1 PARALLEL META-HEURISTICS APPLICATIONS

TEODOR GABRIEL CRAINIC
École des sciences de la gestion
Université du Québec à Montréal
and
Centre de recherche sur les transports
Université de Montréal
ontreal, Canada

NOURREDINE HAIL
Centre de recherche sur les transports
Université de Montréal
Montreal, Canada

1.1 INTRODUCTION

Parallel meta-heuristics have been applied to a multitude of fields by people with different backgrounds and active in different scientific communities. This illustrates the importance of parallel meta-heuristics. This phenomenon has also resulted, however, in what may be called a lack of dissemination of results across fields. This is unfortunate but not exclusive to the parallel meta-heuristics field. This chapter aims to contribute toward filling this gap.

The parallel meta-heuristic field is a very broad one. Parallel meta-heuristics may potentially be applied to any decision problem in whichever field, as indeed it is the case with sequential meta-heuristics. Yet, the space available for this chapter imposes hard choices and limits the presentation. We have therefore selected a number of topics that we believe representative due to their significant methodological impact and broad practical interest: graph coloring and partitioning, Steiner tree

problems, set covering and partitioning, satisfiability and max-sat problems, quadratic assignment, location and network design, traveling salesman and vehicle routing problems.

We do not pretend to be exhaustive. We have also restricted to a minimum the presentation of general parallel computation issues as well as that of the parallel meta-heuristic strategies. The reader may consult a number of surveys, taxonomies, and syntheses of parallel meta-heuristics, of which quite a few address the "classical" meta-heuristics, Simulated Annealing, Genetic Algorithms, and Tabu Search, while some others address the field in more comprehensive terms:

***PSA*:** Azencott [4], Greening [71, 70], Laursen [84], Ram, Sreenivas, and Subramaniam [119];

***PGA*:** Cantù-Paz [21], Lin, Punch, and Goodman [91], Mühlenbein [104], Shonkwiler [133];

***PTS*:** Crainic, Toulouse, and Gendreau [41], Glover and Laguna [69], Voß [162];

***General*:** Barr and Hickman [9], Crainic [33], Crainic and Toulouse [37, 38], Cung *et al.* [43], Holmqvist, Migdalas, and Pardalos [74], Pardalos *et al.* [114], Verhoeven and Aarts [160],.

Surveying the literature, one notices an uneven distribution of work among the various areas, both in number of papers and in the variety of methods used. Indeed, in several fields the number of different meta-heuristics that are applied appears quite limited as is the number of parallelization strategies. As a result, general trends and conclusions are difficult to identify, even within a given area. It appears, however, that several areas would benefit from a broader investigation and critical comparison of meta-heuristics and parallel strategies. This chapter has been conceived with the aim to offer both a starting point and an incentive to undertake the research required to address these challenges.

The Chapter is organized as follows. Section 1.2 introduces the elements we use to describe and classify each contribution. Sections 1.3 to 1.12 present parallel meta-heuristic contributions to the areas identified above. Section 1.13 concludes the chapter.

1.2 PARALLEL META-HEURISTICS

To survey applications of parallel meta-heuristics requires that one defines some criteria to classify the parallel meta-heuristic strategies. Our approach is one of conceptual, algorithmic design, the computer implementation plying second violin.

We adopt for this Chapter a classification that is sufficiently general to encompass all meta-heuristic classes without, on the one side, erasing the specifics of each while, on the other side, avoiding a level of detail incompatible with the scope and dimension limits of the chapter. The classification is based on the work of Crainic, Toulouse, and Gendreau [41]) and Crainic [33]) with considerations from Crainic and

Toulouse [37, 38]. Note that Verhoeven and Aarts [160] and Cung *et al.* [43] present classifications that proceed of the same spirit as Crainic, Toulouse, and Gendreau [41].

Parallel/distributed computing applied to problem solving means that several processes work simultaneously on several processors with the common goal of solving a given problem instance. One then has to determine how the global problem-solving process is controlled and how information is exchanged among the various processes. More than one solution method may be used to perform the same task and thus a taxonomy has also to specify this characteristic. The first dimension, *Search Control Cardinality*, thus explicitly examines how the global search is controlled: either by a single process (as in master-slave implementations) or collegially by several processes that may collaborate or not. The two alternatives are identified as *1-control (1C)* and *p-control (pC)*, respectively.

The dimension relative to the type of *Search Control and Communications* addresses the issue of how information is exchanged. In parallel computing, one generally refers to *synchronous* and *asynchronous* communications. In the former case, all concerned processes have to stop and engage in some form of communication and information exchange at moments (number of iterations, time intervals, specified algorithmic stages, etc.) exogenously determined, either hard-coded or determined by a control (master) process. In the latter case, each process is in charge of its own search, as well as of establishing communications with other processes, and the global search terminates once each individual search stops. To reflect more adequately the quantity and quality of the information exchanged and shared, as well as the additional knowledge derived from these exchanges (if any), we refine these notions and define four classes of Search Control and Communication strategies, *Rigid (RS)* and *Knowledge Synchronization (KS)* and, symmetrically, *Collegial (C)* and *Knowledge Collegial (KC)*.

The third dimension indicates the *Search Differentiation*: do search threads start from the same or different solutions and do they make use of the same or different search strategies? The four cases considered are: *SPSS, Same initial Point/Population, Same search Strategy*; *SPDS, Same initial Point/Population, Different search Strategies*; *MPSS, Multiple initial Points/Populations, Same search Strategies*; *MPDS, Multiple initial Points/Populations, Different search Strategies*. Obviously, one uses "point" for neighbourhood-based methods such as Simulated Annealing, Tabu Search, Variable Neighbourhood Search, GRASP, Guided Local Search, etc., while "population" is used for Evolutionary methods (e.g., Genetic Algorithms), Scatter Search, and Colony methods (e.g., ant colonies). When the initial population is a singleton (e.g., an ant), the term *single evolutionary point* is also used, as well as, in the PGA-context, fine-grained, massive, or global parallelism.

Typically, 1-control strategies implement a classical master-slave approach that aims solely to accelerate the search. Here, a "master" processor executes a sequential meta-heuristic but dispatches computing-intensive tasks to be executed in parallel by "slave" processes. The master receives and processes the information resulting from the slave operations, selects and implements moves or, for population-based methods, selects parents and generates children, updates the memories (if any) or

the population, and decides whether to activate different search strategies or stop the search.

In the context of neighbourhood-based search, the operation most widely targeted in such approaches is the neighbourhood evaluation. At each iteration, the possible moves in the neighbourhood of the current solution are partitioned into as many sets as the number of available processors and the evaluation is carried out in parallel by slave processes. For population-based methods, it is the fitness evaluation that is most often targeted in such 1C/RS/SPSS strategies.

Probing or look-ahead strategies to the 1C/KS class with any of the search differentiation models identified previously. For neighbourhood-based methods, such an approach may allow slaves to perform a a number of iterations before synchronization and the selection of the best neighbouring solution from which the next iteration is initiated (one may move directly to the last solution identified by the slave process or not). For population-based methods, the method may allow each slave process to generate child solutions, "educate" them through a hill climbing or local search procedure, and play out a tournament to decide who of the parents and children survive and are passed back to the master.

Multi-search or *multi-thread* parallel strategies for meta-heuristics have offered generally better performances, in terms of solution quality and computing times, than the methods introduced above. Historically, independent and synchronous co-operative multi-search methods were proposed first. The emphasis currently is on asynchronous communications and co-operation. Most applications of such strategies generally fall into the *pC* category.

Independent multi-search methods belong to the pC/RS class of the taxonomy. Most implementations start several independent search processes, all using the same search strategy, from different, randomly generated, initial configurations. No attempt is made to take advantage of the multiple threads running in parallel other than to identify the best overall solution once all processes stop. This definitively earns independent search strategies their Rigid Synchronization classification. (Note that, in general, the implementations designate a processor to collect the information and verify stopping criteria.) This parallelization of the classic sequential multi-start heuristic is easy to implement and may offer satisfactory results. Co-operative strategies often offer superior performance, however.

pC/KS co-operative strategies adopt the same general approach as in the independent search case but attempt to take advantage of the parallel exploration by synchronizing processors at pre-determined intervals. In a master-slave implementation, the master process than collects the information and usually restarts the search from the best solution. Note that one can overcome the limitations of the master-slave architecture by, for example, empowering each search process to initiate synchronization of all other searches (e.g., using a broadcast) or a pre-specified subset (e.g., processes that run on neighbouring processors). Here, as in the more advanced co-operation mechanisms indicated bellow, *migration* is the term used to identify information exchanges in PGA.

Asynchronous co-operative multi-thread search methods belong to the pC/C or pC/KC classes of the taxonomy according to the quantity and quality of the infor-

mation exchanged and, eventually, on the "new" knowledge inferred based on these exchanges. Most such developments use some form of *memory* for inter-thread communications (the terms *pool*, *blackboard*, and *data warehouse* are also used sometimes). Each individual search thread starts from (usually) a different initial solution and generally follows a different search strategy. Exchanges are performed asynchronously and through the pool. The information exchanged may be simply a "good" solution, a solution and its context (e.g., memories recording recent behaviour of solution attributes), or a comprehensive history search. Some co-operation mechanisms, most migration-based population methods for example, do not keep any trace of information exchanges. Most others keep at least the solutions exchanged. Memories recording the performance of individual solutions, solution components, or even search threads may be added to the pool and statics may be gradually built. Historically, *adaptive* memory mechanisms relied on building a set of "good" partial solutions extracted from "good" solutions, while *central* memory ones kept all solutions exchanged. The differences between the two approaches tend to become more and more blurred. Various procedures may also be added to the pool to attempt to extract information or to create new informations and solutions based on the solutions exchanged. Co-operative multi-thread strategies implementing such methods belong to the pC/KC class.

One of the goals of memory-based co-operation strategies is to increase the control on the complex information diffusion process generated by chains of inter-process information exchanges. (It is indeed well-known that unrestricted access to shared information may have quite a negative impact on the global behaviour of co-operative searches.) A different approach to co-operation has been proposed recently with the same goals. The mechanism is called *diffusion* and in its original presentation was based on a *multi-level* paradigm. In this setting, each search works on a version of the problem instance "aggregated" at a given level. Exchanges are then restricted to the processes working at one aggregation level up and down. Exchanges involve solution values and context information that is used by various operators to guide the search at the receiving level. This pC/KC approach is presented in somewhat more details in Section 1.4.4.

We complete this section with a note on *hybrid* methods. The term is much used but its meaning varies widely. In a strict sense, all meta-heuristics are hybrids since they involve at least two methods. Closer to most applications, a hybrid involves at least two methods that belong to different methodological approaches. Thus, for example, using genetic operators to control the temperature evolution in a PSA method yields a hybrid. Notice, however, that by this definition, all evolutionary methods that include an "educational" component, that is an enhancement of new solutions through a hill climbing, a local search, or even a full-blown meta-heuristic, are hybrids. Most co-operative parallel strategies could be qualified as hybrids as well. Since, other than "more than one method is used", the term does not offer any fundamental insight into the design of parallel strategies for meta-heuristics, we do not use it to qualify the contributions reviewed in this chapter.

1.3 GRAPH COLORING

Graph coloring is a well-studied problem with many applications, in particular to the problem of testing printed circuit boards for unintended short circuits, frequency assignment in wireless networks, and time tabling. The parallel meta-heuristic developments for the problem are, however, very limited.

Given a graph, the problem consists in finding the minimum number of colors such that an assignment of colors to graph vertices yields a coloring where adjacent vertices display different colors. The problem is known to be NP-hard.

The unique parallel meta-heuristic contribution we are aware of was proposed by Kokosinski, Kolodziej, and Kwarciany [82]. The authors proposed a co-operative coarse-grained parallel genetic algorithm, often identified in the PGA literature as the *island* framework: each process in the co-operation run the same GA, and all subpopulations were of the same dimension. Co-operation was performed by migrating at fixed intervals, a fixed number of individuals from one island to all the other ones. The arriving individuals randomly replaced the same number of individuals of the receiving subpopulation. Two migration policies were proposed: random and elitist. In the former, the migrating individuals were randomly selected, while in the latter, the best individuals migrate. Experimental results on benchmark test problems indicated that the elitist migration approach performed better than the random one. Optimal solutions were obtained rapidly when the elitist migration strategy was applied between a small number of subpopulations (3 or 5) of size 60.

1.4 GRAPH PARTITIONING

The graph (and hypergraph) partitioning problem may be stated as follows. Given a graph $G = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$), find a partition of the graph into a number of disjoint subsets of vertices such that some conditions are satisfied. Thus, for example, find a p-set partition such that the cardinality of the p subsets of vertices are (nearly) equal.

Graph partitioning is a fundamental problem in combinatorial optimization and graph theory. It appears in many application areas, especially in computer science (e.g. VLSI design). A significant number of parallel meta-heuristics, involving a wide array of meta-heuristic methodologies, have been proposed for the problem.

1.4.1 Fine-grained PGA

Mühlenbein [103] proposed an asynchronous fine-grained PGA approach, where the individuals were placed on a particular topology (grid) and a small fixed-size neighbourhood was assigned to each individual. This neighbourhood usually consisted of the neighbour individuals on the grid, and the application of selection and crossover operators was restricted to this neighbourhood. The author suggested that different hill-climbing strategies could be applied concurrently, but did not implement this strategy.

Collins and Jefferson [31] developed a fine-grained PGA forthe multilevel graph partitioning problem using the massively parallel Connection Machine. The main purpose of this work was to characterize the difference between local and panmictic (i.e., at the level of the entire population) selection/crossover schemes, which are known to converge on a single peak of multimodal functions, even when several solutions of equal quality exist. Tests were performed on a function with two optimal solutions, and the panmictic approach never found both solutions. The method using local selection and crossover consistently found both optimal solutions and appeared more robust. The authors noted that behaviour-changing modifications to the panmictic selection and crossover operators required access to global information, which they claimed was not suitable for parallelization. One may question this conclusion, given the new co-operation mechanisms proposed since and the particular architecture used (that is no longer available).

Talbi and Bessière [146] studied the problem of mapping parallel programs on parallel architectures. The authors proposed a mathematical model related to the graph partitioning problem and a fine-grained parallel genetic algorithm. The initial population was arranged on a torus of transputers such that every transputer held exactly one individual. Each individual thus had four neighboring individuals, and each of them was selected in turn as the second parent for the reproduction operation. Each reproduction produced two offsprings, but only one of them (randomly selected) survived to participate to the replacement operation that replaced the individual with the best of its surviving offsprings. The proposed parallel algorithm was tested for a pipeline of 32 vertices partitioned into eight sub-sets. Near-linear speedup was observed. Moreover, the solution quality increased with the population size. According to the authors, the fine-grained algorithm outperformed simulated annealing and hill-climbing methods.

