

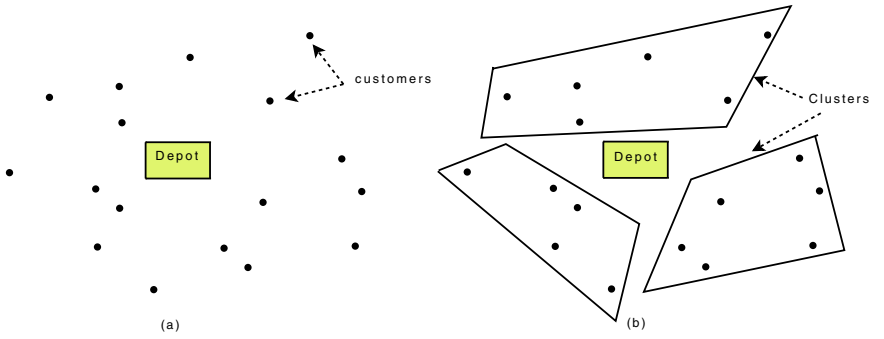

Fig. 1.37 Decomposing an optimization problem using clustering algorithms. (a) Instance of the VRP problem. (b) Clustering the customers and then applying a TSP metaheuristic to the subproblems.

A popular off-line hybrid scheme for expensive objective function consists in approximating the objective function of the problem. Indeed, in many complex real-life applications, the objective function is very expensive to compute. The main objective of this hybrid approach is to improve the efficiency of the search. The approximation can be used either for expensive objective functions or multi-modal functions. A comprehensive survey on objective function approximations may be found in [98]. Data mining approaches are used to build approximate models of the objective function. In this context, previously evaluated solutions are learned by a data mining algorithm to approximate the objective function of other individuals (Fig. 1.38). Many learning algorithms may be used such as neural networks (e.g. multi-layer perceptrons, radial-basis-function networks). The main issue here is to obtain a "good" approximation in terms of maximizing the quality and minimizing the computing time. Many questions arise in the design of this hybrid scheme such as: which proportion of the visited solutions are evaluated using the approximation, and at what time or in which component of the search algorithm the approximation is used.

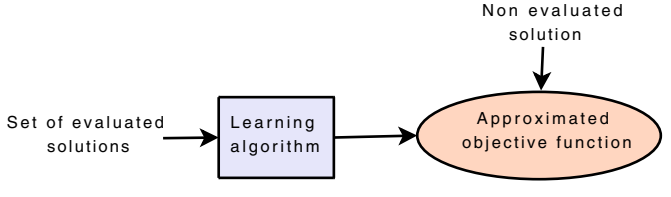

Fig. 1.38 Data mining approach to approximate the objective function

1.5.2.4 High-Level Teamwork Hybrid (HTH)

A HTH approach is a hybrid scheme in which a dynamically acquired knowledge is extracted in parallel during the search in cooperation with a metaheuristic (Fig. 1.39). Any on-line learning algorithms can be used to extract knowledge from informations provided by metaheuristics such as elite solutions, diversified set of good solutions, frequency memory, recency memory, etc. From this input, the data mining agent extracts useful information to be used by metaheuristics to improve the search. Any statistical indicator for landscape analysis of a problem may also be used.

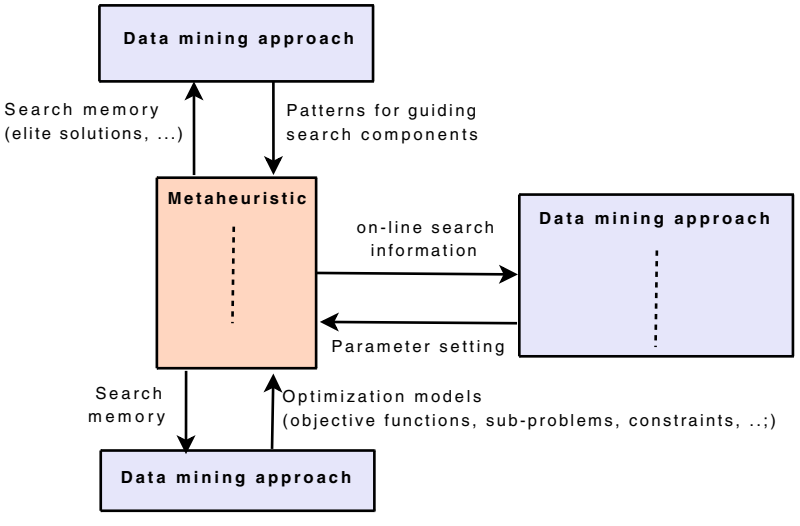

Fig. 1.39 On-line knowledge extraction and its use by a metaheuristic

Example 1.22. **Data mining in population management of a P-metaheuristic:** using the same scheme of cooperation, data mining approaches can be used to manage the population of a P-metaheuristic. Managing a population deals with the intensification and the diversification tasks of a metaheuristic. Diversification may be carried out by injecting new individuals into the population during the search. In order to lead the search to promising search spaces it could be also interesting to regularly introduce individuals that are built based on information of the past encountered high-quality solutions.

Such an approach has been proposed in the CIGAR (Case Injected Genetic AlgoRithm) algorithm [119]. The aim of CIGAR is to provide periodically to the genetic algorithm solutions that suit to similar instances / problems. Hence, a classification task is carried out to find similar instances in a case base. CIGAR has been successfully applied to several problems such as the job-shop scheduling and circuit modeling. For instance, a combination of a GA with the A-priori algorithm has been

used to discover interesting subroutines for the oil collecting vehicle routing problem [51]. The obtained sub-routes are inserted into the new individuals of the population. Another illustrative example is the combination of the GRASP heuristic with A-priori like algorithms to extract promising patterns from elite solutions [145].

1.6 Hybrid Metaheuristics for Multi-objective Optimization

The taxonomy for hybrid metaheuristics presented in this chapter holds in solving multi-objective optimization problems (MOPs). However, the design of hybrid metaheuristics for MOP needs an adaptation for the reason that in multi-objective optimization the main goal consists in generating an approximated set of Pareto solutions whereas in mono-objective optimization a unique "good" solution is aimed to be generated.

1.6.1 Combining Metaheuristics for MOPs

Until the 1990's, the main focus in the metaheuristic field was on the application of pure metaheuristics to MOPs. Nowadays, the use of pure multi-objective metaheuristics is more and more seldom. A skilled combination of concepts of different metaheuristics can provide a more efficient behavior and a higher flexibility when dealing with real-world and large-scale MOPs.

1.6.1.1 Low-Level Relay Hybrids (LRH)

This class of hybrids represents multi-objective hybrid metaheuristics in which a given metaheuristic is embedded into a S-metaheuristic. Few examples belong to this class since S-metaheuristics are not well adapted to approximate the whole Pareto set of a MOP into a single run.

Example 1.23. **An adaptive hybrid metaheuristic:** a multi-objective tabu search hyper-heuristic may be used to optimize the use of different S-metaheuristics [29]. This hybrid approach, tested on timetabling and space allocation, uses a tabu list of S-metaheuristics which is updated by adding the last used S-metaheuristic and/or the worst one, in terms of performance. Hence, this hybrid approach will adapt dynamically the search according to the performance of various S-metaheuristics. More efficient multi-objective S-metaheuristics will be more frequently used during the search.

1.6.1.2 Low-Level Teamwork Hybrids (LTH)

P-metaheuristics (e.g. evolutionary algorithms, scatter search, particle swarm, ant colonies) are powerful in the approximation of the whole Pareto set while

S-metaheuristics are efficient in the intensification of the search around the obtained approximations. Indeed, S-metaheuristics need to be guided to solve MOPs.

Therefore, most efficient multi-objective P-metaheuristics have been coupled with S-metaheuristics such as local search, simulated annealing and tabu search, which are powerful optimization methods in terms of exploitation of the Pareto sets approximations. The two classes of metaheuristics have complementary strengths and weaknesses. Hence, LTH hybrids in which S-metaheuristics are embedded into P-metaheuristics have been applied successfully to many MOPs. Indeed, many state-of-the art hybrid schemes are P-metaheuristics integrating S-metaheuristics.

Example 1.24. **Multi-objective evolutionary local search algorithm:** many multi-objective hybrid metaheuristics proposed in the literature deal with hybridization between P-metaheuristics (e.g. evolutionary algorithms) and S-metaheuristics (e.g. local search). For instance, the well-known genetic local search[11] algorithms are popular in the multi-objective optimization community [167] [96] [94] [75]. The basic principle consists of incorporating a local search algorithm during an evolutionary algorithm search. The local search part could be included by replacing the mutation operator, but it can also be added after each complete generation of the evolutionary algorithm [15]. The classical structure of a multi-objective genetic local search (MOGLS) algorithm is shown in figure 1.40, which depicts the relationships between the evolutionary multi-objective (EMO) component and the local search one.

The local search algorithm can be applied in a given direction (i.e. weighted aggregation of the objectives) [94]. In order to adapt the basic local search algorithm to the multi-objective case, one may take into account the Pareto dominance relation [15]. The algorithm works with a population of non-dominated solutions PO. The hybridization process consists in generating the neighborhood of each solution of the Pareto set approximation PO. The new generated non dominated neighbors are inserted into the approximation Pareto set PO. Solutions belonging to the Pareto set PO and dominated by a new introduced solution are deleted. This process is reiterated until no neighbor of any Pareto solution is inserted into the Pareto set PO. The Pareto local search algorithm is described below:

Algorithm 2. Template of the Pareto guided local search (PLS) algorithm.

Input: an approximated Pareto set PO;
repeat
  $S' = PO$;
  Generate the neighborhood PN_x for each solution x of S' ;
  Let PO be the set of non-dominated solutions of $S' \cup PN_x$;
until PO=S' (the population has reached the local optima)
Output: Pareto set PO

[11] Called also *memetic*.

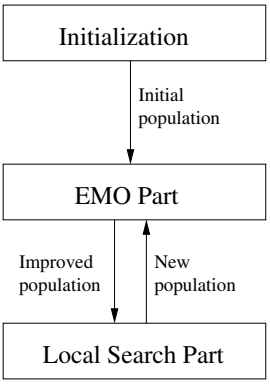

Fig. 1.40 Generic form of multi-objective genetic local search algorithms (MOGLS)

1.6.1.3 High-Level Relay Hybrids (HRH)

In HRH hybrids, self-contained multi-objective metaheuristics are executed in a sequence. A classical HRH for MOP is the application of an intensification strategy (e.g. path relinking, S-metaheuristic) on the approximation of the Pareto set obtained by a P-metaheuristic [53] [106].

Example 1.25. **Target aiming Pareto search - the TAPAS algorithm:** S-metaheuristics can be combined with any multi-objective metaheuristic to improve the quality of a Pareto approximation. First, a multi-objective metaheuristic (e.g. any P-metaheuristic) is used to generate a good approximation P of the Pareto set in terms of diversity. The design of the TAPAS algorithm was motivated by the need to improve the approximation P in terms of convergence towards the optimal Pareto set. Indeed, any S-metaheuristic algorithm can be applied to improve the quality of this approximation [105].

