High Performance Optimization at the Door of the Exascale

Claude Tadonki

High Performance Optimization at the Door of the Exascale

Claude Tadonki
MINES ParisTech - PSL
Département Mathématiques et Systèmes
Centre de Recherche en Informatique (CRI)
35, rue Saint-Honoré
77305, Fontainebleau-Cedex
claude.tadonki@mines-paristech.fr

June 23, 2021

Abstract

The next frontier of high performance computing is the *Exascale*, and this will certainly stand as a noteworthy step in the quest for processing speed potential. In fact, we always get a fraction of the technically available computing power (so-called *theoretical peak*), and the gap is likely to go hand-to-hand with the hardware complexity of the target system. Among the key aspects of this complexity, we have: the *heterogeneity* of the computing units, the *memory hierarchy and partitioning* including the non-uniform memory access (NUMA) configuration, and the *interconnect* for data exchanges among the computing nodes. Scientific investigations and cutting-edge technical activities should ideally scale-up with respect to sustained performance. The special case of quantitative approaches for solving (large-scale) problems deserves a special focus. Indeed, most of common real-life problems, even when considering the artificial intelligence paradigm, rely on optimization techniques for the main kernels of algorithmic solutions. Mathematical programming and pure combinatorial methods are not easy to implement efficiently on large-scale supercomputers because of *irregular control flow*, *complex memory access patterns*, *heterogeneous kernels*, *numerical issues*, to name a few. We describe and examine our thoughts from the standpoint of large-scale supercomputers.

1 Scientific context

The most notorious computing challenges mainly come from combinatorial problems and their applications. As the power of supercomputers is increasing, large-scale scenarios of common problems are under consideration and are expected to enter into routine. As previously stated, most of these problems can be expressed and solved in the standpoint of optimization, combinatorial and/or numerical. Serious efforts are being made to derive more powerful techniques for combinatorial, numerical, and hybrid optimization. At this point, the word "powerful" refers to the complexity in terms of the number of basic steps or any relevant metric. When moving to an implementation on computers, targeting performance through efficiency turns to be a difficult task, which is exacerbated by the specific complexity and constraints of modern computing systems. The case of linear programming is particularly illustrative of what can appears as disconcerting. Indeed, the traditional *simplex* method, which is known to have an exponential (worst case) complexity yields more efficient implementations than the polynomial *ellipsoid* method. We think that similar facts will come up with large-scale optimization on *exascale* systems. Fundamental methods for solving problems are computer agnostic, thus, implementation efforts mainly try to map an existing method onto a given computing system. A full and consistent optimization solution is likely to be a mix of several distinct components from the computing standpoint. Beside linear and non-linear algebra kernels, there are pure combinatorial modules, all orchestrated by at a higher level following the rules of the global method being so implemented.

2 Technical context

High Performance Computing (HPC) aims at providing powerful computing solutions to scientific and real life problems. Many efforts have been made on the way to faster supercomputers, including generic and customized configurations. The advent of multicore architectures is noticeable in the HPC history, because it has brought the underlying parallel programming concept into common considerations. Based on multicore processors, probably enhanced with acceleration units, current generation of supercomputers is rated to deliver an increasing peak performance, the *Exascale* era being the current horizon. However, getting a high fraction of the available peak performance is more and more difficult. The Design of an efficient code that scales well on a supercomputer is a non-trivial task. Manycore processors are now common, and the scalability issue in this context is crucial. Code optimization requires advanced programming techniques, taking into account the specificities and constraints of the target architecture. Many challenges are to be considered from the standpoints of efficiency and expected performances. The current faster supercomputer, the *Supercomputer Fugaku*, has a peak of nearly 0.5 exaflops with 82% for the sustained performance on LinPack, and the average sustained performance for the top 5 machines is 75%. We can see that the increasing available power goes alongside with a better efficiency, most likely because of more efficient memory systems and a faster connection between the compute nodes. It is important to keep in mind that an effective HPC solution comes from a skillful combination of methods, programming, and machines [117]. The topic of *Optimization* is a very nice illustration of this observation because it has provided cutting-edge methods for solving (large-scale) problems, and the question of their efficient mapping onto large-scale supercomputers is crucial and challenging. We now present an overview of the fundamental aspects of optimization, this part comes from our work[117] and is provided here in the intention of a self-contained report.

3 Foundations and background

Operations research is the science of decision making. The goal is to derive suitable mathematical models for practical problems and study effective methods to solve them as efficient as possible. For this purpose, *mathematical programming* has emerged as a strong formalism for major problems. Nowadays, due to the increasing size of the market and the pervasiveness of network services, industrial productivity and customers services should scale up with a whooping need and a higher quality requirement. In addition, the interaction between business operators has reached a noticeable level of complexity. Consequently, for well established companies, dealing with optimal decisions is critical to survive, and the key to achieve this purpose is to exploit recent operation research advances. The objective is to give a quick and accurate answer to practical instances of critical decision problems. The role of operation research is also central in cutting-edge scientific investigations and technical achievements. A nice example is the application of the *traveling salesman problem* (TSP) on *logistics*, *genome sequencing*, *X-Ray crystallography*, and *microchips manufacturing*[5]. Many other examples can be found in real-world applications[108]. A nice introduction of combinatorial optimization and complexity can be found in [105, 39].

The noteworthy increase of supercomputers capability has boosted the enthusiasm for solving large-scale combinatorial problems. However, we still need powerful methods to tackle those problems, and afterward provide efficient implementation on modern computing systems. We really need to seat far beyond brute force or had hoc (unless genius) approaches, as increasingly bigger instances are under genuine consideration. Figure 1 displays an overview of a typical workflow when it comes to solving optimization problems.

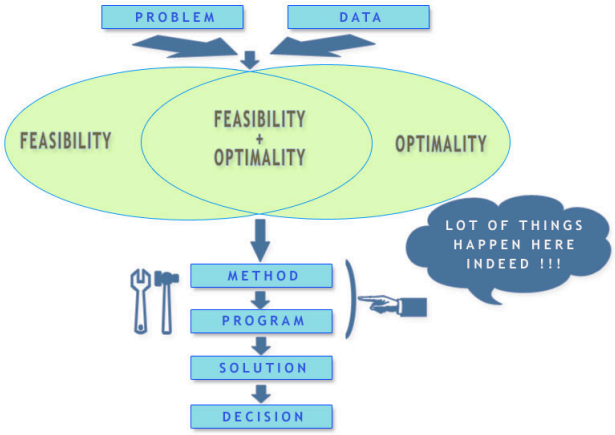


Figure 1: Typical operation research workflow

Most of common combinatorial problems can be written in the following form

$$\begin{cases} \text{minimize} & F(x) \\ \text{subject to} & P(x) \\ & x \in S, \end{cases} \tag{1}$$

where F is a polynomial, $P(x)$ a predicate, and S the working set, generally $\{0,1\}^n$ or $\mathcal{Z}^n$. The *predicate* is generally referred to as *feasibility constraint*, while F is the known as the *objective function*. In the case of a pure feasibility problem, F could be assumed to be constant. An important class of optimization problem involves a linear objective function and linear constraints, thus the following generic formulation

$$\begin{cases} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \end{cases} \tag{2}$$

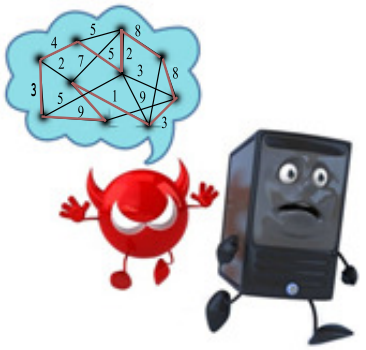
where $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $p \in \{0, 1, \cdots, n\}$. If $p = 0$ (resp. $p = n$), then we have a so-called *linear program* (resp. *integer linear program*), otherwise we have a *mixed integer program*. The corresponding acronyms are LP, ILP, and MIP respectively. In most cases, integer variables are *binary* $0-1$ *variables*. Such variables are generally used to indicate a choice. Besides linear objective functions, quadratic ones are also common, with a *quadratic term* proportional to $x^t Q x$. We now state some illustrative examples.