Maruyama, Hirose, and Konagaya [97] implemented an asynchronous fine-grained PGA on a cluster of workstations and a Sequent Symmetry computer in an attempted to adapt the fine-grained strategy to coarse-grained parallel computer architectures. Each processor had an active individual and a buffer of several suspended individuals. Active individuals were sent to all other processors, which randomly selected one individual among those received to replace one of the suspended individuals according to the fitness function. Crossover consisted of replacing part of the active individual by a part of one of the suspended individuals. Only one offspring was produced, the other parts of the active and suspended individuals being rejected. Mutation was then applied and the modified active individual was compared to the suspended individuals. If it could not survive, it was replaced by one of the suspended individuals according to the fitness function. Tests were performed with 15 processors for the Sequent Symmetry and 6 processors for the cluster of workstations. The authors reported near linear speed-ups for the same quality of solution on both types of coarse-grained architecture.

1.4.2 Coarse-grained PGA

Cohoon *et al.* [28] and Cohoon, Martin, and Richards [29, 30] compared the independent multi-search PGA (i.e., without migration) and a pC/KS co-operative strategy where migration operators were applied at regular intervals. The latter strategy outperformed the independent search approach.

Diekmann *et al.* [49] proposed a 1C/RS PGA for the p-partitioning problem implemented according to the master-slave model. Numerical experimentation on a MIMD machine using up to 64 processors showed sub-linear speedups. Yet, the solution quality of the parallel algorithm outperformed that of the sequential version. This performance was increasing with the number of processors. The authors also observed a strong dependence of the solution quality on the value of p: the larger, the better.

Lin, Punch, and Goodman [91] presented several coarse-grained genetic algorithms based on different co-operation schemes, obtained by varying the Control and Communication and the Search Differentiation strategies. Static and dynamic communications have been considered. In the static model, communications are defined by the physical topology, rings, meshes, etc. The dynamic model allows several degrees of freedom in the choice of the process to communicate with by, for example, taking into account the Hamming distance between the two processes in the given computer architecture. The migration may be synchronous, after a fixed number of generations or when a convergence threshold is attained, or asynchronous (elitist). Two Search Differentiation strategies were implemented: either the same strategy for all processes or different strategies for each by varying the genetic operators, encoding method, an so on. Computational experiments were conducted for eight instantiations of the previous PGA model. Subpopulations were of equal size, randomly generated initially, and placed in a ring. Communications were triggered at fixed intervals. Numerical results indicated that the pC/C/MPDS strategy, i.e., different GA search strategies with asynchronous migrations toward a subpopulation dynamically selected based on Hamming distance, offered the best performance. Super-linear speedup was observed for 5, 10, and 25 subpopulations.

Hidalgo *et al.* [73] studied a graph partitioning problem issued of a particular circuit design application. The authors proposed a two-level hierarchical parallelization. A pC/KS/MPDS multi-thread co-operative method, which synchronized at regular intervals to improve the best individual, run at the first level. The fitness computation was performed at the second level in a master-slave implementation. Experimental results showed good performance for up to eight processors. The overhead due to the parallelization of the fitness becomes significant for larger numbers of processors.

1.4.3 Parallel Simulated Annealing

Durand [54] addressed the issue of error tolerance for parallel simulated annealing methods that implement 1C/KS strategies with domain decomposition. In such strategies, the problem variables (i.e., vertices of the graph partitioning problem) are partitioned into subsets and distributed among a number of processors. A master-slave

approach was used on a shared-memory system. To initiate the search, the master processor partitioned the vertices into a number of initial sets, and sent each set to a processor, together with the initial values for the temperature and the number of iterations to be executed. Each slave processor then executed the simulated annealing search at the received temperature on its allocated set of variables, and sent its partial configuration of the entire solution to the master processor. Once the information from all slaves was received, the master processor merged the partial solutions into a complete solution and verified the stopping criterion. If the search continued, it generated a new partition of the variables such that each set is different from the one at the previous partition, and sent them to the slave processors together with new values for the number of iterations and the temperature. The author tested different levels of synchronization to measure the impact on the errors generated. As expected, they conclude that 1) the error is small at frequent synchronization levels but the cost in computation efficiency is high; and 2) the error increases as the frequency of synchronizations decreases and the number of processors increases.

An alternative to the decomposition-based strategies is to move to multi-search approaches, where each processors runs its own cooling schedule. Lee and Lee [86, 88, 87] examined a pC/RS independent search model as well as several co-operation variants where the SA threads interact synchronously and asynchronously at fixed or dynamic intervals. For the graph partitioning problem, dynamic interval exchange strategies generally performed best. Asynchronous and synchronous co-operative multi-thread SA outperformed the other parallelizations in terms of solution quality and running time. The pC/C strategy, where threads exchange asynchronously through a central memory obtained solutions of equal or better quality compared to the synchronous parallel schemes.

Laursen [83] proposed a different co-operation mechanism. Noting that several SA threads form a population, he proposed a scheme based on the selection and migration operators of parallel genetic algorithms. Each processor concurrently runs k simulated annealing procedures for a given number of iterations. Processors are then paired and each processor migrates (copies) its solutions to its paired processor. Thus, after the migration phase, each processor has $2k$ initial solutions and this number is reduced to k by selection. These new k solutions become the initial configurations of the k concurrent simulated annealing threads, and the search restarts on each processor. Pairing is dynamic and depends on the topology of the parallel machine. For example, in a grid topology, processors can pair with any of their corner neighbours. Three co-operation strategies were tested: no migration, global, and local (stepping-stone). Global migration corresponded to a Knowledge Synchronization strategy: the best states were brought to a given processor, which then chose the best among the best and broadcast them to all processors. This strategy suffered a 10% to 20% overhead in communication cost and produced very bad solutions. As expected, the independent search (no migration) approach was the fastest strategy but produced lower quality solutions compared to the local migration strategy, which incurred only a 2% overhead. Because processors are dynamically paired and neighbourhoods overlap, in the local co-operation scheme information propagates in the network of

processors similarly to the stepping-stone coarse-grained model for parallel genetic methods.

1.4.4 Multi-level Co-operation

Toulouse, Thulasiraman, and Glover [158]; see also Toulouse, Glover, and Thulasiraman 1998[157] proposed a new co-operation mechanism and applied it to the graph and hypergraph partitioning problems with great success (Ouyang *et al.* citeo+toulouse+tgd00,o+toulouse+tgd00a. Their approach is currently the best available for this problem.

The mechanism is called *multi-level co-operative search*, belongs to the pC/KC with potentially any Search Differentiation strategy (the authors used MPSS), and is based on the principle of controlled diffusion of information. Each search process works at a different level of aggregation of the original problem (one processor works on the original problem). The aggregation scheme ensures that a feasible solution at at any level is feasible at the more disaggregated levels. Ouyang *et al.* [111, 112] analyze various aggregation operators for the graph partitioning problem.

Each search communicates exclusively with the processes working on the immediate higher and lower aggregation levels. Improved solutions are exchanged asynchronously at various moments dynamically determined by each process according to its own logic, status, and search history. Received solutions are used to modify the search at the receiving level. An incoming solution will not be transmitted further until a number of iterations have been performed, thus avoiding the uncontrolled diffusion of information. The approach is very successful for graph partitioning problems and starts now to be applied to other fields.

Banos *et al.* [7, 6] have also used the multi-level approach as a basis for constructing a co-operative search, but used a pC/RS/MPSS strategy, implemented in a master-slave configuration. Each search thread consisted of a simulated annealing algorithm, enhanced of a simple tabu search to avoid SA cycling, and worked at a given aggregation level. Periodically, each search thread sent its best solution to the master, which selected the overall best and broadcasted it back to the threads, which then continued their search. Computational results reveal that the parallel algorithm obtained solutions as good or better that the sequential version in shorter computing times.

1.5 STEINER TREE PROBLEM

The Steiner tree problem, also identified sometimes simply as the Steiner problem (Verhoeven and Severens [161], or as the Steiner problem in graphs or the Steiner minimal tree (Martins, Ribeiro, and Souza [96], has many applications, including VLSI design and telecommunication network design (e.g., multicast routing).

Consider a graph $G = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$), and a non-negative weight function w that associates a positive value $w(e)$ to every edge $e \in \mathcal{E}$. Let $\mathcal{X} \subset \mathcal{V}$ such that $\mathcal{V} \setminus \mathcal{X} \neq \emptyset$. The Steiner tree problem then consists in finding

a minimum weighted subtree of G spanning all terminal vertices in $\mathcal{X}$. The set of non-terminal vertices of the minimum tree are called *Steiner Set*. The Steiner tree problem is NP-Hard. Many meta-heuristics belonging to various types, e.g., tabu search, simulated annealing, GRASP, genetic algorithm, and local search, have therefore been proposed. A few of these methods have been parallelized.

1.5.1 Parallel GRASP

Martins, Ribeiro, and Souza [96], Martins *et al.* [95] proposed several GRASP procedures for the Steiner problem. The parallel versions of these methods have also been proposed. The authors used a pC/RS/MPSS parallelization strategy implemented according to a master-slave model. Each thread run the same GRASP procedure with a different initial seed for a number of iterations equal to those of the sequential version divided by the number of available processors. The best solution was collected at the end. The strategy achieved high solution quality for the problem instances tested, as well as good speedups. The Martins *et al.* [95] implementation achieved comparable results, in terms of solution quality, compared to the best known tabu search with path-relinking method of Bastos and Ribeiro [10].

1.5.2 Parallel Local Search

Verhoeven and Severens [161] proposed sequential and parallel local search methods for the Steiner tree problem based on a novel neighborhoods, the authors claimed "better" than those known in the literature. The parallel strategy followed a 1C/KS model. Computational results indicated that good speed-ups could be obtained without loss in solution quality. Actually, the proposed parallel algorithm outperformed in terms of speedup the parallel GRASP of Martins, Ribeiro, and Souza [96].

1.6 SET PARTITIONING AND COVERING

Consider a set $\mathcal{S}$ of cardinality n and a collection $\mathcal{C} = (\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_m)$ of m subsets of $\mathcal{S}$. A weight c_j is associated with each subset $\mathcal{S}_j$. A partition of $\mathcal{S}$ is a subset of $\mathcal{C}$ such that all elements of $\mathcal{S}$ are included and each of them belongs to exactly one set of the partition. A cover of $\mathcal{S}$ is a subset of $\mathcal{C}$ such that each element of $\mathcal{S}$ are belongs to at least one subset in the cover. The weight of a partition or of a cover is the sum of the weights of the included sets. The set partition problem consists of finding a partition of $\mathcal{S}$ with minimum total weight, while a cover of $\mathcal{S}$ of minimum total weight is the goal of the set covering problem.

The set partitioning and covering problems may be also cast as 0-1 linear optimization problems $\min \sum_{j=1,m} c_j x_j$ subject to $Ax < (=) 1$ and $x_j \in \{0, 1\}$, where line of the matrix A corresponds to an element of $\mathcal{S}$, each column to a subset $\mathcal{S}_j,\ j = 1, \cdots, m$ of $\mathcal{C}$, and each decision variable $x_j,\ j = 1, \cdots, m$ indicates if the corresponding subset is to be part of the optimal partition (cover).

The literature on the set partitioning and covering problems is very rich. The two problems appear in many application domains, routing and scheduling, location and design, production, capital investment, image coding, and so on. Exact, heuristic, and meta-heuristic solution methods have been proposed, including a number of parallel meta-heuristics summarized in the following.

1.6.1 Set Partitioning Applications

We identified only two contributions aimed at parallel meta-heuristics for the set partitioning problem.

Levine [89] addressed the set-partitioning problem applied to the airline crew scheduling problem and proposed two versions of a coarse-grained, island parallel genetic algorithm: with (co-operative search) and without (independent search) migration. A simple co-operative mechanism was proposed: the best chromosome in a subpopulation migrates to a neighboring subpopulation at fixed intervals, while the chromosome to delete is randomly selected. A MPSS Search Differentiation strategy was selected, the initial populations for the islands being randomly and independently generated. Numerical results indicated that the two parallel versions generally offered similar solution quality, with a slight advantage to the method integrating migration. Both methods outperformed the sequential approach.

Czech [45] has studied a single-depot vehicle routing problem where each route cannot serve more than a fixed, small number of customers. The author formulated the problem as a set-partitioning problem, and proposed two parallel simulated annealing algorithms, based on the independent and co-operative multi-search paradigms, respectively. The co-operation is of the pC/KS type. The SA processes transmit their best solutions every n steps. Each thread then starts again the search after updating its best solution. The results reported show that both parallel methods obtain better results than the sequential version in terms of solution quality. Moreover, the co-operative method outperformed the independent multi-thread search.

1.6.2 Set Covering Applications

We identified four contributions to the parallel meta-heuristic for set covering literature: two island-based co-operative PGA methods, one ant colony, and one proposing randomized approaches.

Calegari *et al.* [20] have considered the problem of selecting the best set of radio-transmitter locations such that a maximum covering area with an optimal cost (or minimum number of radio transmitters) is achieved, and stated it as a variant of the set covering problem. The authors proposed a pC/KS co-operative approach according to an island model with a small population (two or four individuals) associated to each island (and processor). The islands were arranged on an oriented ring, and migrations were only allowed between neighboring islands. The authors choose this arrangement to minimize the amount of migrations and the communication overload due to migrations between remote islands. Once a new generation is computed, a copy of the best individual of each island was sent to the next island on the ring.

Each island thus received a new individual that replaced a randomly selected local individual. The proposed co-operative multi-thread search performed well in terms of computational time compared to the sequential genetic algorithm. The speedup was almost linear and the efficiency reached 93%.

Solar, Parada, and Urrutia [137] presented a 1C/KS parallel method based on a coarse-grained genetic algorithm. The authors used an island model, where each island contains one subpopulation. Initial subpopulation were independently and randomly generated. A master-slave scheme was selected to implement the proposed approach. Each slave run a standard GA on its island. After performing the computations of each generation, each slave sent its best individual to the master. Once the master received all the best individuals from the slaves, it selected the overall best and broadcast it. Each slave replaced its worst local solution with the individual sent by the master, and launched again its GA. The authors reported numerical results that showed that the parallel approach could reach near-optimal solutions. Errors ranged from 3.3% to 10% of the optimal solution value for almost all problems tested. They also concluded that the parallel approach is more efficient than the corresponding sequential one in the number of generations required to obtain a certain quality of solution. The authors claimed that the solution quality of the proposed algorithm is better than that of both tabu search and simulated annealing. Unfortunately, this comparison is not very helpful because no details were given regarding the solution quality versus computational time for each of these approaches.

Rahoual, Hadji, and Bachelet [118] presented two parallel approaches based on a combination of an ant colony system and a local search heuristic. The sequential method, called AntsLS, consisted in launching the ant searches and applying a local search to each solution found. This sequence was repeated until a stopping criterion based on a convergence threshold was reached. The first parallel approach was a pC/RS independent multi-thread search, where each thread run AntsLS and the best solution is collected at the end. The second method represented a direct parallelization of AntsLS according to a 1C/KS strategy in a a master-slave implementation. Each slave holds one ant process. The master synchronizes the searches of all ants. Once a solution is found by every slave, it is sent to the master. A pheromone updating is then performed by the master. The master also updates the best solution. The numerical results show a high performance of the independent approach in both solution quality and speedup. Not surprisingly, the 1C/RS approach did not perform well due to high communication times. It is noteworthy that the two methods do not parallelize the same algorithm and cannot therefore be compared.