In the TAPAS algorithm, a S-metaheuristic l_i (e.g. tabu search[12]) is applied to each solution s_i of the initial Pareto set P. A specific mono-objective function o_i is defined for each search l_i. The defined objective function o_i must take into account the multiplicity of the S-metaheuristics invoked. Indeed, two S-metaheuristics should not examine the same region of the objective space, and the entire area that dominates the Pareto approximation P should be explored in order to converge towards the optimal Pareto front. The definition of the objective o_i is based on the partition of the objective space O according to the approximation P (see figure 1.41):

$$A_D = \{s \in O / \exists s' \in P, s' \prec s\}$$
$$A_{ND} = \{s \in O / \forall s' \in P, (s' \not\prec s) \text{ and } (s \not\prec s')\}$$
$$A_S = \{s \in O / \nexists s' \in P, s \prec s'\}$$
$$A_P = \{s \in O / \exists s_1, s_2 \in P, (s \prec s_1) \text{ and } (s \prec s_2)\}$$

[12] An efficient S-metaheuristic for the target problem should be selected.

Each solution $s_i \in P$ is associated with a part A_S^i of A_S. If l_i is able to generate a feasible solution in A_S^i, then the approximation is improved according to the convergence, without decreasing the diversity.

To guide the search, a goal g_i is given to each S-metaheuristic l_i, with g_i being the point that dominates all points of A_S^i. In cases where certain coordinates of g_i cannot be defined (e.g. the extremities of P), a lower bound for the missing coordinates should be used. For an objective f_m, the goal g_p is computed as follows:

$$f_m(g_p) = arg\ min_{\{f_m(s')/(s' \in P)\ \text{and}\ (f_m(s') < f_m(s))\}} (f_m(s') - f_m(s))$$

Then, the objective o_i is stated as follows:

$$min(\sum_{j=1}^{M} |f_j(s) - f_j(g_i)|^r)^{1/r}$$

When a S-metaheuristic l_i reaches the goal g_i or when it finds a solution that dominates g_i, it stops and produces an archive a_i which contains all the current solutions that are non-dominated. When all the S-metaheuristics l_i are terminated, a new Pareto approximation set P' is formed by the Pareto union of all a_i. Because P' might be improved by another application of S-metaheuristics, the complete process is iterated until P' does not differ from P.

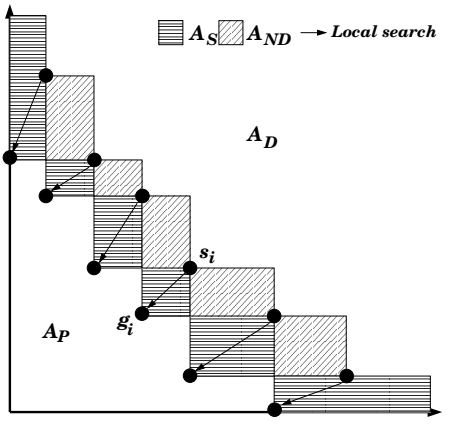

Fig. 1.41 The hybrid TAPAS algorithm for multi-objective optimization: the goal g_i of a solution s_i is defined in function of s_i neighbors in the objective space.

Example 1.26. **Filling the gap of a Pareto approximation with path-relinking:** path relinking can be combined with any multi-objective metaheuristic to intensify the search around a Pareto approximation. First, a multi-objective metaheuristic (e.g. any P-metaheuristic) is used to generate a good approximation of the Pareto set. Then, path relinking concept can be applied to connect the non-dominated solutions

of the approximated Pareto set [16] [18] [97]. The design questions which must be considered are:

- **Selection of the initial and the guiding solutions:** this design question concerns the choice of the pair of solutions to connect. For instance, a random selection from the approximated Pareto set may be applied [16]. Otherwise, some criteria must be used to choose the initial and the guiding solutions: distance between solutions (e.g. distance in the decision or the objective space), quality of the solutions (e.g. best solution according to a reference point or weighted aggregation), etc.
- **Path generation:** many paths may be generated between two solutions. One has to establish which path(s) has to be explored and selected. Among other concepts, a neighborhood operator and a distance measure in the decision space have to be defined. For instance, the shortest paths may be generated according to the selected neighborhood operator [16]. Let us consider x as the current solution and y as the guiding solution. The neighborhood N of x is generated with the following constraint: $\forall z \in N, d(z,x) < d(y,x)$. From this neighborhood, only the non-dominated solutions may be selected to be potential solutions of the future paths (see figure 1.42). The process is iterated, until a complete path from x to y is generated. Many paths may also be considered. However, generating all possible paths may be computationally expensive. Moreover, the non-dominated solutions may also be selected to participate to a Pareto local search algorithm as shown in figure 1.43 [16].

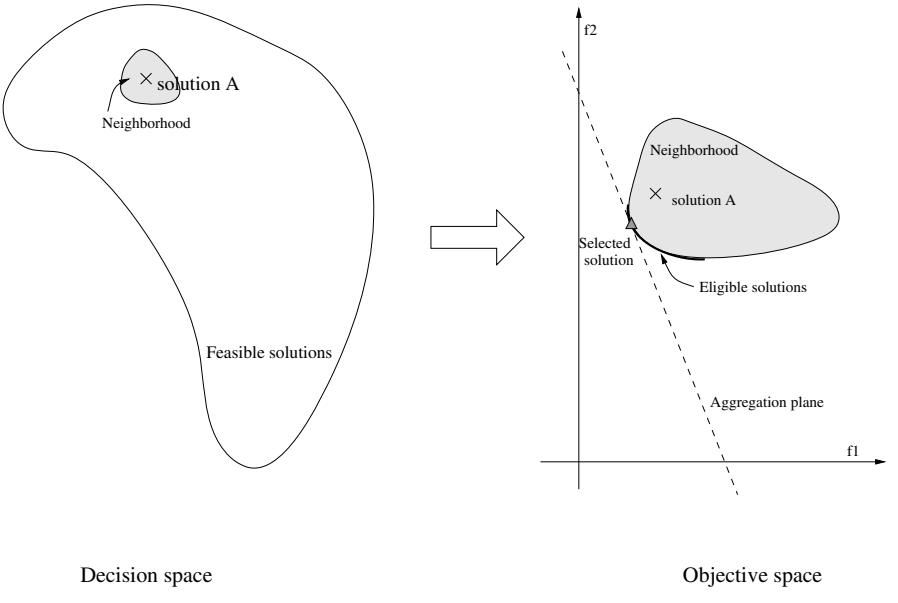

Fig. 1.42 Path Relinking algorithm filling the gap between two non-dominated solutions of an approximation Pareto set: neighborhood exploration

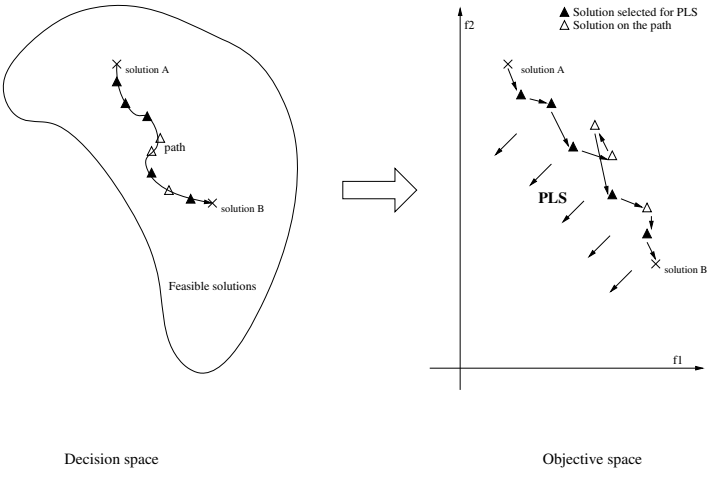

Fig. 1.43 Path relinking algorithm combined with a Pareto local search (PLS) algorithm

1.6.1.4 High-Level Teamwork Hybrid (HTH)

As previously shown in this chapter, HTH hybrids scheme involves several self-contained multi-objective metaheuristics performing a search in parallel and cooperating to find a Pareto set approximation.

Example 1.27. **Cooperative multi-objective evolutionary algorithms:** a growing interest is dedicated to design and implement parallel cooperative metaheuristics to solve multi-objective problems. The majority of designed parallel models in the literature are evolutionary algorithms [80] [101] [126] [147]. In multi-objective evolutionary algorithms, the individuals are selected from either the population, the Pareto archive or both of them. In the multi-objective island model, different strategies are possible. For instance, the newcomers replace individuals selected randomly from the local population that do not belong to the local Pareto archive. Another strategy consists in ranking and grouping the individuals of the local population into Pareto fronts using the non-dominance relation. The solutions of the worst Pareto front are thus replaced by the new arrivals. One can also make use of the technique that consists in merging the immigrant Pareto front with the local one, and the result constitutes the new local Pareto archive. The number of emigrants can be expressed as a fixed or variable number of individuals, or as a percentage of individuals from the population or the Pareto archive. The choice of the value of such parameter is crucial. Indeed, if it is low the migration process will be less efficient as the islands will have the tendency to evolve in an independent way. Conversely, if the number of emigrants is high, the EAs will likely to converge to the same solutions (premature convergence).

Although most of works on parallel multi-objective metaheuristics are related to evolutionary algorithms, there are also proposals related to alternative methods,

such us tabu search [7], simulated annealing [5] [34], ant colonies [54] and memetic algorithms [15].

1.6.2 Combining Metaheuristics with Exact Methods for MOP

Another recent popular issue is the cooperation between multi-objective metaheuristics and exact optimization algorithms. Some hybrid schemes mainly aim at providing Pareto optimal sets in shorter time, while others primarily focus on getting better Pareto set approximations. In a multi-objective context, only few studies tackle this type of approaches. The main interest is to adapt the classical mono-objective hybrids presented in sections 1.3 and 1.4 to multi-objective optimization.

Example 1.28. **Combining branch and bound with multi-objective metaheuristics:** an investigation of several cooperative approaches combining multi-objective branch and bound [177] and multi-objective metaheuristics can be considered for MOPs [13]. Let us consider the bi-objective flow-shop scheduling problem, a multi-objective metaheuristic which approximates the Pareto set of the problem [14], and a bi-objective branch and bound which has been designed to solve the bi-objective flow-shop scheduling problem [116].