Example 1 The Knapsack Problem (KP)*[79]. The Knapsack Problem is the problem of choosing a subset of items such that the corresponding profit sum is maximized without having the weight sum to exceed a given capacity limit. For each item type i, either we are allowed to pick up at most* 1 *(binary knapsack)[35], or at most m_i (bounded knapsack), or whatever quantity (unbounded knapsack). The bounded case may be formulated as follows(3):*

$$\begin{cases} \textit{maximize} & \displaystyle\sum_{i=1}^{n} p_i x_i \\ \textit{subject to} & \\ & \displaystyle\sum_{i=1}^{n} w_i x_i \leq c \\ & x_i \leq m_i \quad i = 1, 2, \cdots, n \\ & x \in \{0,1\}^{n \times n} \end{cases} \tag{3}$$

Example 2 The Traveling Salesman Problem (TSP) *[5]. Given a valuated graph, the Traveling Salesman Problem is to find a minimum cost cycle that crosses each node exactly once (tour). Without lost of generality, we can assume positive cost for every arc and a zero cost for every disconnected pair of vertices. We formulated the problem as selecting a set of arcs (i.e.* $x_{ij} \in \{0,1\}$*) so as to have a tour with a minimum cost(4). Understanding how the way constraints are formulated implies a tour is left as an exercise for the reader.*

$$\begin{cases} \textit{minimize} & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \textit{subject to} & \\ & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1,\cdots,n, i \neq j \\ & \displaystyle\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1,\cdots,n, i \neq j \\ & x \in \{0,1\}^{n\times n} \end{cases} \tag{4}$$

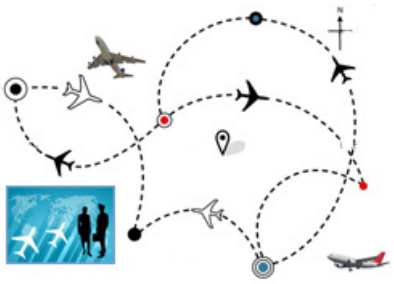


The TSP has an a priori $n!$ *complexity. Solving any instance with* $n = 25$ *using the current world fastest supercomputer (FUGAKU/0.5 exaflops) might require years of calculations.*

Example 3 The Airline Crew Pairing Problem (ACPP) *[127]. The objective of the ACPP is to find a minimum cost assignment of flight crews to a given flight schedule. The problem can be formulated as a set partitioning problem(5).*

$$\begin{cases} \textit{minimize} & c^T x \\ \textit{subject to} & Ax = 1 \\ & x \in \{0,1\}^n \end{cases} \tag{5}$$

In equation (5), each row of A *represents a flight leg, while each column represents a feasible pairing. Thus,* a_{ij} *tells whether or not flight* i *belongs to pairing* j*.*



In practice, the feasibility constraint is mostly the heart of the problem. This the case for the TSP, where the feasibility itself is a difficult problem (the *Hamiltonian cycle*). However, there are also notorious cases where the dimension of the search space S (i.e. n) is too large to be handled explicitly when evaluating the objective function. This is the case of the ACPP, where the number of valid pairings is too large to be included into the objective function in one time. We clearly see that we can either focus on the *constraints* or on *components* of the objective function. In both cases, the basic idea is to get rid of the part that makes the problem difficult, and then reintroduce it progressively following a given strategy. Combined with the well known *branch-and-bound* paradigm[26], these two approaches have led to well studied variants named *branch-and-cut*[126] and *branch-and-price*[12] respectively.

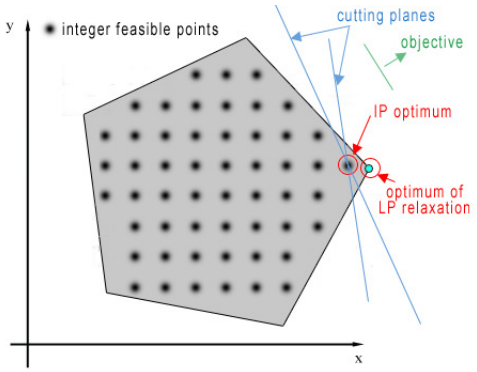


Figure 3: Integer programming & LP

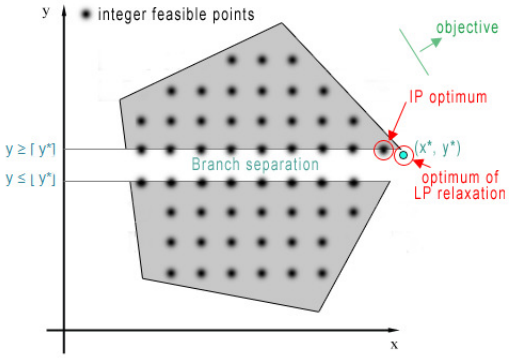


Figure 4: Branch-and-bound & LP

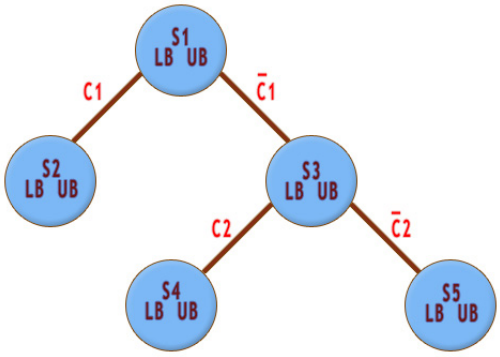


Figure 2: Branch-and-bound overview

The key ingredient of this connection between discrete and continuous optimization is *linear programming* (LP). Indeed, applying a *linear relaxation* on the exact formulation of a combinatorial problem, which means assuming continuous variables in place of integer variables, generally leads to an LP formulation from which lower bounds can be obtained (upper bounds are obtained on feasible guests, mostly obtained through heuristics). LP is also used to handle the set of constraints in a *branch-and-cut*, or to guide the choice of new components (*columns*) of the objective function in the *branch-and-price* scheme. Figure 3 depicts the linear relaxation of an IP configuration, while Figure 4 provides a sample snapshot of an LP driven branch-and-bound.

Linear programming has been intensively studied and has reached a very mature state, even

from the computing standpoint. Nowadays, very large scale LP can now be routinely solved using specialized software packages like CPLEX[135] or MOSEK[138].