Fiorenzo Catalano and Malucelli [57] discussed several general schemes that lead to approximate approaches for the Set Covering Problem. These schemes embedded two constructive heuristics: a greedy algorithm, which at each iteration added a set to the partial solution according to associated probabilities, and a randomized primal-dual approach (Beasley [13]). The authors proposed synchronous and asynchronous parallelizations of these schemes. The first synchronous approach is a variant of the 1p/KS/MPSS strategy and was used for the randomized primal-dual method. It was implemented according to a master-slave scheme. The master held the reduced cost information and updated the Lagragian multipliers, as well as up-

dates the best solution following synchronization, at regular intervals, when slaves sent their best solutions. Slaves received from the master the information required to run the procedure, but also exchanged and updated information regarding recent set operations. The second synchronous approach followed a pC/KS scheme where independent searches regularly exchange information. It was used with both randomized procedures. The authors also proposed a pC/KS/MPDS co-operative procedure implemented in a master-slave framework.. The master received and updated best solutions and search information. Each thread run one of the sequential randomized procedure and exchanged with the master. The parallel algorithms performed well.

1.7 SATISFIABILITY PROBLEMS

Satisfiability problems are boolean problems of central importance in various fields such artificial intelligence, automated reasoning, computer design, data base search, computational complexity, and so on. Loosely speaking, the problem consists in finding an assignment of variables that evaluate TRUE a given formula.

More precisely, consider a set of variables x_1 , ..., x_n , a set of clauses C_1 , ..., C_m, and operators $\wedge$ (AND), $\vee$ (OR), and NOT. A variable may be either TRUE or FALSE. A literal is defined as "a variable or its negation". A clause is a finite disjunction of one or more literals. Thus, for example, a clause with three literals, $C = x_1 \vee x_2 \vee$ NOT x_3, will be true if at least one of the literals is true.

The satisfiability problem SAT consists in determining if there is an assignment of variables that evaluate the formula $C_1 \wedge C_2 \wedge \bullet\bullet\bullet \wedge C_m$ to TRUE. The maximum satisfiability problem, MAX-SAT, refers to finding a truth assignment of variables such that the number of TRUE clauses is maximized.

1.7.1 Parallel Genetic Algorithms

Wilkerson and Nemer-Preece [163] proposed two coarse-grain PGA, an independent and a co-operative multi-search. An initial sequential phase reduced the search space by assigning a TRUE or FALSE value to each variable that appeared in the same literal (positive or negative form) in all the clauses. The remaining variables were used to generate the initial populations for an island model with 2^p processors, where $p \in \{2, 3, 4, 5\}$, represented the highest ranking variables according to the Jeroslaw-Wang rule. A different assignment of TRUE or FALSE values to the p variables was performed for each processor. Then, an initial population was randomly generated for each processor by considering the rest of the variables. the same genetic algorithm was used for all islands. A pC/KS co-operation mechanism was implemented. At fixed intervals (number of iterations), each processor broadcast its best individual to all other processors, the best new individual replacing the worst individual of the receiving population. Experimental results showed that the co-operative model outperformed the independent strategy, achieving super-linear speedups on some problem instances.

Folino, Pizzuti, and Spezzano [60, 59] proposed a fine-grained PGA based on a diffusion model in a Cellular Automata framework. Every cell contained one individual and interacted with the neighbour displaying the best fitness. The offspring survived to replace the parent and be enhanced through a local search (Selman, Levesque, and Mitchell [131], Selman, Kautz, and Cohen [130]), if it had a better fitness. Comparative experimentation on hard 3-SAT problems (Mitchell, Selman, and Levesque [100]) showed that the proposed method outperformed the parallel version of local search used to enhance surviving offspring. Additional results reported later (Folino, Pizzuti, and Spezzano [61]) displayed almost linear speedup and high quality solutions. Moreover, the parallel GA outperformed a Simulated Annealing (Spears [139]) and a Genetic Algorithm (Marchiori and Rossi [94]) developed for the SAT problem.

1.7.2 Parallel Simulated Annealing

Sohn and Biswas [136] and Sohn [135] proposed a 1C/RS PSA method for the -LSAT problem, a variant of the SAT problem where L is the ratio of the number of clauses to the number of variables. The algorithm was implemented according to a master-slave scheme. At each temperature and step, p iterations were distributed by the master among p slaves, one iteration per slave. The master received all the "accepted" solutions of the slaves and, in order to avoid errors associated to simultaneous evaluation of solution, it kept the solution of the slave with the smallest index in the list of processors. The algorithm was implemented on a large-scale distributed-memory multiprocessor machine. The authors reported high quality solutions (i.e., , the quality increasing with the number of processors. Almost linear speedup was reported, particularly when the processors number was below 100.

1.7.3 Parallel GRASP

Pitsoulis, Pardalos, and Resende [116] proposed independent multi-search parallel GRASP method for the MAX-SAT problem. Different seeds were used for each thread in the construction phase to favor the exploration of different search spaces by the independent threads. The maximum number of iterations (Resende and Feo [123]) was divided among the processors. The authors reported high quality solutions in almost linear speedups.

1.7.4 Parallel Ant Colony Systems

Assume a positive weight w_i is associated with each clause i. The weighted MAX-SAT problem then refers to finding a truth assignment of variables such that the total weight of true clauses is maximized. Drias and Ibri [50] propose a sequential ant colony system for MAX-SAT, as well as two parallelizations. Similar to coarse-grained strategies for population methods, both parallelizations are based on dividing the colony into several sub-colonies, each assigned to a different processor. In the 1C/KS/MPSS strategy, implemented according to the master-slave model, each slave

sent to the master its best solution once its ants finished searching. The master then updated the pheromone and launched a new search phase.

The second parallel followed a pC/KS/MPSS strategy. Each process executes the same ant colony method on its sub-colony. When the search is over, it broadcasts its best solution to all other sub-colonies and requests their current best solutions. A "local" synchronization is thus generated within an asynchronous-communication environment. The process then selects the overall best solution, updates the pheromone, and restarts the search. Numerical tests showed the superiority of the co-operative strategy over the first approach. Both parallel methods outperformed, however, Moreover, the scatter search proposed by Drias and Khabzaoui [51] and the sequential GRASP presented in Resende and Feo [124].

1.8 QUADRATIC ASSIGNMENT

One of the most difficult problem in combinatorial optimization, the quadratic assignment problem (QAP) may be simply stated as follows. Given A and B, two square matrices of dimension n, find a permutation p to minimize $\sum_{i=1,\ldots,n} \sum_{j=1,\ldots n} A_{ij} B_{p(i)p(j)}$. The QAP has many applications, facility location problems in particular. Neighbourhoods based on swapping elements in the permutation have been particularly popular, even though their dimensions grow very fast.

1.8.1 Tabu Search

Among the first parallelizations of tabu search for the QAP, as for the other applications with large neighbourhoods and relatively small computing efforts required to evaluate and perform a given move, one finds the 1C/RS/SPSS strategy that targets the neighbourhood evaluation. At each iteration, the possible moves in the neighbourhood of the current solution are partitioned into as many sets as the number of available processors and the evaluation is carried out in parallel by slave processes.

Chakrapani and Skorin-Kapov [22, 24, 25] proposed and studied such algorithms for the Connection Machine CM-2, a massively parallel SIMD machine. At the time, the authors reported that they either attained or improved the best known solutions for benchmark problems, in a significantly smaller number of iterations.

Taillard [142, 144] used a different implementation and a ring of 10 transputers. There was no specific master processor. Following the initial partition of the set of possible moves and their assignment to different processors, each processor evaluated the pair-wise interchange moves and identified the best one. It then broadcast it to all other processors, which then performed all the normal tasks of a "master" selecting and implementing the move, making the necessary adjustments and updates, partitioning the neighbourhood, etc. Load balancing through partition of the neighbourhood was acknowledged as critical, but no indication was given on how it was performed. On several problem sets proposed in the literature (essentially, the same set used by Chakrapani and Skorin-Kapov, with problem instances up to size 100, Taillard reported very good solutions, improving the best known values of

many of the problems tested and obtaining suboptimal solutions (conjectured but not proven to be optimal) for problems up to size 64.

Battiti and Tecchiolli [11, 12] proposed an independent multi-thread PTS, where the independent tabu searches started the exploration of the solution domain from different randomly-generated initial solutions. Each tabu search was including a hashing procedure to dynamically modify the length of the tabu lists and thus react to the behaviour of the search. The authors then proceeded to derive probability formulas for the success of this pC/RS/MPDS global search, which tended to show that the independent search parallelization scheme was efficient - the probability of success increased, while the average success time decreased with the number of processors - provided the tabu procedure did not cycle.

De Falco *et al* [47] proposed a pC/KS co-operative approach. At each iteration, each search thread performed a local search from its best solution. Best solutions were then exchanged between searches that run on neighbouring processors. Local best solutions were replaced with imported ones only if the latter solutions were better. The authors experimented on several architectures and reported that they obtained better solutions when cooperation was included compared to an independent thread strategy. Super-linear speedups were reported.

Talbi, Hafidi and Geib [147] (see also Talbi *et al.* [149] and Talbi, Hafidi and Geib [148]) presented independent multi-thread strategies based on tabu search hybridized with simulated annealing for the intensification phase. An interesting feature of this contribution was the dynamic-loading mechanism, which allowed to use the available resources of a heterogeneous network of single and multi-processors computers. The computational results showed very promising performance of the proposed dynamic loading mechanism in terms of the scheduling overhead. Overhead was low (0.09%) comparing to the total execution time. Moreover, the authors claimed very high solution quality compared to the literature.

1.8.2 Genetic Algorithm

Mühlenbein [101, 101] proposed a fine-grained PGA for the QAP. Individuals were arranged into two equal subsets placed on two rings, such that each individual had two neighbours on each ring. A hill-climbing heuristic was applied to individuals and GA operators were applied to the resulting local optimum. It is interesting to recall that this work was part of a larger body of contributions where hill-climbing heuristics were embedded into genetic algorithms to improve ("educate") individuals and the impact of this hybridization on GA behaviour and performance was addressed (e.g., Mühlenbein, Gorges-Schleuter, and Krämer [106, 107], Mühlenbein [102, 104, 105]).

1.8.3 Simulated annealing

Boissin and Lutton [17] developed a parallel SA that uses a domain decomposition strategy on a massively parallel computer. Experiments performed on a 16K Connection Machine produced interesting performances. Laursen [83] applied to the QAP

the same co-operation mechanism described in Section 1.4.3. The strategy suffered a 10% to 20% communication overhead and produced very bad solutions.

1.8.4 Parallel GRASP

Li, Pardalos, and Resende [90] and Pardalos, Pitsoulis, and Resende [115] proposed the same pC/RS/MPSS parallelization strategy for large-scale QAP that will later be applied to the Steiner problem (i.e., Martins, Ribeiro, and Souza [96] and Martins *et al.* [95]; see Section 1.5.1). Recall that the strategy calls for equal distribution of a maximum number of iterations among the available GRASP threads. Numerical experiments revealed almost linear speedup, around 62 for 64 processors.

Pardalos, Li, and Murthy [113] proposed a pC/RS/MPSS independent multi-search method. Computational results on benchmark problem instances showed that best known solutions were generally found. The speedup results were more varied.

1.8.5 Parallel Scatter Search

Cung *et al.* [44] introduced the first parallel scatter search method for the QAP. The authors proposed an independent multi-thread scatter search, where the different searches used different parameter settings. The computational results showed encouraging speedups, but no improvement in solution quality compared to the sequential algorithm. According to the authors, the main reason that could explain this is the rapid convergence of the small-sized subpopulation of each search.

1.8.6 Parallel Ant Colony Systems

According to our best knowledge, the first parallel implementation of an ant colony system for the QAP was proposed by Talbi *et al.* [150, 151]. The method consisted in a 1C/KS parallelization of ANTabu, which is a combination of an ant colony system and tabu search. In a master-slave implementation, the master kept and updated the pheromone matrix and the best solution. At each iteration, the master spread the current pheromone matrix among all the ants. Each ant performed its search to construct a complete solution, launched a tabu search to improve that solution, and sent the final solution to the master. Computational tests conducted with 10 ants indicated the method offered good performances, in terms of solution quality, compared to the independent multi-thread tabu search proposed by Talbi, Hafidi, and Geib [147].

1.9 LOCATION PROBLEMS

Location problems may be defined in continuous or discrete space (two dimensions are generally used, but not always) or on graphs. The objective is to select, locate, a number of points to cover optimally given zones or points, while satisfying various constraints (topological, demand, capacity, and so on and so forth). Location

problems are broadly applied from marketing and economic research, topolitical districting, from transportation and logistics to telecommunications and production. We present in this section parallel meta-heuristics proposed for three particular classes of location problems.

1.9.1 Simple Plant Location Problem

The Simple Plant Location Problem is a classical discreet location problem. We are given a set of customer points with known demands and a set of potential plant (or warehouse) locations. A fixed cost for opening the facility is associated to each potential plant site. Transportation costs are associated to each customer-plant site pair. The objective is to select the plant locations such that customer demand is satisfied at minimum total system, opening and transportation, cost.

Kohlmorgern, Schmeck, and Haase [81] proposed two fine-grained parallel versions based on the island and the diffusion models for the uncapacitated warehouse location problem. The initial population was divided into 1, 4, 16, 64, 256, and 1024 islands on a parallel machine with 16K processors, each processor holding one individual. Individuals were arranged in a two-dimensional grid and had eight neighbours. In the first version, a sequential GA was launched in each island (for each sub-population). Individuals were exchanged between two neighboring island after a pre-determined number of iterations. The migration rate was fixed as well. The second model was based on selecting partners at given distances. Each processor selected a neighboring partner according at the given distance in one of the eight possible directions A wheel selection rule was used. Experimental results indicated that the performance of the second model increased with the number of neighbors. Best results were found when an elitist selection procedure was used. No results were given for the island model.

1.9.2 Location with Balancing requirements

The multicommodity location problem with balancing requirements emerged in the design of intermodal transportation systems. We are given a set of vehicle/commodities, a set of potential locations for vehicle depots and two sets of "customers", one that supply and another that requests known quantities of given vehicle types. Commodity-specific transportation costs are associated to the customer-to-depot, depot-to-customer, and depot-to-depot movements (there are no customer-to-customer movements). The latter are more efficient (bulk shipments) and cost less that the other two. Fixed opening costs characterize the depot locations. The objective is to design the system such that the total cost is minimized while satisfying requests at supply and demand points. The problem is NP-Hard and is represented as a fixed-cost, mixed-integer formulation with a multicommodity network structure.