Three hybrid schemes combining an exact algorithm with a multi-objective metaheuristic may be considered [13]:

- **Metaheuristic to generate an upper bound:** the first HRH hybrid exact scheme is a multi-objective exact algorithm (e.g. branch and bound) in which the Pareto set approximation is used to speedup the algorithm (Fig. 1.44). The Pareto set approximation is considered as a good upper bound approximation for the multi-objective exact algorithm. Hence, many nodes of the search tree can be pruned by the branch and bound algorithm. This is a multi-objective adaptation of a classical cooperation found in the mono-objective context (see section 1.3). The time required to solve a given problem instance is smaller if the distance between the Pareto front approximation and the Pareto optimal front is small. If the distance is null, the exact algorithm will serve to prove the optimality of the Pareto set approximation. Even if this hybrid approach reduces the search time needed to find the Pareto optimal set, it does not allow to increase considerably the size of the solved instances.
- **Exact algorithm to explore very large neighborhoods:** in this hybrid heuristic approach, the exact multi-objective algorithm is used to explore large neighborhoods of a Pareto solution. The main idea is to reduce the search space explored by the exact algorithm by pruning nodes when the solution in construction is too far from the initial Pareto solution.

  Let us consider a permutation based representation for the bi-objective flow-shop scheduling problem, and an insertion neighborhood operator. The exact algorithm is allowed to explore the neighborhood of the initial Pareto solution in which the solutions are within a distance less or equal to δ_{max} (Fig. 1.45). The

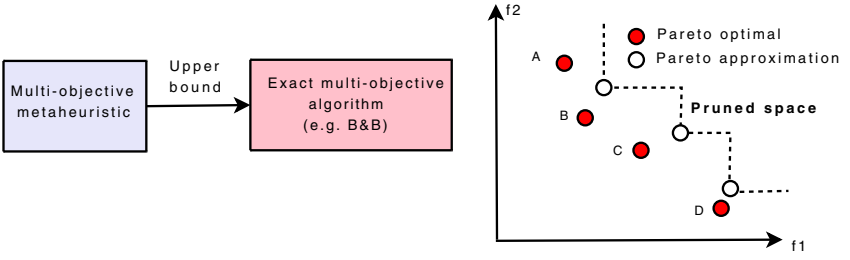

Fig. 1.44 A HRH exact hybrid scheme in which a multi-objective metaheuristic generates an upper bound Pareto set to an exact multi-objective algorithm (e.g. branch and bound).

size of the insertion-based neighborhood is $\Theta(n^2)$, where n represents the number of jobs. Hence, the size of the search space explored by the exact algorithm is exponential and may be approximated by $\Theta(n^{2\delta_{max}})$. Then, the distance δ_{max} must be limited, especially for instances with large number of jobs.

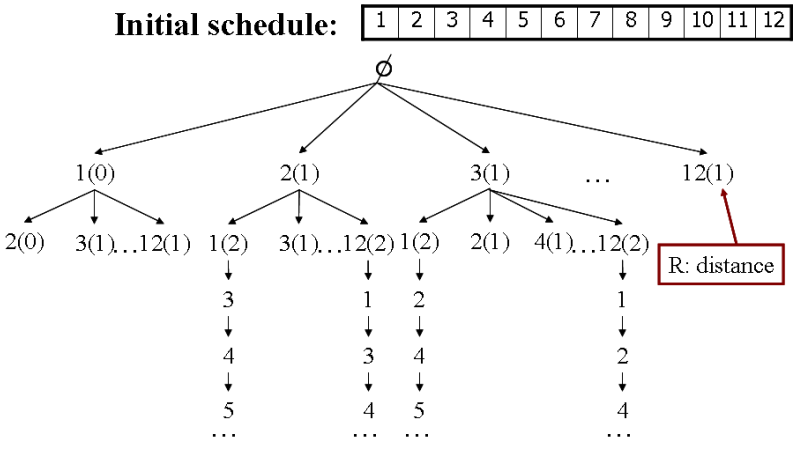

Fig. 1.45 A hybrid heuristic scheme in which an exact algorithm explores very large neighborhoods of the multi-objective metaheuristic

- **Exact algorithm to solve subproblems:** in this hybrid heuristic approach, the exact multi-objective algorithm solve subproblems which are generated by the multi-objective metaheuristic. A given region of the decision space is explored by the exact algorithm. Figure 1.46 shows an example of such hybridization. Let us consider an initial Pareto solution composed of 10 jobs $(a,b,...,i,j)$ which is obtained by the multi-objective metaheuristic. Subproblems of a given size (e.g. 4) are explored by the exact algorithm (e.g. the subproblem defined by the non-freezed jobs d,e,f,g). The first phase consists in placing the three first jobs at the beginning of the schedule. Moreover, the branch and bound algorithm places the three last jobs at the end of the schedule (a job j placed in queue is symbolized

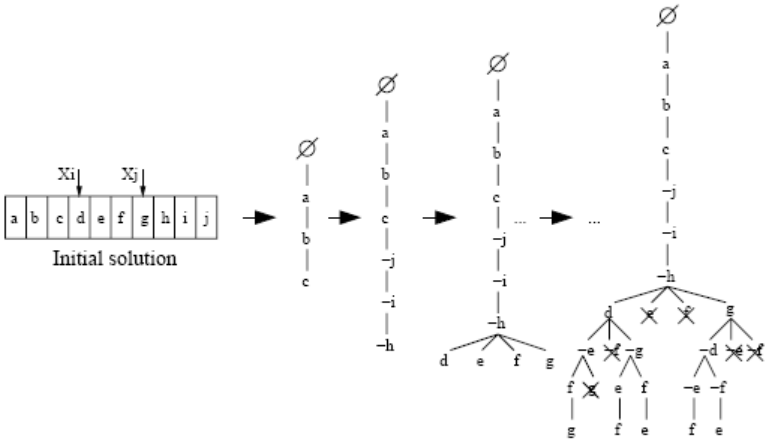

Fig. 1.46 A hybrid heurisitic scheme in which an exact algorithm solves subproblems generated by a multi-objective metaheuristic

by $-j$). Then, the branch and bound multi-objective algorithm is applied on the remaining non-freezed jobs to generate all Pareto solutions in this subspace.

The main parameters which have to be defined for an efficient hybrid scheme are:

- **Partition sizes:** the cardinality of the Pareto set approximation obtained by a multi-objective metaheuristic varies according to the target MOP and instance. For the BOFSP problem, the size of the Pareto set approximation varies between several tens and two hundred solutions. Moreover, the size of partitions must be also limited according to the efficiency of the exact method at hand. For the BOFSP, it may be fixed to 25 jobs for 10 machines instances and 12 jobs for the 20 machines instances, so each exact method can be performed in several seconds or some minutes [13].
- **Number of partitions for each solution:** enough partitions of the complete schedule have to be considered to treat each job at least once by the exact method. Moreover, it is interesting to superpose consecutive partitions to allow several moves of a same job during optimization. Then, a job which is early scheduled could be translated at the end of the schedule by successive moves. On the other side, more partitions are considered, more important the computational time is. For instance, for the BOFSP, 8 partitions for the $50-$jobs instances, 16 partitions for the $100-$jobs and 32 partitions for the $200-$jobs instances may be considered [13].

Example 1.29. **Combining branch and cut with multi-objective metaheuristics:** this example investigates the solution of a multi-objective routing problem, namely the bi-objective covering tour problem (BOCTP), by means of a hybrid HRH strategy involving a multi-objective metaheuristic and a single-objective branch-and-cut algorithm. The BOCTP aims to determine a minimal length tour for a subset of

nodes while also minimizing the greatest distance between the nodes of another set and the nearest visited node. The BOCTP can be formally described as follows (Fig. 1.48): let $G = (V \cup W, E)$ be an undirected graph, where $V \cup W$ is the vertex set, and $E = \{(v_i, v_j)/v_i, v_j, V \cup W, i < j\}$ is the edge set. Vertex v_1 is a depot, V is the set of vertices that can be visited, $T \subseteq V$ is the set of vertices that must be visited $(v_1 \in T)$, and W is the set of vertices that must be covered. A distance matrix $C = (c_{ij})$, satisfying triangle inequality, is defined for E. The BOCTP consists of defining a tour for a subset of V, which contains all the vertices from T, while at the same time optimizing the following two objectives: (i) the minimization of the tour length and (ii) the minimization of the cover. The cover of a solution is defined as the greatest distance between a node $w \in W$, and the nearest visited node $v \in V$.

The BOCTP problem has been extended from the mono-objective covering tour problem (CTP). The CTP problems consists in determining a minimum length tour for a subset of V that contains all the vertices from T, and which covers every vertex w from W that is covered by the tour (i.e. w lies within a distance c from a vertex of the tour, where c is a user defined parameter). A feasible solution for a small instance is provided in figure 1.48. One generic application of the CTP involves designing a tour in a network whose vertices represent points that can be visited, and from which the places that are not on the tour can be easily reached. In the bi-objective covering tour problem BOCTP, the constraint on the cover has been replaced by an objective in which the covering distance is minimized [107].

Let us consider a multi-objective metaheuristic to solve the BOCTP problem which approximates the Pareto set [107], and a branch and cut algorithm to solve the mono-objective CTP problem [76]. The branch and cut algorithm may be considered as a black box optimization tool whose inputs are a subset of V, the set W, and a cover, and whose output is the optimal tour for the CTP. The branch and cut algorithm first relaxes the integrality conditions on the variables and the connectivity constraints of the integer linear programming model. Integrality is then gradually restored by means of a branch and bound mechanism. Before initiating branching at any given node of the search tree, a search is conducted for violated constraints, including the initially relaxed connectivity constraints and several other families of valid constraints. Several classes of valid inequalities have been considered such as dominance constraints, covering constraints, sub-tour elimination constraints, and 2-matching inequalities [76].

In the hybrid approach, the multi-objective metaheuristic generates a Pareto set approximation, which is used to build subproblems; these subproblems are then solved using the branch and cut algorithm (Fig. 1.48). Subproblem construction is a key point of the cooperative design, given that prohibitive computational times result if the subsets of V are too large. By limiting their size and giving the branch and cut algorithm access to the information extracted from the Pareto set approximation, the method makes solving the subproblems relatively easy for the branch-and-cut algorithm. Two procedures for building the subproblems can be considered [107]:

- **One objective improvement by an exact algorithm:** the main purpose of the first construction procedure is to improve the solutions found by the multi-

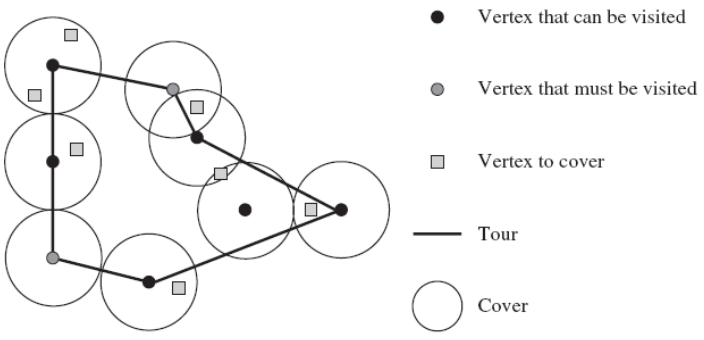

Fig. 1.47 The covering tour problem: an example of a solution

objective metaheuristic in terms of the tour length objective without modifying the cover value. It accomplishes this goal by investigating the possibility that some elements of the set of visited vertices $\tilde{V}$ can be replaced by sets of vertices $R \subseteq V \backslash \tilde{V}$ so that the cover value $\tilde{c}$ provided by the couple (v_t, v_c) remains unchanged (Fig. 1.48a). A vertex $v_k \in \tilde{V}$ can be replaced by a set R if and only if: (i) No subset of R can replace v_k; (ii) No vertex from R can provide a better cover: $\forall v_i \in R, c_{tc} \leq c_{ic}$; (iii) There must be a vertex from $\tilde{V}$ or from R that can replace v_k for every vertex of W that can be covered by v_k. Therefore, $\forall v_l \in W \backslash \{v_c\}$, such that $c_{kl} \leq \tilde{c}$, where the following condition must be true: $\exists v_n \in R \cup (\tilde{V} \backslash \{v_k\}), c_{nl} \leq \tilde{c}$.