Branch-and-bound and its variants can be applied to a mixed integer programming formulation by means of basic techniques like *Bender decomposition*[17] or *Lagrangian relaxation*[87]. Figure 5 depicts the basic idea behind these two approaches.

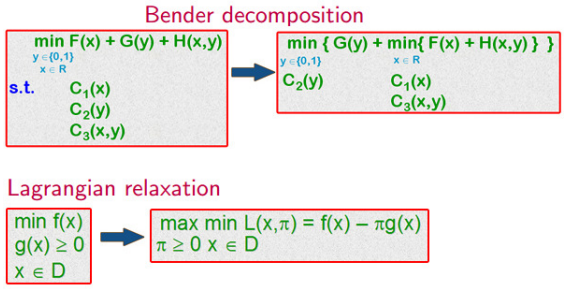


Figure 5: Bender decomposition & Langrangian relaxation

The later is likely to yield *non-differentiable optimization* (NDO) problems. Several approaches for NDO are described in the literature[65], including an oracle-based approach[7], which we will later describe in details as it illustrates our major contribution on that topic. Figure 6 gives an overview of an oracle based mechanism.

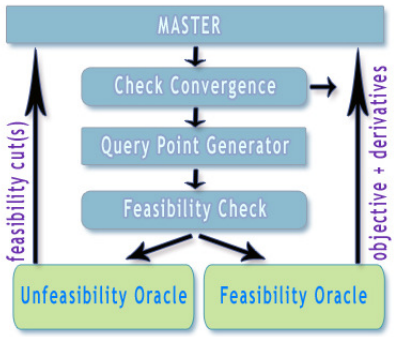


Figure 6: Oracle based optimization workflow

From the methodological point of view, optimization (both continuous and combinatorial) has been so far subject to intensive and fruitful investigations. New optimization paradigms or improved variants of classical techniques have reached an acceptable level of maturity, and have proved successful on number of notorious practical problems. However, in some cases, the expected level of performance can be achieved only through parallel implementation on large-scale supercomputers, especially with intractable (but pertinent) combinatorial problems. The idea is to combine the advantages of mathematically powerful algorithms with the capability of machines that have several processors. The existence of commercial multiprocessor computers has created substantial interest in exploring the use of parallel processing for solving optimization problems even for basic issues. The challenge is to find a suitable way to implement the aforementioned techniques (somehow irregular) on modern supercomputers (mostly tailored for regular computations) with an acceptable efficiency. We now provide technical details on how this can be tackled and what has been done.

4 Parallel optimization

Processing with supercomputers is mainly parallel computing. Theoretical complexity studies the intrinsic difficulty of the optimization problems and classify them accordingly. There is an important set of common problems that can be solved or approximated in polynomial time. However, as some of them are (recursively) solved to get the solution of more difficult problems, improvements are still expected whenever there is a room for that. A good example of this is the *shortest paths problem*, which appears as a sub-problem for the *multicommodity flow problem*[9]. Many other combinatorial problems are (known to be) difficult, thus the basic expectation with supercomputers is to be able to solve them in a reasonable time through an efficient parallelization of a chosen method. The main point with large-scale supercomputers (thus exascale ones) is the huge number of computing units, which implies a larger and deeper parallelism. The global topic of Optimization has mainly two components: *continuous optimization* and *discrete optimization*. However, because pure combinatorial problems might be too difficult to solve only from the combinatorial standpoint, many approaches have developed a bridge between the discrete universe and the continuous universe through geometric, analytic, and algebraic techniques such as *global optimization*, *semidefinite programming*, and *spectral theory*. Mixed integer programming formulations involve differentiable or non-differentiable objective functions. Non-differentiable configurations might come from the consideration of a *Lagrangian relaxation* approach, which moves a subset of the constraints (usually the harder ones) into the objective function. The efforts in the design of efficient algorithms for common combinatorial problems has lead to useful connections among problems (i.e. a solution for one can be used to construct a solution for another). As consequence, there is a set of reference optimization problems for which improved solutions are continuously tracked by researchers. For this purpose, *parallel computing* applied to all previously mentioned optimization paradigms is clearly worth considering.

An important set of discrete optimization problems are NP- complete[61]; hence their time complexity increases exponentially for all known algorithms. Consequently, parallel processing cannot achieve polynomial complexity on these problems without using at least an exponential number of processors (not counting data exchanges). However, the average-time complexity of heuristics and sub-optimal algorithms for a wide range of problems are polynomial[82, 129]. Significant advances have been made in the use of powerful heuristics and parallel processing to solve large scale discrete optimization problems. Number of problem instances that were considered computationally intractable on sequential machines are routinely solved on server-class symmetric multi-processors and workstation clusters. In conjunction with the increasing power of supercomputers, cutting-edge methods in optimization are expected to cope with very large-scale problems.

We get a direct impact of parallel computing in numerical optimization through the advances in parallel numerical algebra[29, 52, 49], with some of them being implemented into effective frameworks[3, 11, 2, 25, 124]. Encapsulating parallel linear algebra routines into optimization codes [43, 114] is a nice way to provide their power to the users without additional efforts. This is still a very critical and challenging topic since parallelizing the linear algebra kernels of optimization algorithms is not an easy task, and moving on the exascale era will made it more complex. For instance, matrix factorization updating process in quasi-Newton methods or active set strategies involves vector-vector operations that are not easy to parallelize efficiently [41]. According to Schnabel[112], parallel computing can be performed in numerical optimization through three levels:

- ◇ parallelization of the function and/or the derivative evaluations;
- ◇ parallelization of the linear algebra kernels;
- ◇ modifications of the basic algorithms in order to increase the degree of parallelism.

For the first two levels, several approaches have been developed [29, 106, 107, 54, 69] and we might also expect some outputs for heterogeneous systems. For most of interior point (IP) methods in linear programming (LP), quadratic programming (QP), and nonlinear programming, the

kernel is the solution of a special linear system [4, 19]. As the iterates approach the boundary of the feasible set or the optimal solution, the system becomes more and more ill-conditioned. Suitable strategies have been developed within a modified Cholesky factorization framework and successfully used in specialized codes as CPLEX[135], LOQO[137] and GUROBI[136]. Thus, efficient parallel versions of these strategies are highly desired, but challenging on large-scale supercomputers, especially for sparse systems. The paper of Durazzi and Ruggiero [50] presents a parallel approximated IP method for QP, based on a preconditioned Conjugated Gradient algorithm. D'Apuzzo and Marino [41] have proposed a parallel Potential Reduction algorithm for the convex quadratic bound constrained problem. A parallel decomposition approach is considered by Zanghirati and Zanni [132] for large scale QPs. Blomwall [23] has proposed a parallel implementation of a Riccati-based primal IP algorithm for multistage stochastic programming. Most of these contributions together with more recent ones consider conventional parallel architectures, the question is how well are they adaptable for complex systems (i.e. heterogeneous with NUMA memory for instance).