The problem served to illustrate the parallel tabu search taxonomy introduced by Crainic, Toulouse, and Gendreau [41], as well as a thorough comparison of various parallelization strategies based on this taxonomy (Crainic, Toulouse, and Gendreau [40, 39]).

The authors implemented and compared a 1C/RS/SPSS and a 1C/KS/SPSS method. In a master-slave implementation, the first method had slaves evaluate candidate moves only, while in the second, called *probing*, slaves also performed a few local search iterations. The second strategy performed marginally better. However, both methods were outperformed by p-Control implementations that attempt a more thorough exploration of the solution space. A decomposition approach, which partitioned the vector of decision variables and performed a search on each subset, was also part of the study (Crainic, Toulouse, and Gendreau [40]). It performed poorly, mainly because of the nature of the class of problems considered; multicommodity location with balancing requirements requires a significant computation effort to evaluate and implement moves, resulting in a limited number of moves that may be performed during the search.

As far as we can tell, Crainic, Toulouse, and Gendreau [41] proposed the first co-operative central memory strategy for tabu search as part of their taxonomy. Other than this pC/KC/MPDS strategy, the authors also implemented and compared an independent multi-thread pC/RS/MPDS approach, pC/KS synchronous co-operations (varying the synchronization mechanisms and the Search Differentiation strategies), and broadcast-based asynchronous pC/C co-operative strategies. The authors report that the parallel versions achieved better quality solutions than the sequential ones and that, in general, asynchronous methods outperformed synchronous strategies. The independent multi-search and the asynchronous co-operative approaches offered the best performance.

Crainic and Gendreau [34] report the development of a hybrid search strategy combining the co-operative multi-thread parallel tabu search method of Crainic, Toulouse and Gendreau [39] with a genetic engine. The genetic algorithm initiates its population with the first elements from the central memory of the parallel tabu search. Asynchronous migration (migration rate = 1) subsequently transfers the best solution of the genetic pool to the parallel tabu search central memory, as well as solutions of the central memory towards the genetic population. The hybrid appears to perform well, especially on larger problems where the best known solutions are improved. It is noteworthy that the genetic algorithm alone was not performing well and that it was the parallel tabu search procedure that identified the best results once the genetic method contributed to the quality of the central memory.

The multicommodity location-allocation problem has also been used to study the impact of co-operation on the global behaviour of the search (Toulouse *et al.* [155], Toulouse, Crainic, and Sansó [153, 154], Toulouse, Crainic, and Thulasiraman [156]). The authors showed that co-operative meta-heuristics with unrestricted access to shared knowledge may experience serious premature "convergence" difficulties, especially when the shared knowledge reduces to one solution only (the overall best or the new best from a given thread). This is due to a combination of factors: There is no global history or knowledge relative to the trajectory of the parallel search and thus each process has a (very) partial view of the entire search; Threads often use similar criteria to access the (same) "best" solution; Each process may broadcast a new best solution after a few moves only and thus disseminate information where only part of the solution has been properly evaluated. The contents of shared data tend then

to become stable and biased by the best moves of the most performing threads and one observes a premature convergence of the dynamic search process. Moreover, the phenomenon may be observed whether one initializes the local memories following the import of an external solution or not.

This study explains several of the anomalies reported in the literature. It has also been the motivation for the development of more advanced co-operation concepts, in particular the central memory and the multi-level diffusion co-operation mechanisms.

Gendron, Potvin, and Soriano [67] proposed a co-operative multi-thread parallel meta-heuristic for the capacitated version of the multicomodity location problem with balancing requirements. The method combined variable neighborhood descent (VND), a version of VNS, and slope scaling (SS). The central memory-based co-operation was implemented via a master-slave architecture and consisted of two phases. In the first phase, the slaves performed SS procedures in parallel and sent the best solutions to the SS and VND memories located in the central memory. In the second phase, a certain proportion of slaves run a VND search, while the others run SS heuristics. VND processes started from SS memory and fed the VND memory, while SS processes started from VND memory and fed the SS memory. The computational tests revealed that the proposed parallel search was more diversified and thus performed better with an increase in the number of slaves.

1.9.3 The p-median Problem

Given a set of potential locations for p facilities and a set of locations for their many users, the p-median problem is to locate simultaneously the p facilities in order to minimize the total transportation cost for satisfying the demand of the users, each supplied from its closest facility. The p-median problem is one of the fundamental models in (discrete) location theory and a classical combinatorial optimization formulation with a broad application range, including cluster analysis, data mining, and so on. Despite the apparent simplicity of its mathematical expression, the p-median problem is difficult to solve. It belongs to the NP-hard class of problems and exact solution methods cannot address realistically sized problem instances in most cases of interest.

1.9.3.1 Parallel Variable Neighbourhood Search García-López *et al.* [62] proposed the first parallelizations for the p-median problem and among the first for VNS in general. The authors introduced and compared three strategies. The first approach was a 1C/RS parallelization that attempted to reduce computation time by parallelizing the local search phase within a sequential VNS. The second one implemented an independent multi-search strategy, pC/RS/MPSS, which run an independent VNS procedure on each processor, the best solution being collected at the end. The third method applied a pC/RS synchronous co-operation mechanism through a classical master-slave approach. The master processor run a sequential VNS. The current solution was sent to each slave processor that modifies it randomly to obtain an initial solution from which local search was started. The solutions were passed on to the master that selected the best and continued the algorithm. The authors tested

their methods using the TSPLIB problem instances with 1400 customers only. Not surprisingly, the last two strategies found better solutions, with the third approach using marginally less iterations than the second one.

Crainic *et al.* [36] presented a pC/C/MPSS co-operation mechanism that allowed each individual search access to the current overall best solution without disturbing its normal proceedings. The parallel procedure was implemented in an asynchronous master-slave scheme and may be summarized as follows. Individual VNS processes communicated exclusively with a central process called *central memory* or *master*. There were no communications among individual VNS processes. The master kept, updated, and communicated the current overall best solution. Solution updates and communications were performed following messages from the individual VNS processes. The master also initiated and terminated the algorithm. To initiate the process, a parallel reduced VNS (no local search phase) was executed until a number of unsuccessful trials were observed.

Each search process implemented the same VNS meta-heuristic (local search used First Improvement, Fast Interchange, and $k_{max} = p$). It proceeded with the "normal" VNS exploration for as long as it improved the solution. When the solution was not improved, it was communicated to the master (if better than the one at the last communication) and the overall best solution was requested from the master. The search was then continued starting from the best overall solution in the current neighborhood. Computational results on TSPLIB problem instances with up to 11849 customers showed that the co-operative strategy yielded significant gains in terms of computation time without loosing on solution quality. The quality of the solutions obtained was, in fact, comparable to that of the best results in the literature (when available).

1.9.3.2 Parallel Scatter Search García-López *et al.* [63] proposed three parallel scatter search methods for the p-median problem. The first parallelization was a 1C/RS synchronous parallel scatter search, where the neighborhood was divided into several disjoints subsets that were assigned to processors, one subset per processor. Each processor run a local search on its corresponding subset and returned the result to the master running the sequential scatter search. The second parallelization, called Replicated combination scatter search (RCSS), also lead to runtime reductions. It consisted of partitioning the reference set and running separate scatter searches on each one of them. The third parallelization follows an independent multi-search strategy by running the scatter search in parallel for several populations. According to the reported computational results, the simple 1C/RS parallelization achieved super-linear speedup, while the best values were found by the independent multi-search procedure.

1.10 NETWORK DESIGN

Network design problems are a generalization of location formulations. They are defined on graphs containing nodes, connected by links, either undirected edges or

directed arcs. Links may have various characteristics, such as length, capacity, and cost. In particular, fixed costs may be associated to some or all links, signaling that as soon as one chooses to use that particular arc, one has to incur the fixed cost, in excess of the utilization cost which is in most cases related to the volume of traffic on the link. Recall that when the fixed costs are associated to nodes, one obtains the formulation of a location problem. Such representations are generally used to model the cost of constructing new facilities, offering new services, or adding capacity to existing facilities. In network design problems, the aim is to choose links in a network, along with capacities, eventually, in order to enable demand to flow between their origin and destination at the lowest possible system cost, i.e., the total fixed cost of selecting the links plus the total variables cost of using the network. Network design has a wide variaty of applications in transportation, logistics, telecommunication, production, and so on.

We found few parallel-meta-heuristic developments for the fundamental network design formulations. Most contributions are dedicated to various aspects of telecommunication network design as illustrated in this section.

1.10.1 Multicommodity Network Design

Crainic and Gendreau [35] proposed a pC/KS/MPDS co-operative multi-thread parallel tabu search for the fixed cost, capacitated, multicommodity network design (CMND) problem. In their study, the individual tabu search threads differed in their initial solution and parameter settings. Communications were performed asynchronously through a central memory device. The authors compared five strategies of retrieving a solution from the pool when requested by an individual thread. The strategy that always returns the overall best solution displayed the best performance when few (4) processors were used. When the number of processors was increased, a probabilistic procedure, based on the rank of the solution in the pool, appeared to offer the best performance. The parallel procedured improves the quality of the solution and also required less (wall clock) computing time compared to the sequential version, particularly for large problems with many commodities (results for problems with up to 700 design arcs and 400 commodities are reported). The experimental results also emphasized the need for the individual threads to proceed unhindered for some time (e.g., until the first diversification move) before initiating exchanges of solutions. This ensures that local search histories can be established and good solutions can be found to establish the central memory as an *elite candidate* set. By contrast, early and frequent communications yielded a totally random search that was ineffective. The co-operative multi-thread procedure also outperformed an independent search strategy that used the same search parameters and started from the same initial points.

Crainic, Toulouse, and Li [42] proposed a multi-level diffusion parallel algorithm for the CMND problem. Each individual search thread involved in cooperation made use of the cycle-based tabu search proposed by Crainic, Gendreau, and Ghamlouche [68], and explored a given level of aggregation of the graph. Aggregation was performed by variable fixing. Sets of elite solutions were built at each level and

local memories recorded the frequency of variables in elite solutions. Exchanges occurred at regular intervals and involved solutions, as well as context information (e.g., memories). Computational results on a set of benchmark problem instances indicated that this approach yielded good quality solutions comparable to those obtained by the current best meta-heuristics for the problem.

1.10.2 Telecommunication Network Design

Sleem *et al.* [134] studied the design of cost-effective ATM networks to provide low end-to-end delays and such that link are well used within their capacity limitations. A 1C/RS/MPSS parallelization strategy was implemented on a distributed (network of workstations) and a multi-processor computer. A first GA was applied to an initial population. The resulting population was divided into subpopulations, one per processor, and a second GA was applied to each subpopulation. The resulting subpopulations were then returned to the master process that reconstituted the whole population and restarted the process. Surprisingly, numerical results indicated better performance on the distributed network than on the parallel machine. The general performance of the parallel algorithm, compared to the serial method, was mediocre due to the overhead associated to exchanging large quantities of data (subpopulations).

Flores, Cegla, and Caceres [58] proposed parallel GA methods for the multi-ojective telecommunication network design. The authors proposed a pC/RS/MPSS co-operative coarse-grained island strategy. Each subpopulation performed its own evolutionary algorithm. Following each generation, an elite migration was triggered from each subopopulation to all other subpopulations. The numerical experimentation was used exclusively to compare this parallel strategy applied to two GA, SPEA (Strenght Pareto Evolutionary Algorithm) and NSGA (Non-dominated Storing Genetic Algorithm) that differed in the fitness evaluation procudure. The former dominated the latter.

Ribeiro and Rosseti [125, 126, 127] proposed pC/RS/MPSS parallelization strategies for a GRASP procedure for the 2-path network design. This problem appears often in telecommunication network design. It aims to identify a minimum cost design that includes a path with at most two edges for each origin-destination pair in the network. The authors made use of the same pC/RS/MPSS parallelization strategy that was applied for quadratic assignment, Steiner, and other problems (e.g., Sections 1.5.1 and 1.8.4). The authors compared several parallel versions of the sequential GRASP on a set of small-sized problem instances. They concluded that the version integrating a two-way path-relinking improvement phase performed best. The results also showed that versions that include the path-relinking improvement phase outperformed a "pure" GRASP.

1.10.3 Mobile Network Design

A core problem in the design of wireless networks is the selection of the best set of radio transmitter locations to maximize the covered area at a minimum cost (or

minimum number of transmitters). The problem may be formulated as a set covering problem.

Calégari *et al.* [20] proposed a co-operative PGA based on the island framework with small subpopulation size. Numerical results on data from a real-life problem, showed that the parallel algorithm performed well, in terms of computational time, in comparison with the sequential genetic algorithm (the speedup was almost linear and the efficiency is high).

Tongcheng and Chundi [152] propose a co-operative coarse-grained parallel genetic algorithm for the same problem, and implemented it on three different topologies: ring, bi-oriented ring, and torus. Each thread run a GA enhanced with a local search applied to 20% of the population after each generation. Migration involves neighboring nodes which exchange their best chromosome. Each process sent its best chromosome to all its neighboring nodes, where it replaced the worst individual. The proposed strategy performed well on the torus topology in terms of solution quality and number of generations to reach a good solution. The authors performed a limited comparison between their algorithm implemented with 16 subpopulation of size 10 and the method proposed by Calégari *et al.* [20] applied to 40 subpopulation of size 4. The results were close with a slight advantage to the latter method. The authors concluded that their approach performed well even when the number of processors and individuals was small.

Towhidul Islam, Thulasiraman, and Thalasiram [159] present a parallel ant colony algorithm for the all-pair routing problem in a wireless network with mobile nodes (MANET). In MANET, nodes are mobile and can instantaneously and dynamically form a network as needed. The topology of the nodes can thus substantially change during a short period of time. This makes the problem of determining the best route for data between all pairs of nodes very difficult. The most significant applications of this type of network appear in the military domain and in major disaster situations. According to our knowledge, this is the first time that a parallel meta-heuristic is applied to the all-pair routing problem (the recent study of Gunes, Sorges, and Bouazizi [72] does not address the all-pair routing problem and does not use parallel computing). The parallel ant colony system proposed by Towhidul Islam, Thulasiraman, and Thalasiram applied a domain-decomposition strategy. The network was decomposed into subgraphs. Each ant was assigned to "its" processor and received a subgraph on which to search for the shortest paths between all pairs of nodes. The ants communicated during their searches. Unfortunately, no details were given on how this communication was realized. It was reported, however, that it required between 93% and 97% of the total runtime. A speedup of 7 could, however, be reached when 10 computers were used.

1.11 THE TRAVELING SALESMAN PROBLEM

A well-known, even outside of scientific circles, and fundamental problem in graph theory and combinatorial optimization, the Traveling Salesman Problem (TSP) may be summarized as follows. Given a set of points and distances between all pairs of

points, find a Hamiltonian tour through all points. The TSP appears prominently in many applications, transportation, distribution, logistics, telecommunications, production planning, and so on. The TSP is NP-Hard. In recent years, however, advances in mixed-integer programmoing have resulted in codes that may address very large problem instances. Yet, it is still interesting to survey the parallel meta-heuristic field, since the TSP has offerred a rich environment for developments, many of which were later applied to other problems. Moreover, it often appears as a subproblem in many applications where meta-heuristics are the solution method of choice.