Replacing a node of $\tilde{V}$ by a subset R tends to become easier as the cardinality of R increases. However, in practice, condition (i) limits the candidate subsets. The larger the R set, the higher the cost of the test. Certainly, if the size of the set used for the branch and cut algorithm is very large, the algorithm will require too much computational time. Therefore, in practice, the cardinality of R is limited to one or two elements.

For each solution s of the Pareto set approximation, a problem is built as follows. The set V_I of vertices that can be visited is created by the union of $\tilde{V}$ and all subsets of V with a cardinality of 1 or 2 that can replace a vertex of $\tilde{V}$. The set W of vertices that must be covered remains unchanged. Here, the parameter c is equal to the cover of s.

- **Region exploration by an exact algorithm:** in the first construction procedure it is unlikely that all the feasible covers corresponding to Pareto optimal solutions will be identified. These unidentified solutions must always be situated between two solutions of the approximation, although not always between the same two solutions. Thus, it is reasonable to assume that new Pareto optimal solutions may be discovered by focusing searches in the area of the objective space between two neighboring solutions. The second procedure aims to build sets of vertices in order to identify potentially Pareto optimal solutions whose cover values were not found by the multi-objective metaheuristic. Let A and B be two neighboring solutions in the approximation sets found by the evolutionary algorithm (i.e. there

are no other solutions between A and B). A (resp. B) is a solution with a cover c_A (resp. c_B) which visits the vertices of the set V_A (resp. V_B). Assuming that $c_A < c_B$, the branch and cut algorithm can be executed on a set V_{II}, built according to both V_A and V_B, with the first cover $\tilde{c}$ which is strictly smaller than c_B as a parameter (Fig. 1.48b). If $\tilde{c}$ is equal to c_A, there is no need to execute the branch and cut algorithm.

It appears that neighboring solutions in the Pareto set have a large number of vertices in common. Thus, V_{II} contains V_A and V_B. This inclusion insures that the branch and cut algorithm will at least be able to find the solution A, or a solution with the same cover but a better tour length in cases for which the tour on V_A is not optimal. The following process is used to complete V_{II}: for every feasible cover c, so that $c_A < c < c_B$, vertices are added to V_{II} in order to obtain a subset of V_{II} with c as a cover. The algorithm below provides the procedure for constructing the set V_{II}.

Algorithm 3. Construction of the set V_{II}.

$V_{II} = V_A \cup V_B$;
for all c so that $c_A < c < c_B$ **do**
  for $v_l \in W$ **do**
    $V_{II} = \cup V_{II}\{v_k \in V \backslash V_{II} / c_{kl} \leq c\}$
  end for
end for

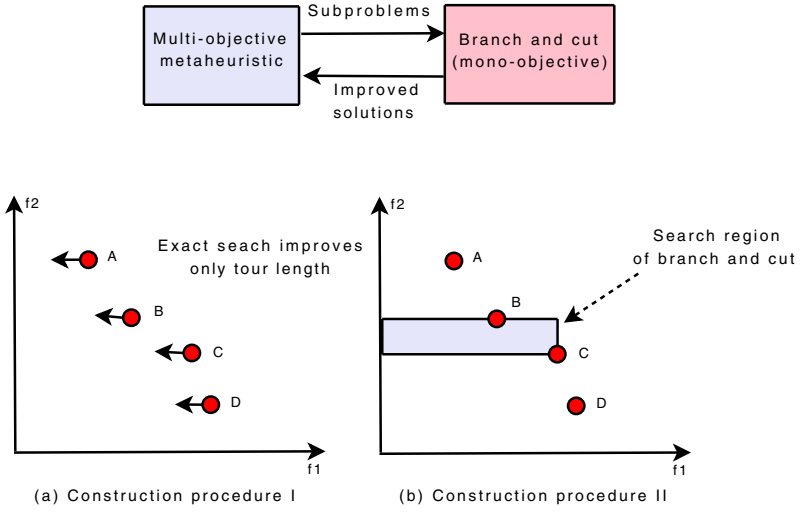

Fig. 1.48 Combining a mono-objective branch and cut algorithm and a multi-objective metaheuristic to solve the bi-objective covering tour problem

1.6.3 Combining Metaheuristics with Data Mining for MOP

Most of the classical combinations of metaheuristics with machine learning and data mining techniques (e.g. feature selection, classification, clustering, association rules) which have been applied to mono-objective optimization (see section 1.5) can be generalized to multi-objective optimization:

- Search operators (e.g. recombination operators in P-metaheuristics, neighborhoods in S-metaheuristics).
- Optimization models (e.g. approximation of the objectives functions, generation of sub-problems, new constraints).
- Parameter setting of the metaheuristics.

Example 1.30. **Search operators:** integrating knowledge into search operators is the most popular scheme in this class of hybrids. For instance, in a P-metaheuristic (e.g. evolutionary algorithm), a set of decision rules describing the best and worst individuals of the current population may be extracted. The extracted rules are incorporated into the crossover operator of an evolutionary algorithm to generate solutions sharing the characteristics of non-dominated solutions and avoiding those of dominated solutions.

This principle can be applied to multi-objective optimization in the following way [103]: a set of rules that describes why some individuals dominate others (positive rules) and why some individuals are dominated by others (negative rules[13]) are extracted using the *C4.5 classifier*. Offsprings that match the positive rules and do not match the negative rules are generated. The obtained results indicate that those learnable evolution models allow to speedup the search and improve the quality of solutions.

Parameter setting: in a multi-objective metaheuristic, the efficiency of an operator may change during the execution of the algorithm: an operator may offer a better convergence at the beginning of the metaheuristic, but this convergence may be improved later with another operator. The success of an operator may also depend on the instance of the problem. This motivates the use of adaptive operator probabilities to automate the selection of efficient operators. The adaptation can be done by exploiting information gained, either implicitly or explicitly, regarding the current ability of each operator to produce solutions of better quality [173]. Other methods adjust operator probabilities based on other criteria, such as the diversity of the population [45]. A classification of adaptation on the basis of the used mechanisms, and the level at which adaptation operates may be found in [89].

Example 1.31. **Adaptive mutation in multi-objective evolutionary algorithms:** let us consider a multi-objective evolutionary algorithm in which the choice of the mutation operators is done dynamically during the search. The purpose is to use simultaneously several mutation operators during the EA, and to change automatically the probability selection of each operator according to its effectiveness [15].

[13] Negative knowledge.

So the algorithm always uses more often the best operators than the others. Let us remark that a similar approach could be defined with other operators (e.g. crossover, neighborhoods, hybrid strategies).

Initially, the same probability is assigned to each mutation operator: $Mu_1, \ldots, Mu_k$. Those probabilities are equal to the same ratio $P_{Mu_i} = 1/(k * P_{Mu})$, where k is the number of mutation operators, and P_{Mu} is the global mutation probability. At each iteration, the probabilities associated to the mutation operators are updated according to their average progress. To compute the progress of the operators, each mutation Mu_i applied to the individual I is associated with a progress value:

$$\Pi(I_{Mu_i}) = \begin{cases} 1 \text{ if } I \text{ is dominated by } I_{Mu_i} \\ 0 \text{ if } I \text{ dominates } I_{Mu_i} \\ \frac{1}{2} \text{ otherwise (non comparable solutions)} \end{cases}$$

where I_{Mu_i} is the solution after mutation (Fig. 1.49).

At the end of each generation of the EA, an average progress $Progress(Mu_i)$ is assigned to each operator Mu_i. Its value is the average progress of $\Pi(I_{Mu_i})$ computed with each solution modified by the mutation Mu_i:

$$Progress(Mu_i) = \frac{\sum \Pi(I_{Mu_i})}{\|Mu_i\|}$$

where $\|Mu_i\|$ is the number of applications of the mutation Mu_i on the population. The new selection probabilities are computed proportionally to these values:

$$P_{Mu_i} = \frac{Progress(Mu_i)}{\sum_{j=1}^{k} Progress(Mu_j)} \times (1 - k \times \delta) + \delta$$

where δ is the minimal selection probability value of the operators.

This approach of progress computation compares two solutions with their dominance relation. However, a comparison only between I and I_{Mu_i} is not sufficient. Firstly, if the two individuals I and I_{Mu_i} are non comparable, the quality of the mutation cannot be evaluated. For instance, in figure 1.50, the progress Π of the two mutation operators applied on the solution $\triangle$ is the same $(1/2)$. However, the observation of the whole Pareto front shows that the second mutation operator performs better since the generated solution by the second mutation operator is Pareto optimal whereas the solution generated by the first is not.

Secondly, if the generated individual dominates the initial individual, the progress realized cannot be measured with precision. For instance, in figure 1.51, the progress Π of the two mutation operators applied on the solution $\triangle$ is the same (1). However, the observation of the whole population shows that the second mutation operator performs much better than the first one.