Regarding the third level, multi-directional search strategies [89] provide a high-level parallelism which can be exploited through a concurrent execution of the minimization processes. Adhoc or application specific algorithms are also concerned, particularly when large-scale instances are considered [31, 81]. Another case study in statistical model selection is analyzed by Gatu and Kontoghiorghes [62]. As many fields in numerical analysis, several algorithms in numerical optimization have been revisited because of parallelism considerations. In [56], approaches to expose parallelism through appropriate partitioning of mathematical programs are reported. Interior point strategies, because of their direct possibility of parallel implementation [42, 73], have received much attention compare to active set algorithms, and have stimulated intensive researches in order to understand and overcome their weak scaling on large supercomputers. Developments in object oriented software for coding and tuning linear algebra algorithms at a high level of abstraction are provided in [125, 130]

As previously said, many techniques have so far been developed to provide a bridge between continuous and discrete formulations. Recent successes based on such approaches include IP methods for discrete problems, the Goemans-Williamson relaxation of the maximum cut problem, the Chvatal cuts for the traveling salesman problem, and the Gilbert-Pollak's conjecture, to name a few. Parallel algorithms for discrete optimization problems can be obtained in many different ways including the classical *domain decomposition*. SPMD (Single Program Multiple Data) parallelization attempts to enlarge the exploration of the solution space by initiating multiple simultaneous searches towards the optimal solution. These approaches are well implemented by clustering methods. Byrd et al. [31, 30] and Smith and Schnabel[115] have developed several parallel implementations of the clustering method. Parallelization of classical paradigms have also been explored: *parallel dynamic programming*[63], *branch and bound*[26, 45], *tabu search, simulated annealing*, and *genetic algorithms*[109]. In the paper of Clementi, Rolim, and Urland [37], randomized parallel algorithms are studied for *shortest paths, maximum flows, maximum independent set*, and *matching problems*. A survey of parallel search techniques for discrete optimization problems are presented in [71]. The most active topics are those involved with searching over trees, mainly the *depth-first* and the *best-first* techniques and their variants. The use of parallel search algorithms in games implementation has been particularly successful, the case of IBM's Deep Blue [113] is illustrative. This topic is very active in *Artificial Intelligence* for which the question of efficient parallelization stands as one of the major HPC applications.

We now discuss what appears to us crucial points on the way to the Exascale when considering efficient implementations in both continuous optimization and combinatorial optimization.

5 Critical Numerical and Performance Challenges

Let first point out and describe the main issues when seeking an efficient implementation of the aforementioned paradigms on large-scale supercomputers.

- ◇ *Computing unit:* The generic compute node is likely to be a many-core processor. Seeking efficiency and scalability with many-core processors is a hard task [116]. As with any shared-memory system, the way to go is through the shared-memory paradigm. Thereby, we avoid explicit data exchanges, but there is more pressure on main memory accesses with a heavy concurrency that will be the main culprit of weak scalability. Vectorization is to be considered at the level of the linear algebra kernels, and this requires a suitable data organization.

- ◇ *Memory system:* One critical point here is the management of shared variables. Optimization techniques are likely to be iterative, so the access to these variables is repeated accordingly. For read-only accesses, the performance will depend on how good we are with memory caching, this aspect should be investigated deeply considering all iteration levels. For write accesses, the main issue is concurrency, with a special focus on iterative (in-place) updates. The case of non uniform memory access (NUMA) needs a special attention as most of many-core processors follow this specificity along with the corresponding packaging of the cores. It is likely that exascale machines will be equipped with such processors.

- ◇ *Numerical sensitivity:* A part from accuracy concerns, it is common to consider a lower precision in order to reach a faster flops through wider SIMD and also to speed-up the memory accesses. Iterative methods usually consider this adaptation under a global mixed-precision scheme. The main drawbacks with lower precision come from the potential lost of accuracy, which might led to wrong numerical results or slower convergence.

- ◇ *Heterogeneity:* The tendency with top class supercomputers is heterogeneity. The most common configuration is the classical CPU-GPU conjunction. GPUs has reached enough maturity to be considered for most of common computing tasks including those form linear algebra. It is likely that this will be and remain the typical scenario of the GPU consideration for the implementation of optimization algorithms. However, the well-known problem of CPU/GPU data exchanges is to be seriously considered in the standpoint of an iterative process. In case of high-precision computation with GPU, there might be some concerns about accuracy. In addition, a trade-off between accuracy and performance should be skillfully considered.

- ◇ *Synchronization:* Considering the notorious case of the *brand-and-bound*, which also stands as a typical connection between continuous optimization and combinatorial optimization, all active explorations running in parallel share some common variables (concurrent updates, critical values, ...) and conditions (termination/pruning, numerical/structural, ...). Synchronize in the context of a large-scale supercomputer is costly and the effect on the scalability is noteworthy.

- ◇ *Data exchanges:* This is the main source of a serious time overhead with distributed memory parallelism, which also generally includes the aforementioned mechanism (synchronization). A more general optimization scheme has several levels of iteration (of different natures), which yields a complex communication flow and topology. This aspect is certainly the most hindering on the way to parallel efficiency, as it consumes the major part of the global overhead.

- ◇ *Load balance:* Active subproblems might have different complexities and numerical characteristics, thus yielding unequal loads for the corresponding tasks. Beside the computing load, there is also some numerical characteristics that might impact the local runtime complexity on the compute nodes. This aspect is hard to fix without changing the way the computation is organized. The way a given (sub)problem is solved in optimization sometimes depends on

its specific structure, this makes difficult to predict the choice that will be made at runtime and thus complicates any prediction.

Many optimization problems are based on an objective function that is implicit or non-differentiable. To solve them with gradients-based approaches, we need to deal with an *oracle* that can return for a given point of the search space the evaluation of the objective function and the corresponding derivatives. Oracle-based Optimization is a powerful tool for general purpose optimization. To make this approach successful from the performance and numerical standpoints, it is important to (i) keep the number of calls to the oracle as low as possible, especially if it involves solving a difficult combinatorial problem; and (ii) take care of numerical issues that might extend the number of necessary iterations or lead to divergence. Oracle-based continuous optimization is better addressed with generic approaches so as to offer the possibility to treat most common combinatorial problems. However, number of important aspects still need to be seriously considered. We list some of them.

- ◇ The core of an oracle-based method in continuous optimization involves solving a linear system for the search direction used to get the next query point. It is crucial to have the solution accurate enough to be meaningful and keep us to the track. As we get close to the boundaries of the search space or to the optimal solution, the principal matrix of the linear system becomes ill-conditioned, thus making difficult the computation of the required solution. This fact severely increases the associated computational cost, unless we chose to sacrifice the accuracy, which will extend the number of outer iterations towards the solution. Thus, it is important to carefully address this issue, which belongs to the more general topic of solving ill-conditioned linear systems. However, there is probably a way to exploit the specific structure of the principal matrix in this case. The topic here is mainly that of *linear systems solving*, which has been extensively studied but remains difficult to make it as scalable as desired, especially with sophisticated iterative methods. Inter-processors communication and global synchronization mechanisms are what we should care about for this aspect on *exascale* systems.

- ◇ The query point generator of the cutting planes method looks for a guess within the localization set that corresponds to the polyhedral defined by the cuts accumulated so far. A good management of these cuts is crucial and their number linearly increases with the number of iterations. If the dimension of the problem is huge, or if we have already performed a large number of iterations, then the required memory space necessary to keep all the cuts will become significant, and this might slowdown the global memory efficiency, especially with complex memory systems like NUMA ones. One way to fight against this problem is to eliminate redundant cuts, or to keep only the minimal set of the cuts that corresponds to the same (or equivalent) polyhedral of the localization set. Doing this is not trivial as there are many valid selections. Another way is to aggregate the cuts instead of eliminating them. We could also weight the cuts according to their importance within the localization set. All theses thoughts have to be studied deeply, even through an experimental approach. However, we need to be careful as we could destroy the coherence of the localization set, thus impacting the convergence.