The next subsections examine parallel meta-heuristics for the TSP according to the basic methodology. The last subsection group the contributions to a variant of the problem, the Traveling Purchaser Problem.

1.11.1 Parallel Simulated Annealing

Felten, Karlin, and Otto [55] proposed a 1C/KS/MPSS strategy based on the domain decomposition idea, and experimented with 64 cities using up to 64 processors of a hypercube computer. An initial tour was randomly generated and partitioned into p subsets of adjacent cities. Each subset was then assigned to a processor, which performed local swaps on adjacent cities for a number of iterations. This was followed by a synchronization phase where cities were rotated among processors. Parallel moves did not interact due to the spatial decomposition of the decision variables. Moreover, each synchronization ensured the integrity of the global state. Hence, there was no error and almost linear speedups were observed.

Similar developments were proposed by Allwright and Carpenter [2], but communication overhead impaired performances. Jeong and Kim [76, 77] improved the performance by working on the implementation to reduce this overhead. Numerical results showed their method to be ten times faster than the previous one. A different approach, based on parallelizing the speculative computation method of Witte, Chamberlain and Franklin [164, 165] was presented by Nabhan and Zomaya [109]. The authors managed to reduce the overhead by communicating only the moves actually performed rather than whole solutions. The method produced modest results, however.

Bevilacqua [16] proposed an adaptive strategy to parallelize simulating annealing methods and used the TSP as benchmark. According to the status of the search, the adaptive strategy executes either k iterations of the sequential SA or $2k$ iterations of an independent multi-search SA, with k a parameter to be calibrated. Computational results indicated the proposed method may yield high quality solutions. For the instances tested, deviation of the order of 1% to 2.5% from the optimal solution were observed.

Sanvicente-Sánchez and Frausto-Solís [129] proposed a pC/C/MPSS strategy that decomposed the the sequential SA along temperature lines. A SA was launched at each temperature. Once its iterations are finished, a SA process broadcast its best solution to all processes working at lower temperature. On receiving a solution, a process verified if it was better than its best. In the affirmative, it restarted its search from the imported solution. It continued, otherwise. The algorithm was tested

on a network of workstations with few machines on a few test problem instances. Computation times were reduced, for the same solution quality, compared to the sequential version. It would be surprising if such an approach could perform on a larger number of processors.

Diekmann, Lüling, and Simon [48] proposed a synchronous pC/KS/MPSS co-operative multi-search. The parallel method performed as well as the sequential SA in terms of solution quality, while achieving a speedup of 85 over 121 processors on a parallel machine (OCCAM-2) with 320 transputers. The authors noticed that the solution quality was independant of the numebr of processors.

Ram, Sreenivas, and Subramaniam [119] proposed two variants of a pC/RS/MPDS synchronous multi-search strategy and applied it to the TSP and the Jop-Shop problem. The first parallel approach was a direct implementation of the strategy according to a master-slave model. Slaves run different simulated annealing threads and synchronized at predefined moments to exchange information. The second parallel approach consisted in executing a sequential genetic algorithm first, and then to start the previous PSA, each thread starting from a different individual (solution) from the last generation of the GA. Experimental tests were conducted on a limited number of processors and the two parallel procedures performed well compared to the sequential SA, in terms of both solution quality and computation time. Using the GA to initiate the parallel search seemed to be beneficial.

Miki *et al.* [99] presented a different SA-GA combination within a parallel scheme. The authors proposed pC/RS independent multi-search strategy, where each thread corresponds to a SA, where an adaptive temperature schedule is controled by a genetic algorithm. Computational results showed good speedups (from 20 to 26.60 on 32 processors).

1.11.2 Parallel Genetic Algorithms

1.11.2.1 Fine-grained Methods Mühlenbein, Gorges-Schleuter, and Krämer [106, 107] proposed a fine-grained parallel GA for the TSP. The individuals of the population were placed on a planar grid, each on a processor. Consequently, each individual had thus 4 neighbours and neighbourhoods overlapped allowing diffusion of informaiton. This work was part of the effort to study the introduction into the GA framework of individual enhancement procedures (a fast version of the 2-opt local search procedure in this case).

Knight and Wainwright [80] and Mutalik *et al.* [108] proposed a pC/KS/MPSS coarse-grained co-operative GA, called HYPERGEN, where an initial population was evenly distributed among the processors of a hypercube. Migration occurred at predetermined moments. The proposed algorithm was tested for few (three) TSP test problems, but with different parameter settings: population size, migration rate, and interval migration. The authors observed that, on their test problems, implementations with large subpopulations needed less interaction (migration exchanges) than those with smaller subpopulations, for equivalent solution quality. The achieved solution quality was comparable to the best known solutions at the time.

Logar, Corwin, and English [92] developed a massively parallel genetic algorithm on a MasPar MP-1 parallel computer for the TSP. The 2048 processors of the SIMD machine were arranged in a toroidal grid. Each processor had 16 KB memory and 8 neighboring processors. The authors used all 2048 processors to created an equal number of islands with 10 individuals each. They compared different migration strategies for their pC/MPSS method. The variant that triggered migration at each generation offered the best performance with a no-so-impressive speedup of 145.

Chen, Flann and Watson [26] proposed a massively parallel algorithm for a GA that included a simulated annealing method to determine the surviving offspring. The method is similar to a cellular GA, except that mating is not limited to neighbouring individuals. According to the authors, one of the most striking remarks was the inversely proportional relationship between the number of processors and the time needed to reach a near-optimal solution.

Kohlmorgern , Schmeck, and Haase [81] focused on studies regarding the capability of different processor topologies to prevent demes of fine-grained parallel genetic algorithms to be dominated by the genotype of strong individuals. The population was distributed, one individual per processor, on the processors of a massively parallel computer 16k processing elements. p subpopulations of 1, 4, 16, 64, 256, and 1024 were defined for the purpose of the study. The authors noticed that solution quality increased with the number of subpopulations. They also reported good speedups, but slow convergence.

1.11.2.2 Coarse-grained Methods Sena, Megherbi, and Isern [132] discussed a parallel implementation of PGA that combined strict synchronization and population decomposition ideas. A master-slave platform was used to realize this implementation. Slaves run the same GA on subpopulations that were randomly generated initially. After every generation, slaves sent their best solutions to the master. The master synchronized and controlled the operations. It stooped the search when the best solution reached a certain threshold. The master could also stop or let continue slaves evolve their populations, depending on the quality of solution received. Experimental results pointed to relations between the population size and the number of slave GA processes. Indeed, an optimum combination seemed to exist between these two parameters for best performance. The authors also noticed that as the size of the population increases, the performance of the proposed PGA improved with an increase in the number of slaves. Performances were poor for small population sizes due to communication overhead.

Baraglia, Hidalgo, and Perego [8] started from a GA enhanced with a Lin-Kernighan heuristic for “educating" new individuals and proposed a coarse-grained parallelization. The population was partitioned into a few subpopulations, each of which evolved in parallel on different processors. An elitist migration occurred at predefined period between all subpopulations. According to the authors, the proposed algorithm performed well on a number of TSP benchmarks. It required less iterations to reach the optimal solution than the sequential version.

Katayama, Hibarayashi, and Narihisa [78, 79] studied the impact on island-based parallel genetic algorithms of crossover and selections operators. The pC/KS/MPSS

strategy had the initial population partitioned into subpopulations and the same local search (2-opt) enhance GA run starting from each. Co-operation was performed by elit migration at fixed intervals defined as a number of iterations. Numerical results (first paper) showed that the proposed PGA was robust relative to the three crossover operators MPX, ERX, and CSEX, but that the PGA using CSEX dominated the others. The results (second paper) also showed that the performance of the parallel approach is more sensitive to the choice of crossover operator than to that of the selection operators.

1.11.3 Tabu search

Malek *et al.* [93] proposed a pC/KS/MPDS co-operative parallel strategy where the individual search threads were tightly synchronized. The authors implemented acording to a master-slave model. Their method proceeded with one master that controled the co-operation and four slaves that run serial tabu search algorithms with different tabu conditions and parameters. Slaves were stopped after a specified time interval, solutions were compared, bad areas of solution space were eliminated, and the searches were restarted with a good solution and an empty tabu list. Note that long-term diversification memories were disabled in order to strictly implement this strategy. This implementation was part of a comparative study of serial and parallel simulated annealing and tabu search algorithms for the TSP. The authors reported that the parallel tabu search implementation outperformed the serial one and consistently produced comparable or better results than sequential or parallel simulated annealing.

Chakrapani and Skorin-Kapov [23] proposed the same general 1C/RS/SPSS strategy used for the QAP (Chakrapani and Skorin-Kapov [22, 24, 25]; Section 1.8.1) based on distributing the neighbourhood evaluation. The Connection Machine was again used for experimentation. The authors reported that, for the same quality of solution as the sequential version, near-linear speedups were achieved using a relatively small number of processors.

De Falco *et al* [47] also applied to the TSP their pC/KS co-operative approach used for QAP. Recall that the authors implemented a multi-thread strategy, where each process performed a local search from its best solution. When done, processes synchronized to exchange best solutions with processes that run on neighbouring processors. Local best solutions were replaced with imported ones only if the latter were better. The authors indicated that better solutions were obtained when co-operation was included compared to an independent thread strategy. Super-linear speedups were reported.

Fiechter [56] proposed pC/KS/MPSS co-operative method for the TSP that included an intensification phase during which each process optimizes a specific slice of the tour. At the end of the intensification phase, processes synchronized to recombine the tour and modify (shift part of the tour to a predetermined neighbouring process) the partition. To diversify, each process determined from among its subset of cities a candidate list of most promising moves. The processes then synchronized to exchange these lists, so that all build the same final candidate list and apply the same moves. A master-slave model was used for implementation. Fiechter reported

near-optimal solutions to large problems (500, 3000, and 10000 vertices) and almost linear speedups (less so for the 10000 vertex problems).

1.11.4 Parallel Ant Colony Systems

Stutzle [140] implemented an independent multi-search pC/RS/MPSS approach, where each thread was an enhanced ant colony system (Stutzle and Hoos 1997). For a similar total run time, i.e., the parallel method run for the time of the sequential method divided by the number of processors, this approach achieved high performance on a number of TSP instances.

Bullnheimer, Kotsis, and Strauß [19] presented for the TSP two parallel ant colony systems: synchronous and partially asynchronous algorithms. The 1p/KS parallel synchronous method was implemented in a master-slave platform, where each slave held one ant. After each iteration, each slave sent its tour and the trace trail to the master. After updating the trails, the master sent the new information back to the slaves, which restarted the seach. In order to reduce the communication overhead, a partially asynchronous strategy was also proposed. In this approach, a certain number of ants was assigned to each slave, which performed their search independently of other slaves. The master triggered the trail updating by a global synchronization of all slaves at regular intervals (number of iterations).

In 2000, Middendorf, Reischle, and Schmeck [98] discussed information exchange strategies for multi-colony ant algorithms. They focused on the impact of information exchange on running time and solution quality. Colonies were of different sizes and each was assigned to one processor arranged in a ring. Synchronization took place after several generations and the best ants were exchanged. This is a very simple co-operation mechanism and, not surprisingly, the authors reported that the solution quality was better when the colonies did not exchange too much information.

Randall and Lewis [120] examined several parallelization strategies, including multi-colony co-operation: Parallel Independent Ant Colonies (pC/RS/MPDS), Parallel Interacting Ant Colonies (pC/KS/MPDS), Parallel Ants (1C/KS), Parallel Evaluation of Solution Elements (1C/RS), and Parallel Combination of Ants and Evaluation of Solution Elements (1C/KS). A simple synchronization with broadcasting was used as co-operation mechanism in the second strategy. The authors selected Parallel Ants for the experiments dedicated to the TSP. he authors implemented this approach according to a master-slave model. Each slave launched an ant search separately from other ants. The master received information from slaves (pheromone, solutions), updated pheromones, and restart ed the search for each ant. The computational tests revealed that the performance, in terms of speedup and efficiency, was acceptable for larger problems (number of cities > 200). However, one of the disadvantages of the approach is the large amount of communication required in maintaining and updating the pheromone matrix.

1.12 VEHICLE ROUTING PROBLEMS

The *vehicle routing problem (VRP)* is one of the central problems in operations research and combinatorial optimization with numerous applications in the fields of transportation, telecommunications, production planning, etc. The VRP may be briefly described as follows. Given one or more depots, a fleet of vehicles, homogeneous or nor, and a set of customers with known or forecast demands, find a set of closed routes, originating and ending at one of the depots, to service all customers at minimum cost, while satisfying vehicle and depot capacity constraints. Other constraints may be added to this core problem, e.g., time restrictions, yielding a rich set of problem variants. Most VRP problems are NP-Hard and exact solution methods address limited-size problem instances only.

1.12.1 Parallel Meta-heuristics for the VRP

Rego and Roucairol [121] proposed a tabu search approach for the VRP based on ejection chains and an independent multi-thread parallel version where each thread used a different set of parameter settings but started from the same solution. The method was implemented in a master-slave setting, where each slave executed a complete sequential tabu search. The master gathered the solutions found by the threads, selected the overall best, and reinitialized the threads for a new search. Low-level parallelism was used to accelerate the move evaluations of the individual searches, as well as in a post-optimization phase. Experiments showed the method to be competitive on a set of standard VRP problems (Christofides, Mingozzi, and Toth, [27]).

Ochi *et al.* [110] (see also Drummond, Ochi, and Vianna [52, 53]) proposed a pC/KS/MPSS coarse-grained PGA based on the island model for the vehicle routing problem with heterogeneous fleet. A petal decomposition procedure was used to build the initial population. The population was then divided into several disjoint subpopulations. Each GA thread evolves a subpopulation and triggers migration when subpopulation renewal is necessary. An island in this case would broadcast its need and receive the best individual of every other island. The incoming individuals would replace the worst individuals of the receiving population. Computational tests show encouraging results in terms of solution quality and computing effort.

Alba and Dorronsoro [1] addressed the VRP in which the routes have to be limited by a predefined travel time and proposed a fine-grained, cellular PGA. The population was arranged in a 2 dimentional toroidal grid, each individual having 4 neighbors. Binary tournament selection was applied when selecting the mate for the first parent. Crossover was applied for these parents, then mutation and local search for the offspring. Two local search procedures were tested, 2-opt and 2-opt+λ-Interchange, with $\lambda \in \{1, 2\}$. Elitist replacement was used. The authors compared their algorithm to classical heuristics, the tabu search of Rochat and Taillard [128], the genetic algorithms of Prins and Taillard [117] and Berger and Barkaoui [15], the ant algorithms of Bullnheimer, Hartl, and Strauß[18] and Reimann, Doerner, and Hartl [122]. Computational results on benchmark problem instances showed

high performance quality for both local search versions. Best performance (solution quality and rapidity) was observed for 2-opt+1-Interchange.