These problems can be tackled in the case of evolutionary algorithms using selection by ranking. The progress value can be replaced by:

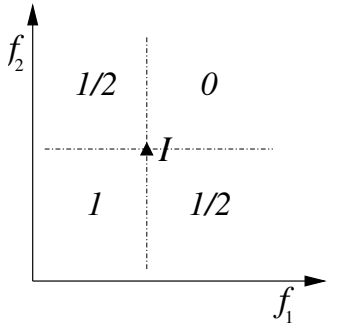

Fig. 1.49 Progress value of $\Pi(I_{Mu_i})$ for mutation operators

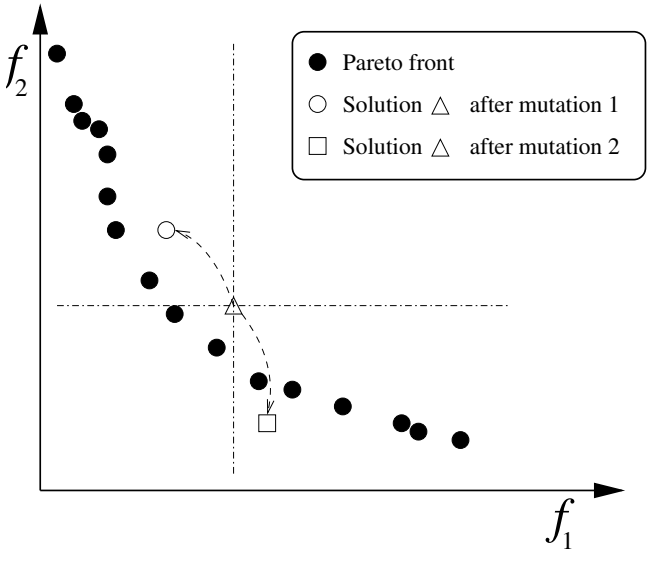

Fig. 1.50 Evaluation of the quality of the mutation operators

$$\Pi(I_{Mu_i}) = \left(\frac{R_k I}{R_k I_{Mu_i}}\right)^{\beta}$$

where $R_k I_{Mu_i}$ is the rank of the solution after mutation, $R_k I$ is the rank of the solution before mutation, and β is how much the progress made by mutation operators is encouraged (e.g. $\beta = 2$).

The evaluation of the progress of the mutation operators can be still improved by supporting the progresses realized on good solutions. In fact, these progresses are generally more interesting for the front progression than progresses made on bad solutions (Fig. 1.52). So an elitist factor $E_f I_{Mu_i}$ has been introduced into the last progress indicator to favor progresses made on good solutions:

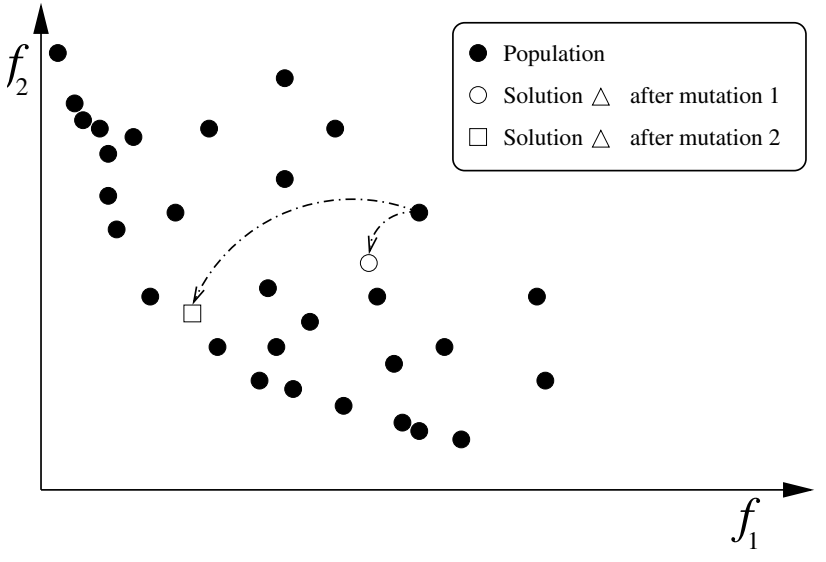

Fig. 1.51 Computing the progress realized by different mutation operators

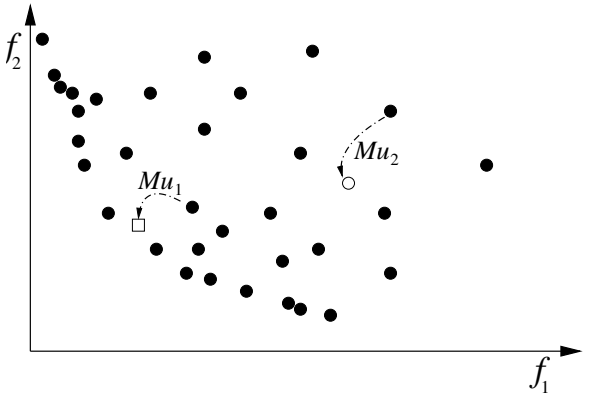

Fig. 1.52 Progress realized by mutation operators on good quality solutions

$$\Pi(I_{Mu_i}) = E_f I_{Mu_i} \times \left(\frac{R_k I_{Mu_i}}{R_k I}\right)^{\beta}$$

with $E_f I_{Mu_i} = (R_k I_{Mu_i})^{-1}$. Then, the average progress of a mutation Mu_i is defined as follows:

$$Progress(Mu_i) = \frac{\sum \Pi(I_{Mu_i})}{\sum E_f I_{Mu_i}}$$

Some hybrid schemes are specific to multi-objective metaheuristics such as introducing data mining tasks in the search component dealing with elitism.

Example 1.32. **Clustering archives in multi-objective metaheuristics:** a classical approach using data mining approaches in the population management of multi-objective metaheuristics is the application of clustering algorithms on the archive. The objective is to produce a set of well diversified representatives Pareto solutions in a bounded archive. An archive is often used to store Pareto solutions and the clustering is then performed to avoid a bias towards a certain region of the search space and to reduce the number of Pareto solutions. Such a bias would lead to an unbalanced distribution of the Pareto solutions. For instance, a hierarchical clustering can be applied using the average linkage method [183].

1.7 Conclusions and Perspectives

The efficient solving of complex problems must involve ideas from different paradigms: metaheuristics, mathematical programming, constraint programming, machine learning, graph theory, parallel and distributed computing, and so on. Pure metaheuristics are not generally well suited to search in high-dimensional and complex landscapes. Hybrid metaheuristics represent actually the most efficient algorithms for many classical and real-life difficult problems. This is proven by the huge number of efficient hybrid metaheuristics proposed to solve a large variety of problems.

Nowadays, combining metaheuristics becomes a common strategy to solve optimization problems. Hybrid algorithms will constitute competitive candidates for solving difficult optimization problems in the future years. As we have developed a unified view of metaheuristics which is based on their key search components, one can say that designing a mono-objective or multi-objective metaheuristic can be reduced to select the most suited search components and combining them. This design approach is naturally a hybrid one, and it is not under the control of a single paradigm of metaheuristics[14].

A unified taxonomy, based on a hierarchical (low level versus high level, relay versus teamwork) and flat classification (homogeneous/heterogeneous, global/partial, general/specialist), has been developed to describe in terms of design and implementation the different hybridization schemes of metaheuristics with:

- **Metaheuristics:** combining P-metaheuristics with S-metaheuristics has provided very powerful search algorithms. Pure P-metaheuristics such as evolutionary algorithms, scatter search, and ant colonies are generally not well suited to fine-tuned search in highly combinatorial spaces. P-metaheuristic are more efficient in terms of diversification (i.e. exploration) in the search space. Hence, they need to be combined with more intensification-based (i.e. exploitation-based) search algorithms which are generally based on S-metaheuristics (e.g. local search, tabu search).

[14] Using this design approach, it is worthwhile to speak about hybrid metaheuristics as any metaheuristic will be a hybrid one!.

- **Mathematical programming:** in the last decade, there has been an important advance in designing efficient exact methods in the operations research community (e.g. integer programming). There are many opportunities to design hybrid approaches combining metaheuristics and exact methods. Indeed, the two approaches have complementary advantages and disadvantages (e.g. efficiency and effectiveness).
- **Constraint programming:** over the last years, interest on combining metaheuristics and constraint programming has risen considerably. The availability of high-level modeling languages and software solvers for constraint programming will lead to more hybrid approaches which capture the most desirable features of each paradigm.
- **Data mining:** nowadays, using metaheuristics to solve data mining and machine learning problems becomes common. But the challenge is the incorporation of machine learning and data mining techniques into metaheuristics. The major interest in using machine learning and data mining techniques is to extract useful knowledge from the history of the search in order to improve the efficiency and the effectiveness of metaheuristics. Both positive and negative knowledge must be extracted. In fact, most of the actual works focus only on positive knowledge [128].

The main drawback of hybridization is the introduction of new parameters which define the hybrid scheme. The setting of those parameters is non trivial. A crucial question that has to be addressed in the future is an aid for the efficient design of hybrid metaheuristics, in which the automatic setting of parameters must be investigated [73] [25]. Indeed, it will be interesting to guide the user to define the suitable hybrid scheme to solve a given problem. It will be also interesting to define "adaptive" cooperation mechanisms which allows to select dynamically the optimization methods according to convergence or other criteria such as diversity. Some approaches such as the COSEARCH [164] or "hyper-heuristics" [28] have been proposed to deal with this problem. Those approaches are dedicated to choose the right heuristic for the right operation at the right time during the search. It must be noted that the those hybrid approaches operate in the heuristic space, as opposed to most implementations of meta-heuristics, which operate in the solution space. This principle is relatively new, although the concept of "optimizing heuristics" is not a recent one.

Using the software framework ParadisEO, it is natural to combine metaheuristics which have been developed under the framework to design S-metaheuristics (under ParadisEO-MO), P-metaheuristics (under ParadisEO-EO), and multi-objective metaheuristics (under ParadisEO-MOEO). Still a work to do for combining metaheuristics with exact optimization and machine learning algorithms. The coupling of software frameworks dealing with the three classes of algorithms (i.e. metaheuristics, exact and machine learning algorithms) is an important issue for the future. This enables to reduce the complexity of designing and implementing hybrid approaches and make them more and more popular.

It will be also interesting to deeply explore parallel models for hybrid methods. Parallel schemes ideally provide novel ways to design and implement hybrid

algorithms by providing parallel models of the algorithms. Hence, instead of merely parallelizing and finely tuning a sequential hybrid algorithm which has limited capabilities to be parallelized, teamwork hybrid schemes are inherently suited to parallel environments.