- ◇ Cutting planes methods are iterative, and the convergence is monitored by the calculation of the gap between the best solution found so far and the estimated lower/upper bound (ideally the optimal value of the objective function, but we don't have it). The process converges if: (a) the gap is below the tolerance threshold; (b) we have reached the maximum number of iterations (over the expectation); (c) a null gradient is returned by the oracle; (d) an incoherent information is provided by the oracle or calculated internally; (e) an unexpected critical issue (hardware/system or numerical) has occurred. The main focus here is the lower/upper bound estimation. This is usually obtained from the localization set (the cuts + the objectives), which might become heavy and numerically sensitive over the time (again, because of the large number of collected cuts and the global configuration). If the estimation

of the lower/upper bound is good enough, then we will perform more additional iterations or never converge (even if we should, either because we are already at the optimum or there is no further improvement). It is therefore important to address this problem and look for a robust approach. It makes sense to assign this calculation to single computing unit and broadcast the result to the whole computing system.

◇ Regarding the Newton linear system that is solved during inner iterations to get the search direction for the next query point, updating the principal matrix takes a serious overhead. Indeed, at each iteration, the matrix of the generated cuts is updated from A to $[A, u]$, where u is the new incoming cut, then we solve a linear system based on a principal matrix of the form

$$A \times diag(s^2) \times A^T, \tag{6}$$

where s is the vector of the so-called *slack variables* and $s^2 = (s_i^2)$. It is quite frustrating to solve this system from scratch at each iteration. Indeed, the principal matrix (6) seems to have a suitable form for a direct Cholesky factorization. The dream here is to keep on the desired factorization by means of efficient updates, thus a quadratic complexity instead of the cubic one for the factorization restarted from scratch. The current state-of-the-art in matrix computation, to the best of our knowledge, does not provides the aforementioned Cholesky update, this remains to be investigated including the parallelization from the perspective of running on a (large-scale) supercomputer.

◇ About the branch-and-bound, an important method for solving combinatorial problems (including approximations), very popular for MIP formulations, the main research direction from the HPC standpoint is through an efficient parallelization of the paradigm itself. Branch-and-bound is likely to yield an irregular computation scheme with an unpredictable path to the solution, thus making very challenging for efficient parallelization, especially on large-scale supercomputers. Among critical issues, we mention: *heavy synchronization, irregular communication pattern, huge amount of memory to handle the generated cuts*, *load unbalanced* and/or *non-regular memory accesses*. The management of the implicit recursion of the whole is difficult to keep scalable as the number of processors increases. An on-the-fly rescheduling of the tasks might be necessary at some points in order to adapt to branching mispredictions or severe load imbalance. In conjunction with continuous optimization, there are effective generic frameworks for the branch-and-bound associated with continuous optimization solvers [28], such frameworks should be made parallel at design time.

6 Conclusion

Optimisation is a central topic, which combines advances in applied mathematics and technical computing. Powerful methods have been developed and are still improved to solve difficult but relevant real-life problems. As the power of supercomputers is significantly increasing, there is an instinctive desire for being able to routinely solve large-scale problems. This raises the challenge of efficient implementations of cutting-edge optimization techniques on large-scale supercomputers. Parallel optimization is the main topic involved in this context, and the main concern is scalability. Ideally, the most powerful optimization methods should be scalable enough to yield the most efficient solutions to the target problems. However, the global and internal structures of modern supercomputers make them not easy to program efficiently, especially with too specific approaches like the ones from optimization. On the way to the exascale, this will be exacerbated by the complexity of the systems, but the efforts are worth it.

References

[1] C. J. Adcock and N. Meade, *A simple algorithm to incorporate transaction costs in quadratic optimization*, European Journal of Operational Research, **7**, 85-94, 1994.

[2] E. Agullo, B. Hadri, H. Ltaief and J. Dongarra, *Comparative study of one-sided factorizations with multiple software packages on multi-core hardware*, SC'09: International Conference for High Performance Computing, 2009

[3] E. Agullo, J. Dongarra, B. Hadri, J. Kurzak, J. Langou, J. Langou, H. Ltaief, P. Luszczek, and A. YarKhan, *PLASMA: Parallel Linear Algebra Software for Multicore Architectures, Users? Guide*, `http://icl.cs.utk.edu/plasma/`, 2012.

[4] E. D. Andersen, J. Gondzio, Cs. Mészáros, and X. Xu, *Implementation of interior point methods for lardge scale linear programming*, T. Terlaky (Ed), Interior-point Methods of Mathematical Programming, Kluwer Academic Publishers, pp. 189-252, 1996.

[5] Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J., *The Traveling Salesman Problem - A Computational Study*, Princeton Series in Applied Mathematics, 2006.

[6] F. Babonneau and J.-P. Vial. ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems. Technical report, Logilab, Hec, University of Geneva, June 2005.

[7] Babonneau, Frédéric and Beltran, Cesar and Haurie, Alain and Tadonki, Claude and Vial, Jean-Philippe, *Proximal-ACCPM: a versatile oracle based optimization method*, 9, 69–92, 2007.

[8] F. Babonneau, O. du Merle, and J.-P. Vial. Solving large scale linear multicommodity flow problems with an active set strategy and Proximal-ACCPM. *Operations Research*, 54(1):184–197, 2006.

[9] F. Babonneau, *Solving the multicommodity flow problem with the analytic center cutting plane method*, PhD thesis, University of Geneva, `http://archive-ouverte.unige.ch/unige:396`, 2006.

[10] Bader, D.A., Hart, W.E., Phillips, C.A., *Parallel Algorithm Design for Branch and Bound*, In: Greenberg, H.J. (ed.) Tutorials on Emerging Methodologies and Applications in Operations Research, ch. 5, Kluwer Academic Press, Dordrecht, 2004.

[11] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. Curfman McInnes, B. Smith, and H. Zhang, *PETSc Users Manual. Revision 3.2*, Mathematics and Computer Science Division, Argonne National Laboratory, September 2011

[12] Barnhart et. al. , *Branch and Price : Column Generation for Solving Huge Integer Programs*,Operation Research, Vol 46(3), 1998.

[13] C. Barnhart, A. M. Cohn, E. L. Johnson, D. Klabjan, G. L. Nemhauser, P. H. Vance, *Airline Crew Scheduling*, Handbook of Transportation Science International Series in Operations Research & Management Science Volume 56, pp 517-560, 2003.

[14] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[15] C. Beltran, C. Tadonki, J.-Ph. Vial, *Semi-Lagrangian relaxation* , Computational Management Science Conference and Workshop on Computational Econometrics and Statistics, Link, Neuchatel, Switzerland, April 2004 .

[16] C. Beltran, C. Tadonki, and J.-P. Vial. Semi-lagrangian relaxation. Technical report, Logilab, HEC, University of Geneva, 2004.

[17] J. F. Benders, *Partitioning procedures for solving mixed-variables programming problems*, Numerische Mathematik 4(3), pp. 238?252, 1962.