1.12.2 Vehicle Routing with Time Constraints

Also known as the *Vehicle Routing Problem with Time Windows (VRPTW)*, this problem specifies that service at customer sites must take place within given time intervals. Most time constraints specify that service cannot begin before a certain moment (but vehicles may wait "outside", in most cases) and must be over by a given deadline. In soft-constrainted versions, the time limits may be violated at a penalty.

Czech and Czarnas [46] proposed a pC/KS/MPSS co-operative multi-thread PSA implemented on a master-slave platform. The master sent the initial solution to the salves. It was also in charge of controlling the annealing procedure temperature schedule, collecting the best local solution from each slave after n^2 iterations for each temperature level (n was the number of customers), and updating the global best solution. Each slave run a SA algorithm with the same parameters. Each slave j co-operated with slaves $j-1$ and $j+1$ (slave 1 co-operated with slave 2 only) by exchanging best solutions. Co-operation was triggered every n iterations. Computational tests with few (five) showed good performance, in terms of solution quality, compared to the best-known solutions of the Solomon benchmarks.

Berger and Berkaoui [14] presented a low-level parallel hybrid GA that used two population. The first one aimed to minimize of the total traveled distance, while the second aimed to minimize the violation of the time window constraints. A different fitness function was associated with each population. A master-slave platform was applied, where the master controled the execution of the algorithm and coordinated the genetic operations. The slave concurrently executed the reproduction and mutation operators. Computational tests were conducted on a cluster of heterogeneous machines (19 computers). The authors compared their algorithm to the best-known methods in the literature for Solomon's benchmark. Their results showed that the proposed technique was very competitive.

Taillard [143] proposed a pC/KS/MPSS parallel tabu search based on domain decomposition. The domain was partitioned and vehicles were allocated to the resulting regions. Once the initial partition was performed, each subproblem was solved by an independent tabu search. All processors stopped after a number of iterations that varies according to the total number of iterations already performed. The partition was then modified by an information exchange phase, during which tours, undelivered cities, and empty vehicles were exchanged between adjacent processors (corresponding to neighbouring regions). This approach did allow to address successfully a number of problem instances. The synchronization inherent in the design of the strategy hindered its performance, however.

Rochat and Taillard [128] proposed what may be considered as the first fully developed adaptive memory-based approach for the VRPTW. The adaptive memory contained tours of good solutions identified by the tabu search threads. The tours were ranked according to attribute values, including the objective values of their respective solutions. Each tabu search process then probabilistically selected tours

in the memory, constructed an initial solution, improved it, and returned the corresponding tours to the adaptive memory. Despite the fact that it used a rather simple tabu search, this method produced many new best results at publication time. Taillard *et al.* [145] and Badeau *et al.* [5] refined this method by enriching the neighbourhood and the intensification phase and by adding a post-optimization procedure. The authors reported 14 new best solutions for the standard data set (Solomon 1987).

Gehring and Homberger [64] (see also Homberger and Gehring [75]) proposed a pC/KS/MPDS co-operative parallel strategy where concurrent searches were performed with differently configured two-phase meta-heuristics. The first phase tried to minimize the number of vehicles by using an evolutionary meta-heuristic, while the second phase aimed to minimize the total traveled distance by means of a tabu search. The parallel meta-heuristic was initiated on different threads with different starting points and values for the search time available for the first and second search phases. Threads co-operated by exchanging solutions asynchronously through a master process. For now, this approach has produced, on average, the best solution for the Solomon problems with respect to, first, the number of vehicles and, second, the total distance. Results were also presented on larger instances, generated similarly to the original Solomon problems, but varying in size from 200 to 1000 customers. It is worth mentioning, however, that this method is rather time consuming compared to other meta-heuristics, tabu search in particular.

Le Bouthiller and Crainic [85] proposed a central memory pC/KS/MPDS parallel meta-heuristic where several tabu search and genetic algorithm threads co-operate. In this model, the central memory constituted the population common to all genetic threads. Each genetic algorithm had its own parent selection and crossover operators. The offspring were returned to the pool to be enhanced by two tabu search procedures. The central memory followed the same rules as in the work of Crainic and Gendreau [35]. Experimental results show that without any particular calibration, the parallel meta-heuristic obtained solutions whose quality is comparable to the best meta-heuristics available, in almost linear speedups.

1.12.3 Dynamic Problems

Gendreau, Laporte, and Semet [66] addressed the deployment problem for a fleet of emergency vehicles and proposed a parallel tabu search based on domain decomposition. A master-slave implementation was performed where each slave addressed a sub-problem associated to a vehicle. Computation tests showed high solution quality as indicated by territory coverage measures.

Attanasio *et al.* [3] addressed the multi-vehicle dial-a-ride-problem and proposed two parallel strategies based on a multi-thread tabu search. a pC/C/SPDS and pc/C/MPSS strategies. In the pC/C/SPMS approach, each processor run a different tabu search strategy from the same initial solution. Once a processor found a new best solution, it broadcast it. Re-initilization searches were then launched. Every κ iterations, a diversification procedure was applied to the first half of the processors, while an intensification was run on the remaining ones. The pC/KS/MPSS strategy consists of running various tabu search algorithms from different starting points.

Each processor run the same tabu search algorithm with the best known parameter settings. Moreover, every η iterations, processors exchanged information in order to perform a diversification procedure. According to the computational results, both the pC/C/SPMS and pC/C/MPSS strategies outperformed the sequential tabu search of Cordeau and Laporte [32].

Gendreau *et al.* [65] proposed a co-operative multi-thread parallel tabu search method for real-time routing and vehicle dispatching problems. The authors followed and adaptive memory approach. In an interesting development, the authors also exploited parallelism within each search thread by decomposing the set of routes along the same principles proposed in Taillard's work [143]. Very good results were obtained. This line of research is continuing very strongly.

1.13 SUMMARY

We have presented a survey of parallel meta-heuristic methods applied to a rather broad set of problems: graph coloring and partitioning, Steiner tree problems, set covering and partitioning, satisfiability and max-sat problems, quadratic assignment, location and network design, traveling salesman and vehicle routing problems.

This survey it is certainly not comprehensive. Important topics could not be covered and not all published contributions in the topics covered could be surveyed. The scope of the chapter is sufficiently broad, however, to allow us to draw some conclusions and share a number of thoughts on the subject of parallel meta-heuristic applications.

The survey illustrates the richness of contributions to the development of parallel meta-heuristics as well as that of their applications to many important problems in science and practice. It also illustrate the fact that this richness notwithstanding, one finds only a somewhat limited number of fundamental principles regarding how to design parallel meta-heuristic procedures. We summarized these principles in the taxonomy section presented at the beginning of the chapter. To sum up, it appears that asynchronous co-operation enhances the performance of parallel meta-heuristics independently of the methodology used in the initial sequential method. This conclusion is strongly supported by the results obtained by multi-thread co-operative strategies.

The survey also illustrate that not all application fields have been studied with comparable fervor. Indeed, many important topics have seen only a few of contributions. Even for topics for which the number of contributions is more numerous, these are not evenly distributed among meta-heuristic classes. Without trying to completely explain the phenomenon, one may observe correlations between the methodologies selected and the scientific field of most of the researchers that have addressed it. Interesting research avenues and promising developments may thus go unexplored, and appropriate tools may be missing in some areas. It should be a challenge of the profession to explore as comprehensively as possible as many problem types as possible. While taking up this challenge, one should make sure that methods are compared

across methodological approaches and that such comparisons are performed fairly, that is, all algorithmic developments are at the same level of sophistication.

To conclude, parallel meta-heuristics offer versatile and powerful tools to address large and complex problems. Many fascinating research avenues are open. Some address issues related to the design of parallel meta-heuristics. Others concern the application of these designs to specific problems and the selection of the most appropriate. We hope that this chapter has contributed to illustrate these opportunities and challenges.

Acknowledgments

Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada, and by the Fonds FQRNT of the Province of Québec.

REFERENCES

1. Alba, E. and Dorronsoro, B. Solving the vehicle routing problem by using cellular genetic algorithms. In *EvoCOP*, pages 11–20, 2004.

2. Allwright, J.R.A. and Carpenter, D.B. A Distributed Implementation of Simulated Annealing for the Travelling Salesman Problem. *Parallel Computing*, 10:335–338, 1989.

3. Attanasio, A., Cordeau, J.F., Ghiani, G., and Laporte, G. Parallel tabu search heuristics for the dymanic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30:377–387, 2004.

4. Azencott, R. *Simulated Annealing Parallelization Techniques*. John Wiley & Sons, New York, NY, 1992.

5. Badeau, P., Guertin, F., Gendreau, M., Potvin, J.-Y., and Taillard, É.D. A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. *Transportation Research C: Emerging Technologies*, 5(2):109–122, 1997.

6. Banos, R., Gil, C., Ortega, J., and Montoya, F.G. A Parallel Multilevel Metaheuristic for Graph Partitioning. *Journal of Heuristics*, 10(4):315–336, 2004.

7. Banos, R., Gil, C., Ortega, J., and Montoya, F.G. Parallel Heuristic Search in Multilevel Graph Partitioning. In *Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 88–95, 2004.

8. Baraglia, R., Hidalgo, J.I., and Perego, R. A Parallel Hybrid Heuristic for the TSP. In Boers, E.J.W., Cagnoni, S., Gottlieb, J., Hart, E., Lanzi, P.L., Günther, R., Smith, R., and Tijink, H., editors, *Applications of Evolutionary Computing. Proceeding of EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM*, volume

2037 of *Lecture Notes in Computer Science*, pages 193–202. Springer-Verlag, Heidelberg, 2001.

9. Barr, R.S. and Hickman, B.L. Reporting Computational Experiments with Parallel Algorithms: Issues, Measures, and Experts Opinions. *ORSA Journal on Computing*, 5(1):2–18, 1993.

10. Bastos, M.P. and Ribeiro, C.C. Reactive Tabu Search with Path-Relinking for the Steiner Problem in Graphs. In S. Voß, S. Martello, C. Roucairol, and Osman, I.H., editors, *Meta-Heuristics 98: Theory & Applications*, pages 31–36. Kluwer Academic Publishers, Norwell, MA, 1999.

11. Battiti, R. and Tecchiolli, G. Parallel Based Search for Combinatorial Optimization: Genetic Algorithms and TABU. *Microprocessors and Microsystems*, 16(7):351–367, 1992.

12. Battiti, R. and Tecchiolli, G. The Reactive Tabu Search. *ORSA Journal on Computing*, 6(2):126–140, 1994.

13. Beasley, J.E. Randomized Heuristic Schemes for the Set Covering Problem. *Naval research Logistics*, 37:151–164, 1990.

14. J. Berger and M. Barkaoui. A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows. *Computers & Operations Research*, 31(12):2037–2053, 2004.

15. Berger, J. and Barkaoui, M. A hybrid genetic algorithm for the capacitated vehicle routing problem. In E. Cantu-Paz, editor, *GECCO03*, pages 646–656. Springer-Verlag, 2003.

16. Bevilacqua, A. A Methodological Approach to Parallel Simulated Annealing. *Journal of Parallel and Distributed Computing*, 62:1548–1570, 2002.

17. Boissin, N. and Lutton, J.L. A Parallel Simulated Annealing Algorithm. *Parallel Computing*, 19(8):859–872, 1993.

18. Bullnheimer, B., Hartl, R., and Strauß, C. An Improved Ant System Algorithm for the Vehicle Routing Problem. *Annals of Operations Research*, 89:319–328, 1999.

19. Bullnheimer, B., Kotsis, G., and Strauß, C. Parallelization strategies for the ant system. *Applied Optimization*, 24:87–100, 1998.

20. Calégari, P. , Guidec, F., Kuonen, P., and Kuonen, D. Parallel Island-Based Genetic Algorithm for Radio Network Design. *Journal of Parallel and Distributed Computing*, 47(1):86–90, 1997.

21. Cantú-Paz, E. A Survey of Parallel Genetic Algorithms. *Calculateurs Parallèles, Réseaux et Systèmes répartis*, 10(2):141–170, 1998.

22. Chakrapani, J. and Skorin-Kapov, J. A Connectionist Approach to the Quadratic Assignment Problem. *Computers & Operations Research*, 19(3/4):287–295, 1992.

23. Chakrapani, J. and Skorin-Kapov, J. Connection Machine Implementation of a Tabu Search Algorithm for the Traveling Salesman Problem. *Journal of Computing and Information Technology*, 1(1):29–36, 1993.

24. Chakrapani, J. and Skorin-Kapov, J. Massively Parallel Tabu Search for the Quadratic Assignment Problem. *Annals of Operations Research*, 41:327–341, 1993.

25. Chakrapani, J. and Skorin-Kapov, J. Mapping Tasks to Processors to Minimize Communication Time in a Multiprocessor System. In *The Impact of Emerging Technologies of Computer Science and Operations Research*, pages 45–64. Kluwer Academic Publishers, Norwell, MA, 1995.

26. Chen, H., Flann, N.S., and Watson, D.W. Parallel Genetic Simulated Annealing: A Massively Parallel SIMD Algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 9(2):126–136, 1998.

27. Christofides, N., Mingozzi A., and Toth, P. The Vehicle Routing Problem. In N. Christofides, Mingozzi A., P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. John Wiley, New York, 1979.

28. Cohoon, J., Hedge, S., Martin, W., and Richards, D. Punctuated Equilibria: A Parallel Genetic Algorithm. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, pages 148–154. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

29. Cohoon, J., Martin, W., and Richards, D. Genetic Algorithm and Punctuated Equilibria in VLSI. In Schwefel, H.-P. and Männer, R., editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 134–144. Springer-Verlag, Berlin, 1991a.

30. Cohoon, J., Martin, W., and Richards, D. A Multi-Population Genetic Algorithm for Solving the k-Partition Problem on Hyper-Cubes. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 134–144. Morgan Kaufmann, San Mateo, CA, 1991b.

31. Collins, R.J. and Jefferson, D.R. Selection in Massively Parallel Genetic Algorithms. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 249–256. Morgan Kaufmann, San Mateo, CA, 1991.

32. Cordeau, J.F. and G. Laporte, G. A Tabu Search Heuristics for the Static multi-vehicle Dial-a-ride Problem. *Transportation Research Part B*, pages 579–594, 2003.

33. Crainic, T.G. Parallel Computation, Co-operation, Tabu Search. In C. Rego and B. Alidaee, editors, *Adaptive Memory and Evolution: Tabu Search and Scatter Search*. Kluwer Academic Publishers, Norwell, MA, 2004.

34. Crainic, T.G. and Gendreau, M. Towards an Evolutionary Method - Cooperating Multi-Thread Parallel Tabu Search Hybrid. In S. Voß, S. Martello, C. Roucairol, and Osman, I.H., editors, *Meta-Heuristics 98: Theory & Applications*, pages 331–344. Kluwer Academic Publishers, Norwell, MA, 1999.