References

1. Abbattista, F., Abbattista, N., Caponetti, L.: An evolutionary and cooperative agent model for optimization. In: IEEE Int. Conf. on Evolutionary Computation, ICEC 1995, Perth, Australia, pp. 668–671 (December 1995)
2. Abramson, D., Logothetis, P., Postula, A., Randall, M.: Application specific computers for combinatorial optimisation. In: Australien Computer Architecture Workshop, Sydney, Australia (February 1997)
3. Abramson, D.A.: A very high speed architecture to support simulated annealing. IEEE Computer 25, 27–34 (1992)
4. Aggarwal, C.C., Orlin, J.B., Tai, R.P.: An optimized crossover for the maximum independent set. Operations Research 45, 226–234 (1997)
5. Agrafiotis, D.K.: Multiobjective optimization of combinatorial libraries. Technical report, IBM J. Res. and Dev. (2001)
6. Aiex, R.M., Binato, S., Ramakrishna, R.S.: Parallel GRASP with path relinking for job shop scheduling. Parallel Computing 29, 393–430 (2003)
7. Al-Yamani, A., Sait, S., Youssef, H.: Parallelizing tabu search on a cluster of heterogeneous workstations. Journal of Heuristics 8(3), 277–304 (2002)
8. Applegate, D., Cook, W.: A computational study of the job-shop scheduling problem. ORSA Journal on Computing 3, 149–156 (1991)
9. Apt, K.: Principles of constraint programming. Cambridge University Press (2003)
10. Augerat, P., Belenguer, J.M., Benavent, E., Corberan, A., Naddef, D.: Separating capacity constraints in the CVRP using tabu search. European Journal of Operational Research 106(2), 546–557 (1998)
11. Balas, E., Niehaus, W.: Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems. Journal of Heuristics 4(2), 107–122 (1998)
12. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H.: Branch-and-price: column generation for huge integer programs. Operations Research 46(316) (1998)
13. Basseur, M., Lemesre, J., Dhaenens, C., Talbi, E.-G.: Cooperation Between Branch and Bound and Evolutionary Approaches to Solve a Bi-Objective Flow Shop Problem. In: Ribeiro, C.C., Martins, S.L. (eds.) WEA 2004. LNCS, vol. 3059, pp. 72–86. Springer, Heidelberg (2004)
14. Basseur, M., Seynhaeve, F., Talbi, E.-G.: Design of multi-objective evolutionary algorithms: Application to the flow-shop scheduling problem. In: Congress on Evolutionary Computation, CEC 2002, Honolulu, Hawaii, USA, pp. 1151–1156 (May 2002)
15. Basseur, M., Seynhaeve, F., Talbi, E.-G.: Adaptive mechanisms for multi-objective evolutionary algorithms. In: Congress on Engineering in System Application, CESA 2003, Lille, France, pp. 72–86 (2003)
16. Basseur, M., Seynhaeve, F., Talbi, E.-G.: Path Relinking in Pareto Multi-Objective Genetic Algorithms. In: Coello Coello, C.A., Aguirre, A.H., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 120–134. Springer, Heidelberg (2005)
17. Beasley, J.E.: OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society 41(11), 1069–1072 (1990)
18. Beausoleil, R.P.: Mutiple criteria scatter search. In: 4th Metaheuristics International Conference (MIC 2001), Porto, Portugal, pp. 539–544 (2001)

19. Belding, T.: The distributed genetic algorithm revisted. In: Eshelmann, D. (ed.) Sixth Int. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo (1995)
20. Belew, R.K., McInerny, J., Schraudolph, N.N.: Evolving networks: Using genetic algorithms with connectionist learning. In: Langton, C.G., Taylor, C., Doyne Farmer, J.D., Rasmussen, S. (eds.) Second Conf. on Artificial Life, pp. 511–548. Addison-Wesley, USA (1991)
21. Bellman, R.: Dynamic programming. Princeton University Press, NJ (1957)
22. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4, 238–252 (1962)
23. Bertsekas, D.P.: Network optimization: Continuous and discrete models. Athena Scientific, MA (1998)
24. Boese, K.D.: Models for iterative global optimization. PhD thesis. University of California, Los Angeles (1996)
25. Boese, K.D., Kahng, A.B., Muddu, S.: New adaptive multi-start techniques for combinatorial global optimizations. Operations Research Letters 16(2), 101–113 (1994)
26. Braun, H.: On Solving Traveling Salesman Problems by Genetic Algorithms. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 129–133. Springer, Heidelberg (1991)
27. Burke, E.K., Cowling, P.I., Keuthen, R.: Effective Local and Guided Variable Neighbourhood Search Methods for the Asymmetric Travelling Salesman Problem. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 203–312. Springer, Heidelberg (2001)
28. Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., Schulemburg, S.: Hyperheuristics: An emerging direction in modern search technology. In: Handbook of Metaheuristics. Kluwer Academic Publishers (2003)
29. Burke, E.K., Landa Silva, J.D., Soubeiga, E.: Hyperheuristic approaches for multiobjective optimisation. In: 5th Metaheuristics International Conference (MIC 2003), Kyoto, Japan (August 2003)
30. Caseau, Y., Laburthe, F.: Disjunctive scheduling with task intervals. Technical Report LIENS-95-25, Ecole Normale Supérieure de Paris, France (1995)
31. Caseau, Y., Laburthe, F.: Heuristics for large constrained routing problems. Journal of Heuristics 5, 281–303 (1999)
32. Cesta, A., Cortellessa, G., Oddi, A., Policella, N., Susi, A.: A Constraint-Based Architecture for Flexible Support to Activity Scheduling. In: Esposito, F. (ed.) AI*IA 2001. LNCS (LNAI), vol. 2175, pp. 369–390. Springer, Heidelberg (2001)
33. Chabrier, A., Danna, E., Le Pape, C.: Coopération entre génération de colonnes sans cycle et recherche locale appliquée au routage de véhicules. In: Huitièmes Journées Nationales sur la Résolution de Problèmes NP-Complets, JNPC 2002, Nice, France (May 2002)
34. Chang, C.S., Huang, J.S.: Optimal multiobjective SVC planning for voltage stability enhancement. IEE Proceedings on Generation, Transmission and Distribution 145(2), 203–209 (1998)
35. Chelouah, R., Siarry, P.: A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global optimization of multiminima functions. European Journal of Operational Research 161(3), 636–654 (2004)
36. Chen, H., Flann, N.S.: Parallel simulated annealing and genetic algorithms: A space of hybrid methods. In: Davidor, Y., Schwefel, H.-P., Manner, R. (eds.) Third Conf. on Parallel Problem Solving from Nature, PPSN 1994, Jerusalem, Israel, pp. 428–436. Springer (October 1994)
37. Chu, P.C.: A genetic algorithm approach for combinatorial optimization problems. PhD thesis. University of London, London, UK (1997)
38. Chvatal, V.: A greedy heuristic for the set covering problem. Mathematics of Operations Research 4(3), 233–235 (1979)

39. Clearwater, S.H., Hogg, T., Huberman, B.A.: Cooperative problem solving. In: Huberman, B.A. (ed.) Computation: The Micro and the Macro View, pp. 33–70. World Scientific (1992)
40. Clearwater, S.H., Huberman, B.A., Hogg, T.: Cooperative solution of constraint satisfaction problems. Science 254, 1181–1183 (1991)
41. Cohoon, J., Hedge, S., Martin, W., Richards, D.: Punctuated equilibria: A parallel genetic algorithm. In: Grefenstette, J.J. (ed.) Second Int. Conf. on Genetic Algorithms, pp. 148–154. MIT, Cambridge (1987)
42. Cohoon, J.P., Martin, W.N., Richards, D.S.: Genetic Algorithms and Punctuated Equilibria. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 134–141. Springer, Heidelberg (1991)
43. Cohoon, J.P., Martin, W.N., Richards, D.S.: A multi-population genetic algorithm for solving the k-partition problem on hypercubes. In: Belew, R.K., Booker, L.B. (eds.) Fourth Int. Conf. on Genetic Algorithms, pp. 244–248. Morgan Kaufmann, San Mateo (1991)
44. Cook, W., Seymour, P.: Tour merging via branch-decomposition. INFORMS Journal on Computing 15(3), 233–248 (2003)
45. Coyne, J., Paton, R.: Genetic Algorithms and Directed Adaptation. In: Fogarty, T.C. (ed.) AISB-WS 1994. LNCS, vol. 865, pp. 103–114. Springer, Heidelberg (1994)
46. Crainic, T.G., Nguyen, A.T., Gendreau, M.: Cooperative multi-thread parallel tabu search with evolutionary adaptive memory. In: 2nd Int. Conf. on Metaheuristics, Sophia Antipolis, France (July 1997)
47. Crainic, T.G., Toulouse, M., Gendreau, M.: Synchronous tabu search parallelization strategies for multi-commodity location-allocation with balancing requirements. OR Spektrum 17, 113–123 (1995)
48. Crainic, T.G., Toulouse, M.: Parallel strategies for metaheuristics. In: Glover, F.W., Kochenberger, G.A. (eds.) Handbook of Metaheuristics, pp. 475–513. Springer (2003)
49. Cung, V.-D., Mautor, T., Michelon, P., Tavares, A.: A scatter search based approach for the quadratic assignment problem. In: IEEE Int. Conf. on Evolutionary Computation, ICEC 1997, Indianapolis, USA (April 1997)
50. Cung, V.-D., Mautor, T., Michelon, P., Tavares, A.: Recherche dispersée parallèle. In: Deuxième Congrés de la Société Francaise de Recherche Opérationnelle et d'Aide à la Décision, ROADEF 1999, Autrans, France (January 1999)
51. Dalboni, F.L., Ochi, L.S., Drummond, L.M.D.: On improving evolutionary algorithms by using data mining for the oil collector vehicle routing problem. In: Int. Network Optimization Conf., INOC 2003, Paris, France (October 2003)
52. Davis, L.: Job-shop scheduling with genetic algorithms. In: Grefenstette, J.J. (ed.) Int. Conf. on Genetic Algorithms and their Applications, Pittsburgh, pp. 136–140 (1985)
53. Deb, K., Goel, T.: A hybrid Multi-Objective Evolutionary Approach to Engineering Shape Design. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D. (eds.) EMO 2001. LNCS, vol. 1993, pp. 385–399. Springer, Heidelberg (2001)
54. Delisle, P., Krajecki, M., Gravel, M., Gagné, C.: Parallel implementation of an ant colony optimization metaheuristic with OpenMP. In: 3rd European Workshop on OpenMP (EWOMP 2001), pp. 8–12 (2001)
55. Dimitrescu, I., Stutzle, T.: Combinations of local search and exact algorithms. In: Evo Workshops, pp. 211–223 (2003)
56. Dowsland, K.A.: Nurse scheduling with tabu search and strategic oscillation. European Journal of Operational Research 106, 393–407 (1998)
57. Dowsland, K.A., Herbert, E.A., Kendall, G.: Using tree search bounds to enhance a genetic algorithm approach to two rectangle packing problems. European Journal of Operational Research 168(2), 390–402 (2006)
58. Dowsland, K.A., Thomson, J.M.: Solving a nurse scheduling problem with knapsacks, networks and tabu search. Journal of Operational Research Society 51, 825–833 (2000)