[18] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Computational Management Science*, 2:3–19, 2005. Initially appeared in *Numerische Mathematik*, 4: 238-252, 1962.

[19] H. Y. Benson, D. F. Shanno, R.J. Vanderbei, *Interior-point methods for convex nonlinear programming: jamming and comparative numerical testing*, Op. Res. and Fin. Eng., ORFE-00-02-Princeton University, 2000 .

[20] C. Berger, R. Dubois, A. Haurie, E. Lessard, R. Loulou, and J.-P. Waaub. Canadian MARKAL: An advanced linear programming system for energy and environmental modelling. *INFOR*, 30(3):222–239, 1992.

[21] D. Bienstock, *Computational study of a family of mixed-integer quadratic programming problems*, Math. Prog. **74**, 121-140, 1996.

[22] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R. C. Whaley*ScaLAPACK*, `http://www.netlib.org/scalapack`, 2012.

[23] J. Blomwall, *A multistage stochastic programming algorithm suitable for parallel computing*, Parallel Computing, 29, 2003.

[24] R. A. Bosh and J.A. Smith, *Separating Hyperplanes and the Authorship of the Disputed Federalist Papers*, American Mathematical Monthly, Volume 105, No 7, pp. 601-608, 1995.

[25] Bosilca, G., Bouteiller, A., Danalis, A., Faverge, M., Haidar, H., Herault, T., Kurzak, J., Langou, J., Lemariner, P., Ltaief, H., Luszczek, P., YarKhan, A., Dongarra, J. *Distributed Dense Numerical Linear Algebra Algorithms on Massively Parallel Architectures: DPLASMA*, University of Tennessee Computer Science Technical Report, UT-CS-10-660, Sept. 15, 2010.

[26] S. Boyd , J. Mattingley, *Branch and Bound Methods*, Notes for EE364b, Stanford University, Winter 2006-07 (`http://see.stanford.edu/materials/lsocoee364b/17-bb_notes.pdf`).

[27] O. Briant and D. Naddef. The optimal diversity management problem. *Operations research*, 52(4), 2004.

[28] O. Briant, C. Lemaréchal,K. Monneris,N. Perrot,C. Tadonki,F. Vanderbeck,J.-P. Vial,C. Beltran,P. Meurdesoif, *Comparison of various approaches for column generation*, Eigth Aussois Workshop on Combinatorial Optimization, 5-9 january 2004.

[29] A. Buttari, J. Langou, J. Kurzak, and J. Dongarra, *A class of parallel tiled linear algebra algorithms for multicore architectures*, Parallel Computing 35: 38-53, 2009.

[30] R. H. Byrd, et al., *Parallel global optimization for molecular configuration problem*, in Proceedings of the 6th SIAm Conference on Parallel Processing for Scientifi Computation, SIAM, Philadelphia, 1993.

[31] R. H. Byrd, et al., *Parallel global optimization: numerical methods, dynamic scheduling methods, and application to molecular configuration*, in B. Ford and A. Fincham (Eds), Parallel Computation, Oxford University Press, Oxford, pp. 187-207, 1993 .

[32] CPLEX, http://www.ilog.com/products/cplex/

[33] D. Carlson, A. Haurie, J.-P. Vial, and D.S. Zachary. Large scale convex optimization methods for air quality policy assessment. *Automatica*, 40:385–395, 2004.

[34] T. J. Chang, N. Meade, J. E. Beasley, Y. M. Sharaiha, *Heuristics for cardinality constrained portfolio optimization*, Computers & Operation Research, 27, pp. 1271-1302, 2000.

[35] V. Chvátal, *Hard Knapsack Problems*, Operations Research, Vol. 28(6),pp. 1402-1411,1980.

[36] V. Chvatal, *Linear Programming*, W. H. Freeman Compagny, Series of Books in the Mathematical Sciences, 1983.

[37] A. Clementi, J. D. P. Rolim, and E. Urland, *Randomized Parallel Algorithms*, LLNCS, A. Ferreira and P. Pardalos (Eds), pp. 25-48, 1995.

[38] Constantinides G. M. and Malliaris A.G., *Portfolio theory*, Finance ed R.A. Jarrow, V. Maksimovic and W. T. Ziemba, Elsevier-Amsterdam, 1-30, 1995.

[39] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, A. Schrijver, *Combinatorial Optimization*, John Wiley & Sons, 1998.

[40] T. Crainic, B. Le Cun, and C. Roucairol, *Parallel Branch-and-Bound Algorithms*, In: Talbi, E. (ed.) Parallel Combinatorial Optimization, ch. 1, Wiley, Chichester 2006.

[41] M. D'Apuzzo and M. Marino, *Parallel computation issued of an interior point method for solving large bound-constrained quadratic programming problems*, Parallel Computing, 29, 2003.

[42] M. D'Apuzzo, et al., *A parallel implementation of a potential reduction algorithm for box-constrained quadratic programming*, in LLNCS pp. 839-848, Europar2000, Spriger-Verlag, Berlin, 2000.

[43] M. D'Apuzzo, et al., *Nonlinear optimization: a parallel linear algebra standpoint*, Handbook of Parallel Computing and Statistics, E. J. Kontoghiorges (Ed.), New-York, 2003.

[44] M. D'Apuzzo, et al.,*Parallel computing in bound constrained quadratic programming*, Ann. Univ. Ferrara-Sez VII-Sc. Mat. Supplemento al XLV pp. 479-491, 2000.

[45] A. De Bruin, G. A. P. Kindervater, H. W. J. M. Trienekens , *Towards and abstract parallel branch and bound mahcine*, LLNCS, A. Ferreira and P. Pardalos (Eds), pp. 145-170, 1995.

[46] Z. Degrave and M. Peeters, *Benchmark Results for the Cutting Stock and Bin Packing Problem*, Research Report No 9820 of the Quantitative Methods Group, Louvain, Belgique, 1998.

[47] L. Drouet, C. Beltran, N.R. Edwards, A. Haurie, J.-P. Vial, and D.S. Zachary. An oracle method to couple climate and economic dynamics. In A. Haurie and L. Viguier, editors, *Coupling climate and economic dynamics*. Kluwer (to appear), 2005.

[48] L. Drouet, N.R. Edwards, and A. Haurie. Coupling climate and economic models in a cost-benefit framework: A convex optimization approach. *Environmental Modeling and Assessment*, to appear in 2005.

[49] I. S. Duff, H. A. VanDer Vorst, *Developments and trends in the parallel solution of linear systems*, Parallel Computing 25, pp. 13-14, 1999.

[50] C. Durazzi and V. Ruggiero, *Numerical solution of special linear and quadratic programs via a parallel interior-point method*, Parallel Computing, 29, 2003.

[51] O. Du Merle and J.-P. Vial. Proximal ACCPM, a cutting plane method for column generation and Lagrangian relaxation: application to the p-median problem. Technical report, Logilab, University of Geneva, 40 Bd du Pont d'Arve, CH-1211 Geneva, Switzerland, 2002.

[52] Fengguang, S., Tomov, S., Dongarra, J., *Efficient Support for Matrix Computations on Heterogeneous Multi-core and Multi-GPU Architectures*, University of Tennessee Computer Science Technical Report, UT-CS-11-668, June 16, 2011.