35. Crainic, T.G. and Gendreau, M. Cooperative Parallel Tabu Search for Capacitated Network Design. *Journal of Heuristics*, 8(6):601–627, 2002.

36. Crainic, T.G., Gendreau, M., Hansen, P., and Mladenović, N. Cooperative Parallel Variable Neighborhood Search for the p-Median. *Journal of Heuristics*, 10(3):293–314, 2004.

37. Crainic, T.G. and Toulouse, M. Parallel Metaheuristics. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 205–251. Kluwer Academic Publishers, Norwell, MA, 1998.

38. Crainic, T.G. and Toulouse, M. Parallel Strategies for Meta-heuristics. In F. Glover and G. Kochenberger, editors, *Handbook in Metaheuristics*, pages 475–513. Kluwer Academic Publishers, Norwell, MA, 2003.

39. Crainic, T.G., Toulouse, M., and Gendreau, M. Parallel Asynchronous Tabu Search for Multicommodity Location-Allocation with Balancing Requirements. *Annals of Operations Research*, 63:277–299, 1995.

40. Crainic, T.G., Toulouse, M., and Gendreau, M. Synchronous Tabu Search Parallelization Strategies for Multicommodity Location-Allocation with Balancing Requirements. *OR Spektrum*, 17(2/3):113–123, 1995.

41. Crainic, T.G., Toulouse, M., and Gendreau, M. Towards a Taxonomy of Parallel Tabu Search Algorithms. *INFORMS Journal on Computing*, 9(1):61–72, 1997.

42. Crainic, T.G., Toulouse, M., and Li, Y. A Simple Cooperative Multilevel Algorithm for the Capacitated Multicommodity Network Design. Publication CRT-2004-, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 2004.

43. Cung, V.-D., Martins, S.L., Ribeiro, C.C., and Roucairol, C. Strategies for the Parallel Implementations of Metaheuristics. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 263–308. Kluwer Academic Publishers, Norwell, MA, 2002.

44. Cung, V.-D., Mautor, T., Michelon, P., and Tavares, A. A Scatter Search Based Approach for the Quadratic Assignment Problem on Evolutionary Computation and Evolutionary Programming. In T. Baeck, Z. Michalewicz, and X. Yao,

editors, *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 165–170. IEEE Press, 1997.

45. Czech, Z.J. A Parallel Genetic Algorithm for the Set Partitioning Problem. In *8th Euromicro Workshop on Parallel and Distributed Processing*, pages 343–350, 2000.

46. Czech, Z.J. and Czarnas, P. Parallel simulated annealing for the vehicle routing problem with time windows. In *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 376–383, 2002.

47. De Falco, I., Del Balio, R., Tarantino, E., and Vaccaro, R. Improving Search by Incorporating Evolution Principles in Parallel Tabu Search. In *Proceedings International Confonference on Machine Learning*, pages 823–828, 1994.

48. R. Diekmann, R. Lüling, and J. Simon. Problem Independent Distributed Simulated Annealing and its Applications. In R.V.V. Vidal, editor, *Lecture Notes in Economics and Mathematical Systems*, volume 396, pages 17–44. Springer Verlag, Berlin, 1993.

49. Diekmann, R., Lüling, R., Monien, B., and Spräner, C. Combining Helpful Sets and Parallel Simulated Annealing for the Graph-Partitioning Problem. *International Journal of Parallel Programming*, 8:61–84, 1996.

50. Drias, H. and Ibri , A. Parallel ACS for Weighted MAX-SAT. In Mira, J. and Álvarez, J., editors, *Artificial Neural Nets Problem Solving Methods - Proceedings of the 7th International Work-Conference on Artificial and Natural Neural Networks*, volume 2686 of *Lecture Notes in Computer Science*, pages 414–421. Springer-Verlag, Heidelberg, 2003.

51. Drias, H. and Khabzaoui, M. Scatter Search with Random Walk Strategy for SAT and Max-SAT Problems. In L. Monostori, Váncza, J., and A. Moonis, editors, *Proceedings of the 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2001I*, pages 35–44. Springer-Verlag, 2001.

52. Drummond, L.M.A., Ochi, L.S., and Vianna, D.S. A parallel hybrid evolutionary metaheuristic for the period vehicle routing problem. volume 1586 of *Lecture Notes in Computer Science*, pages 183–191. 1999.

53. Drummond, L.M.A., Ochi, L.S., and Vianna, D.S. An asynchronous parallel metaheuristic for the period vehicle routing problem. *Future Generation Computer Systems*, 17:379–386, 2001.

54. Durand, M.D. Parallel Simulated Annealing: Accuracy vs. Speed in Placement. *IEEE Design & Test of Computers*, 6(3):8–34, 1989.

55. Felten, E., Karlin, S., and Otto, S.W. The Traveling Salesman Problem on a Hypercube, MIMD Computer. In *Proc. 1985 of the Int. Conf. on Parallel Processing*, pages 6–10, 1985.

56. Fiechter, C.-N. A Parallel Tabu Search Algorithm for Large Travelling Salesman Problems. *Discrete Applied Mathematics*, 51(3):243–267, 1994.

57. Fiorenzo Catalano, M.S. and Malucelli, F. Randomized Heuristic Schemes for the Set Covering Problem. In M. Paprzyky, L. Tarricone, and T. Yang, editors, *Practical Applications of Parallel Computing*, pages 23–38. Nova Science, 2003.

58. Flores, S.D., Cegla, B.B., and Caceres, D.B. Telecommunication Network Design with Parallel Multi-objective Evolutionary Algorithms. In *IFIP/ACM Latin America Networking Conference 2003*, 2003.

59. Folino, G., Pizzuti, C., and Spezzano, G. Combining Cellular Genetic Algorithms and Local Search for Solving Satisfiability Problems. In *Proceedings of the Tenth IEEE International Conference on Tools with Artificial Intelligence*, pages 192–198. IEEE Computer Society Press, 1998.

60. Folino, G., Pizzuti, C., and Spezzano, G. Solving the Satisfiability Problem by a Parallel Cellular Genetic Algorithm. In *Proceedings of the 24th EUROMICRO Conference*, pages 715–722. IEEE Computer Society Press, 1998.

61. Folino, G., Pizzuti, C., and Spezzano, G. Parallel Hybrid Method for SAT that Couples Genetic Algorithms and Local Search. *IEEE Transactions on Evolutionary Computation*, pages 323–334, 2001.

62. García-López, F., Melián-Batista, B., Moreno-Pérez, J.A., and Moreno-Vega, J.M. The Parallel Variable Neighborhood Search for the p-Median Problem. *Journal of Heuristics*, 8(3):375–388, 2002.

63. García-López, F., Melián-Batista, B., Moreno-Pérez, J.A., and Moreno-Vega, J.M. Parallelization of the Scatter Search for the p-Median Problem. *Parallel Computing*, 29:575–589, 2003.

64. Gehring, H. and Homberger, J. A parallel two-phase metaheuristic for routing problems with time windows. *Asia-Pacific Journal of Operational Research*, 18(1):35–47, 2001.

65. Gendreau, M., Guertin, F., Potvin, J.-Y., and Taillard, É.D. Tabu Search for Real-Time Vehicle Routing and Dispatching. *Transportation Science*, 33(4):381–390, 1999.

66. Gendreau, M., Laporte , G., and Semet, F. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Computing*, 27(12):1641–1653, 2001.

67. Gendron, B., Potvin, J.-Y., and Soriano, P. A Parallel Hybrid Heuristic for the Multicommodity Capacitated Location Problem with Balancing Requirements. *Parallel Computing*, 29:591–606, 2003.

68. Ghamlouche, I., Crainic, T.G., and Gendreau, M. Cycle-based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design. *Operations Research*, 51(4):655–667, 2003.

69. Glover, F. and Laguna, M. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, 1997.

70. Greening, D.R. Asynchronous Parallel Simulated Annealing. *Lectures in Complex Systems*, 3:497–505, 1990.

71. Greening, D.R. Parallel Simulated Annealing Techniques. *Physica D*, 42:293–306, 1990.

72. Gunes, M., Sorges, U., and Bouazizi, I. ARA - The Ant Colony Based Routing Algorithm for MANETs. In *Proceedings of the International Conference on Parallel Processing*, pages 79–85, 2002.

73. Hidalgo, J.I., Prieto, M., Lanchares, J., Baraglia, R., Tirado, F., and Garnica, O. Hybrid Parallelization of a Compact Genetic Algorithm. In *Proceedings of the 11th uromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 449–455, 2003.

74. Holmqvist, K., Migdalas, A., and Pardalos, P.M. Parallelized Heuristics for Combinatorial Search. In A. Migdalas, P.M. Pardalos, and S. Storoy, editors, *Parallel Computing in Optimization*, pages 269–294. Kluwer Academic Publishers, Norwell, MA, 1997.

75. Homberger, J. and Gehring, H. Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows. *INFOR*, 37:297–318, 1999.

76. Jeong, C.-S. and Kim, M.-H. Parallel Algorithm for the TSP on SIMD Machines Using Simulated Annealing. In *Proceedings of the International Conference on Application Specific Array Processors*, pages 712–721, 1990.

77. Jeong, C.-S. and Kim, M.-H. Fast Parallel Simulated Annealing Algorithm for TSP on SIMD Machines with Linear Interconnections. *Parallel Computing*, 17:221–228, 1991.

78. Katayama, K., Hirabayashi, H., and Narihisa, H. Performance Analysis for Crossover Operators of Genetic Algorithm. *Systems and Computers in Japan*, 30:20–30, 1999.

79. Katayama, K., Hirabayashi, H., and Narihisa, H. Analysis of Crossovers and Selections in a Coarse-grained Parallel Genetic Algorithm. *Mathematical and Computer Modelling*, 38:1275–1282, 2003.

80. Knight, R.L. and Wainwright, R.L. HYPERGEN" - A Distributed Genetic Algorithm on a Hypercube. In *Proceedings of the 1992 IEEE Scalable High*

Performance Computing Conference, pages 232–235. IEEE Computer Society Press, Los Alamitos, CA, 1992.

81. Kohlmorgen, U., Schmeck, H., and Haase, K. Experiences with Fine-grained Parallel Genetic Algorithms. *Annals of Operations Research*, 90:203–219, 1999.

82. Kokosinski, Z., Kolodziej, M., and Kwarciany, K. Parallel Genetic Algorithm for Graph Coloring Problem. In Bubak, M., van Albada, G.D., and Sloot, P.M.A., editors, *International Conference on Computational Science*, volume 3036 of *Lecture Notes in Computer Science*, pages 215–222. SpringerVerlag, Heidelberg, 2004.

83. Laursen, P.S. Problem-Independent Parallel Simulated Annealing Using Selection and Migration. In Davidor, Y., Schwefel, H.-P., and Männer, R., editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 408–417. Springer-Verlag, Berlin, 1994.

84. Laursen, P.S. Parallel Heuristic Search – Introductions and a New Approach. In A. Ferreira and P.M. Pardalos, editors, *Solving Combinatorial Optimization Problems in Parallel, Lecture Notes in Computer Science 1054*, pages 248–274. Springer-Verlag, Berlin, 1996.

85. Le Bouthillier, A. and Crainic, T.G. A Cooperative Parallel Meta-Heuristic for the Vehicle Routing Problem with Time Windows. *Computers & Operations Research*, 32(7):1685–1708, 2005.

86. Lee, K-G. and Lee, S-Y. Efficient Parallelization of Simulated Annealing Using Multiple Markov Chains: An Application to Graph Partitioning. In Mudge, T.N., editor, *Proceedings of the International Conference on Parallel Processing*, volume III: Algorithms and Applications, pages 177–180. CRC Press, 1992a.

87. Lee, K-G. and Lee, S-Y. Synchronous and Asynchronous Parallel Simulated Annealing with Multiple Markov Chains. volume 1027 of *Lecture Notes in Computer Science*, pages 396–408. Springer-Verlag, Berlin, 1995.

88. Lee, S.-Y. and Lee, K.-G. Asynchronous Communication of Multiple Markov Chains in Parallel Simulated Annealing. In Mudge, T.N., editor, *Proceedings of the International Conference on Parallel Processing*, volume III: Algorithms and Applications, pages 169–176. CRC Press, Boca Raton, FL, 1992b.

89. D. Levine. A parallel genetic algorithm for the set partitioning problem. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 23–35. Kluwer Academic Publishers, Norwell, MA, 1996.

90. Li, Y., Pardalos, P.M., and Resende, M.G.C. A Greedy Randomized Adaptive Search Procedure for Quadratic Assignment Problem. In *DIMACS Implementation Challenge, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, volume 16, pages 237–261. American Mathematical Society, 1994.

91. Lin, S.-C., Punch, W., and Goodman, E. Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach. In *Sixth IEEE Symposium on Parallel and Distributed Processing*, pages 28–37. IEEE Computer Society Press, 1994.

92. Logar, A.M., Corwin, E.M., and English, T.M. Implementation of Massively Parallel Genetic Algorithms o the MasPar MP-1. In *Proceedings of the 1992 IEEE ACM/SIGAPP Symposium on Applied Computing: Technological Challenges of the 1990's*, pages 1015–1020. ACM Press, Kansas City, Missouri, 1992.

93. Malek, M., Guruswamy, M., Pandya, M., and Owens, H. Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, 21:59–84, 1989.

94. Marchiori, E. and Rossi, C. A Flipping Genetic Algorithm for Hard 3-SAT Problems. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., and Smith, R.E., editors, *Proceedings of the Genetic Evolutionary Computation Conference*, pages 393–400. Morgan Kaufmann, San Mateo, CA, 1999.

95. Martins, S.L., Resende, M.G.C., Ribeiro, C.C., and Parlados, P.M. A Parallel Grasp for the Steiner Tree Problem in Graphs Using a Hybrid Local Search Strategy. *Journal of Global Optimization*, 17:267–283, 2000.

96. Martins, S.L., Ribeiro, C.C., and Souza, M.C. A Parallel GRASP for the Steiner Problem in Graphs. In A. Ferreira and J. Rolim, editors, *Proceedings of IRREGULAR'98 – 5th International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 285–297. Springer-Verlag, 1998.

97. T. Maruyama, T. Hirose, and A. Konagaya. A Fine-Grained Parallel Genetic Algorithm for Distributed Parallel Systems. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 184–190. Morgan Kaufmann, San Mateo, CA, 1993.

98. Middendorf, M., Reischle, F., and Schmeck, H. Information exchange in multi colony ant algorithms. volume 1800 of *Lecture Notes in Computer Science*. Springer-Verlag, Heidelberg, 2000.

99. Miki, M., Hiroyasu, T., Wako, J., and Yoshida, T. Adaptive Temperature Schedule Determined by Genetic Algorithm for Parallel Simulated Annealing. In *CEC'03 - The 2003 Congress on Evolutionary Computation*, volume 1, pages 459–466, 2003.