59. Eby, D., Averill, R., Punch, W., Goodman, E.: Evaluation of injection island model GA performance on flywheel design optimization. In: Int. Conf on Adaptive Computing in Design and Manufacturing, Devon, UK, pp. 121–136. Springer (1998)
60. Engelmore, R.S., Morgan, A.: Blackboard systems. Addison-Wesley (1988)
61. De Falco, I., Del Balio, R., Tarantino, E.: An analysis of parallel heuristics for task allocation in multicomputers. Computing 59(3), 259–275 (1997)
62. De Falco, I., Del Balio, R., Tarantino, E., Vaccaro, R.: Improving search by incorporating evolution principles in parallel tabu search. In: IEEE Conference on Evolutionary Computation, pp. 823–828 (1994)
63. Federgruen, A., Tzur, M.: Time-partitioning heuristics: Application to one warehouse, multi-item, multi-retailer lot-sizing problems. Naval Research Logistics 46, 463–486 (1999)
64. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
65. Feo, T.A., Resende, M.G.C., Smith, S.H.: A greedy randomized adaptive search procedure for maximum independent set. Operations Research 42, 860–878 (1994)
66. Feo, T.A., Venkatraman, K., Bard, J.F.: A GRASP for a difficult single machine scheduling problem. Computers and Operations Research 18, 635–643 (1991)
67. Filho, G.R., Lorena, L.A.N.: Constructive genetic algorithm and column generation: An application to graph coloring. In: APORS 2000 Conf. of the Association of the Asian-Pacific Operations Research Societies within IFORS (2000)
68. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming B 98(23-47) (2003)
69. Fisher, M.L.: An application oriented guide to lagrangian relaxation. Interfaces 15, 399–404 (1985)
70. Fleurent, C., Ferland, J.A.: Genetic hybrids for the quadratic assignment problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 16, 173–188 (1994)
71. Fleurent, C., Ferland, J.A.: Genetic and hybrid algorithms for graph coloring. Annals of Operations Research 63(3), 437–461 (1996)
72. Focacci, F., Laburthe, F., Lodi, A.: Local search and constraint programming. In: Handbook of Metaheuristics. International Series in Operations Research and Management Science. Kluwer Academic Publishers, Norwell (2002)
73. Fonlupt, C., Robillard, D., Preux, P., Talbi, E.-G.: Fitness landscape and performance of metaheuristics. In: Meta-Heuristics - Advances and Trends in Local Search Paradigms for Optimization, pp. 255–266. Kluwer Academic Press (1999)
74. Gao, B., Liu, T.-Y., Feng, G., Qin, T., Cheng, Q.-S., Ma, W.-Y.: Hierarchical taxonomy preparation for text categorization using consistent bipartite spectral graph copartitioning. IEEE Transactions on Knowledge and Data Engineering 17(9), 1263–1273 (2005)
75. Gen, M., Lin, L.: Multiobjective hybrid genetic algorithm for bicriteria network design problem. In: The 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems, Cairns, Australia, pp. 73–82 (December 2004)
76. Gendreau, M., Laporte, G., Semet, F.: The covering tour problem. Operations Research 45, 568–576 (1997)
77. Gilmore, P.C., Gomory, R.E.: A linear programming approach to the cutting stock problem. Operations Research 9, 849–859 (1961)
78. Ginsberg, M.L.: Dynamic backtracking. Journal of Artificial Intelligence Research 1, 25–46 (1993)
79. Golden, B., Pepper, J., Vossen, T.: Using genetic algorithms for setting parameter values in heuristic search. Intelligent Engineering Systems Through Artificial Neural Networks 1, 9–32 (1998)
80. Golovkin, I.E., Louis, S.J., Mancini, R.C.: Parallel implementation of niched Pareto genetic algorithm code for x-ray plasma spectroscopy. In: Congress on Evolutionary Computation, CEC 2002, pp. 1820–1824 (2002)

81. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bulletin AMS 64, 275–278 (1958)
82. Grefenstette, J.J.: Incorporating problem specific knowledge into genetic algorithms. In: Davis, L. (ed.) Genetic Algorithms and Simulated Annealing, Research Notes in Artificial Intelligence, pp. 42–60. Morgan Kaufmann, San Mateo (1987)
83. Gutin, G.M.: Exponential neighborhood local search for the traveling salesman problem. Computers and Operations Research 26(4), 313–320 (1999)
84. Habet, D., Li, C.-M., Devendeville, L., Vasquez, M.: A Hybrid Approach for SAT. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 172–184. Springer, Heidelberg (2002)
85. Hansen, P., Mladenovic, M., Perez-Britos, D.: Variable neighborhood decomposition search. Journal of Heuristics 7(4), 330–350 (2001)
86. Hart, W.E.: Adaptive global optimization with local search. PhD thesis. University of California, San Diego (1994)
87. Harvey, W.D., Ginsberg, M.L.: Limited discrepancy search. In: IJCAI Int. Joint Conference on Artificial Intelligence, pp. 607–613. Morgan Kaufmann (1997)
88. Hindi, K.S., Fleszar, K., Charalambous, C.: An effective heuristic for the CLSP with setup times. Journal of the Operations Research Society 54, 490–498 (2003)
89. Hinterding, R., Michalewicz, Z., Eiben, A.-E.: Adaptation in evolutionary computation: A survey. In: Proceedings of the IEEE Conference on Evolutionary Computation, Indianapolis, USA, pp. 65–69 (April 1997)
90. Hogg, T., Williams, C.: Solving the really hard problems with cooperative search. In: 11th Conf. on Artificial Intelligemce AAAI 1993, pp. 231–236. AAAI Press (1993)
91. Hong, T.-P., Wang, H.-S., Chen, W.-C.: Simultaneous applying multiple mutation operators in genetic algorithm. Journal of Heuristics 6(4), 439–455 (2000)
92. Huberman, B.A.: The performance of cooperative processes. Physica D 42, 38–47 (1990)
93. Husbands, P., Mill, F., Warrington, S.: Genetic Algorithms, Production Plan Optimisation and Scheduling. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 80–84. Springer, Heidelberg (1991)
94. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews 28(3), 392–403 (1998)
95. Jahuira, C.A.R., Cuadros-Vargas, E.: Solving the TSP by mixing GAs with minimal spanning trees. In: First Int. Conf. of the Peruvian Computer Society, Lima, Peru, pp. 123–132 (2003)
96. Jaszkiewicz, A.: Genetic local search for multiple objective combinatorial optimization. Technical Report RA-014/98. Institute of Computing Science, Poznan University of Technology (1998)
97. Jaszkiewicz, J.: Path relinking for multiple objective combinatorial optimization: TSP case study. In: The 16th Mini-EURO Conference and 10th Meeting of EWGT (Euro Working Group Transportation) (2005)
98. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing 9(1), 3–12 (2005)
99. Jin, Y., Sendhoff, B.: Reducing Fitness Evaluations Using Clustering Techniques and Neural Network Ensembles. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 688–699. Springer, Heidelberg (2004)
100. Jog, P., Suh, J.Y., Van Gucht, D.: The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem. In: 3rd Int. Conf. Genetic Algorithms. Morgan Kaufmann, USA (1989)
101. Jones, B.R., Crossley, W.A., Lyrintzis, A.S.: Aerodynamic and aeroacoustic optimization of airfoils via parallel genetic algorithm. Journal of Aircraft 37(6), 1088–1098 (2000)
102. Jourdan, L., Basseur, M., Talbi, E.-G.: Hybridizing exact methods and metaheuristics: A taxonomy. European Journal of Operational Research (2008) (to appear)

103. Jourdan, L., Corne, D.W., Savic, D.A., Walters, G.A.: Preliminary Investigation of the 'Learnable Evolution Model' for Faster/Better Multiobjective Water Systems Design. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 841–855. Springer, Heidelberg (2005)
104. Jourdan, L., Dhaenens, C., Talbi, E.-G.: Using Datamining Techniques to Help Metaheuristics: A Short Survey. In: Almeida, F., Blesa Aguilera, M.J., Blum, C., Moreno Vega, J.M., Pérez Pérez, M., Roli, A., Sampels, M. (eds.) HM 2006. LNCS, vol. 4030, pp. 57–69. Springer, Heidelberg (2006)
105. Jozefowiez, N.: Modélisation et résolution approchée de problèmes de tournées multiobjectif. PhD thesis. University of Lille, Lille, France (2004)
106. Jozefowiez, N., Semet, F., Talbi, E.-G.: Parallel and Hybrid Models for Multi-Objective Optimization: Application to the Vehicle Routing Problem. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 271–280. Springer, Heidelberg (2002)
107. Jozefowiez, N., Semet, F., Talbi, E.-G.: The bi-objective covering tour problem. Computers and Operations Research 34, 1929–1943 (2007)
108. Juenger, M., Reinelt, G., Thienel, S.: Practical problem solving with cutting plane algorithms in combinatorial optimization. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 20, 111–152 (1995)
109. Kamarainen, O., El Sakkout, H.: Local Probing Applied to Scheduling. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 155–171. Springer, Heidelberg (2002)
110. Karp, R.M.: Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane. Mathematics of Operations Research 2, 209–224 (1977)
111. Kim, H., Hayashi, Y., Nara, K.: The performance of hybridized algorithm of genetic algorithm simulated annealing and tabu search for thermal unit maintenance scheduling. In: 2nd IEEE Conf. on Evolutionary Computation, ICEC 1995, Perth, Australia, pp. 114–119 (December 1995)
112. Kim, H.-S., Cho, S.-B.: An efficient genetic algorithm with less fitness evaluation by clustering. In: Congress on Evolutionary Computation, CEC 2001, pp. 887–894. IEEE Press (2001)
113. Kostikas, K., Fragakis, C.: Genetic Programming Applied to Mixed Integer Programming. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 113–124. Springer, Heidelberg (2004)
114. Koza, J., Andre, D.: Parallel genetic programming on a network of transputers. Technical Report CS-TR-95-1542. Stanford University (1995)
115. Krueger, M.: Méthodes d'analyse d'algorithmes d'optimisation stochastiques à l'aide d'algorithmes génétiques. PhD thesis, Ecole Nationale Supèrieure des Télécommunications, Paris, France (December 1993)
116. Lemesre, J., Dhaenens, C., Talbi, E.-G.: An exact parallel method for a bi-objective permutation flowshop problem. European Journal of Operational Research (EJOR) 177(3), 1641–1655 (2007)
117. Levine, D.: A parallel genetic algorithm for the set partitioning problem. PhD thesis. Argonne National Laboratory, Illinois Institute of Technology, Argonne, USA (May 1994)
118. Lin, F.T., Kao, C.Y., Hsu, C.C.: Incorporating genetic algorithms into simulated annealing. In: Proc. of the Fourth Int. Symp. on AI, pp. 290–297 (1991)
119. Louis, S.J.: Genetic learning from experiences. In: Congress on Evolutionary Computations, CEC 2003, Australia, pp. 2118–2125 (2003)
120. Lourenco, H.R.: Job-shop scheduling: Computational study of local search and large-step optimization methods. European Journal of Operational Research 83, 347–367 (1995)
121. Mahfoud, S.W., Goldberg, D.E.: Parallel recombinative simulated annealing: A genetic algorithm. Parallel Computing 21, 1–28 (1995)