[53] A. Ferreira and P. M. Pardalos (Eds.), *Solving Combinatorial Optimization Problems in Parallel*, LLNCS-Springer 1054, 1995 .

[54] A. Ferreira and P. M. Pardalos, *Parallel Processing of Discrete Optimization Problems*, DIMACS Series Vol. 22, American Mathematical Society, 1995.

[55] M. C. Ferris and T.S. Munson, *Interior Point Methods for Massive Support Vector Machines*, Cours/SÈminaire du 3^e cycle romand de recherche opÈrationnelle, Zinal, Switzerland, march 2001.

[56] M. C. Ferris, J. D. Horn, *Partitioning mathematical programs for parallel solution*, Mathematical Programming 80, PP. 35-61, 1998.

[57] Ferris, M., *GAMS: Condor and the grid: Solving hard optimization problems in parallel*, Industrial and Systems Engineering, Lehigh University, 2006.

[58] L. G. Fishbone and H. Abilock. MARKAL, a linear programming model for energy systems analysis: Technical description of the BNL version. *International Journal of Energy Research*, 5:353–375, 1981.

[59] E. Fragnière and A. Haurie. A stochastic programming model for energy/environment choices under uncertainty. *International Journal of Environment and Pollution*, 6(4-6):587–603, 1996.

[60] E. Fragnière and A. Haurie. MARKAL-Geneva: A model to assess energy-environment choices for a Swiss Canton. In C. Carraro and A. Haurie, editors, *Operations Research and Environmental Management*, volume 5 of *The FEEM/KLUWER International Series on Economics, Energy and Environment*. Kluwer Academic Publishers, 1996.

[61] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Morgan Kaufmann, 1979.

[62] C. Gatu and E. J. Kontoghiorghes, *Parallel aalgorithms for computing all possible subset regression models using the QR decomposition*, Parallel Computing, 29, 2003.

[63] M. Gengler, *An introduction to parallel dynamic programming*, LLNCS, A. Ferreira and P. Pardalos (Eds), pp. 86-114, 1995.

[64] A. M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[65] J.-L. Goffin and J.-P. Vial. Convex nondifferentiable optimization: A survey focussed on the analytic center cutting plane method. *Optimization Methods and Software*, 174:805–867, 2002.

[66] J.-L. Goffin and J.-P. Vial. Shallow, deep and very deep cuts in the analytic center cutting plane method. *Mathematical Programming*, 84:89–103, 1999.

[67] J. L. Goffin, A. Haurie, and J. P. Vial, *Decomposition and nondifferentiable optimization with the projective algorithm* Management Science, **37**, 284-302.

[68] J.-L. Goffin, Z. Q. Luo, and Y. Ye. Complexity analysis of an interior point cutting plane method for convex feasibility problems. *SIAM Journal on Optimization*, 69:638–652, 1996.

[69] J. Gondzio, A. Grothey, *Parallel Interior Point Solver for Structured Quadratic Programs: Application to Financial Planning Problems*, RR MS-03-001, School of Mathematics, University of Edinburgh, 2003.

[70] J. Gondzio, O. du Merle, R. Sarkissian and J.P. Vial, *ACCPM - A Library for Convex Optimization Based on an Analytic Center Cutting Plane Method*, European Journal of Operational Research, 94, 206-211, 1996.

[71] A. Grama and V. Kumar, *State-of-the-Art in Parallel Search Techniques for Discretes Optimization Problems*, Personnal communication, 1993 .

[72] M. Guignard and S. Kim. Lagrangean decomposition: a model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39:215–228, 1987.

[73] A. Gupta, G. Karypis, V. Kumar, *A highly scalable parallel algorithm for sparse matrix factorization*, IEEE TPDS 8(5), pp. 502-520.

[74] N. H. Hakansson, *Multi-period mean-variance analysis: Toward a theory of portfolio choice*, Journal of Finance, **26**, 857-884, 1971.

[75] J. Han and M. Kamber, *Data Mining: Concept and Techniques*, Morgan Kaufmann Publishers, 2000.

[76] P. Hansen, N. Mladenovic, and D. Perez-Brito. Variable neighborhood decomposition search. *Journal of Heuristics*, 7:335–350, 2001.

[77] A. Haurie, J. Kübler, A. Clappier, and H. Van den Bergh. A metamodeling approach for integrated assessment of air quality policies. *Environmental Modeling and Assessment*, 9:1–122, 2004.

[78] J. L. Houle, W. Cadigan, S. Henry, A. Pinnamanenib, S. Lundahlc, *Database Mining in the Human Genome Initiative*, `http://www.biodatabases.com/whitepaper01.html`

[79] B. Hunsaker, C. A. Tovey, *Simple lifted cover inequalities and hard knapsack problems*, Discrete Optimization 2(3), pp. 219-228, 2005.

[80] N. J. Jobst, M. D. Horniman, C. A. Lucas, and G. Mitra, *Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints*, Quantitative finance, Vol. 1, p. 1-13, 2001.

[81] L. Jooyounga, et al., *Efficient parallel algorithms in global optimization of potential energy functions for peptides, proteins, and crystals*, Computer Physics Communications 128 pp. 3999-411, 2000.

[82] Judea Pearl, *Heuristics-Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA, 1984.

[83] O. Kariv and L. Hakimi. An algorithmic approach to network location problems. ii: the p-medians. *SIAM Journal of Applied Mathematics*, 37(3):539–560, 1979.

[84] J. Kepner, *MatlabMPI*, `http://www.ll.mit.edu/MatlabMPI/`

[85] M. Kleinberg, C.H. Papadimitriou, and P. Raghavan, *Segmentation Problems*, ACM Symposium on Theory of Computing, 1998, pp. 473-482.

[86] E. K. Lee and J. E. Mitchell , *Computational experience of an interior-point SQP algorithm in a parallel branch-and-bound franmework*, Proc. High Perf. Opt. Tech., 1997.

[87] C. Lemaréchal, *Lagrangian relaxation*, M. Junger and D. Naddef (Eds.): Computat. Comb. Optimization, LNCS 2241, pp. 112?156, 2001.
`http://link.springer.com/content/pdf/10.1007%2F3-540-45586-8_4`

[88] C. Lemaréchal. Nondifferentiable optimization. In G.L. Nemhauser, A.H.G Rinnooy Kan, and M.J. Todd, editors, *Handbooks in Operations Research and Management Science*, volume 1, pages 529–572. North-Holland, 1989.

[89] R. M. Lewis and V. J. Torczon, *Pattern search methods for linearly constrained minimization*, SIAM Journal of Optimization, 10, pp. 971-941, 2000.

[90] D. Li and W. L. Ng, *Optimal dynamic portfolio selection: Multi-period mean-variance formulation*, Math. Finance **10**, 387-406, 2000.

[91] MOSEK, http://www.mosek.com/.

[92] O. L. Mangasarian, R. Setino, and W. Wolberg, *Pattern Recognition via linear programming: Theory and Applications to Medical Diagnosis*, 1990.

[93] O. L. Mangasarian, W.N. Street, and W.H. Wolberg, *Breast Cancer Diagnosis and prognosis via linear programming*, Operation research, Vol. 43, No. 4, July-August 1995, pp. 570-577.