100. Mitchell, D., Selman, B., and Levesque, H. Hard and Easy Distribution of SAT Problems. In Rosenbloom, P. and Szolovits, P., editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 459–465. AAAI Press, Menlo Park, CA, 1992.

101. H. Mühlenbein. Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421. Morgan Kaufmann, San Mateo, CA, 1989.

102. Mühlenbein, H. Evolution in Time and Space - The Parallel Genetic Algorithm. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithm & Classifier Systems*, pages 316–338. Morgan Kaufman, San Mateo, CA, 1991.

103. Mühlenbein, H. Asynchronous Parallel Search by the Parallel Genetic Algorithm. In V. Ramachandran, editor, *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing*, pages 526–533. IEEE Computer Society Press, Los Alamitos, CA, 1991c.

104. Mühlenbein, H. Parallel Genetic Algorithms in Combinatorial Optimization. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research: New Developments in their Interface*, pages 441–456. Pergamon Press, New York, NY, 1992.

105. Mühlenbein, H. How Genetic Algorithms Really Work: Mutation and Hill-Climbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 15–26. North-Holland, Amsterdam, 1992a.

106. Mühlenbein, H., Gorges-Schleuter, M., and Krämer, O. New Solutions to the Mapping Problem of Parallel Systems - the Evolution Approach. *Parallel Computing*, 6:269–279, 1987.

107. Mühlenbein, H., Gorges-Schleuter, M., and Krämer, O. Evolution Algorithms in Combinatorial Optimization. *Parallel Computing*, 7(1):65–85, 1988.

108. Mutalik, P. P., Knight, L. R., Blanton, J. L., and Wainwright, R. L. Solving Combinatorial Optimization Problems Using Parallel Simulated Annealing and Parallel Genetic Algorithms. In *Proceedings of the 1992 IEEE ACM/SIGAPP Symposium on Applied Computing: Technological Challenges of the 1990's*, pages 1031–1038. ACM Press, Kansas City, Missouri, 1992.

109. Nabhan, T.M. and Zomaya, A.Y. A Parallel Simulated Annealing Algorithm with Low Communication Overhead. *IEEE Transactions on Parallel and Distributed Systems*, 6(12):1226–1233, 1995.

110. Ochi, L.S., Vianna, D.S., Drummond, L.M.A., and Victor, A.0. A Parallel Evolutionary Algorithm for the Vehicle Routing Problem with Heterogeneous Fleet. *Future Generation Computer Systems*, 14(3):285–292, 1998.

111. Ouyang, M., Toulouse, M., Thulasiraman, K., Glover, F., and Deogun, J.S. Multi-Level Cooperative Search: Application to the Netlist/Hypergraph Partitioning Problem. In *Proceedings of International Symposium on Physical Design*, pages 192–198. ACM Press, 2000.

112. Ouyang, M., Toulouse, M., Thulasiraman, K., Glover, F., and Deogun, J.S. Multilevel Cooperative Search for the Circuit/Hypergraph Partitioning Problem. *IEEE Transactions on Computer-Aided Design*, 21(6):685–693, 2002.

113. Pardalos, P.M., Li, Y., and Murthy, K.A. Computational Experience with Parallel Algorithms for Solving the Quadratic Assignment Problem. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research: New Developments in their Interface*, pages 267–278. Pergamon Press, New York, NY, 1992.

114. Pardalos, P.M., L. Pitsoulis, T. Mavridou, and Resende, M.G.C. Parallel Search for Combinatorial Optimization: Genetic Algorithms, Simulated Annealing, Tabu Search and GRASP. In A. Ferreira and J. Rolim, editors, *Proceedings of Workshop on Parallel Algorithms for Irregularly Structured Problems, Lecture Notes in Computer Science*, volume 980, pages 317–331. Springer-Verlag, Berlin, 1995.

115. Pardalos, P.M., Pitsoulis, L., and Resende, M.G.C. A Parallel GRASP Implementation for the Quadratic Assignment Problem. In A. Ferreira and J. Rolim, editors, *Solving Irregular Problems in Parallel: State of the Art*, pages 115–130. Kluwer Academic Publishers, Norwell, MA, 1995.

116. L. Pitsoulis, Pardalos, P.M., and Resende, M.G.C. A Parallel GRASP for MAX-SAT. In Wasniewski J., Dongarra, J., Madsen, K., and Olesen, D., editors, *Third International Workshop on Applied Parallel Computing, Industrial Computation and Optimization*, volume 1180 of *Lecture Notes in Computer Science*, pages 575–585. Springer-Verlag, Berlin, 1996.

117. Prins, C. and Taillard, É.D. A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.

118. Rahoual, M., Hadji, R., and Bachelet, V. Parallel Ant System for the Set Covering Problem Source. In *Proceedings of the Third International Workshop on Ant Algorithms*, pages 262–267. Springer-Verlag, London, UK, 2002.

119. Ram, D.J., Sreenivas, T.H., and Subramaniam, K.G. Parallel Simulated Annealing Algorithms. *Journal of Parallel and Distributed Computing*, 37:207–212, 1996.

120. Randall ,M. and Lewis, A. A parallel implementation of ant colony optimisation. *Journal of Parallel and Distributed Computing*, 62:1421–1432, 2002.

121. Rego, C. and Roucairol, C. A Parallel Tabu Search Algorithm Using Ejection Chains for the VRP. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 253–295. Kluwer Academic Publishers, Norwell, MA, 1996.

122. Reimann, M., Doerner, K., and Hartl, R. D-ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem. *Computers & Operations Research*, 31:563–591, 2004.

123. Resende, M.G.C. and Feo, T.A. A GRASP for Satisfiability. In Trick, M.A. and Johnson, D.S., editors, *The Second DIMACS Implementation Challenge, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, volume 26, pages 499–520. American Mathematical Society, 1996.

124. Resende, M.G.C. and Feo, T.A. Approximative Solution of Weighted MAX-SAT Problems Using GRASP. *Discrete Mathematics and Theoretical Computer Science*, 35:393–405, 1997.

125. Ribeiro, C.C. and Rosseti, I. A Parallel GRASP Heuristic for the 2-path Network Design Problem. 4 journée ROADEF, Paris, February 20-22, 2002.

126. Ribeiro C.C. and Rosseti, I. A Parallel GRASP Heuristic for the 2-path Network Design Problem. Third Meeting of the PAREO Euro Working Group, Guadeloupe (Françe), May, 2002.

127. Ribeiro C.C. and Rosseti, I. Parallel grasp with path-relinking heuristic for the 2-path network design problem. AIRO'2002, L'Aquila, Italy, September, 2002.

128. Rochat, Y. and Taillard, É.D. Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1(1):147–167, 1995.

129. Sanvicente-Sánchez, H. and Frausto-Solís, J. . Mpsa: A methodology to parallelize simulated annealing and its application to the traveling salesman problem. volume 2313 of *Lecture Notes in Computer Science*. Springer-Verlag Heidelberg, 2002.

130. Selman, B., Kautz, H. A., and Cohen, B. Noise Strategies for Improving Local Search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 337–343, 1994.

131. Selman, B., Levesque, H., and Mitchell, D. A New Method for Solving Hard Satisfiability Problems. In Rosenbloom, P. and Szolovits, P., editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446. AAAI Press, Menlo Park, CA, 1992.

132. Sena, G.A., Megherbi, D., and Isern, G. Implementation of a Parallel Genetic Algorithm on a Cluster of Workstations: Traveling Salesman Problem, a Case Study. *Future Generation Computer Systems*, 17:477–488, 2001.

133. Shonkwiler, R. Parallel Genetic Algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 199–205. Morgan Kaufmann, San Mateo, CA, 1993.

134. Sleem, A., Ahmed, M., Kumar, A., and Kamel, K. Comparative Study of Parallel vs. Distributed Genetic Algorithm Implementation for ATM Networking

Environment. In *Fifth IEEE Symposium on Computers and Communications*, pages 152–157, 2000.

135. Sohn, A. Parallel Satisfiability Test with Synchronous Simulated Annealing on Distributed Memory Multiprocessor. *Journal of Parallel and Distributed Computing*, 36:195–204, 1996.

136. Sohn, A. and Biswas, R. Satisfiability Tests with Synchronous Simulated Annealing on the Fujitsu AP1000 Massively-parallel Multiprocessor. In *Proceedings of the International Conference on Supercomputing*, pages 213–220, 1996.

137. Solar, M., Parada, V., and Urrutia, R. A Parallel Genetic Algorithm to Solve the Set-Covering Problem. *Computers & Operations Research*, 29(9):1221–1235, 2002.

138. Solomon, M.M. Time Window Constrained Routing and Scheduling Problems. *Operations Research*, 35:254–265, 1987.

139. Spears, W.M. Simulated Annealing for Hard Satisfiability Problems, in Clique, Coloring and Satisfiability. In Johnson, D.S. and Trick, M.A., editors, *Cliques, Coloring, and Satisfiability*, volume 26, pages 533–558. American Mathematical Society, 1996.

140. Stutzle, T. Parallelization Strategies for Ant Colony Optimization. In Eiben, A.E., Back, T., Schoenauer, M., and Schwefel, H.-P., editors, *Proceedings of Parallel Problem Solving from Nature V*, volume 1498 of *Lecture Notes in Computer Science*, pages 722–731. Springer-Verlag, Heidelberg, 1998.

141. Stutzle, T. and Hoos, H. Improvements on the Ant System: Introducing the MAX-MIN Ant System. In Smith, G.D., Steele, N.C., and Albrecht, R.F., editors, *Proceedings of Artificial Neural Nets and Genetic Algorithms*, Lecture Notes in Computer Science, pages 245–249. Springer-Verlag, Heidelberg, 1997.

142. Taillard, É.D. Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computing*, 17:443–455, 1991.

143. Taillard, É.D. Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks*, 23:661–673, 1993.

144. Taillard, É.D. *Recherches itératives dirigées parallèles*. PhD thesis, École Polytechnique Fédérale de Lausanne, 1993.

145. Taillard, É.D., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science*, 31(2):170–186, 1997.

146. Talbi, E-G. and Bessière, P. A Parallel Genetic Algorithm for the Graph Partitioning Problem. In *Proceedings of the ACM International Conference on Supercomputing ICS91*, pages 312–320, 1991a.

147. Talbi, E.-G., Hafidi, Z., and Geib, J.-M. Parallel Adaptive Tabu Search Approach. *Parallel Computing*, 24:2003–2019, 1998.

148. Talbi, E.-G., Hafidi, Z., and Geib, J.-M. Parallel Tabu Search for Large Optimization Problems. In S. Voß, S. Martello, C. Roucairol, and Osman, I.H., editors, *Meta-Heuristics 98: Theory & Applications*, pages 345–358. Kluwer Academic Publishers, Norwell, MA, 1999.

149. Talbi, E.-G., Hafidi, Z., Kebbal, D., and Geib, J.-M. A Fault-Tolerant Parallel Heuristic for Assignment Problems. *Future Generation Computer Systems*, 14:425–438, 1998.

150. Talbi, E.-G., Roux, O., Fonlupt, C., and Robillard, D. Parallel Ant Colonies for Combinatorial Optimization Problems. In J.D.P. Rolim et al., editor, *11th IPPS/SPDP'99 Workshops*, volume 1586 of *Lecture Notes in Computer Science*, pages 239–247. 1999.

151. Talbi, E.-G., Roux, O., Fonlupt, C., and Robillard, D. Parallel Ant Colonies for the Quadratic Assignment Problem. *Future Generation Computer Systems*, 17:441–449, 2001.

152. Tongcheng, G. and Chundi, M. Radio Network Design Using Coarse-Grained Parallel Genetic Algorithms with Different Neighbor Topology. In *Proceedings of the 4th World Congress on Intelligent Control and Automation*, volume 3, pages 1840–1843, 2002.

153. Toulouse, M., Crainic, T.G., and Sansó, B. An Experimental Study of Systemic Behavior of Cooperative Search Algorithms. In S. Voß, S. Martello, C. Roucairol, and Osman, I.H., editors, *Meta-Heuristics 98: Theory & Applications*, pages 373–392. Kluwer Academic Publishers, Norwell, MA, 1999.

154. Toulouse, M., Crainic, T.G., and Sansó, B. Systemic Behavior of Cooperative Search Algorithms. *Parallel Computing*, 21(1):57–79, 2004.

155. Toulouse, M., Crainic, T.G., Sansó, B., and Thulasiraman, K. Self-Organization in Cooperative Search Algorithms. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2379–2385. Omnipress, Madisson, Wisconsin, 1998.

156. Toulouse, M., Crainic, T.G., and Thulasiraman, K. Global Optimization Properties of Parallel Cooperative Search Algorithms: A Simulation Study. *Parallel Computing*, 26(1):91–112, 2000.

157. Toulouse, M., Glover, F., and Thulasiraman, K. A Multi-Scale Cooperative Search with an Application to Graph Partitioning. Report, School of Computer Science, University of Oklahoma, Norman, OK, 1998.

158. Toulouse, M., Thulasiraman, K., and Glover, F. Multi-Level Cooperative Search: A New Paradigm for Combinatorial Optimization and an Application to Graph

Partitioning. In P. Amestoy, P. Berger, M. Daydé, I. Duff, V. Frayssé, L. Giraud, and D. Ruiz, editors, *5th International Euro-Par Parallel Processing Conference*, volume 1685 of *Lecture Notes in Computer Science*, pages 533–542. Springer-Verlag, Heidelberg, 1999.

159. Towhidul Islam, M., Thulasiraman, P., and Thalasiram, R.K. A Parallel Ant Colony Optimization Algorithm for All-Pair Routing in MANETs. In *Proceedings of the International Parallel and Distributed Processing Symposium*, IEEE, 2003.

160. Verhoeven, M.G.A. and Aarts, E.H.L. Parallel Local Search. *Journal of Heuristics*, 1(1):43–65, 1995.

161. Verhoeven, M.G.A. and Severens, M.M.M. Parallel Local Search for Steiner Trees in Graphs. *Annals of Operations Research*, 90:185–202, 1999.

162. Voß, S. Tabu Search: Applications and Prospects. In D.-Z. Du and P.M. Pardalos, editors, *Network Optimization Problems*, pages 333–353. World Scientific Publishing Co., Singapore, 1993.

163. Wilkerson, R. and Nemer-Preece, N. Parallel Genetic Algorithm to Solve the Satisfiability Problem. In *Proceedings of the 1998 ACM symposium on Applied Computing*, pages 23–28. ACM Press, 1998.

164. Witte, E.E., Chamberlain, R.D., and Franklin, M.A. Parallel Simulated Annealing using Speculative Computation. In *Proceedings of the 19th International Conference on Parallel Processing*, pages 286–290, 1990.

165. Witte, E.E., Chamberlain, R.D., and Franklin, M.A. Parallel Simulated Annealing using Speculative Computation. *IEEE Transactions on Parallel & Distributed Systems*, 2(4):483–494, 1991.