122. Maniezzo, V.: Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. INFORMS Journal on Computing 11(4), 358–369 (1999)
123. Mariano, C.E., Morales, E.: A multiple objective ant-q algorithm for the design of water distribution irrigation networks. In: First Int. Workshop on Ant Colony Optimization, ANTS 1998, Brussels, Belgium (1998)
124. Martin, O.C., Otto, S.W., Felten, E.W.: Large-step markov chains for the TSP: Incorporating local search heuristics. Operation Research Letters 11, 219–224 (1992)
125. Mautor, T., Michelon, P.: Mimausa: A new hybrid method combining exact solution and local search. In: Second Int. Conf. on Metaheuristics, Sophia-Antipolis, France (1997)
126. Meunier, H., Talbi, E.-G., Reininger, P.: A multiobjective genetic algorithm for radio network optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000), La Jolla, CA, USA, pp. 317–324. IEEE Press (2000)
127. Michalski, R.S.: Learnable evolution model: Evolutionary processes guided by machine learning. Machine Learning 38(1), 9–40 (2000)
128. Minsky, M.: Negative expertise. International Journal of Expert Systems 7(1), 13–19 (1994)
129. Nagar, A., Heragu, S.S., Haddock, J.: A metaheuristic algorithm for a bi-criteria scheduling problem. Annals of Operations Research 63, 397–414 (1995)
130. Narayek, A., Smith, S., Ohler, C.: Integrating local search advice into a refinment search solver (or not). In: CP 2003 Workshop on Cooperative Constraint Problem Solvers, pp. 29–43 (2003)
131. Nemhauser, G., Wolsey, L.: Integer and combinatorial optimization. Wiley (1999)
132. Nissen, V.: Solving the quadratic assignment problem with clues from nature. IEEE Transactions on Neural Networks 5(1), 66–72 (1994)
133. Nuijten, W., Le Pape, C.: Constraint based job scheduling with ILOG scheduler. Journal of Heuristics 3, 271–286 (1998)
134. Nwana, V., Darby-Dowman, K., Mitra, G.: A cooperative parallel heuristic for mixed zero-one linear programming. European Journal of Operational Research 164, 12–23 (2005)
135. O'Reilly, U.-M., Oppacher, F.: Hybridized crossover-based techniques for program discovery. In: IEEE Int. Conf. on Evolutionary Computation, ICEC 1995, Perth, Australia, pp. 573–578 (December 1995)
136. Patterson, R., Rolland, E., Pirkul, H.: A memory adaptive reasoning technique for solving the capacitated minimum spanning tree problem. Journal of Heuristics 5, 159–180 (1999)
137. Pesant, G., Gendreau, M.: A view of local search in constraint programming. Journal of Heuristics 5, 255–279 (1999)
138. Potts, C.N., Velde, S.L.: Dynasearch-iterative local improvement by dynamic programming. Technical Report TR. University of Twente, Netherlands (1995)
139. Prestwich, S.: Combining the scalability of local search with the pruning techniques of systematic search. Annals of Operations Research 115, 51–72 (2002)
140. Puchinger, J., Raidl, G.R.: Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2005. LNCS, vol. 3562, pp. 41–53. Springer, Heidelberg (2005)
141. Ramsey, C.L., Grefenstette, J.J.: Case-based initialization of genetic algorithms. In: Fifth Int. Conf. on Genetic Algorithms, pp. 84–91 (1993)
142. Rasheed, K., Vattam, S., Ni, X.: Comparison of methods for developing dynamic reduced models for design optimization. In: CEC 2002 Congress on Evolutionary Computation, pp. 390–395 (2002)
143. Renders, J.-M., Bersini, H.: Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways. In: First IEEE International Conference on Evolutionary Computation, pp. 312–317 (1994)

144. Reynolds, R.G., Michalewicz, Z., Peng, B.: Cultural algorithms: Computational modeling of how cultures learn to solve problems-an engineering example. Cybernetics and Systems 36(8), 753–771 (2005)
145. Ribeiro, M., Plastino, A., Martins, S.: Hybridization of GRASP metaheuristic with data mining techniques. Journal of Mathematical Modelling and Algorithms 5(1), 23–41 (2006)
146. Rosing, K.E., ReVelle, C.S.: Heuristic concentration: Two stage solution construction. European Journal of Operational Research 97(1), 75–86 (1997)
147. Rowe, J., Vinsen, K., Marvin, N.: Parallel GAs for multiobjective functions. In: Proc. of the 2nd Nordic Workshop on Genetic Algorithms and Their Applications (2NWGA), pp. 61–70 (1996)
148. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by backpropagating errors. Nature 323, 533–536 (1986)
149. Salami, M., Cain, G.: Genetic algorithm processor on reprogrammable architectures. In: Fifth Annual Conference on Evolutionary Programming, EP 1996. MIT Press, San Diego (1996)
150. Sebag, M., Schoenauer, M., Ravise, C.: Toward civilized evolution: Developing inhibitions. In: Bäck, T. (ed.) Seventh Int. Conf. on Genetic Algorithms, pp. 291–298 (1997)
151. Sefraoui, M., Periaux, J.: A Hierarchical Genetic Algorithm Using Multiple Models for Optimization. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 879–888. Springer, Heidelberg (2000)
152. Sellmann, M., Ansótegui, C.: Disco - novo - gogo: Integrating local search and complete search with restarts. In: The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, Boston, USA (2006)
153. Shahookar, K., Mazumder, P.: A genetic approach to standard cell placement using meta-genetic parameter optimization. IEEE Trans. on Computer-Aided Design 9(5), 500–511 (1990)
154. Shaw, P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: Maher, M.J., Puget, J.-F. (eds.) CP 1998. LNCS, vol. 1520, pp. 417–431. Springer, Heidelberg (1998)
155. Sprave, J.: A unified model of non-panmictic population structures in evolutionary algorithms. In: Proc. of the 1999 Congress on Evolutionary Computation, Piscataway, NJ, vol. 2, pp. 1384–1391. IEEE Press (1999)
156. Stutzle, T., Hoos, H.H.: The MAX-MIN ant system and local search for combinatorial optimization problems: Towards adaptive tools for global optimization. In: 2nd Int. Conf. on Metaheuristics, Sophia Antipolis, France, pp. 191–193. INRIA (July 1997)
157. Suh, J.Y., Van Gucht, D.: Incorporating heuristic information into genetic search. In: 2nd Int. Conf. Genetic Algorithms, pp. 100–107. Lawrence Erlbaum Associates, USA (1987)
158. Taillard, E.: Parallel iterative search methods for vehicle routing problem. Networks 23, 661–673 (1993)
159. Taillard, E.: Heuristic methods for large centroid clustering problems. Journal of Heuristics 9(1), 51–74 (2003)
160. Taillard, E., Voss, S.: POPMUSIC: Partial optimization metaheuristic under special intensification conditions. In: Essays and Surveys in Metaheuristics, pp. 613–629. Kluwer Academic Publishers (2002)
161. Taillard, E.D., Gambardella, L.: Adaptive memories for the quadratic assignment problem. Technical Report 87-97. IDSIA, Lugano, Switzerland (1997)
162. Taillard, E.D., Gambardella, L.M., Gendreau, M., Potvin, J.-Y.: Adaptive memory programming: a unified view of metaheuristics. European Journal of Operational Research 135(1), 1–16 (2001)
163. Talbi, E.-G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics 8, 541–564 (2002)

164. Talbi, E.-G., Bachelet, V.: COSEARCH: A parallel cooperative metaheuristic. Journal of Mathematical Modelling and Algorithms (JMMA) 5(2), 5–22 (2006)
165. Talbi, E.-G., Fonlupt, C., Preux, P., Robillard, D.: Paysages de problèmes d'optimisation et performances des méta-heuristiques. In: Premier Congrés de la Société Francaise de Recherche Opérationnelle et Aide à la Décision ROAD, Paris, France (January 1998)
166. Talbi, E.G., Muntean, T., Samarandache, I.: Hybridation des algorithmes génétiques avec la recherche tabou. In: Evolution Artificielle, EA 1994, Toulouse, France (September 1994)
167. Talbi, E.-G., Rahoual, M., Mabed, M.H., Dhaenens, C.: A Hybrid Evolutionary Approach for Multicriteria Optimization Problems: Application to the Flow Shop. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 416–428. Springer, Heidelberg (2001)
168. Talukdar, S., Baerentzen, L., Gove, A., De Souza, P.: Asynchronous teams: cooperation schemes for autonomous agents. Journal of Heuristics 4(4), 295–321 (1998)
169. Tamura, H., Hirahara, A., Hatono, I., Umano, M.: An approximate solution method for combinatorial optimization-hybrid approach of genetic algorithm and lagrangean relaxation method. Trans. Soc. Instrum. Control Engineering 130, 329–336 (1994)
170. Tanese, R.: Parallel genetic algorithms for a hypercube. In: Proc. of the Second Int. Conf. on Genetic Algorithms, pp. 177–183. MIT, Cambridge (1987)
171. Thiel, J., Voss, S.: Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms. INFOR 32(4), 226–242 (1994)
172. Toulouse, M., Crainic, T., Gendreau, M.: Communication issues in designing cooperative multi-thread parallel searches. In: Osman, I.H., Kelly, J.P. (eds.) Meta-Heuristics: Theory and Applications, pp. 501–522. Kluwer Academic Publishers (1996)
173. Tuson, A., Ross, P.: Adapting operator settings in genetic algorithms. Evolutionary Computation 6(2), 161–184 (1998)
174. Ulder, N.L.J., Aarts, E.H.L., Bandelt, H.-J., Van Laarhoven, P.J.M., Pesch, E.: Genetic Local Search Algorithms for the Traveling Salesman Problem. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 109–116. Springer, Heidelberg (1991)
175. Vasquez, M., Hao, J.-K.: A hybrid approach for the 0-1 multidimensional knapsack problem. In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, pp. 328–333 (2001)
176. Verhoeven, M.G.A., Aarts, E.H.L.: Parallel local search. Journal of Heuristics 1(1), 43–65 (1995)
177. Visée, M., Teghem, J., Pirlot, M., Ulungu, E.L.: Two-phases method and branch and bound procedures to solve knapsack problem. Journal of Global Optimization 12, 139–155 (1998)
178. Voigt, H.-M., Born, J., Santibanez-Koref, I.: Modelling and Simulation of Distributed Evolutionary Search Processes for Function Optimization. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 373–380. Springer, Heidelberg (1991)
179. Voss, S.: Tabu search: Applications and prospects. In: Network Optimization Problems, pp. 333–353. World Scientific, USA (1993)
180. Wang, L.-H., Kao, C.-Y., Ouh-young, M., Chen, W.-C.: Molecular binding: A case study of the population-based annealing genetic algorithms. In: IEEE Int. Conf. on Evolutionary Computation, ICEC 1995, Perth, Australia, pp. 50–55 (December 1995)
181. Wright, A.H.: Genetic algorithms for real parameter optimization. In: Foundation of Genetic Algorithms, pp. 205–218. Morgan Kaufmann (1991)
182. Yagiura, M., Ibaraki, T.: Metaheuristics as robust and simple optimization tools. In: IEEE Int. Conf. on Evolutionary Computation, ICEC 1996, pp. 541–546 (1996)
183. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Trans. on Evolutionary Computation 3(4), 257–271 (1999)