[94] O. L. Mangasarian, *Linear and Non-linear Separation of Patterns by linear programming*, Operations Research, 13, pp. 444-452.

[95] R. Mansini and M. G. Speranza, *Heuristic algorithms for the portfolio selection poblem with minimum transaction lots*, Eur. Jour. Op. Res., **114**, 219-223, 1999.

[96] H. Markowitz, *Portfolio Selection: Efficient Diversification of Investment*, John Wiley & Sons, New-York, 1959.

[97] H. Markowitz, *Portfolio Selection*, The Journal of Finance **1**, 77-91, 1952.

[98] A. Migdalas, G. Toraldo, and V. Kumar, *Nonlinear optimization and parallel computing*, Parallel Computing 29, pp. 375-391, 2003.

[99] J. Mossin, *Optimal multiperiod portfolio policies*, J. Business, **41**, 215-229, 1968.

[100] Y. Nesterov and A. Nemirovsky. *Interior Point Polynomial Algorithms in Convex Programming: Theory and Applications*. SIAM, Philadelphia, Penn., 1994.

[101] Y. Nesterov and J.-P. Vial. Homogeneous analytic center cutting plane methods for convex problems and variational inequalities. *SIAM Journal on Optimization*, 9:707–728, 1999.

[102] Y. Nesterov. Complexity estimates of some cutting plane methods based on the analytic center. *Mathematical Programming*, 69:149–176, 1995.

[103] Y. Nesterov. *Introductory Lectures on Convex Optimization, a Basic Course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, 2004.

[104] P.S. Pacheco, *Parallel Programming with MPI*, Morgan Kaufmann, 1997.

[105] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*, Prentice-Hall 1982.

[106] P. M. Pardalos, A. T. Phillips, and J. B. Rosen, *Topics in Parallel Computing in Mathematical Programming*, Science Press, 1993.

[107] P. M. Pardalos, M. G. C. Resende, and K. G. Ramakrishinan (eds), *Parallel Processing of Discrete Optimization Problems*, DIMACS Series Vol. 22, American Mathematical Society, 1995.

[108] Paul A. Jensen and Jonathan F. Bard, *Operations Research - Models and Methods*, John Wiley and Sons , 2003.

[109] Per. S. Lauren, *Parallel heuristic search - Introduction and new approach*, LLNCS, A. Ferreira and P. Pardalos (Eds), pp. 248-274, 1995.

[110] G. Reinelt. Tsplib, 2001. http://www.iwr.uni-heidelberg.de / groups / comopt / software / TSPLIB95.

[111] P. A. Samuelson, *Lifetime portfolio selection by dynamic stochastic programming*, Rev. Econ. Stat. **51**, 239-246, 1969.

[112] R. B. Schnabel, *A view of the limitation, opportunities, and challenges in parallel nonlinear optimization*, Parallel Computing 21(6), pp. 875-905, 1995.

[113] Scott Hamilton and Lee Garber, *Deep Blue's hardware-software synergy*, IEEE Computer, 30(10), pp. 29-35, 1997.

[114] Y. Shinano, T. Fujie, *ParaLEX: A Parallel Extension for the CPLEX Mixed Integer Optimizer*, Recent Advances in Parallel Virtual Machine and Message Passing Interface Lecture Notes in Computer Science Volume 4757, pp 97-106, 2007.

[115] S. L. Smith, R. B. Schnabel, *Centralized and distributed dynamic scheduling for adaptative parallel algorithms*, in P. Mehrotra, J. Saltz, R. Voight (Eds), Unstructured Scientific Computation on Scalable Multiprocessors, MIT Press, pp. 301-321, 1992.

[116] C. Tadonki, *Scalability on Manycore Machines*
https://www.cri.ensmp.fr/people/tadonki/talks/Scalability.pdf

[117] C. Tadonki, *High Performance Computing as Combination of Machines and Methods and Programming*
HDR Thesis, Université Paris Sud-Paris XI, 2013.

[118] C. Tadonki and J.-P. Vial, *Efficient algorithm for linear pattern separation*, (to appear in) International Conference on Computational Science, ICCS04 (LNCS/Springer), Krakow, Poland, June 2004 .

[119] C. Tadonki, C. Beltran and J.-P. Vial , *Portfolio management with integrality constraints*, Computational Management Science Conference and Workshop on Computational Econometrics and Statistics, Link, Neuchatel, Switzerland, April 2004 .

[120] C. Tadonki, J.-P. Vial, *Efficient Algorithm for Linear Pattern Separation*, International Conference on Computational Science, ICCS04 (LNCS/Springer), Krakow, Poland, June 2004.

[121] C. Tadonki, *A Recursive Method for Graph Scheduling*, International Symposium on Parallel and Distributed Computing (SPDC), Iasi, Romania, July 2002

[122] C. Tadonki, *Parallel Cholesky Factorization*, Workshop on Parallel Matrix Algorithm and Applications (PMAA), Neuchatel, Switzerland, August 2000.

[123] E.-G. Talbi (Editor), *Parallel Combinatorial Optimization*,Wiley Series on Parallel and Distributed Computing,2006.

[124] S. Tomov R. Nath P. Du J. Dongarra, *MAGMA: Matrix Algebra on GPU and Multicore Architectures*, `http://icl.cs.utk.edu/magma`, 2012.

[125] R. A. Van de Geijn, *Using PLAPACK*, The MIT Press, 1997.

[126] Vance et. al. , *Using Branch-and-Price-and-Cut to solve Origin-Destination Integer Multi-commodity Flow problems*,Operation Research, Vol 48(2), 2000.

[127] P. H. Vance, A. Atamturk, C. Barnhart, E. Gelman, and E. L. Johnson, A. Krishna, D. Mahidhara, G. L. Nemhauser, and R. Rebello, *A Heuristic Branch-and-Price Approach for the Airline Crew Pairing Problem*, 1997.

[128] M.S. Viveros, J.P. Nearhos, M.J. Rothman, *Applying Data Mining Techniques to a Health Insurance Information System*, 22nd VLDB Conference, Mumbai(Bombay), India,1996, pp. 286-294.

[129] B. W. Wah, G.-J. Li, and C. F. Yu, *Multiprocessing of combinatorial search problems*, IEEE Computer, June 1985.

[130] R. C. Whaley, et al., *Automated empirical optimizations of software and the ATLAS project*, Parallel Computing 27, pp. 3-35, 2001.

[131] M. R. Young, *A minimax portfolio selection rule with linear programming solution*, Management Science **44**, 673-683, 1992.

[132] G. Zanghirati and L. Zanni, *A parallel solver for large quadratic programs in training support vector machines*, Parallel Computing, 29, 2003.

[133] T. Zariphoulou, *Investment-consumption models with transactions costs and Marko chain parameters*, SIAM J. Control Optim **30**, 613-636, 1992.

[134] X. Y. Zhou and D. Li, *Continuous-Time mean-variance portfolio selection: A stochastic LQ framework*, Appl. Math. Optim. **42**, 19-33, 2000.

[135] *CPLEX*, `http://www.ilog.com/products/cplex`

[136] *GUROBI*, `https://www.gurobi.com`

[137] *LOGO*, `http://www.orfe.princeton.edu/ loqo/`

[138] *MOSEK*, `http://www.mosek.com/`